Zaigham Mahmood *Editor*

Fog Computing

Concepts, Frameworks and Technologies

Springer

Fog Computing

Zaigham Mahmood
Editor

Fog Computing

Concepts, Frameworks and Technologies

Springer

Editor
Zaigham Mahmood
Debesis Education
Derby, UK

and

Shijiazhuang Tiedao University
Hebei, China

ISBN 978-3-319-94889-8 ISBN 978-3-319-94890-4 (eBook)
https://doi.org/10.1007/978-3-319-94890-4

Library of Congress Control Number: 2018946577

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To
Rehana Zaigham Mahmood
For her Love and Support

Preface

Overview

Fog Computing is a distributed computing environment that extends the Cloud Computing paradigm to fully support the Internet of Things (IoT) vision. It aims to push the intelligence, processing and storage of data closer to the *Edge* of the network to provide computer-related services more immediately and near to the interconnected smart *things* that form part of the IoT. For this reason, Fog Computing is sometimes also referred to as Edge Computing. The two terms are often used interchangeably; however, there are subtle differences. Fog Computing market is currently valued at $22.3 million in 2018; it is expected to expand at an explosive rate to grow to around $205 million over the next five years. With such high level of connectivity of smart devices, Fog Computing appears to be the next big thing for the IoT vision.

Although the Cloud paradigm is highly attractive for data-intensive processing requirements, there are a number of inherent limitations that the Fog vision aims to resolve: by reducing the Internet/service latency, increasing the context awareness, conserving the network bandwidth, improving the quality of provision, enhancing the operational efficiency, providing the improved user experience and speeding up the real-time processing as well as storing of sensitive data close to where the data are generated. Fog paradigm also provides better support for mobility and the use of mobile communication.

In brief, whereas in Cloud Computing, response times are in minutes and often days, data storage periods are more longer time and often permanent, and location coverage is too wide and often global; in Fog Computing, response times are in milliseconds or sub-seconds, data storage periods are transient, and location coverage is local and with increased awareness. These benefits are mainly due to the localization of nodes in the Fog architecture that supports vertically isolated latency-sensitive applications by providing ubiquitous, scalable, layered and federated network connectivity. Whereas the Cloud environments are often geographically a long way away from the end-user devices and rely heavily on the wider Internet bandwidths, the Fog services are much closer to end users with dense geographical distribution. Fog Computing is currently being successfully deployed in numerous commercial and industrial application scenarios that require both Fog localization and Cloud globalization, including: smart grids, smart homes and cities, connected vehicles, self-drive cars and trains, traffic light system, healthcare management and numerous other cyber-physical systems.

With this background, this book aims to investigate development approaches, architectural mechanisms and measurement metrics for building smart adaptable environments and discuss the key-related topics such as device connectivity, security and interoperability and communication methods. Thirty-four researchers and practitioners of international repute have presented latest research on frameworks, technologies, current trends and suggestions for further enhancement of the IoT, Fog Computing and related paradigms.

Hopefully, *Fog Computing: Concepts, Frameworks and Technologies* will fill a gap with respect to Fog Computing environment and architecture, and extend the existing body of knowledge in this field.

Objectives

The aim of this volume is to present and discuss the state of the art in terms of concepts, frameworks, underlying technologies, methodologies, opportunities and issues, as well as present future directions and developments. The core objectives include:

- Capturing the state-of-the-art research and practice with respect to the frameworks and technologies relevant to Fog Computing.
- Discussing the Internet of Things (IoT) vision in terms of Fog Computing and other related distributed computing paradigms.
- Developing a complete reference for students, researchers and practitioners of distributed computing systems and Fog/Edge environments.
- Identifying further research directions and technological innovations in relation to distributed computing environments such as the IoT.

Organization

There are thirteen chapters in *Fog Computing: Concepts, Frameworks and Technologies*. These are organized into three parts:

Part I: Concepts and Principles

This section has a focus on concepts and underlying principles relating to the Fog Computing vision. There are four contributions. The first chapter introduces the concepts and principles of Fog Computing and discusses the related IoT paradigms such as Cloud Computing. The second contribution presents a detailed literature analysis and outlines the characteristics, models, theories and different application scenarios relating to Fog Computing. The next chapter presents the key issues surrounding the Cloud and Fog paradigms and proposes emerging dimensions discussing the Fog Computing landscape in Africa. The final contribution explores the contributing factors for the adoption of Fog Computing, using the TOE framework approach, with a special focus on developing countries.

Part II: Frameworks and Technologies

This part of the book comprises five chapters that focus on methodologies and technologies. The first chapter provides an analysis of sensordata formats in the context of smart cities and develops a data retrieval tool for filtering the open datasets. The next contribution discusses the performance issues and presents means of achieving high performance in the case of Fog environments. The third chapter analyses users' expectations in terms of QoE aspect of Fog environments and suggests User Expectations Metrics to determine the QoE. The next contribution proposes software engineering approaches for Fog-basedarchitectural design using UML and the four-step-rule-set method. The final submission presents a data utility model for developing data-intensiveapplications for Fog environments.

Part III: Advanced Topics and Future Directions

There are four chapters in this section that focus on ongoing research on advanced topics and future directions. There are four chapters here. The first one elaborates on the analysis of sensitive data using context-aware systems for cyber-threat intelligence. The next contribution presents a novel software engineering framework for service and Cloud Computing that is equally applicable to the Fog paradigm. Chapter 3 presents data and computation movement between the Edge and the Cloud architectures using the DITAS approach to improve data management in Fog environments. The final contribution of the book discusses SDN-based heterogeneous vehicular networking to help overcome the bottlenecks posed by the traditional networking architectures.

Target Audiences

The current volume is a reference text aimed at supporting a number of potential audiences, including the following:

- *Communication engineers, software developers* and *network security specialists* who wish to adopt the newer approaches to ensure the security of data and devices and seamless connectivity in the IoT environments.
- *Students* and *lecturers* who have an interest in further enhancing the knowledge of technologies, mechanisms and frameworks relevant to the Fog/Edge Computing vision from a distributed computing perspective.
- *Researchers* and *practitioners* in this field who require up-to-date knowledge of the current practices, mechanisms, approaches and limitations relevant to the Fog/Edge Computing and the Internet of Things vision.

Derby, UK Zaigham Mahmood
Hebei, China
June 2018

Acknowledgements

The editor acknowledges the help and support of the following colleagues during the review, development and editing phases of this text:

- Prof. Zhengxu Zhao, Shijiazhuang Tiedao University, Hebei, China
- Dr. Alfredo Cuzzocrea, University of Trieste, Trieste, Italy
- Prof. Jing He, Kennesaw State University, Kennesaw, GA, USA
- Josip Lorincz, FESB-Split, University of Split, Croatia
- Aleksandar Milić, University of Belgrade, Serbia
- Dr. S. Parthasarathy, Thiagarajar College of Engineering, Tamil Nadu, India
- Daniel Pop, Institute e-Austria Timisoara, Univ. of Timisoara, Romania
- Dr. Pethuru Raj, Reliance Jio Infocomm, Bangalore, India
- Dr. Muthu Ramachandran, Leeds Beckett University, Leeds, UK
- Dr. Lucio Agostinho Rocha, State University of Campinas, Brazil
- Dr. Saqib Saeed, University of Dammam, Saudi Arabia
- Prof. Claudio Sartori, University of Bologna, Bologna, Italy
- Dr. Mahmood Shah, University of Central Lancashire, Preston, UK
- Igor Tomičić, University of Zagreb, Pavlinska 2, 42000 Varazdin, Croatia
- Dr. Fareeha Zafar, GC University, Lahore, Pakistan

I would also like to thank the contributors to this book: 34 authors and co-authors, from academia as well as industry from around the world, who collectively contributed 13 well-researched chapters. Without their efforts in developing quality contributions, conforming to the guidelines and meeting often the strict deadlines, this text would not have been possible.

Grateful thanks are also due to the members of my family—Rehana, Zoya, Imran, Hanya, Arif and Ozair—for their continued support and encouragement. Every good wish, also, for the youngest and the most delightful in our family: Eyaad Imran Rashid Khan and Zayb-un-Nisa Khan.

Derby, UK Zaigham Mahmood
Hebei, China
June 2018

Other Springer Books by Zaigham Mahmood

Contents

Editor and Contributors

Contributors

Ricardo Abreu Cachapuz Bilanciai Group, Braga, Portugal

Maya Anderson IBM Research, Haifa University Campus, Haifa, Israel

David Bermbach IS Eng Research, TU Berlin, Berlin, Germany

Pratima Bhagat CTIM, National Institute of Industrial Engineering (NITIE), Mumbai, India

Kelvin Joseph Bwalya School of Consumer Intelligence and Information Systems, University of Johannesburg, Johannesburg, South Africa

Cinzia Cappiello Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy

Pethuru Raj Chellaiah Reliance Jio Cloud Services (JCS), Bangalore, India

Nuno Ferreira ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal; i2S—Insurance Knowledge, S.A., Porto, Portugal

David Garcia-Perez Atos Spain SA, Barcelona, Spain

Ronen I. Kat IBM Research, Haifa University Campus, Haifa, Israel

Attila Kertesz Software Engineering Department, University of Szeged, Szeged, Hungary

Ricardo J. Machado CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Somayya Madakam FORE School of Management, Information Technology, New Delhi, India

Adnan Mahmood Communications Research Group, Faculty of Engineering, Universiti Malaysia Sarawak, Kota Samarahan, Sarawak, Malaysia; Emerging Networks Laboratory, Telecom. Software and Systems Group, Waterford Institute of Technology, Waterford, Ireland

Zaigham Mahmood Debesis Education, Derby, UK; Shijiazhuang Tiedao University, Hebei, China

Achilleas Marinakis NTUA—National Techinical University of Athens, Zografou Campus, Athens, Greece

Raquel Martins CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Francisco Morais CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Vrettos Moulos NTUA—National Techinical University of Athens, Athens, Greece

Frank Pallas IS Eng Research, TU Berlin, Berlin, Germany

Jaime Pereira CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Tamas Pflanzner Software Engineering Department, University of Szeged, Szeged, Hungary

Pierluigi Plebani Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy

J. Pushpa Visvesvaraya Technological University, Belgaum, Karnataka, India

Pethuru Raj Site Reliability Engineering (SRE) Division, Reliance Jio Infocomm Ltd., Bangalore, Karnataka, India

Muthu Ramachandran School of Computing, Creative Technologies, and Engineering, Leeds Beckett University, Leeds, UK

Helena Rodrigues CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Nuno Santos CCG/ZGDV Institute, Guimarães, Portugal; ALGORITMI Centre, School of Engineering, University of Minho, Guimarães, Portugal

Arif Sari Department of Management Information Systems, School of Applied Sciences, Girne American University, Canterbury, UK

P. Beaulah Soundarabai CHRIST Deemed to be University, Bangalore, India

Bvuma Stella School of Consumer Intelligence and Information Systems, University of Johannesburg, Johannesburg, South Africa

Stefan Tai IBM Research, Haifa University Campus, Haifa, Israel

Monica Vitali Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy

Hushairi Zen Communications Research Group, Faculty of Engineering, Universiti Malaysia Sarawak, Kota Samarahan, Sarawak, Malaysia

About the Editor

Prof. Dr. Zaigham Mahmood is a published author of twenty-two books. Seven of these are dedicated to electronic government, and the other fifteen focus on the subjects of Cloud Computing, Fog Computing, data science, big data, Internet of Things, smart cities, project management and software engineering, including the textbook *Cloud Computing: Concepts, Technology and Architecture* which is also published in Korean and Chinese languages. Additionally, he is developing two new books to appear later in 2018. He has also published more than 100 articles and chapters and organized numerous conference tracks and workshops.

He is the editor-in-chief of *Journal of E-Government Studies and Best Practices* as well as the series editor-in-chief of the IGI book series on *E-Government and Digital Divide*. He is a senior technology consultant at Debesis Education, UK, and a professor at the Shijiazhuang Tiedao University in Hebei China. He further holds positions as a foreign professor at NUST and IIU in Islamabad, Pakistan. He has served as a reader (associate professor) at the University of Derby, UK, and a professor extraordinaire at the North-West University, Potchefstroom, South Africa. He is also a certified Cloud Computing instructor and a regular speaker at international conferences devoted to Cloud Computing and e-government. His specialized areas of research include distributed computing, project management and e-government.

Part I
Concepts and Principles

Chapter 1
Fog Computing: Concepts, Principles and Related Paradigms

Zaigham Mahmood and Muthu Ramachandran

Abstract Fog Computing, sometimes also referred to as Edge Computing, extends the Cloud Computing paradigm to lower latency, improve location awareness, provide better support for mobility and increase business agility. There is necessarily a requirement for these attributes in this age of the Internet of Things (IoT) where, according to one estimate, there will be close to 50 billion interconnected smart devices by 2020, and the amount of Big Data generated by these devices is expected to grow to around 200 exabytes per year by 2020. The core characteristic of the Fog Computing architecture is that it provides compute and data analytics services more immediately and close to the physical devices that generate such data, i.e. at the *Edge* of the network, and thus bypassing the wider Internet. In this chapter, we discuss the concepts and principles of Fog paradigm as well as the related paradigms and technologies, present the difference between the Cloud and Fog architectures and briefly discuss the OpenFog Reference Architecture. Hopefully, this chapter will set a scene for the various Fog-related topics presented in the rest of this book.

1.1 Introduction

Ubiquitous deployment of smart devices (such as mobile phones, tablets, sensors, motors, relays and actuators) connected though the Internet of Things (IoT) is estimated to reach 50 billion units by 2020 [1]. The data generated by these devices as well data from newly connected factories, homes communities, cars, hospitals, social and media and more is expected to grow from 1.1 zettabytes (or 89 exabytes)

Z. Mahmood (✉)
Debesis Education, Derby, UK
e-mail: dr.z.mahmood@hotmail.co.uk

Z. Mahmood
Shijiazhuang Tiedao University, Hebei, China

M. Ramachandran
Leeds Becket University, Leeds, UK

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_1

per year in 2016 to around 2.3 zettabytes (or 194 exabytes) per year by 2020 [2]. Managing such amounts of data, as well as data generated by social media technologies (such as Facebook, Twitter), is one of the biggest challenges, which the traditional IoT- and Cloud-based architectures are unable to cope with. The reasons being: the large scale, variety and velocity of data often known as Big Data; heterogeneity of the IoT devices; differing connectivity protocols and lack of communication standards; and high latency of the Cloud-based environments and systems. One solution is to *decentralise applications, management and data analytics into the network itself using a distributed and federated compute model* [3, 4].

Fog Computing [1], a term first put forward by Cisco, is sometimes also referred to as Edge Computing [4], Mist Computing, Fogging or Cloudlets. Although there are some subtle differences between these different terms, at a higher level, the terms can be regarded as synonyms. The term 'Fog Computing' or 'Edge Computing' means that rather than hosting and working from a centralised Cloud environment, Fog systems operate on network ends. It refers to placing some intelligence, processes and resources at the edge of the Cloud, instead of establishing channels for Cloud storage and utilisation [5, 6]. Fog Computing appears to be the next big thing for the Internet of Everything (IoE). According to one research [7], the Fog Computing market is currently valued at $22.3 million in 2017 and is expected to expand at an explosive rate and grow to $203.5 million over the next 5 years.

In this chapter, we attempt to present, first, the various definitions of this emerging paradigm known as Fog Computing and characterise some core aspects of Fogging; then, we link it up to the Cloud paradigm discussing the limitations and inherent difficulties of Cloud environments and how Fogging may possibly address at least some of the related issues. We also articulate the subtle differences between Fog Computing and Edge Computing and also suggest a layered approach to visualise where exactly Cloud, Fog and Mist Computing may be placed in a wider context of a Cloud–Fog-based system serving smart end-user devices in a distributed IoT environment.

1.2 Fog Computing

Fog Computing is a way of providing compute and storage services more immediately and close to the physical devices of an organisation [5], i.e. at the *Edge* of the Cloud network. Fog Computing can really be thought of as a way of providing services more immediately, but also as a way of bypassing the wider Internet, whose speeds are largely dependent on bandwidth and carriers [5].

NIST Special Publication 800-191 [8] defines it as horizontal, physical or virtual resource paradigm that resides between smart end devices (that generally reside within the organisations) and traditional Cloud Computing or connected data centres. According to the OpenFog Consortium [9], Fog Computing is *a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a Cloud-to-Thing continuum*. It is a highly virtualised platform that provides compute, storage and networking services between end devices and traditional Cloud Computing data centres [4, 10].

The Fog paradigm provides reduced latency and context awareness because of the localisation of Fog nodes and supports *vertically isolated latency-sensitive applications by providing ubiquitous, scalable, layered and federated network connectivity* [4]. Fog nodes deploy and provision same types of services as provisioned by Cloud Computing, viz: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). Additionally, the Fogging architecture uses one or more collaborative end-user clients or near-organisation Edge devices to carry out a substantial amount of communication, control, configuration, measurement and management services. It is a paradigm that extends Cloud Computing services to the edge of the network. The distinguishing characteristic being that: whereas Cloud environment may be geographically a long way away from the organisation, often not even knowing where the Cloud-based services actually reside, and relying heavily on the wider Internet bandwidths, Fog services are much closer to the end-users, with dense geographical distribution, and much better support for mobility.

As presented in Fig. 1.1, Fog Computing characteristics include the following [4]:

- Low latency—because of closeness of Fog nodes to the on-premise end-point devices, resulting in response and analysis in a much quicker time frame,
- Rich and heterogeneous end-user support—because of proximity of Edge devices to the compute nodes,

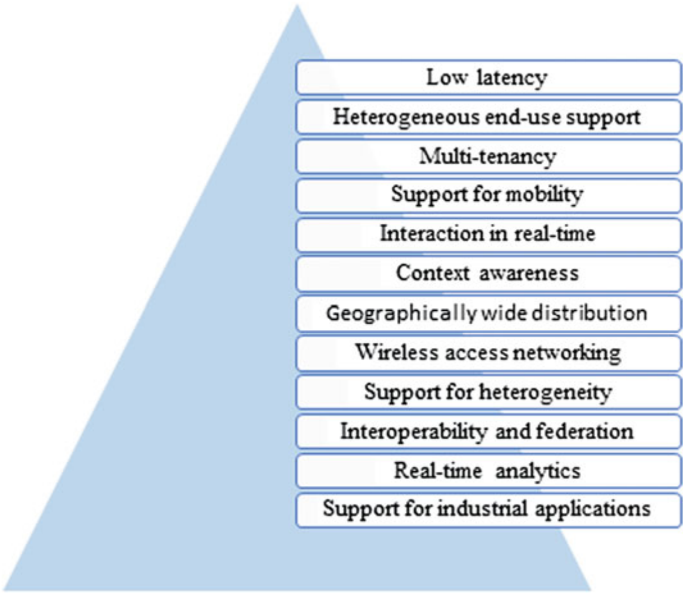

Low latency
Heterogeneous end-use support
Multi-tenancy
Support for mobility
Interaction in real-time
Context awareness
Geographically wide distribution
Wireless access networking
Support for heterogeneity
Interoperability and federation
Real-time analytics
Support for industrial applications

Fig. 1.1 Fog Computing characteristics

- Multi-tenancy in controlled environment—because of highly virtualised distributed platform,
- Better support for mobility—because of more direct communication between the Fog applications and the mobile devices,
- Interaction in real time—as opposed to batch processing as, for example, in case of Cloud-based applications,
- Context awareness—as the devices and nodes in the environment have knowledge and understanding of the environment,
- Geographical distribution—as Fog environment is geographically distributed, so it plays active role in delivery of high quality of streaming services,
- Wireless access networking—that is more appropriate for wireless sensing devices that require time-distributed analysis and communication,
- Support for heterogeneity—as the Fog nodes come in different form factors, and deployed in a variety of distributed environments,
- Seamless interoperability and better federation—for better communication between devices from various vendors and across various domains,
- Analytics in real time—which is easily possible because of ingestion and processing of data close to the sources,
- Support for a wide variety of industrial applications—through processing and analysis in real time.

These characteristics are useful when devising Fog Computing applications or environments such as smart cities, smart home, smart grids, smart health and eminent applications such as emergency response systems for flood monitoring and recovery. Varshney and Simmhan [11] also discuss three essential characteristics—resource; physical presence and access; and mobility—that distinguishes Fog Computing from the characteristics of Edge and Cloud Computing. In addition, the wider Fog paradigm is composed mainly of two other related technologies [1]:

- Cloudlets: These are applications located on the Edge of the network [12], mainly to respond to low latency in machine communications. A Cloudlet is a Fog node, resource-rich computer—or cluster of computers (a data centre in a box), that is well connected to the Internet and available for use by localised mobile devices. Fog nodes can be either physical or virtual elements and are tightly coupled with the smart end devices or access networks. These have four major attributes: self-managing, having computer power, low end-to-end latency and having based on Cloud-related technologies. In the network architecture, Cloudlets reside in the middle of the three-part *mobile device—Cloudlet—Cloud* structure.
- Mobile Edge Computing (MEC): This is a technology related to mobile networking, within the radio access network (RAN) and in close proximity to mobile subscribers [13]. It is a network architecture that enables Cloud Computing capabilities and an IT service environment at the edge of the cellular network. The underlying idea is that related processing is closer to the cellular

customers that in turn helps to reduce network congestion and increase application performance.

As for the advantages of Fogging, a closer look at the Fog model suggests that it is about taking decisions as close to the data and sources of generation of data as possible. In this context, Hadoop and other Big Data solutions have already started the trend to bring processing closer to the location of data. Fogging is about doing the same on a larger scale. There are obvious advantages in taking decisions as close to where the data generation taking place and not needing for certain data to be processed in the Cloud [5]. This, in turn, resolves the issues of security and privacy as well—as the valuable and sensitive data should not really be travelling to and through the Cloud Computing networks. Some of the advantages of Fogging include:

- Dense geographical distribution and support for mobility,
- Low latency, location awareness and improved QoS,
- Greater business agility and reduced operating costs.

These advantages of Fog Computing concepts and principles need to be considered when designing applications with some of the key service design principles of loose coupling, scalability and business process modelling and simulation. This will also help to predict the Fog Computing characteristics discussed in this section. However, there are some issues to be considered that are similar to Cloud Computing research issues such as those relating to security and privacy, resource management, shared boundaries in case of distributed environments and communication protocols.

1.2.1 Fog Computing Issues

Although the promise of Fog paradigm is attractive, it is important to note and understand the different issues that come with the use and deployment of the Fog Computing approach. Besides the issues inherited from Cloud Computing, some of the Fog-specific issues refer to the following:

1.2.1.1 Security and Privacy

Most security-related issues of distributed computing environments apply to Fog paradigm, and nature of some of these issues are subtly different because of the Fog nodes residing at the edge of the network, close to the devices in the network. Main issues relate to client authentication at different levels of gateway, as well as at the level of networked devices. As an example, consider a smart metre that is connected and has an IP address. A malicious user can easily tamper with this device or report false reading or spoof an IP address. As another example, consider a gateway that serves a number of Fog devices. These may be easily compromised or replaced by fake ones or by connecting to malicious access points that provide deceptive SSID as a legitimate one. Also, as the man-in-the-middle attack in the Fog environment is simple to launch, it can be difficult to address. Although, techniques such as signature or anomaly-based intrusion detection, multicast authentication and Diffie–Hellman key exchange can be used to counteract, such issues at the local level are matters of serious concern. The main issues can be grouped into the following categories:

- **Advance Persistent Threats (APT)**: These refer to unauthorised users gaining access to systems or networks and remain there for a long time without being detected
- **Access Control Issues (ACI)**: These refer to unauthorised users managing to install malicious software to gain unauthorised access to cause malicious damage
- **Account Hijacking (AH)**: Here, the intention of the attack is to gain access to user accounts using phishing techniques for malevolent aim to compromise the system
- **Denial of Service (DoS)**: Here, the objective of the hacker is to disable and render inaccessible the entire system or parts of the system resulting in the interruption as authorised user's access
- **Data Breaches (DB)**: This refers to unauthorised or illegal viewing, access, deletion, modification or retrieval of data by an individual attacker or malicious application
- **Data Loss (DL)**: This refers to an event or situation that results in data being corrupted, deleted or made unavailable by an unauthorised user or a malicious application
- **Malicious Insider (MI)**: This is an authorised user who uses his access permissions for harmful, unethical or illegal activities to cause damage or harm to the system resources
- **Insufficient Due Diligence (IDD)**: This refers to lacking in the required standard of care and not fulfilling the legal obligation, as a result of which damage or failures may be caused
- **Shared Technology Issues (STI)**: These refer to multi-tenancy when many users share the same resources that can result in compromising the system by threats such as DoS, DL and DB.

1.2.1.2 Fog Network Topology and Location Awareness of Nodes

Fog networks are heterogeneous where management and maintenance of the connectivity of a wide variety of diverse devices are not easy. In this context, design and the arrangement of nodes therein, consisting of a wide variety of devices with varied communication protocols, are one of the key issues. Architecture is grounded in virtualisation technology and that itself has certain inherent limitations in terms of security and shared boundaries. Relevant issues, here, relate to network scalability (horizontal, vertical, up and down), topological arrangement, virtualisation, redundancy, measurement of performance, monitoring and management and operational costs.

Although network topology issues can be managed using techniques such as software-defined networking (SDN) and network function virtualisation (NFV), the performance and scalability of virtualised network appliances are yet another serious key concern [14].

It is important that interface mechanisms between nodes are dynamic and the nodes themselves have an additional layer that have ambient intelligence (AmI) embedded so that the nodes become location and context aware. However, application-aware provisioning can be easily compromised because of the bandwidth, storage and latency.

1.2.1.3 Resource Management

In any network, there are issues concerning resource discovery, end-point determination, resource allocation and resource sharing. Fog architecture is no different. The most critical problem is designing resource management techniques that determine which modules of analytics applications are pushed to each edge device to minimise the latency and maximise throughput. We, therefore, need an evaluation platform that enables the quantification of performance of resource management policies on an IoT or Fog Computing infrastructure in a repeatable manner.

1.2.1.4 Interoperability

Another key challenge in facilitating Fog Computing is building the necessary level of interoperability to support the access of Fog-based resources. This requires a collaborative approach between the different elements of Fog infrastructure. One solution towards achieving the interoperability is the need for an open architecture that can significantly reduce the cost of developing Fog applications, increase the adoption of Fog Computing model and ultimately increase the quality and innovation of Fog Computing services [15].

1.2.1.5 Other Issues

There are also some serious issues due to the multi-tenant nature of the Fog environment. Such issues revolve around security and privacy (as mentioned above) and service-level agreements. Lack of appropriate tools for the measurement of connectively, capacity, reliability, effectiveness and delay are also of serious concerns [14]. Here, the issues are the same as for the Cloud paradigm because of similarity due to the nature of distributed computing environments.

To resolve, or at least minimise, the inherent issues of distributed computing environments, we need new approaches that satisfy, at least, the following requirements [13]:

- Minimise latency,
- Conserve network bandwidth,
- Address security and privacy concerns,
- Collect data securely from different environments,
- Manage data processing effectively.

1.3 Cloud Paradigm Versus Fog Computing

After briefly introducing the Fog Computing concepts, characteristics and issues in the preceding section, the first part of the current section is about the Cloud paradigm. Here, first, we present what this paradigm entails—mainly for the sake of completeness. Later, in the section, we articulate the differences between the Cloud and the Fog models.

1.3.1 Cloud Computing

Cloud Computing is a generic term for anything that involves delivering hosted services over the Internet [16]. It is a paradigm based on a pay-as-you-go approach. Gartner [16] defines Cloud Computing as *a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies*. It is an all-inclusive solution in which computing resources (hardware, software, networking, storage and so on) are provided rapidly to users as demand dictates [12, 17]. Cloud Computing is the practice of using a network of remote servers hosted on the Internet to store, manage and process data, rather than a local server or a personal computer. It is generally a heavyweight and dense form of computing power that promises the following benefits:

- Cost saving with respect to capital investment—as organisations can lease or deploy Cloud-based resources, whether virtualised hardware, software, storage or network capabilities from the Cloud providers,
- Reduction in costs with respect to developing and delivering IT services—as all manner of services (software, hardware, networking, storage, etc.) are already available in the Cloud environment,
- Reduction in management responsibilities and thus allowing key personnel to focus more on production and innovation—as the maintenance and deployment of services is the responsibility of Cloud owners/providers,

- Increased business agility to allow enterprises to meet the needs of rapidly changing markets—as the latest technologies can be easily provisioned on a pay-as-you-go basis from the Cloud providers.

Some of the core characteristics of the Cloud paradigm include [10, 17]:

- On-demand self-service—to enable users to consume computing capabilities (e.g. applications, server time, network storage) as and when required,
- Resource pooling—to allow combining computing resources (e.g. hardware, software, processing, network bandwidth) to serve multiple consumers,
- Rapid elasticity and scalability—to allow functionalities and resources to be rapidly and automatically provisioned and scaled up or down as demand dictates,
- Measured provision to optimise resource allocation—to determine usage for billing purposes,
- Extension to existing on-premise hardware and application resources, e.g. through Cloud bursting—to reduce the cost of additional resource provisioning.

Cloud-based services (software, hardware, networking, servers, virtualisation, security, etc.) come in different varieties; however, these are generally classified as three types: software services, platform services and infrastructure services. These are generally abbreviated as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS), respectively. More specifically, these can also be of the following variety (and many more):

- Storage-as-a-Service,
- Database-as-a-Service,
- Security-as-a-Service,
- Communication-as-a-Service,
- Management/Governance-as-a-Service,
- Integration-as-a-Service,
- Testing-as-a-Service,
- Business Process-as-a-Service.

Cloud environments come in different varieties, and these may be deployed in a number of ways, more generally as:

- Public or External Clouds—These are owned, managed and hosted by third parties. Cloud providers assume the responsibilities of installation, management, provisioning and maintenance. This variety of Clouds provides a much greater level of efficiency of pooling of resources.
- Private or Enterprise Clouds—These are proprietary networks, often data centres, owned and managed by the organisations, and residing within the enterprise. Thus, the enterprises can take better control of all aspects of the provision and functioning.
- Hybrid Clouds—These are combination of private and public Clouds, where the management responsibilities are split between the enterprise and the public

Cloud providers. The main advantage is that the organisations can keep the sensitive data within the private Cloud and the rest in the public Cloud.

Although the Cloud paradigm presents numerous benefits and that is exactly the reason that it is an attractive model for enterprises, there are also numerous inherent issues. Some of these relate to the following:

- Data governance and service management,
- Product and process monitoring,
- Infrastructure reliability and system availability,
- Information and visualisation security,
- Business continuity,
- High latency and bandwidth bottlenecks,
- Data transmission across existing broadband speeds.

1.3.2 Cloud Versus Fog Computing Comparison

In this subsection, we compare the two models and look into the similarity and differences and also discuss how some of the issues inherent in the Cloud paradigm may be resolved, or at least, minimised by the Fog paradigm.

Fog Computing is a distributed computing paradigm that extends Cloud Computing to the edge of the network—as a compliment to the Cloud solution, to adjust to the emerging IoT vision. It facilitates the operation of compute, storage and networking services between end devices and Cloud Computing data centres. Fog Computing typically involves components of an application running both in the Cloud as well as in the Edge devices in the Fog, i.e. in smart gateways, routers or dedicated Fog devices. Refer to Fig. 1.2 where the bottom layer has the enterprise smart sensor devices that are accessing resources and compute power from the

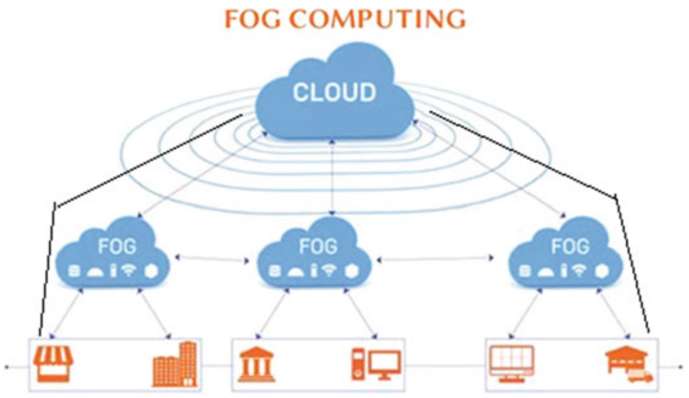


Fig. 1.2 Three-layer Cloud–Fog enterprise model

Table 1.1 Fog Computing versus Cloud Computing

	Fog Computing	Cloud Computing
Response time	Milliseconds, sub-seconds	Minutes, days, weeks
Data storage period	Transient	Months and years
Applications	e.g. M2M	e.g. Data analytics
Location coverage	Local	Global

middle layer as well as resources and compute power directly from the Cloud; the Fog environment in the middle layer is also linked with the Cloud environment in the top layer. Table 1.1 illustrates the benefits of storage and processing ‘closer to home’ rather than in a geographically distant Cloud environment.

To summarise, the Cloud requires a huge amount of bandwidth, the Internet is inherently unreliable, and wireless networks have limitations. By using Fog Computing, the amount of bandwidth required is much reduced. It allows, essentially, transmitted data to bypass the Internet, keeping it as local as possible, on the smart devices in the Fog environment. The most valuable data may still be transmitted through Cloud networks, but much of the traffic, especially the sensitive data, could be kept off of those networks, freeing up bandwidth for everyone else using the Cloud.

Similar to Cloud Computing, Fog Computing provides storage, compute and applications to be consumed by end-users. However, Fog Computing has a bigger proximity to end-users and bigger geographical distribution [18], as mentioned before. Compared to the Cloud paradigm, Fog Computing emphasises proximity to end-users and client objectives, local resource pooling, latency and the backbone bandwidth reduction to achieve better quality of service (QoS) and Edge analytics/stream mining, resulting in superior user experience [19]. Thus, Fog Computing extends the Cloud model to the edge of the network to address applications and services that do not fit the paradigm of the Cloud due to technical and infrastructure limitation such as the following [16]:

- Applications requiring lower and predictable latency,
- Geographically widely distributed applications and processing,
- Faster mobility and mobile applications,
- Large-scale distributed control systems requiring faster processing.

There are certain inherent issues in Cloud Computing, as discussed before. Fog Computing is highly suited to resolving at least some of these, e.g. reducing the need for bandwidth by not sending every bit of information over Cloud channels and instead aggregating it at certain access points [5]. This type of distributed strategy, in turn, also lowers costs, improves efficiencies, and so the QoS. More interestingly, it is another approach to dealing with the emerging and much popular concept of Internet of Things (IoT).

1.4 Fog Computing Versus Edge Computing

Fog Computing and Edge Computing are often used to mean the same architecture, and therefore, even in this contribution, we have regarded the terms as interchangeable; however, a subtle distinction can be made. In this section, we aim to highlight the differences between the two architectures.

Although Fog and Edge Computing both refer to having intelligence, processing and storage at the edge of the network, i.e. closer to the sources of data, the main difference is to do with exactly where the intelligence and processing are placed. Whereas Fog Computing pushes intelligence down to the local area network level of the network architecture, processing data in a Fog node or the IoT gateway, Edge Computing places the intelligence, processing and communication capabilities of an Edge gateway directly into the smart devices like programmable automation controllers (PAC) [20]. Besides, Edge Computing is an older expression predating the Fog Computing term. While Edge Computing is typically referred to the location where services are instantiated, Fog Computing implies distribution of the communication, computation and storage resources and services on or close to devices and systems in the control of end-users [4].

In Fog Computing, data communication between the data generating devices and the Cloud environment requires a number of steps [21] including:

- Communication is first directed to the input/output points of a programmable automation controller (PAC) that runs a control system program to perform automation.
- It is then sent to a protocol gateway that converts data to an understandable format such as HTTP.
- Data is then transmitted to a Fog node on the local network that performs the required analysis and organises data transmission to the Cloud for storage, etc.

Thus, in the Fog environment, there are many links and so many potential points of failures.

In Edge Computing, the communication is much simpler and there are potentially less points of failures. Here, data generating devices are physically connected to PACs for onboard automation as well as for parallel processing and analysis of data. Again, it is PACs that determine which data is to be stored locally or sent to the Cloud. Thus, apart from reducing possibility of failures, there is saving of time and streamlining of communication, as well reduction in the complexity of architecture [21, 22].

In Fog Computing, there is a single centralised device responsible for processing data from different end-points in the network. In the Edge architecture, IoT data is collected and analysed directly by the connected devices, so every network node participates in the processing. Shariffdeen [22] suggests that whereas Fogging is much preferred by the service providers and data processing companies, Edge architecture is more preferred by middle-ware companies that own the backbone and radio networks. Table 1.2 compares Fog vs the Edge Computing in terms of advantages.

Table 1.2 Advantages of Fog versus Edge Computing

	Fog Computing	Edge Computing
Advantages	• Location awareness, low latency, QoS—but requires large resources • Data closer to the user—but not for systems that require limited data • Integration of distributed data with Cloud services	• All nodes participate so reduced delays • Real-time local analysis • Lower operating costs and network traffic • Improved performance

According to [2, 9], Fog works with the Cloud but Edge is defined by the exclusion of Cloud. Fog has a hierarchical and flat architecture with several layers forming a network, whereas Edge is often limited to separate nodes (in addition to compute power) that do not form a network. Fog also addresses networking, storage, control and acceleration. Fog Computing has extensive peer-to-peer interconnect capability between nodes, where Edge runs its nodes in silos, requiring data transport back through the Cloud for peer-to-peer traffic [9].

The Fog model architecture consists of three main segments [23]:

- IoT devices: these are connected devices that generate and transit large amounts of a variety of structured and semi-structured data.
- Fog network: that receives real-time data from IoT devices using a diverse variety of communication protocols and performs real-time analysis.
- Cloud environment: that receives data for storage from Fog nodes and also performs analysis for business intelligence.

Edge Computing architecture consists of the following components [23]:

- Edge devices: these are connected smart devices (sensors, actuators, etc.) that generate, analyse and take other relevant actions.
- IoT Gateway: that has responsibility for connecting Edge devices with the Cloud environment; deals with varied protocols and stores peripheral data.
- Cloud environment: that receives data from gateway, analyses and sends instructions back to the gateway.

Design for resource-constrained Fog Computing applications requires well-established principles of service computing and to follow a software engineering approach to building Fog Computing applications that are safe and secure. Therefore, OpenFog consortium has established a reference architecture which provides an open, distributed and secure platform for developing Fog Computing applications. This architecture is briefly outlined in the following section.

1.5 Fog Computing Reference Architecture

In order to develop a Fog Computing architecture, a collaboration by the name of OpenFog Consortium, comprising the joints of industry (such as Cisco, Dell, Intel and Microsoft), research institutions such as Princeton University and users, was founded in November 2015. This consortium is an independent non-profit organisation run under the direction of its board of directors; its committees and working groups are run by its membership. Their deliberations have resulted in, what is now called as the OpenFog Reference Architecture (OpenFog RA) that aims to help business leaders, software engineers and system designers in developing and maintaining hardware, software and system elements necessary for Fog Computing.

The OpenFog RA is built upon a set of eight core principles called Pillars, viz:

- Security: safety, trust,
- Scalability: scalable performance, capacity, reliability, security, hardware, software,
- Openness: composability, interoperability, communication, location transparency,
- Autonomy: of discovery, orchestration, management, security, operation, cost savings,
- RAS: reliability, availability, serviceability,
- Agility: innovation, affordability,
- Hierarchy,
- Programmability: adoptive infrastructure, efficient deployment, effective operations, enhanced security.

In determining the relevant pillars, ISO/IEC/IEEE standards have been followed.

Figure 1.3 shows the OpenFog layered architectural logical view of the IoT system from a computational perspective. The hierarchical layers are deployed in the Fog–Cloud model as illustrated in Fig. 1.4.

In Fig. 1.4, we note that:

- Enterprise Systems (ESs) in the model designated as {1} reside only in the Fog environment and are independent of the Cloud services.
- Model {2} uses the Cloud for ESs needed for business support at strategic level, e.g. strategic decisions making.
- Model {3} suggests that local Fog infrastructure is used for time-sensitive processing, while the Cloud provision is accessed for operational and business support-related information processing.
- The last model, referred to as {4}, is employed in scenarios such as connected cars etc.

The discussion in this section, so far, has referred to an abstract logical higher-level architecture as proposed by the OpenFog RA. For various stakeholder perspectives in terms of the other pillars and for more detailed information, reader is

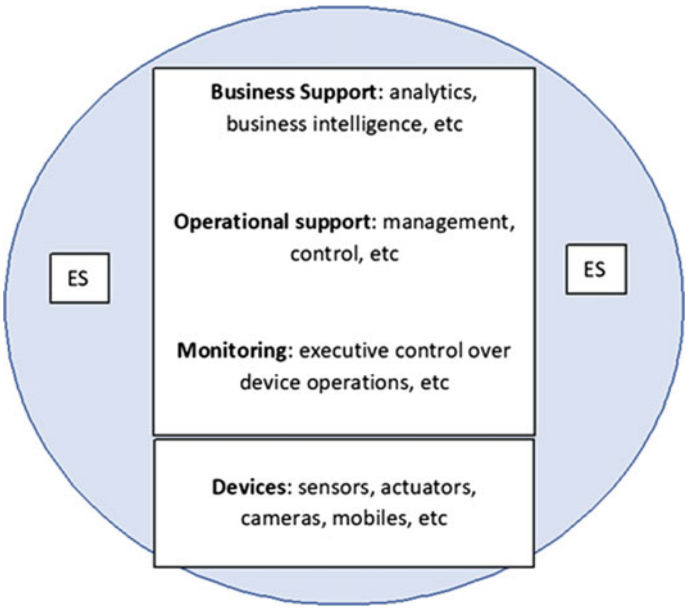
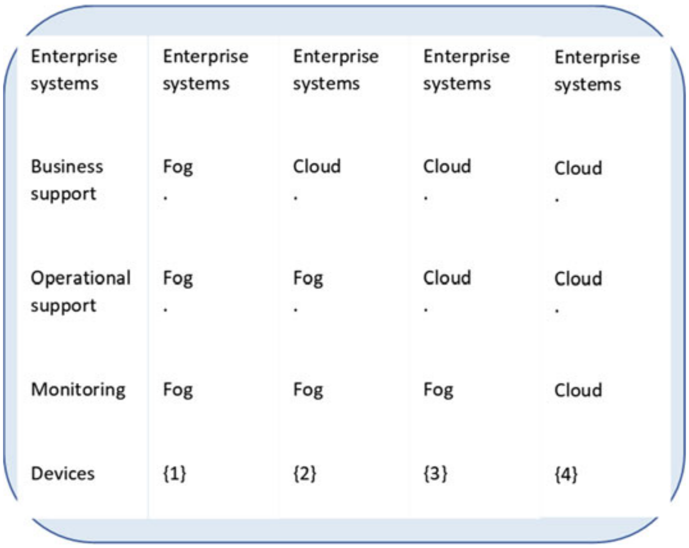

Business Support: analytics, business intelligence, etc
Operational support: management, control, etc
Monitoring: executive control over device operations, etc
Devices: sensors, actuators, cameras, mobiles, etc
ES
ES

Fig. 1.3 OpenFog layered architectural view—ES refers to enterprise systems for implementation of various systems within the hierarchy



Enterprise systems	Enterprise systems	Enterprise systems	Enterprise systems	Enterprise systems
Business support	Fog .	Cloud .	Cloud .	Cloud .
Operational support	Fog .	Fog .	Cloud .	Cloud .
Monitoring	Fog	Fog	Fog	Cloud
Devices	{1}	{2}	{3}	{4}

Fig. 1.4 OpenFog hierarchy Fog deployment (in Fog–Cloud) models. Adapted from [24]

referred to the OpenFog Consortium Reference Architecture paper [24]—as the fuller description, here, is out of the scope of this chapter.

At this point, it is worth noting that the OpenFog RA offers a number of distinct advantages which the OpenFog Consortium refer to as SCALE [24] that are as follows:

- Security: additional security to ensure safe and trusted transactions,
- Cognition: awareness of client-centric objectives to enable autonomy,
- Agility: rapid innovation and affordable scaling under common infrastructure,
- Latency: real-time processing and cyber-physical system control,
- Efficiency: dynamic pooling of resources from participating end-user devices.

1.6 Fog Computing Application Scenarios

Having discussed the differences between Cloud and Fogging, there are many commercial and industrial applications that require both Fog localisation and Cloud globalisation, particularly for analytics and Big Data processing and manipulation.

Technology giants, such as IBM, are the driving force behind Fog Computing and to link it to the concept of IoT. Most of the buzz around Fog Computing has a direct correlation with IoT. The fact that everything from cars to thermostats to pumps to smart energy metres are gaining Web intelligence means that direct user-end computing and communication may become much more important than ever. The following are some of the practical example applications where Fog Computing is already being applied [5]:

- **Smart grids**: Fogging, for the same reason as above, provides fast machine-to-machine (M2M) handshakes and human-to-machine interactions (HMI), resulting in a more efficient cooperation with the wider Cloud provision. Fog devices collect the local information and collectively take real-time decisions based on 360° view of what is happening in the environment.
- **Smart homes and cities**: Fog Computing enables getting sensor data at all levels of the activities of homes as well as from the entire cities, integrating the mutually independent network entities within the homes and cities and faster processing to create more adaptive user environments for better human conditions and the quality of living.
- **Connected vehicles**: Fogging provides an ideal architecture for vehicle-to-vehicle (V2V) communication, because of proximity of devices embedded in cars, roads and access points. Fogging, with context awareness, makes real-time interactions between cars, access points and traffic lights much safer and more efficient.
- **Self-drive cars**: These vehicles rely entirely on automated input to perform navigation. Thus, a slow response when vehicles are moving at 60 mph can be dangerous or even fatal, so real-time processing speed and immediate decisions

are required. Fog networks are especially suitable for applications that require a response time of less than a second, according to Cisco [7].

- **Traffic light system**: Fogging is suitable for building smart traffic light systems that change signals based on surveillance of incoming traffic to prevent accidents or reduce congestion. Data can also be sent to the Cloud for longer-term analysis. The communication between vehicles and access points are being improved with the arrival of 3G and 4G and more powerful WiFi.
- **Healthcare management**: Cloud Computing market for healthcare has already reached in excess of $5.4 billion [5]. Fog Computing is helping to speed up the process by localising the device connectivity and proximity of devices to the patients and user community.
- **Medical wearables**: These are increasingly being used by healthcare providers to monitor patient conditions, to provide remote telemedicine and even to guide on-site staff and robots in procedures as delicate as surgery. Thus, reliable real-time data processing is crucial for these types of applications [7].
- **IoT and cyber-physical systems (CPSs)**: Fogging has a vital role to play in CPSs (integration of system's physical and computational elements) and IoT (interlink physical objects). The combination of these is already changing the world comprising computer-based control systems, physical reality and engineered systems.

Other application scenarios include: rail safety, power restoration from smart grid networks, smart parking metres, self-drive cars and trains, air traffic control, cyber security, IoT cyber-physical systems, machine-to-machine communication and human–computer interaction.

1.7 Future of Fog Computing

Attractive nature of Fog and Edge Computing will result in the development of new business models, thus helping the industries to grow more efficiently and much faster. As a result, new vendors and new industries will come on board with new offerings and new architectural approaches to networking.

One exciting area of development is Fog-as-a-Service (FaaS) where a Fog service provider deploys interconnected Fog nodes to blanket a regional service area [25]. This, in turn, will provide opportunities for creating new applications and services that cannot be easily developed using the current host-based and Cloud-based platforms; for example, Fog-based security services would address many challenges that we are currently facing in the IoT environment.

The emergence of 5G technologies, development of smart city applications and building the distributed computing environments with embedded ambient intelligence will all help to revolutionise the quality of life and assisted living through better and more efficient health monitoring and predictions, recycling and waste management systems, connecting people, smart wearables, tourism, smart buildings, smart transportation and adaptable smart homes.

1.8 Conclusion

Fog Computing is becoming an attractive paradigm for reasons of proximity of processing, storage and data analytics to the devices that generate and exchange data. In this chapter, we have discussed the Fog Computing paradigm in some detail, compared it with the Cloud Computing model and presented the OpenFog Reference Architecture, albeit only briefly. We have illustrated the usefulness of Fog paradigm as an extension of the Cloud architecture and also presented some use cases. The aim in this contribution has been to provide some general background information with some critical analysis so that the chapter serves as a foundation for the more detailed accounts of the more specialised Fog-related topics articulated in the other chapters in this book.

References

1. Evans D (2011) The internet of things how the next evolution of the internet is changing everything. Cisco White Paper, https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
2. Adams F (2017) OpenFog reference architecture for fog computing. https://knect365.com/Cloud-enterprise-tech/article/0fa40de2-6596-4060-901d-8bdddf167cfe/openFog-reference-architecture-for-Fog-computing
3. NIST (2018) Fog computing conceptual model, NIST special publ (SP) 500-325. https://doi.org/10.6028/NIST.SP.500-325
4. NIST (2017) The NIST definition of fog computing, NIST Special Publ 800-191 (Draft)
5. Banafa A (2014) What is fog computing? https://www.ibm.com/blogs/Cloud-computing/2014/08/Fog-computing/
6. Bar-Magen Numhauser J (2012) Fog Computing introduction to a New Cloud Evolution. Escrituras silenciadas: paisaje como historiografía. University of Alcala, Spain, pp 111–126. ISBN 978-84-15595-84-7
7. Rasmussen R (2017) How fog computing will shape the future of IoT applications and cybersecurity. https://www.informationsecuritybuzz.com/articles/fog-computing-will-shape-future-iot-applications-cybersecurity/
8. NIST (2018) The NIST definition of fog computing. https://csrc.nist.gov/csrc/media/publications/sp/800-191/draft/documents/sp800-191-draft.pdf
9. Hardesty L (2017) Fog computing group publishes reference architecture. https://www.sdxcentral.com/articles/news/Fog-computing-group-publishes-reference-architecture/2017/02/
10. Mahmood Z (2011) Cloud computing: characteristics and deployment approaches. In: 11th IEEE international conference on computer and information technology
11. Varshney P, Simmhan Y (2017) Demystifying fog computing: characterizing architectures, applications and abstractions. In: E/ACM 1st international conference on fog and edge computing (ICFEC), IEEE 2017
12. Amrhein D, Quint S (2009) Cloud computing for the enterprise: Part 1: capturing the cloud, developer works. IBM Report, www.ibm.com/developerworks/websphere/techjournal/0904_amrhein/0904_amrhein.html
13. Cisco (2015) Fog computing and the internet of things: extend the cloud to where the things are. White Paper, Cisco. Available at: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

14. Yi S, Li C, Li Q (2012) A survey of fog computing: concepts, applications and issues. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.702.7773&rep=rep1&type=pdf
15. Gorlatova M (2017) Reference architecture. http://pages.di.unipi.it/throughtheFog/wp-content/uploads/sites/13/2017/02/gorlatova.pdf
16. Cearley DW (2010) Cloud computing: key initiative overview. Gartner Report, 2010
17. Mahmood Z (2011) In: Mahmood Z, Hill R (eds) Cloud computing for enterprise architectures. Springer, Berlin
18. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on mobile cloud computing, ser. MCC'12. ACM, 2012, pp 13–16
19. Cisco (2013) Fog computing, ecosystem, architecture and applications. Cisco Report RFP-2013-078
20. Greenfield D (2016) Fog computing vs. edge computing: what's the difference? https://www.automationworld.com/Fog-computing-vs-edge-computing-whats-difference
21. OPTO-22 (2018) Fog computing vs edge computing. http://info.opto22.com/Fog-vs-edge-computing
22. Shariffdeen R (2017) Fog computing vs edge computing. https://medium.com/@rshariffdeen/edge-computing-vs-Fog-computing-5b23d6bb049d
23. Shah H (2017) Edge computing and fog computing for enterprise IoT. https://www.simform.com/iot-edge-Fog-computing/
24. OpenFog (2017) OpenFog reference architecture for fog computing. https://www.openFogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL-1.pdf
25. McKendrick J (2016) Fog computing: a new IoT architecture? https://www.rtinsights.com/what-is-Fog-computing-open-consortium/

Chapter 2
Fog Computing in the IoT Environment: Principles, Features, and Models

Somayya Madakam and Pratima Bhagat

Abstract Internet of Things, abbreviated as IoT, is a distributed computing environment that is full of promise, helping to shape the future of the world. IoT is a network of smart devices capable of processing and storage. It integrates the existing and upcoming technologies to ensure enhanced provision of quality of life for human beings. IoT applications are immense, touching the varied fields of aviation, medicine, home automation, manufacturing, mining, marine zone, agricultural fields, forests, transportation, and citizens' security, to name but a few. IoT has already helped tag the tigers from forests and cows from farming fields for identification and tracking. Smart wearable technologies have been embedded into the human body for health monitoring. Recently, IoT has become the backbone of a rising new distributed computing paradigm, called Fog Computing, also known as Edge Computing. Fog Computing is the latest distributed computing model that extends the Cloud Computing vision. The main advantage of the Fog paradigm is that it brings partial computation and storage right at the *edge* of the network, thus helping to reduce latency and the Internet bandwidth bottlenecks. With this background, this book chapter presents a detailed literature analysis using the Web of Science and Relecura software. This chapter also outlines fundamentals of Fog Computing such as its characteristics, models, theories, and different applications. Moreover, various upcoming technologies and frameworks have also been illustrated for better understanding of the Fog Computing phenomenon.

S. Madakam (✉)
FORE School of Management, Information Technology, New Delhi, India
e-mail: somayya@fsm.ac.in

P. Bhagat
CTIM, National Institute of Industrial Engineering (NITIE), Mumbai, India

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_2

2.1 Introduction

Multinational companies are spending enormous amounts of money on salaries for the technology experts in planning, designing, coding, developing hardware, software, and networking products and services. These are direct products or technologies embedded into physical *things* to bring operational efficiency in objects and systems, in relation to the Internet of Things (IoT) vision. It has been widely misunderstood that IoT technologies are limited to sensors and radio frequency identification. This is certainly not the case. Sophisticated technologies such as Bluetooth, Wi-fi, ZigBee, EPC, Barcode, QR codes, IPv4/6, Microcontrollers, Sensors, Actuators, Pumps, Social media, 4G/5G, Artificial Intelligence, Robotics, Ambience intelligence, Web 3.0, Big Data Analytics, Cloud Computing, and many more are under the IoT umbrella [1]. In fact, the IoT technologies extend even further to cover all the existing software together with existing operating systems such as Windows 16, Linux, SuSE Linux, Macintosh, Ubuntu, Android, Sun Solaris, Unisys, CP/M, and upcoming Urban Operating Systems. Electronic hardware devices including computers, laptops, servers, tablets, and smartphones are the backbone of the IoT environment. These technologies are helping human life become more comfortable, modifying public utility services to make them better fit for the purpose, home automation, educational performance, good governance practices, automated manufacturing process, defence services, and healthcare—all in a big way.

Most of these technologies are already in existence and in the public domain, except the technologies that are in advanced version usages. Using these technologies, even objects such as the tables, chairs, curtains, windows, fridges, washing machines, clothes, and glasses can be connected to the Future Internet (FI). Even the mountains, lakes, and forests can get connected via the Wi-Fi or local wired connections for monitoring and securing them from willful damage. In fact, natural or man-made inanimate objects can become capable of interacting with the *Homo sapiens*! European countries, Singapore, Korea, Japan, China, and other South Asian countries are proactively developing newer IoT technologies for better business, effective operational efficiency and improved day-to-day human life. Several collaborative projects in designing the hardware, software, and networking products are underway. Some researchers are working on IoT architectures, their inter-operability, and security mechanisms during data transmission among IoT devices. Many multinational corporates, firms, organizations, and universities have entered into a common understanding for developing IoT products under the umbrella of the IPSO alliance. Another significant factor is the security of devices and data transmission, for which many multinational corporate giants are in the line, developing better high standard security software and mechanisms. Even International cyber laws are coming into practice for the security of personal data. A humongous amount of data is being generated by different objects getting connected to the Internet. Benefits are numerous, however, computational issues, storage scarcity, lack of speed capacity, and security concerns are a few of the challenges that face the new technologies. In order to solve these issues, a new technology christened as "Fog Computing" has been gradually gaining foothold in the market. Importantly, widespread adoption of the Fog Computing paradigm requires the development of new network architectures and middleware platforms deploying new emerging communication technologies such as 5G, edge analytics tools for IoT, big data, and machine learning of novel context-aware management approaches. Fog Computing is proposed to enable computing directly at the edge of network, closer to the devices that generate and require processing of data, to deliver new applications for billions of connected devices [2].

The organization of this book chapter is as follows. The next section discusses the existing literature in some detail. The third section focuses on principles, characteristics, and concepts of Cloud and Fog Computing. The next section is devoted to discussing the models and Cloud/Fog architectures, and the last but not the least section presents the conclusion.

2.2 Literature Review—Analysis

2.2.1 Methodology

The present literature review analysis on Fog Computing was purely dependent on secondary data. The secondary data was collated from the Web of Science, which is maintained by M/s. Thomson and Reuters. It is now known as a part of Clarivate Analytics. Importantly, only the indexed articles in the Web of Science database were considered for this study. The research articles were collected in February 2018. This Web of Science database is widely acknowledged by researchers as one of best, yielding empirically-proven articles, manuscripts and whitepapers for academicians, researchers, and corporate personnel alike. In total, 746 articles were obtained from the Web of Science database regarding Fog Computing. The articles published during the period from 2011 to 10 February 2018 were included in this study. The keywords used to collate the article were "Fog Computing" and "Edge Computing". The purpose of this analysis was to analyse the Fog Computing paradigm, with emphasis on the concepts, principles, models, and prominent contributors in this domain. The literature analysis will help in understanding the fundamentals of Fog Computing through explanatory figures, statistics, and tables.

2.2.2 Literature Analysis Using Web of Science

The phenomenon of Fog Computing traces its beginning to the year 2011. Soon after, computer scientists, IT experts, engineers, and mobile computing developers started developing Fog Computing-related hardware, software, and networking

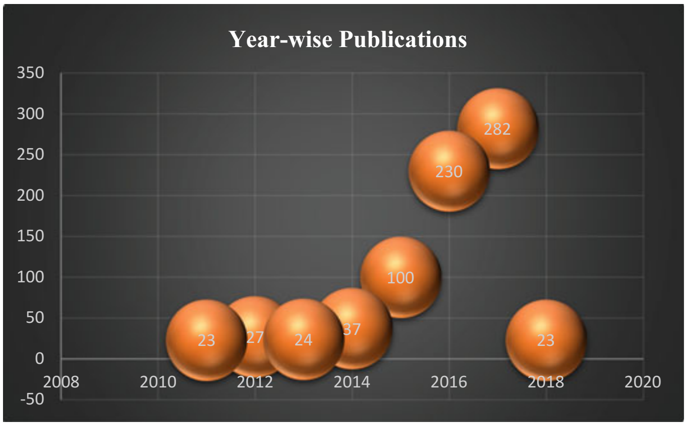


Fig. 2.1 Year—wise publications

components in a bigger way. Figure 2.1 demonstrates the year-wise publications on this topic. 282 articles were published in 2017, which was the highest figure obtained during 2011–2018. It is likely that this figure for the year 2018 goes up as the present study has only considered for the period of 1 January–11 February 2018. The year 2016 followed next, with 230 articles, followed by 100 publications in the year 2015. The period between 2011 and 2017 showed a steady growth rate in the number of publications, which indicates that the phenomenon is gaining popularity, touching human life through more and more applications. Remarkably, Fog Computing is an interdisciplinary subject. It marries all the elements involved in providing human comfort, connecting several allied subjects. In the last 7 years, the applications developed relating to this phenomenon of Fog Computing include Computer Science, Engineering, Telecommunications, Radiology, Nuclear Medicine, Medical Imaging, Optics, Oncology, Instruments Instrumentation, Transportation, Chemistry, Meteorology Atmospheric Sciences, Electrochemistry, Imaging Science, Photographic Technology, Physics, Automation Control Systems, Neurosciences, Remote Sensing, Materials Science, Science Technology, Cardiovascular System, Cardiology, and Energy Fuels to name a few. Figure 2.2 provides the percentage-based classification of publications as per research area and subject.

The top 25 authors were Marin-Tordera E (10), Masip-Bruin X (9), Aazam M. (8), Huh EN (8), Baccarelli E (6), Chen Y (6), Chiang M (6), HA S (6), Hong CS (6), Ivanov S (6), Li Q (6), Li W (6), Liljeberg P (6), Naranjo PGV (6), Rahmani AM (6), Tenhunen H (6), Zhang T (6), Zomaya AY (6), Bruschi R (5), Buyya R (5), Chen N (5), Cheng XZ (5), Lago P (5), Li JH (5), and Liu AF (5). Numbers in parenthesis refer to number of published articles by the authors. This data is represented in Fig. 2.3.

Figure 2.4 represents the geographical distribution of research centres where the published articles were produced. The USA is conducting tremendous research in this field, with 167 publications during the period being considered in this study. China comes a close second, with 139 periodicals to its credit. Interestingly, third in

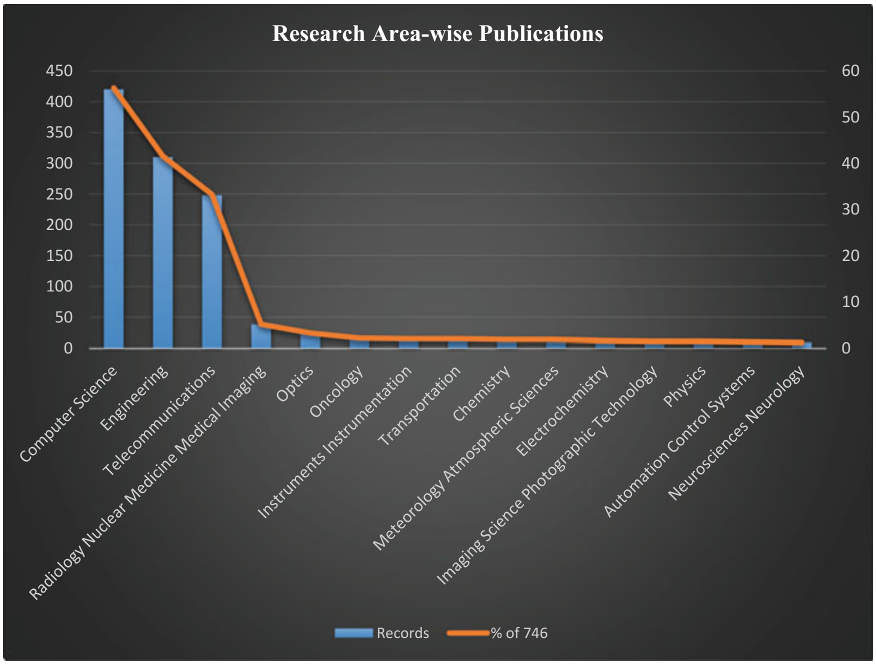


Fig. 2.2 Research areawise- publications

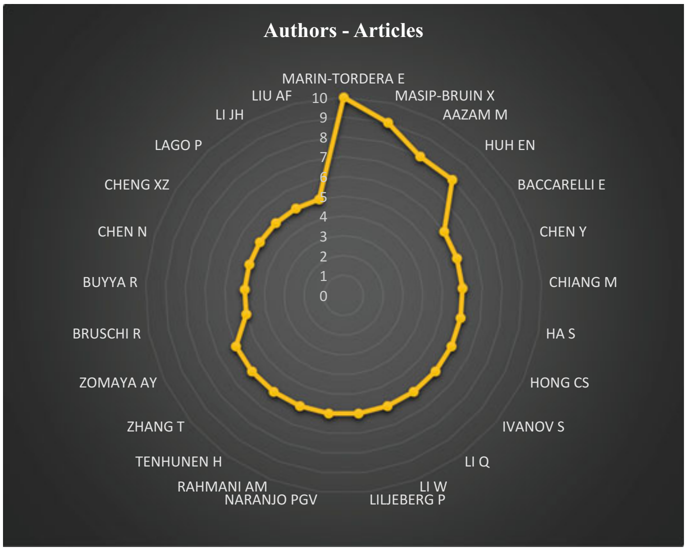


Fig. 2.3 Author wise—publications

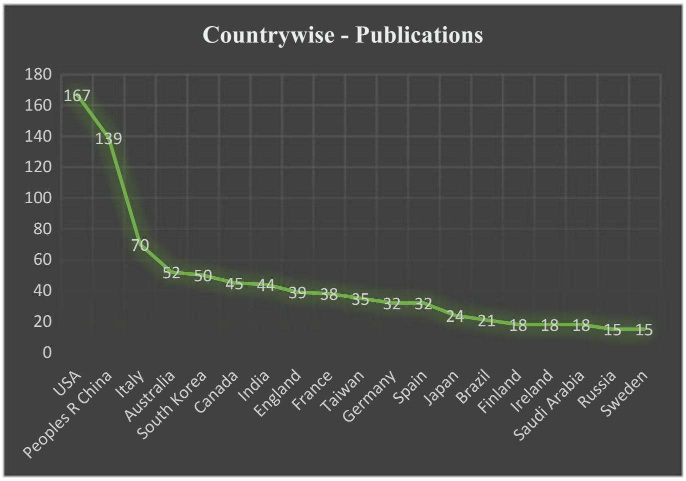


Fig. 2.4 Countrywide—publications

line is Italy with 70 publications. The others in line are Australia (52), South Korea (50), Canada (45), India (44), England (39), France (38), Taiwan (35), Germany (32), Spain (32) Japan (24), Brazil (21), Finland (18), Ireland (18), Saudi Arabia (18), Russia (15), and Sweden (15). These are the top 20 countries contributing significantly to Fog Computing research. Thus, research and development in Fog Computing is gaining popularity across countries. The International Journal of Fog Computing, through its publications, aims to provide an international forum for researchers to discuss and address these challenges and propose innovative solutions, products, theories, and concepts for seamless future computing infrastructures now and future.

2.2.3 *Patent Analysis Using Relecura Software*

Several multinational IT-related companies started working in planning, designing, and developing the innovative Fog Computing technological products and services to increase their business. These firms are working towards new hardware, software and networking products, which shall possess embedding and interoperable capabilities. On the other side, the companies are filing their new innovative products and then striving to obtain Intellectual Property Rights in the form of patents, being pioneers for particular technologies. Some individual scientists and companies are selling their patents to other companies and R&D centres to convert them into real-time products. In this scenario, authors are conducting Fog Computing patent analysis to discuss the state-of-the-art of Fog Computing products and services at the global level. This is likely to help other competitors innovate accordingly, reduce monopoly, and increase consumer convenience. For patent analysis, Relecura software was used to categorise companies' portfolios into multi-level classifications, determine the strengths and gaps in the portfolio, and plan for adequate patent coverage. This software helps track companies' needs for in-licensing or acquiring patents. This will be used to identify patents for sale or licensing and generate leads for the same. Patent documents contain important research results. However, they are lengthy and rich in technical terminology such that it takes plenty of human efforts for analysis. Automatic tools for assisting patent engineers or decision-makers in patent analysis are in great demand. Thus, patent analysis is a unique emerging management tool all around the globe for addressing the strategic management of business technologies, new products, and service development processes. Table 2.1 and Fig. 2.5 illustrate patent analysis using the Relecura software.

Table 2.1 Countrywide patent analysis on fogging

Year	China	USA	Korea	Germany	Japan	Taiwan	India	EP	WIPO
2009	2	0	0	0	0	0	0	0	0
2010	4	0	1	0	1	1	0	0	0
2011	1	1	1	1	0	3	0	0	0
2012	20	0	0	0	0	1	2	2	2
2013	13	0	1	1	0	0	0	0	0
2014	14	5	1	0	0	0	0	0	0
2015	13	1	3	0	1	0	0	1	1
2016	15	1	0	0	0	0	0	0	0
2017	31	4	1	0	3	0	0	0	0
2018	0	1	0	0	0	0	0	0	0

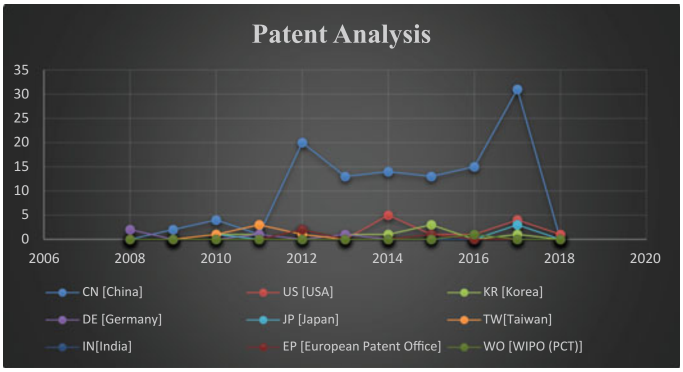


Fig. 2.5 Country wise—patent analysis on fogging

Figure 2.5 depicts the number of patents filed by different countries across the globe in the field of Fog Computing. The European countries, EP [European Patent Office] won 2 patens in 2012 and 1 in 2015, whereas WO [WIPO (PCT)] also obtained 3 patents in the same period. In contrast, China bagged a good number of patents compared to the other countries. The USA bagged 13 patents from the year 2011 onwards. Clearly, Fog/Edge Computing technologies are attracting the attention of researchers, entrepreneurs, and academicians all over the world.

2.3 Principles

Fog Computing is a hyper term that is often used as a substitute to Cloud Computing, though there are differences, as will be illustrated in the discussion below. Fog Computing is a new concept that extends the cloud computing model to the edge of the computing networks, much closer to where the devices, that generate data, exist. Fogging provides computation, data storage, and application services to the end user. This facilitates a new variety of applications and services [3]. As the Internet of Things (IoT) phenomenon matures into reality, a vast number of devices are going to connect to the Future Internet (FI). Every physical thing and object that will be connected to the Future Internet shall generate a huge volume of data each nanosecond, in the range of Giga-, Tera-, Peta-, Exa- and Zetta-bytes, and perhaps even in Yottabytes. Hence, the existing storage infrastructure with cloud computing may not be capable of carrying out the required processing in the required time. Data transmission bandwidth does not possess the required calibre to accommodate all this data on a single occasion, given the voluminous data generated by multiple heterogeneous devices, and fast data transmission rates. Further, unnecessary data transmissions in the unwanted networks of Wide Area Networks, State Area Networks, Metropolitan Area Networks, Local Area Networks, and Home Area Networks are also to be processed. Given this scenario, the concept of Fog Computing comes to the rescue. Cloud computing has established itself as an alternative of Information Technology (IT) infrastructure and service model. However, as with all logically centralized resources and service provisioning infrastructures, the cloud does not efficiently handle local issues involving a large number of networked elements and it is not responsive enough for many applications that require the immediate attention of a local controller [4]. As we discussed above, the traditional cloud-based infrastructures are not enough for the current demands of IoT applications. Two major issues are the limitations in terms of latency time and network bandwidth. The study conducted by Bierzynski et al. [5] reveal that in recent years, the concepts of Fog Computing or Edge Computing were proposed to improve these limitations by moving data processing capabilities closer to the network edge.

The term Fog Computing is frequently associated with the networking multinational company "Cisco". It is believed that Mr. Ginny Nichols, Cisco's product line manager coined the term, and hence he has been credited as the father of Fog

Computing. While "Cisco Fog Computing" is a registered name, Fog Computing is open to the community at large. Cisco recently delivered the vision of Fog Computing to enable applications on billions of connected devices to run directly at the network edge. Customers can develop, manage, and run software applications on the Cisco framework of networked devices, including hardened routers and switches. The OpenFog Consortium was founded in November 2015 by members from Cisco, Dell, Intel, Microsoft, ARM and the Princeton University. The main intention of the "OpenFog Consortium" mission was to develop open reference architecture and convey the business value of Fog Computing. Fog Computing is arguably the next frontier for accelerating IoT connections, offering increased speed, performance, and a multitude of other benefits. Fog Computing is a highly virtualized platform that provides computing, storage, and networking services between end devices and traditional Cloud Computing Data Centres, typically, but not exclusively located at the edge of the network. This means that Fog Computing provides data, computing, storage, and application services to end users. Some believe that Prof. Salvatore J. Stolfo, coined "Fog Computing" as "fogging".

Fog Computing is defined as a disseminated computing infrastructure in which the applications and services are handled either at the network edge or in a remote data centre cloud. Therefore, in this requirement, the networking "Switches" and "Mugs" and smart devices themselves become the calculating devices to carry out the computing process within nanoseconds, for local computing. The concept of Fog Computing was introduced by Cisco [6], as an analogy to Cloud Computing, as clouds are far away from earth, but the fog is closer. Hence, the term Fog Computing is considered to mimic the limited form of cloud behaviour, located nearer to end users. Fog Computing was specifically introduced to facilitate latency-sensitive applications and broader IoT vision. To mitigate real-time, mobility, and location awareness, edge technologies such as fog, mobile edge, cloudlets, and micro-data centres are foreseen as the Internet of Everything (IoE) enabling technologies [7]. That means that Fog Computing is open to the community at large. The choice of the word "fog" is meant to convey the idea that the advantages of cloud computing should be brought closer to the data source. In meteorology, "fog" is simply a cloud that is close to the ground. The cloud is migrating to the edge of the network, where routers themselves may become the virtualization infrastructure, in an evolution labelled as "the fog" [8]. With this technology, every smart building/home can have their own Fog Computing facilities for smart homeowners using smart mobiles or devices.

Fog Computing is better suited than cloud computing for meeting the demands of numerous emerging applications such as self-driving cars, traffic lights, smart homes, etc. However, it cannot replace cloud computing totally—as cloud will still be preferred for high-end batch processing jobs that are very common in the global business world. Hence, we can arrive at the conclusion that Fog Computing and Cloud Computing will complement each other while having their own advantages and disadvantages. Fog (Edge) computing plays a crucial role in the Internet of Things environment.

New studies related to privacy, security, system reliability, and security mechanisms in the Fog Computing platform are topics for fundamental research and need to be explored. Fog Computing will grow in helping the emerging network paradigms that require faster processing with less delay, whereas cloud computing would help the business community meet their high-end computing demands and lower their costs based on a utility pricing and the reliability of the networks of smart devices. Combining the reliability requirements of grid and cloud paradigms with the reliability requirements of networks of sensor and actuators, it follows that designing a reliable Fog Computing platform is a feasible model. Hence, the Fog Computing paradigm can be perceived as a non-trivial extension of the cloud [9]. This puts some types of transactions and resources at the edge of a communication network rather than establishing channels for cloud storage and utilisation. Fog Computing exponents debate that it can reduce the need for bandwidth by simply not sending every bit of information over cloud channels, and instead, gathering it at certain access points, such as modern routers. This computation allows for a more strategic compilation of data that may not be needed in cloud storage right away, if at all. By using this kind of distributed strategy, company's project managers can ensure lower costs, quicker processing and improved efficiencies of systems. In the Fog Computing environment, processing takes place in a data hub on a smart device like smart or android mobile, or in a smart router or mugs or network gateways. This reduces the amount of data sent to the cloud via a network. It must be especially noted that the fog networking complements do not replace cloud computing at any point of time; fogging allows for short-term analytics at the edge, whereas the cloud computing technology performs resource-intensive and longer-term analytics. Initially, while the edge devices and sensors are where the data is created and composed, they do not have the computation and storage resources to perform advanced analytics and machine learning tasks. Though cloud servers have the power to do these, they are often too far away to process the data and respond in a timely manner. In addition, having all endpoints connecting to and sending raw data to the cloud over the Internet can have privacy, security and legal implications, especially when dealing with sensitive data subject to regulations in different countries. In this scenario, fogging is the best available method.

Fog Computing has many advantages for services in numerous fields, such as Wireless Sensor Networks, Smart Grid, Internet of Things, and Software-Defined Networks (SDNs). However, some general issues associated with Fog Computing are privacy, trust, and security service migration among Fog Computing devices. The Global Fog Computing Market revenue is expected to reach \$556.72 million by 2023, growing at a CAGR of around 61.63% during the forecast period 2017–2023.

2.3.1 Characteristics

Fog Computing extends the cloud computing paradigm to the edge of the network, thus enabling a new breed of applications and services [6]. Characteristics of Fog Computing include:

- Low latency and location awareness,
- Widespread geographical distribution,
- Mobility,
- A large number of network nodes,
- Predominant role of wireless access,
- The strong presence of streaming and real-time applications,
- Heterogeneity.

Some authors have defined the Fog Computing physiognomies as follows:

- The main characteristic being that it is an edge location, with location awareness, and low latency, which means that Fog Computing supports endpoints with the finest services at the edge of the network.
- Secondly, its Geographical distribution has been very attractive in that the services and applications in the fog are widely distributed across spaces and nations. In sharp contrast to the more centralized Cloud, the services and applications targeted by Fog Computing demand widely distributed deployments. The Fog, for instance, will play an active role in delivering high quality streaming to moving vehicles, through proxies and access points positioned along highways and tracks [10].
- The third attribute is support for mobility, i.e., using LISP protocol fog devices to provide mobility techniques like decoupling host identity to location identity. That means it is essential for many Fog and Edge computing applications to communicate directly with mobile and other smart or computational devices. Hence, the support for mobility techniques, such as the LISP protocol, which decouple host identity from location identity and require a distributed directory system are required.
- Ability for Real-time interactions in another attribute and an essential requirement. Fog Computing requires real-time interactions for speedy service without any interruptions.
- Sky is the limit for connectivity of heterogeneous devices in the network. Fog nodes can be deployed in a wide variety of environments. These environments include the heterogeneity objects or devices such as computers, servers, laptops, printers, pagers, and smartphones as well as physical objects including both living and non-living things. Hence, one may well imagine a home where everything—from appliances to pet animals—is connected to the Internet.
- Interoperability is one of the vital networking communication characteristics in Fog Computing. This explains how different hardware devices or IoT objects interact and cooperate during the data transmission process by software through

Table 2.2 Difference between cloud and fog computing

Requirements	Cloud computing	Fog/edge computing
Latency time	High	Low
Delay Jitter	High	Very low
Access	Fixed and wireless	Mainly wireless
Service location	Within the internet	At the edge of the network
No. of server nodes	Few	Very large
Distance (client–server)	Multiple hops	Only one hop
Security	Undefined	Can be defined
Location awareness	No	Yes
Support for mobility	Limited	Supported
Geo distribution	Centralised	Distributed
Availability	99.99%	Highly volatile/redundant

various networks. Fog components must be able to interoperate in order to give a wide range of services such as platform-independent streaming.

- The end object is to improve the quality of life of general public. To do so, one will have to overcome bandwidth limitations, increasing computing capabilities and storages in these heterogeneous environments, and fogging technology is the best method to fulfil these requirements.

Table 2.2 delineates the differences between Cloud and Fog Computing [11].

2.3.2 *Concepts*

The Fog Computing technology offers a number of benefits in the computing and communication arena. As defined by Cisco in one of their research papers, "by moving data processing and delivery closer to the requesting devices through the expansion of their geographic network footprint and operating in node-based fog architecture rather than in either cloud or traditional systems, response times to customers can be improved almost exponentially". Nearness is the major contributor to network efficiency, and the effect of moving data towards the consumer along a communication network path through such fog nodes simply cannot be overstated. The increased network efficiency through proximity multiplies many of the benefits of the classic architecture while avoiding the drawbacks of such. As part of this movement towards network nodes, congestion on the network is likewise drastically reduced. In the traditional architectural paradigm, the total of processing is being demanded in a rather centralised system. Essentially, standard architectures are glorified bottlenecks in all but a few specific use cases, where data is trapped in a looping hold pattern. Thus, Fog Computing gets around this by distributing this functionality across a much wider space. While cloud distribution can indeed be negated by some solutions, it is problematic, as it also means that network response as a whole suffers, even with multiplexing and other load balancing solutions. Users may still face bottlenecks, but the concept of load balancing has been amply made use of, resulting in quicker data delivery and better network efficiency. As more nodes become necessary, dormant nodes can be spun and utilised. Fog Computing also boasts of impressive scaling abilities. Working with nodes implies geographical nearness to the data request. This also indicates a boost in security—data is encoded as it moves towards the edge of the network; and changing node relations and presence means that the user attack surface is ever changing. End devices, fog and cloud form a three-layer hierarchical service delivery model, supporting a range of applications such as web content delivery [12], augmented reality [13], and big data analysis [14].

However, several challenges are also associated with the Fog/Edge Computing. The major disadvantage of the Fog technology and architecture is that it intensifies the complexity of the communication network, and thus, adds a certain amount of overhead in business terms—both directly and indirectly. Furthermore, implementing Edge Computing means that the user is introducing an ever-increasing number of points of failure. While privacy and security are benefited in a way by this constant change, the maintenance and probability aspects are not. Besides, the security and privacy should be addressed in every layer while designing the fog Computing system [15]. In old-style solutions, the user has a single point of failure with centralised effort to repair, maintain, and identify potential issues in the Fog Computing architecture. By spreading out the computational load, users are spreading out that effort and responsibility, which can make for stress on the entire Fog Computing process. Fog is another layer of a distributed network environment and is closely associated with cloud computing and the Internet of Things.

2.4 Models/Architectures

No standard architectures and tested models have been derived from empirical research. However, some of the Fog Computing architectures have been described in the following sections for better understanding of the Fog vision.

2.4.1 A Generic Fog Computing Architecture

Mind Commerce [22] has outlined a generic Fog Computing architecture that consists of the following elements:

- IoT endpoints: which include end devices like sensors, gateways, edge devices, and other physical computing objects that perform the required functions and provide apps that are installed on the end devices. These devices are generally connected to a data storage. The IoT endpoints collect the data from other different devices; process some data in real-time in the IP network data processing units. The rest of the data is stored in a Cloud environment for further processing.

- IP (Internet Protocol) Network: that provides distributed intelligence and is linked to IoT End-point devices (e.g. sensors) to receive data from and also to the Cloud environment to send required data for storage purposes and future analysis. Purpose is mainly to provide communication links. the IP network utilises distributed intelligence for supporting intelligence representation e.g. big data mining etc. Distributed Intelligence is considered as a collective intelligence that provides an effective theoretical framework for understanding what humans can achieve and how artefacts, tools, and socio-technical environments can be designed and evaluated to empower human beings.
- Centralised Control and Mediation Unit: which is a cloud-based unit in the Cloud environment where the remaining data, which was not processed in the data processing unit, is stored and analysed, as required. This unit is used for storage, queries, and business analytics.

2.4.2 Fog Computing Environmental Model

Zhu et al. [16], in their seminal research paper "Improving website performance using edge servers in Fog Computing architecture" structured the Fog Computing Environmental Model. This model has been presented in Fig. 2.6. They have divided the entire Fog Computing environment into three layers as follows:

Centralised Intelligence: Cloud Computing

Here, Data Centre Cloud may be employed for services for application hosting, management. The Core Networking and Services are responsible for Internet Protocols (IP), Multiprotocol Label Switching (MPLS), Quality of Service, Multicast, Data Security, Network Services, and Mobile Packet Core functions.

Distributed Intelligence: Fog Computing

This layer is the primary one at the edge level. It may also be called Multi-Service Edge. In this layer, 3G/4G/Long-Term Evolution (LTE), Wi-Fi, Ethernet, and Programmable Logic Controller (PLC) technologies and other such can be encountered. This is much like the field area network.

Distributed Intelligence: End-Point Computing

It can alternatively be called the Smart Things Network. This layer consists of several embedding systems, objects and sensors. This indicates that smart and less smart things like vehicles and machines are connected via wired and wireless connections.

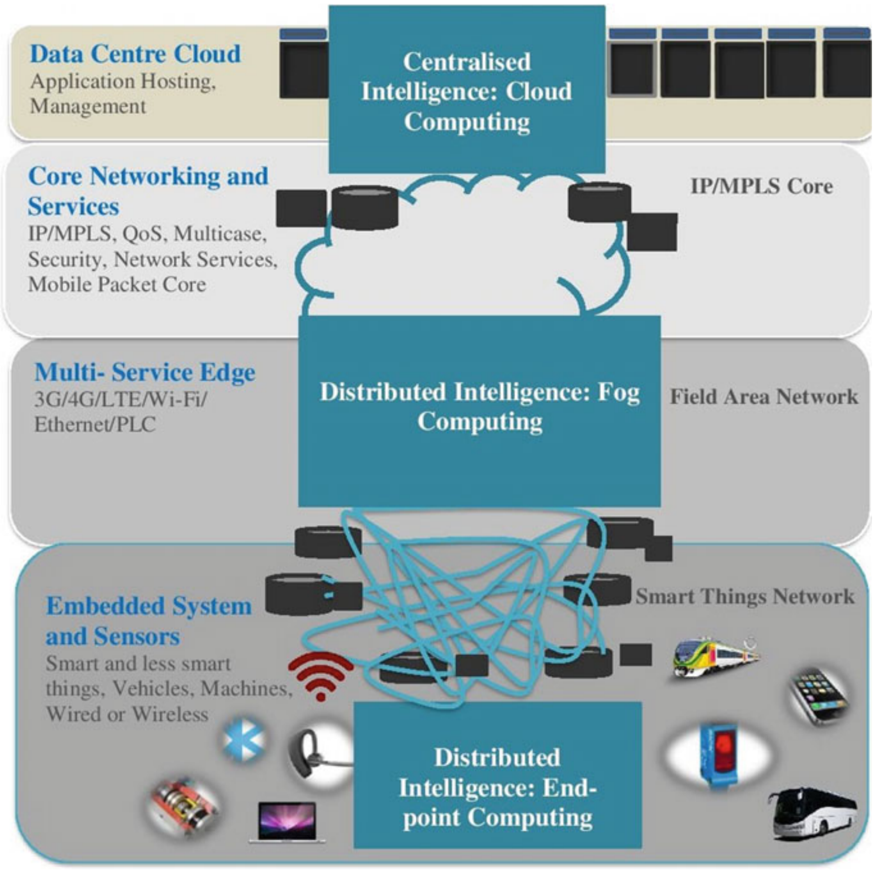

Data Centre Cloud
Application Hosting, Management
Centralised Intelligence: Cloud Computing
Core Networking and Services
IP/MPLS, QoS, Multicase, Security, Network Services, Mobile Packet Core
IP/MPLS Core
Multi- Service Edge
3G/4G/LTE/Wi-Fi/ Ethernet/PLC
Distributed Intelligence: Fog Computing
Field Area Network
Embedded System and Sensors
Smart and less smart things, Vehicles, Machines, Wired or Wireless
Smart Things Network
Distributed Intelligence: End-point Computing

Fig. 2.6 Fog computer architecture (adapted from [16])

2.4.3 *A Fog Computing Architecture by Joud Khattab [21]*

The Fog Computing model (Fig. 2.7) by Joud Khattab was published in August, 2017. In this approach, the Fog Architecture has been divided into seven layers, as briefly articulated below.

IoT Sensors and Actuators

The bottom-most layer of the architecture in Fig. 2.7 is formed by IoT sensors and actuators. This layer is geographically distributed and possesses sensing, communication, and application objectives, directing the fetched values to the next layers via gateways for further processing. The IoT actuators play an important role in this layer as they are responsible for controlling the mechanism or IoT system. They fetch the data from the remote sensors and convert them into mechanical or

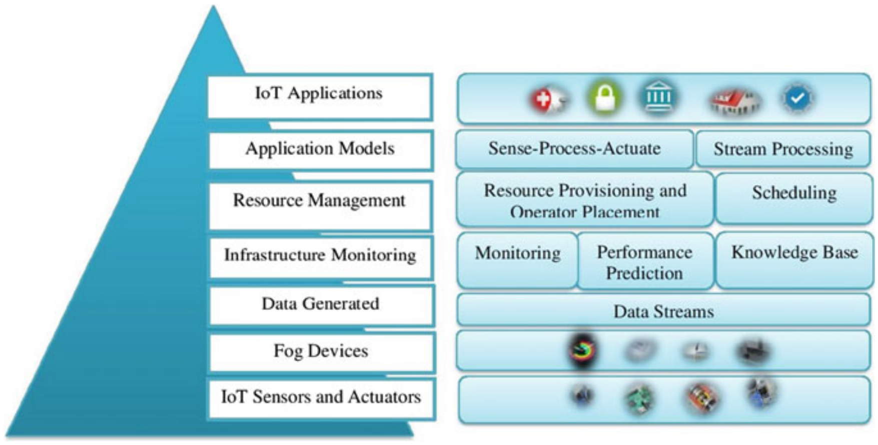


Fig. 2.7 Fog computer architecture by Joud Khattab (adapted from [21])

electrical moments. The actuator usually responds to the sensors that sense the changes in the environmental deviations.

Fog Devices

The next layer of the architecture has the "Fog Devices" that consist of gateways for connecting the sensors to the Internet, edge devices and physical computing devices. This layer is capable of hosting the different sorts of application modules.

Data Generation

The collected data from different sensors, RFID and other IoT devices is processed further through this layer; of this, some data can be used for real-time processing; some of the rest can be sent to a Cloud environment. An enormous amount of data is generated at this layer.

Infrastructure Monitoring

This layer is mainly responsible for the monitoring of resource utilisation, availability of sensors, actuators, and IP network. It monitors and keeps track of the services and applications deployed on the infrastructure. Depending on the requirement, the monitoring component supplies the gathered information to other services and application objectives.

Resource Management

The core layer of Fog architecture is resource management. It consists of computational components that systematically manage resources in such a way that resource wastage is virtually minimised to zero. This layer has a uniform and programmable interface for seamless resource management of hardware and controlling. It provides generic Application Process Interfaces (APIs) for monitoring, and controlling physical resources such as CPU, Memory, network and IoT devices. In this layer, the placement and scheduler components are the most important parts as they keep track of the state of available resources.

Application Model

This layer of Fog architecture consists of the sense-process-actuate model in which the sensor publishes measured data either periodically or in an event-based manner. Applications running on Fog devices subscribe to and process data coming from the sensors. Finally, insights obtained from the processing are translated into actions forwarded to actuators. Two relevant models are briefly described below.

- Sense-process-actuate model: The information collected by sensors is transmitted as data streams, which is acted upon by applications running on fog devices and the resultant commands are sent to actuators.
- Stream-processing model: The stream-processing model has a network of application modules running on fog devices that continuously process data streams emitted from sensors. The information mined from the incoming streams is stored in data centres for large-scale and long-term analytics. We consider stream-processing model as a subcategory of the sense-process-actuate model [17].

IoT Applications

No field, societal or commercial, has been left untouched by the Internet of Things (IoT) applications. IoT scenarios includes healthcare, manufacturing, governance, urbanisation, home automation, citizen security, education, transportation, defence and many more [18].

2.4.4 Fog Computing Tree Model

Figure 2.8 shows another proposed architecture for the Internet of Things (IoT) environment. This has been called the “Tree Model”. It describes the IoT architecture consisting of three layers. It has been developed based on the IoT Forum Architecture, ITU Architecture, and European FP7 Research Project ARM models of IoT Architectures. The model is an addition to IoT-ARM (Architectural Reference Model) by the FP7 Research Project. The approach describes the components including connectivity among them and process flow. IoT “Tree Model” is like a tree structure with roots, trunk and unlimited leaves [19], [20]. The three components of the Tree Model are the Processing, Transportation and Application Layers, as elaborated below.

Processing Layer

For every tree, roots are the base and they are essential for the growth and entire life of the tree. Similarly, in an IoT architecture, Processing Layer is the principal layer

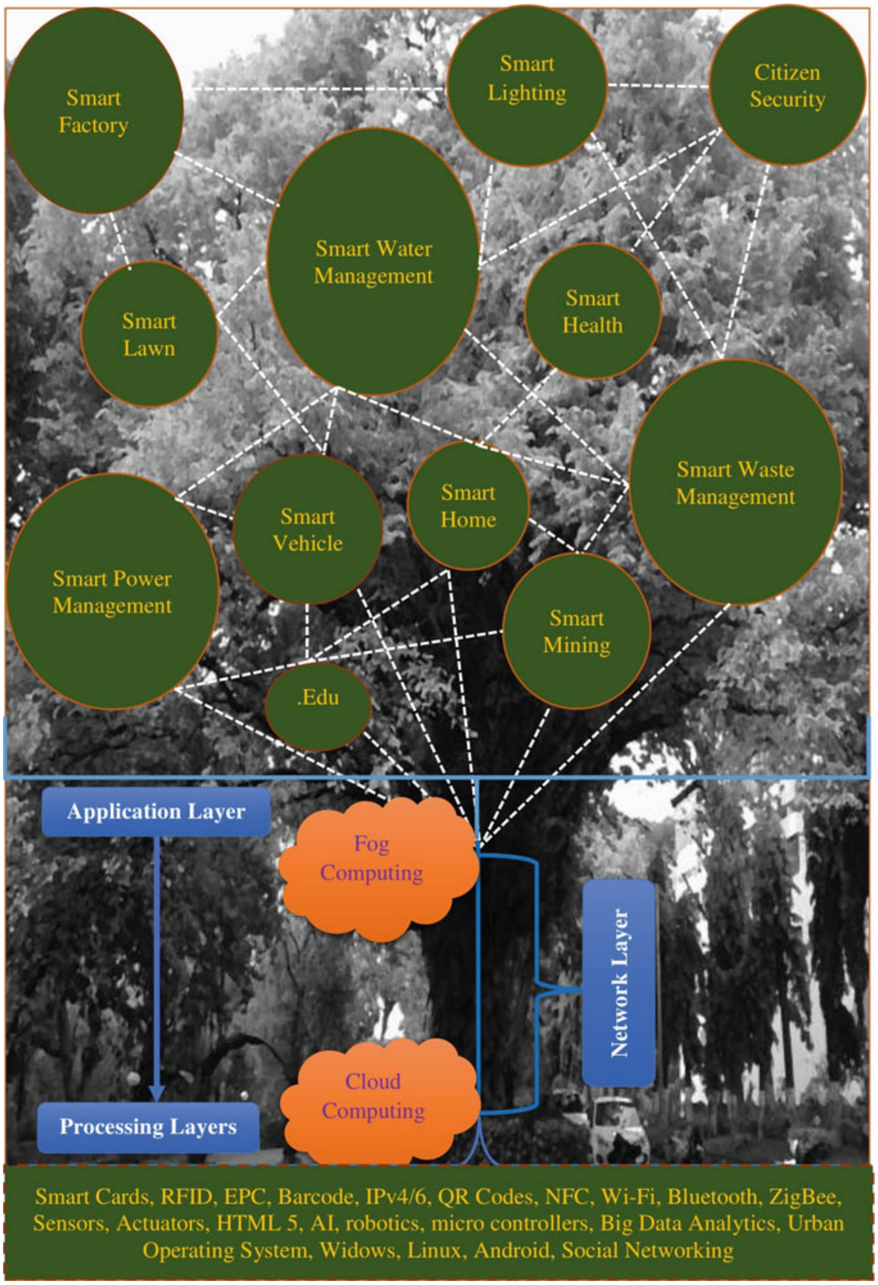


Fig. 2.8 Fog computing view of IoT—A tree model

in the global IoT world, with interoperable capabilities. This layer generally has the computing and embedding abilities with different physical objects or things. The relevant technologies help in the identification of the objects, securing them, and monitoring the events. Of course, these technologies are also capable of conducting automation based on the protocols defined in the network connection and may end up controlling even human beings.

A few of the technologies involved in the Processing Layer include Smart Cards, Radio Frequency Identification, Electronic Product Codes, Barcode, IPv4/6, QR Codes, NFC, Wi-Fi, Bluetooth, ZigBee, Sensors, Actuators, Hyper Text Mark-up Language 5, robotics, microcontrollers, Big Data Analytics, Urban Operating Systems, Windows, Linux, Android, and Social Networking.

Network Layer

The Networking Layer is placed between the Processing and Application Layer. However, there are no hard and fast rules that fixate this sequence and in Fog Computing, the Processing Layer and Application Layer are likely to be embedded into each other. In the Network layer, there are sub-networks consisting of Home Area Network (HAN), LAN, MAN, SAN, WAN, and overlapping.

HAN is a new and relatively unknown network. All the home appliances including towels, shirts, utensils, windows, doors, tables, fans, fridges, radiators and so on can get connected to the Internet through smart mobile/devices for easy automation. To ensure ease of application, this Network Layer has high bandwidth capabilities, and high data rate speeds for the transmission of yottabytes of data. The 5G mobile and NG9-1-1 technologies are soon likely to be in place to further enhance the capabilities. Fog Computing will occur via switches, thermostats, actuators and smart mobile devices of this networking layer.

Application Layer

The Application layer is similar to leaves of the “Tree”. Like the uncountable leaves on a tree, the IoT Application layer consists of a gamut of applications including transportation, security, governance, education, health, pharmacy, logistics, manufacturing and process industries. Beside these, Application Layers support the technologies for improved Quality of Life in urban areas in public utility services. These services include power management, water management, waste management and irrigation management. In fact, even mining processes can be monitored through this and the monitoring of natural ecosystems including ponds, forest, lakes, and mountains also is possible, too. IoT has brought about a tremendous number of services for round-the-clock provision of Quality of Life.

2.5 Conclusion

Fog Computing-based systems are becoming important in our daily lives. It is amply clear from the above discussions that Fog Computing will, in future, be omnipresent and pervading across all disciplines, social and commercial. Through this, even earthen things are likely to get high calibre computation capabilities, through the inclusion of basic electronic devices and embedded processing capabilities. The beauty of this technology is that all smart devices, including smart phones, already possess this computation technology. The user-oriented applications are already fully loaded in almost all these mobiles. These applications include Google, music, album, movies, camera, calendar, maps, alarms, Facebook, Twitter, navigation, voice speech, Hangouts, voice dialler, WhatsApp, notes, Email and so on. Edge technologies can be used to assist free-view videos by offering virtual view synthesis and related multi-view video incentives. All these applications require local storage and high-speed real-time processing, along with high-speed connected bandwidth to perform edge computing. Even in the case of home automation, smart devices are coming on board with facilities such as computation, residential tracking, sensing and adapting the IoT environments through Ambient Intelligence (AmI), as well as manipulation of databases in real-time situations. All the IT-related companies are developing IoT products and services to interoperable Fog Computing hardware, software, and networking. However, researchers still have much work to do on the fundamentals of Fogging and its allied applications to develop standard Fogging models and architectures.

References

1. Madakam S, Date H (2016) Security mechanisms for connectivity of smart devices in the Internet of Things. In: Mahmood Z (ed) Connectivity frameworks for smart devices. Springer, Cham, pp 23–41
2. Tseng YH, Lin CJ, Lin YI (2007) Text mining techniques for patent analysis. Inf Process Manage 43(5):1216–1247
3. Jain A, Singhal P (2016). Fog computing: driving force behind the emergence of edge computing. In: System modeling and advancement in research trends (SMART), international conference on IEEE, pp 294–297
4. Dang TD, Hoang D (2017) A data protection model for fog computing. In: Fog and mobile edge computing (FMEC), 2017 second international conference on IEEE. IEEE, pp 32–38
5. Bierzynski K, Escobar A, Eberl M (2017) Cloud, fog, and edge: cooperation for the future? In: Fog and mobile edge computing (FMEC), 2017 second international conference on IEEE. IEEE, pp 62–67
6. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role on the Internet of Things. In: Proceedings of the first edition of the MCC workshop on mobile cloud computing. ACM, pp 13–16
7. Bilal K, Erbad A (2017) Edge computing for interactive media and video streaming. In: Fog and mobile edge computing (FMEC), 2017 second international conference on IEEE. IEEE, pp 68–73

8. Vaquero LM, Rodero-Merino L (2014) Finding your way in the fog: towards a comprehensive definition of fog computing. ACM SIGCOMM Comput Commun Rev 44 (5):27–32
9. Madsen H, Albeanu G, Burtschy B, Popentiu-Vladicescu FL (2013) Reliability in the utility computing era: towards reliable fog computing. In: Systems, signals and image processing (IWSSIP), 2013 20th international conference on IEEE. IEEE, pp 43–46
10. Waheetha R, Sowmya F (2016) Fog computing and its applications. Int J Adv Res Basic Eng Sci Technol (IJARBEST) 2(19), October 2016
11. Abdelshkour M (2015) IoT, from cloud to fog computing. Cisco Blog-Perspect. https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing
12. Zhu J, Chan DS, Prabhu MS, Natarajan P, Hu H, Bonomi F (2013) Improving web sites performance using edge servers in fog computing architecture. In: Service oriented system engineering (SOSE), 2013 7th international symposium on IEEE. IEEE, pp 320–323
13. Ha K, Chen Z, Hu W, Richter W, Pillai P, Satyanarayanan M (2014) Towards wearable cognitive assistance. In: Mobisys. ACM
14. Zao JK, Gan TT, You CK, Chung CE, Wang YT, Rodríguez Méndez SJ, et al (2014) Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology. Front Human Neurosci 8:370
15. Yi S, Qin Z, Li Q (2015) Security and privacy issues of fog computing: a survey. In: International conference on wireless algorithms, systems, and applications. Springer, Cham, pp 685–695
16. Zhu J, Chan DS, Prabhu MS, Natarajan P, Hu H, Bonomi F (2013) Improving web sites performance using edge servers in fog computing architecture. In Service oriented system engineering (SOSE), 2013 7th International Symposium on IEEE. IEEE, pp 320–323
17. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R (2017) iFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw Pract Exper 47(9):1275–1296
18. Madakam S, Ramaswamy R (2015) 100 new smart cities (India's smart vision). In Information technology: towards new smart world (NSITNSW), 2015 5th National Symposium on IEEE. IEEE, pp 1–6
19. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener Comput Syst 29(7):1645–1660
20. Khan R, Khan SU, Zaheer R, Khan S (2012) Future internet: the internet of things architecture, possible applications and key challenges. In Frontiers of information technology (FIT), 2012 10th international conference on IEEE. IEEE, pp 257–260
21. Khattab J (2017) Fog computing. https://www.slideshare.net/joudkhattab/fog-computing-78498759
22. Mind Commerce (2017) Computing at the edge series: fog computing and data management. http://www.mindcommerce.com/files/FogComputingDataManagement.pdf

Chapter 3
Dichotomy of Fog Computing in the Realm of Cloud Computing: Exploring the Emerging Dimensions

Kelvin Joseph Bwalya and Bvuma Stella

Abstract There is much hype about the emergence of Cloud and Fog Computing in achieving a truly pervasive access to information and computing resources. Given the increased need for the instantaneous access and analysis of data with huge dimensions (big data, with respect to volume, velocity and varied formats) and the emergence of predictive analytics, individuals and businesses will experience over-reliance on Fog Computing. Despite the promise, scholars and researchers are finding it difficult to distinguish between Cloud and Fog Computing and find little time in investigating the key issues that affect these emerging computing models. In this context, the aim of this chapter is to explore the formulaic definitions and key issues surrounding Cloud and Fog Computing. Using a meta-analysis of recent literature on Cloud Computing literature and authors' experience, this chapter presents a development projectile and emerging dimensions of computing. The chapter articulates how Fog Computing can be used at the centre of instantaneous ubiquitous data and information management which is a key requirement for competitive business entities in any given setup. At the end of the chapter, we discuss Fog Computing landscape in Africa.

3.1 Introduction

Many organisations are transforming their organisation's data management roles to include Cloud and Fog Computing as a progressive reliant data and information platform. Further, the Cloud Computing environment offers scenarios for seamless and elastic technology resource sharing and provision unlocking opportunities for ubiquitous information management [1]. It cannot be overemphasised that Cloud Computing is a paradigm shift to the traditional computing technology model where heterogeneous end-users are able to access computing resources and information

K. J. Bwalya (✉) · B. Stella
School of Consumer Intelligence and Information Systems,
University of Johannesburg, Johannesburg, South Africa
e-mail: kbwalya@uj.ac.za

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_3

using identical operations [2]. Cloud Computing provides virtually shared IT resources with desired flexibility and modularity reducing the cost of providing requisite IT services. Given this fact, many organisations, even in the context of Africa such as the University of Johannesburg, have migrated many of their information management tasks to the Cloud where its employees use thin clients to access Cloud services.

Cloud Computing applications are rapidly developing given the ever-increasing business needs and technological advances rather than user needs. It cannot be overemphasised that Cloud Computing enables the sharing of a pool of resources such as servers, storage, networks, applications, and a variety of services, thereby completely transforming the way computing is perceived [3]. Cloud Computing can be used especially in smart applications where the computation battery need not be flat at any time. For example, consider a situation where an old man is let to live alone in an isolated building. For someone to monitor his health and safety, one can mount sensors throughout his house and on his body. These sensors are able to automatically send signals on sleeping routines, health activity and his general safety to relevant audience for monitoring or for action. This scenario depicts the three typical layers of the Internet of Things (IoT) upon which many Cloud Computing applications are designed: Sensing layer; Network layer; and Application layer.

Despite a lot of progress made in Cloud and Fog Computing research and practice, there are still serious issues such as limitation in the provision of appropriate and adequate levels of seamless resource, information interchange and dynamic elasticity, guaranteed service-level agreements (SLA) and the quality of service (QoS) that need to be addressed. Other inherent issues include the appropriateness of service and programming interface which in many instances causes lock-in, privacy and trust issues, inadequate bandwidth, etc. [1]. Further, problems such as lack of requisite mobility support, unreliably low latency, location awareness and agent identify prevent Cloud Computing from being used at a large scale [4].

The main objective of this chapter is to locate Fog Computing within the broader field of applied information systems, present synoptic articulation of gaps in research and practice of Fog Computing, present the current status of Cloud and Fog Computing in Africa (ICT infrastructure development status (fibre networks, virtual networks, etc.), explore the existing legal and regulatory frameworks, etc., define key terms in emerging Fog Computing research and practice domains, and to present prior work done in this field, thereby articulating the gaps and grey areas, etc., that need to be explored.

It is worth noting that this chapter does not claim to present novel ideas but merely re-emphasizes many of the concepts that are currently being discussed in the research community worldwide. The ‘novelty’ of this chapter is that it punches into the future dimensions of Cloud Computing in as far as Cloud Computing implementation in the developing world is concerned. Further, the chapter is vital to advancing scholarly discourse in that it explores a Cloud Computing architecture which can be used in real-world setups.

The arrangement of the chapter is as follows: the first section discusses the fundamental concepts of Cloud Computing highlighting the key characteristics, the second part discusses the key attributes of Fog Computing and thereafter compares it with Cloud Computing, the third part discusses the design aspects of Cloud and Fog Computing, the next section discusses the different issues that need to be overcome in the Cloud/Fog Computing solutions such as legal aspects, and the last part explores the status of Cloud/Fog Computing in Africa.

3.2 Key Tenets of Cloud Computing

Appreciating the key tenets of any phenomenon emanates from understanding in detail what the phenomenon entails in any given context. It is important, therefore, to understand what Cloud Computing entails in order to delve into its different domains.

Several researchers have attempted to advance different definitions of what Cloud Computing entails given the context in which they operate. Although this is the case, there is a lack of agreed global definition of what Cloud Computing entails. However, many people have adopted the NIST definition as one that fits many contextual nuances. Therefore, many definitions for Cloud Computing have been based on the NIST SP-800-145 definition: *A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model promotes availability and is composed of five essential characteristics, three service models and four deployment models.*

The major tenet of Cloud computing has to do with the drive to exploit opportunities of how to manage, process and integrate ashore and afloat data, information and different network resources into the different business processes of the organisation. The end-result is a significant reduction into IT costs that would have otherwise been incurred by an organisation. Depending on the model of Cloud Computing utilised and in what contextual setting the implementation is pursued, in some circumstances Cloud Computing may prove to be an expensive option especially in situations where the company offering Cloud services is corrupt. Figure 3.1 below highlights the three basic levels of Cloud computing which articulates the basic elements of Cloud Computing.

The layer with the highest degree of abstraction is the one detailing the essential characteristics of Cloud computing applications. One of the key concepts upon which Cloud Computing is hinged is resource pooling which entails that the Cloud resources are accessible using a multi-tenant model by many end-users simultaneously. Cloud Computing uses dynamic provisioning which allows a single resource to be accessed by multiple applications simultaneously. The Cloud services are measured or timed and controlled to ensure that there is no abuse of the service.

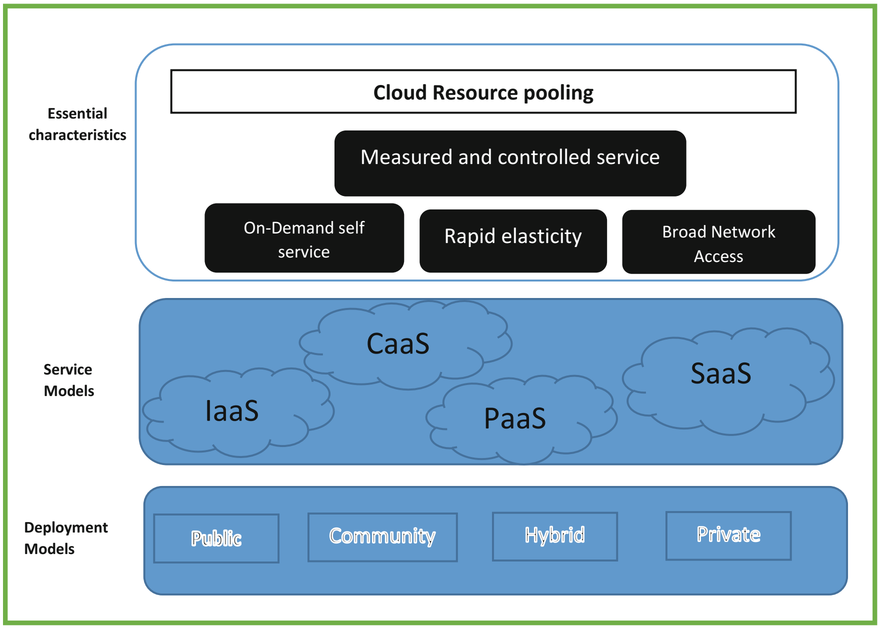


Fig. 3.1 Key elements of Cloud Computing paradigm

Each Cloud service needs to be offered immediately (on-demand self-service) in many cases without the intervention of a human being. The Cloud services also need to rapidly elastic and allow requisite broad network access—this means that the Cloud services need to be vendor neutral and heterogeneous network configurations should be able to access the services using identical operations or open interfaces. The middle layer articulates some of the different service models that are used. As espoused in Fig. 3.1, there are different service models for Cloud computing. These service models allow the consumers to access Cloud services, resources and applications in different ways. Some of them are articulated below [5]:

- **Cloud Platform as a Service (cPaaS)**: Using this model, programming tools and languages are deployed into the Cloud infrastructure to create a platform or interface for accessing the different resources provided in the Cloud. Unlike in the cSaaS, the consumer does not go to control the Cloud infrastructure (network, servers, storage, operating systems, etc.). However, the consumer can be given a degree of control of the deployed applications. Force.com, Google* App Engine, Red Hat* OpenShift*, VMware Cloud Foundry, and Windows Azure are examples of cPaaS. Consumer and the producer are both given adequate privileges to customise applications of the Cloud interface or service. An example of cPaaS is Salesforce1 Platform that gives opportunities to end-user apps based on the salesforce platform limiting them in getting user interfaces based on applications from other vendors.

- **Cloud Software as a Service (cSaaS)**: Here, the consumer is given rights to access applications of the provider of a service running on Cloud infrastructure through a thin client like Web-based mail. Using this model, the consumer is not mandated to control or manage the Cloud infrastructure. In general, both the end-user (consumer) and the producer have very limited control of the interface upon which the Cloud Computing service is accessed. Amazon* Elastic MapReduce, Cetas* by VMWare* analytics solutions, Google* BigQuery services, Rackspace* Hadoop* service, and Windows Azure* HDInsight* are examples of cSaaS.
- **Cloud Infrastructure as a Service (cIaaS)**: In this model, the consumer has the privilege to provide processing, storage, networks, etc., and also be able to deploy and run arbitrary software. The consumer is not mandated to manage the Cloud infrastructure. The cIaaS accords end-users maximum flexibility to customise their Cloud services and therefore provides opportunities for the development of a consistent user interface commensurate with the user characteristics. Amazon* Web Services, Citrix* CloudPlatform, Windows Azure* and Microsoft* System Center, OpenStack* software; Rackspace*, Savvis*, Verizon* Terremark* and VMware vCloud* Suite are examples of cIaaS.
- **Cloud Communication as a Service (cCaaS)**: This is a key platform for interactions between producers and consumers of Cloud or Fog Computing resources. It provides collaborative and audio/video communication services, unified communications, instant messaging, email, data sharing in dynamic platforms such as Web conference, etc.
- **Cloud Network as a Service (cNaaS)**: Service providing the main space for managing Internet access of Cloud resources (speed, availability, etc.). It includes virtual private networks (VPNs) and Cloud Computing services which are generally based on flexible and on-demand bandwidth.

The service models articulate the different usability needs of the consumers in the light of their privileges with regard to resource interaction and management. However, despite the many advantages brought about by the different service models, it’s not automatic that anyone employing any of these models will benefit from the implementation. There is a need for contextual other than formulaic models in order to realise their full potential. Contextual service models are going to be designed baring the contextual nuances in the area in which they are to be employed. Other than the service models, deployment models are worth considering when designing Cloud service applications. Deployment models depict how Cloud infrastructure is made available to the different sets of consumers. Some of the common deployment models are briefly discussed below, mainly for the sake of completeness:

- **Community Cloud**: Several organisations are able to share the Cloud infrastructure in order to share Cloud resources for the benefit with regard to shared concerns. Given a situation where a community uses the cPaaS model, the service-oriented-architecture (SOA) applets are specific to the overall requirements of the community, e.g. industry-specific, business-process-specific. This allows members of the community to easily access Cloud service applications as the underlying programming models have been made and they only have to plug in their service needs.

- **Private Cloud**: In this deployment model, the Cloud infrastructure can be deployed on-sight or off-sight and is managed by the organisation or a third-party entity solely for the organisation. This deployment model entails that no one other than the members of the organisation have the privilege to access the Cloud network. Such kind of configuration is synonymous to an intranet which restricts the access of the network resources to outsiders.
- **Public Cloud**: The general public has access to the Cloud infrastructure and its resources or services. The need of such a network to design the Cloud service applications on open and interoperable interfaces cannot be overemphasised.
- **Hybrid Cloud**: This is a composition of two or more Clouds retaining unique attributes from each of the Clouds involved. The Cloud uses standard or proprietary technology enabling data and application portability. In most cases, this is a powerful Cloud which is very competitive given the context in which it operates.

The different service and deployment models need to be considered when designing Cloud computing services regardless of the context in which this is implemented. It worth emphasising that following the standard design principles and considering the local contextual characteristics when designing Cloud Computing will culminate into improved cost efficiencies and innovation, improved access to network or Fog Computing environment [6].

3.3 Cloud Versus Fog Computing

Just like Cloud computing as discussed above, in order to understand the contours of Fog Computing, one first needs to understand what Cloud Computing entails. Good-enough understanding of the logical and conceptual outlay of Cloud Computing is important before attending to understand; let alone define what Fog Computing is. The understanding is desired to articulate the differences between the two concepts and articulate why there is a need for such a delineation.

Both the Cloud and Fog Computing are conceptually based on the understanding of the Internet of Things (IoT). There has been a continuous push from technology innovators to augment IoT applications with Cloud Computing. The IoT articulates a connectivity environment which allows different resources to be accessed using an interplanetary network hinged on either of the following: Personal Area Network (802.15), Local Area Network (802.11), Metropolitan Area Network (802.16) and the Wide Area Network (802.16). Both Cloud and Fog Computing environments need to consider the basic building blocks that are needed in designing dynamic and competitive application environments. Some of these technical building blocks are discussed below:

- Service Oriented Architecture (SOA)—this provides a library of tested and functional software applets that can be configured together to become useful application.
- eXtensible Markup Language (XML)—this allows the use of identifier tags that are used to transport information of any kind to any designated application based on the Internet.
- Application Programming Interfaces (APIs)—these are technical tags that can be used to direct applets concerning the Internet.

Fog Computing is a system-level architecture that brings together storage, processing and networking closer to the end-users with the Cloud-to-thing consortium in edge computing [7]. Fog Computing takes virtualisation to another level using edge computing. Virtualisation enables virtual machine instances or states to be established on the heterogeneous wireless components. Cloud Computing largely focusses on making resources available through a virtually connected core network, whereas Fog Computing focusses on the provision of elastic resources/services at the edge of the network enabling new applications designed upon new and emerging technologies. An example of edge technology is a commercial edge router which is advantageous over the traditional routers by having built-in network storage capacity, increased processing speed, etc [4]. Fog Computing is an extension of Cloud Computing with the differences in the network configuration where the latter is implemented using edge of the network and the former is implemented using the core of the network [4]. A common understanding of Fog Computing is that heterogeneous non-disparate pervasive devices communicate and can potentially coordinate amongst themselves in a scenario where some of them perform storage and processing tasks.

Fog Computing is implemented using different configurations. In a situation where Fog Computing is used at the server point in the processing and transition of information, it significantly reduces the data latency and facilitates seamless transmission of information. In this regard, the Fog server allows dynamic customizable optimization owing to the client device and local context characteristics. The Fog server can further sniff the user experience, knowledge which can be used to improve the Web page rendering and surfing experience.

Synonymously, computing paradigms such as Mobile Cloud Computing (MCC) and Mobile Edge Computing (MEC) are used in place of Fog Computing applications. An example of MEC application could be a Cloud server mounted at the end of a mobile network performing tasks that could not be traditionally executed in a conventional network configuration [4]. The Google Glass, Microsoft HoloLens and the Sony SmartEyeglass are examples of some of the popular products designed on the Fog Computing conceptualisation [4]. Fog Computing opens opportunities for enhanced capabilities with regard to data processing of large data sets. This processing can achieve big data acquisition, aggregation and preprocessing enabling data transportation and thereby balancing computing power. Further, Fog Computing extends the benefits that can be extracted from the Internet of Things.

3.4 Promise of Cloud and Fog Computing

There is undoubted agreement amongst experts that the implementation of Cloud and Fog Computing culminates into tangible benefits in the realm of flexibility and convenience. The potential of Fog Computing in changing the way the end-user computing is perceived cannot be overemphasised. However, there is also a line of analysis which has revealed that Cloud or Fog Computing does not automatically culminate into gains and reduction of costs as in some cases the gains may be less than the costs of the entire model. Further, the ‘opportunity cost’ of engaging in Cloud Computing is sometimes high in certain contain contextual settings. The concept of the opportunity cost kicks in especially in resource-constrained environments like Africa when it comes to considering whether to consider Fog Computing or not. The unplanned high costs in Cloud Computing can be the transfer of very large data sets to the Cloud and storing it there over longer periods of time can be very costly to the company in the long run. Further, movement of data sets between the company and the Cloud can culminate into huge costs in terms of bandwidth especially if the throughput is low and data latency is high. The key advantages of Cloud and Grid Computing include the following:

- Possibility of large enterprises in offloading a portion of their information management roles to the Cloud and small enterprises including start-ups are able to translate some of their business domains without incurring huge start-up infrastructure costs.
- Accords chance to system developers to concentrate on the core of designing business logic having not to worry about the question of infrastructure management and scalability.
- Service consumers can access services ubiquitously translating into increased convenience.

3.5 Platform Design in Cloud and Fog Computing

Researchers and practitioners have attempted bringing out the key principles that need to guide the design and implementation of Cloud and Fog Computing. For example, the OpenFog Reference Architecture articulates principles that guide the configuration of storage, network, computation, control and accelerators [7]. The following principles are core to designing the Fog and Grid Computing applications:

- Designs should observe the principles of reliability, availability and serviceability. The quality of Cloud services should be in such a way that it always accessible to the end-users. However, the need for universal accessibility of Fog applications opens up such applications to the numerous security issues and challenges. Real-world Fog Computing applications enable its exposure to the different security dimensions to the security rules.

- Tactical and strategic decision-making provides agile applications that are cardinal in the transformation of data and wisdom. Cloud services should be agile in such a way that they allow changes in configurations and service models whenever there is a justified change in the needs of the customers.
- Fully Cloud-enabled computational systems which observe autonomy at all levels.
- Virtualisation and multi-tenant. End-users should access Cloud services. Simultaneously without going through a reduction in the quality of the services.
- Flexibility and autonomy and observation of the true value of data.
- Specific to deployment needs and observation of all desired values of privacy.
- Because innovations change, and there are new services and applications that emerge everyday, Cloud services should observe a higher degree of scalability.

Understanding Fog Computing goes down to understanding the Fog node architecture. The Fog node is the smallest unit of analysis and its understanding can be used in the understanding of the Fog architecture. The architectural configuration of the server computing node is shown in Fig. 3.2.

Each of the different elements of the Fog node needs to be considered when designing Fog services on the edge network. Some of the Cloud Computing technologies include:

- Hadoop Distributed File System (HDFS), Apache Hadoop [8], Dryad—a distributed execution engine used for coarse grain data applications [9], Amazon EC2, GoGrid [10], and ElasticHosts [11], MapReduce—lightweight runtime programming model [12]. These technologies have unlocked exciting opportunities in parallel computing especially with regard to handling large data sets [13].

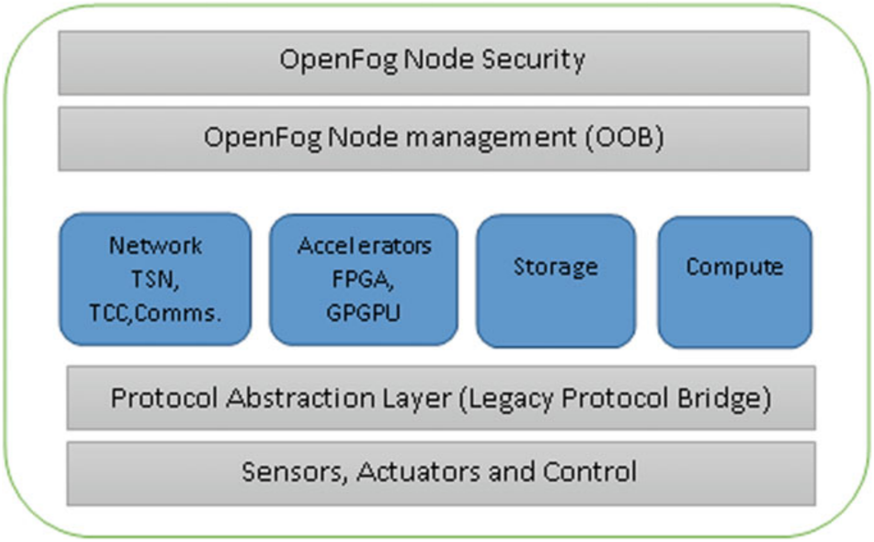

OpenFog Node Security
OpenFog Node management (OOB)
Network TSN, TCC,Comms.
Accelerators FPGA, GPGPU
Storage
Compute
Protocol Abstraction Layer (Legacy Protocol Bridge)
Sensors, Actuators and Control

Fig. 3.2 Architectural arrangement of the Fog node (Adapted from [7])

- Open-source Cloud infrastructure such as Nimbus [14] and Eucalyptus [15], and the open-source virtualization software stacks such as Xen Hypervisor [16], Manjrasoft Aneka provides itself as a platform for creating scalable application in different Clouds in a seamless and elastic manner [13].
- Some of the commercial technology products espoused on Cloud computing include the Cloudlet, Cisco IOx.

The NIST Cloud Computing Reference Architecture is a high-level abstract model focusing on 'what' Cloud services need to provide, disregarding the 'how' to design. It provides a controlled vocabulary of Cloud Computing by articulating the Cloud Computing taxonomy. The architecture is not tied to any specific Cloud Computing or technology product. The conceptual Reference Model is hinged by the Cloud carrier which is further connected to the Cloud provider, Cloud broker, Cloud consumer and the Cloud auditor. Each of these parts of the reference model has certain attributes that need to be considered when design Cloud or Grid applications. The Cloud auditor has the security audit, privacy impact audit and the performance audit. These are different audit systems that focus on ensuring that there is integrity in the over Cloud application provision. The Cloud provider focuses on service orchestration by considering the physical resource later which houses the hardware and facility. The service layer considers the different service models of Cloud Computing such as the Platform as a Service. The service orchestration is generally the resource abstraction and control layer. The Cloud service management part houses the business support, provisioning or configuration and portability or interoperability of platforms for accessing Cloud or Grid Computing applications. The Cloud broker housed the service intermediation, aggregation and arbitrage.

Technologies such as system on chip (SoC) which can embed the whole system on a chip (i.e. CPU, memory, timers, external interfaces) and system in package (SiP) embedding circuits in a single unit and extensively used in smart phones can be used in the implementation of Fog Computing [17].

Web 2.0 applications are used to provide interactive and flexible information sharing, user-centred and collaboration design, and application composition providing enhanced user experiences. This is made possible by the use of technology standards such as XML, Web services, asynchronous JavaScript. In the light of Cloud and Fog Computing, these technologies facilitate the building of applications leveraged on the contribution of content by both users and producers. Users OF Web 2.0 platforms do not necessarily have to install all new software releases but can benefit from these by simply interacting with the Cloud applications made possible by deployment of lightweight programming models, uses loose coupling by integrating and synthesizing existing applications and services, and allowing universal access to the Internet through a dynamic and user-friendly platform.

As a key reference model for Cloud and Fog Computing systems, service-oriented computing allows system and application development to be achieved by adopting 'services' as the main building block. Service-oriented computing is the cornerstone for the development of interoperable, evocable applications and services which are low cost. The key ingredients of a service are location transparency (the actual location of a resource should not be made known to the user or consumer of that resource), loosely coupled, programme language independent and reusable.

Cloud Computing enables us to carefully consider the concept of Web services inferred by metadata standard the Web Services Description Language (WSDL) and the interaction platform the Simple Object Access Protocol (SOAP) allowing different applications to access the Web Service platforms. Interactivity in the Web Service platforms is achieved by the SP.NET and ADO.NET.

Resources managed and offered by/to multiple stakeholders with some form of a metered service at different granularities given a desired SLA conforms to a Cloud. In the EU, the OpenStack and OpenNebula consortia have delved into the development of open-source platform and application development which are cardinal to developing future-fit Cloud and Fog Computing applications [1].

3.6 Issues in Cloud and Fog Computing

In order for the objectives of Fog Computing to be accomplished, there is a need to understand the different issues that come with Fog Computing implementation. Some of these issues refer to the following:

1. **Fog network**: One of the key ingredients for a dynamic Fog Computing implementation is the network itself which is at the centre of connecting all the different heterogeneous components (devices and gadgets) accessing specified resources. Issues such as achieving network scalability, reduction of costs can be managed using emerging techniques such as Software-Defined Networking (SDN) and Network Function Virtualisation (NFV). The SDN allows wired ports on the Fog server to select optimum frequency on the wireless interfaces to support privilege traffic reservation, network virtualisation, etc. The key function of the NFV is to replace the network functions with virtual machine instances. The performance of virtualised network appliances is a key concern that needs urgent attention [4].
2. **Quality of Service (QoS)**: QoS is a very important measure of network service quality in a Fog Computing network. It is measured using connectively, capacity, reliability and delay. QoS is achieved by continuously measuring the four metrics [4].
3. **Interface and programming model**: Because resources in a Fog Computing network are accessed using virtual networks, there is need for robust and dynamic interfacing and programming models. Current thinking points to application-centric computing to have a brighter future in Fog Computing in order to provide scenarios where components in Fog Computing environment will be application-aware allowing different types of optimisations. Currently, it is difficult to build compatible applications which can be accessible over

multiple heterogeneous platforms. Although some authors, such as Hong et al. [18], have developed high-level programming models for emerging internet applications, it is important that more general schemes need to be built.

4. **Computation offloading**: It saves computational effectiveness by extending the battery lifetime or facilitating a distributed shared memory on computational-extensive tasks.
5. **Provisioning and resource management**: When the end node in Fog Computing environment moves, application-aware provisioning is compromised due to dynamism brought into the metrics such as bandwidth, storage and latency. Fog Computing and the IoT have a future in playing critical roles in mobile crowd-sourcing or sensing applications. Resource discovery in highly dynamic information resource environments, and sharing resource discovery thereof, is of critical important in Fog Computing environments. Resource sharing is a key attribute that needs to be carefully considered in heterogeneous resource environments to overcome the different challenges brought about by Fog Computing [4].
6. **Privacy and security**: Although a lot of effort has been dedicated by various researchers on security issues in Cloud computing, there are few interventions and effort dedicated to Fog Computing. Customised systems on intrusion detection systems can be deployed in geo-dispersed Fog Computing environments to detect suspected malicious activities such as denial of service (DoS), port scanning. There is need to explore security issues in both hardware and software layers.

Another key challenge in facilitating Fog Computing is building the necessary level of interoperability to support access of Fog Computing resources. This requires a collaborative approach given the multi-dimensionality of the different entities of Fog Computing. One solution towards achieving the interoperability in Fog Computing environment is the need for an open architecture which will not depend on the make and type of technology innovation accessing the Fog Computing environment. An open infrastructure will significantly reduce the cost to develop Fog applications, will provide a common platform upon which Fog applications will be developed, increase penetration and adoption of Fog Computing and ultimately increase the quality and innovation of Fog Computing [7].

One of the key challenges of Fog Computing has been multi-tenancy—the existing of heterogeneous agents accessing resources in the Fog Computing environment separated by varying levels of security, service-level agreements (SLAs), governance, policies, etc.

In a multi-tenant environment, the issue of security cannot be overemphasised. In order to implement desired levels of security, one of the key requirements is hinged on identify and access management (IAM). IAM involves understanding the different component of IAM as espoused in the ISO/IEC 24760-1 such as Entity (item that exists inside a system e.g. a device, SIM card), Identity (sum of attributes defining an entity), Identifier (unique identity related to an entity), Credential (this is a representation of an entity that is used to achieve authentication of an entity, e.g. password, pin, password, smartcard).

The problems of Fog Computing can be dwarfed by continuously engaging into the issues of efficiency and effectiveness. The key assumption in achieving dynamic security management in loud computing environments is ensuring that there is robust identity management.

3.7 Legal Dimensions of Cloud and Fog Computing

Law and order is very important in ensuring that there is sanity in the Cloud and Fog Computing environments given the proliferation of Cloud service providers in any given setting, especially that a multi-tenant model is utilised. The following are some of the legal dimensions that need to be considered in Cloud and Fog Computing environments:

1. There is a growing wave of competition in the provision of Cloud services prompting some Cloud service providers to delivery implement service applications that lead to consumer lock-in, blockage and dependency. This is a problem brought about by the varying data formats and interfaces upon which Cloud services and resources can be accessed. There is need for legal frameworks to protect the consumer of Cloud services given the environment in which Cloud Computing operates. Further, legal frameworks should encourage the provision of Cloud services on open interfaces so that issues of user lock-in are going to be left in the past [19].
2. There is need for legal provisions where if either a consumer or provider wish to part company with one another, they should do so without penalties, loss of data or other unforeseen repercussions [19].
3. Legal instruments need to be made with regard to ensuring that there are minimum quality levels espoused in the service-level agreements (SLAs) for all Cloud and Fog service providers. This can wipe away many rudimentary and unreliable data service management seen in many of the Cloud or Fog service provides today.
4. Service providers need to be monitored to ensure that the data they may hold in the storage confines of the virtual servers is not used at all costs to the disadvantage of the owner of the data.
5. Other practical issues need to be considered upon the design of Cloud computing legal setups such as physical location of the data, legal status of the Cloud service provider in the area it is providing the service and whether it is authorised to use an infrastructure located outside the country of origin, what will happen to the data upon the expiry of the contract, etc.

It is important to note that all the points given above are hinged on the need to enable interoperability and reversibility in Cloud computing. Because of many Cloud service providers offering all sorts of services, there is a need for Cloud service to interoperate. In any given context, it is important that there is encouragement to adopt the regulatory and technical aspects to Cloud Computing as espoused in the international coordination effort (*see*: www.itu.int/en/ITU-T/focusgroups/Cloud/Pages/default.aspx).

3.8 Realisation of Cloud and Fog Computing in Africa

Although Cloud and Fog Computing implementation are relatively new, African countries are jumping onto the bandwagon in as far as its implementation is concerned. In conjunction with the international players, Africa is implementing various Cloud and Fog Computing projects given the appropriateness of the Cloud Computing model to the African context. However, as aforementioned, technical shortcomings and regulatory inefficiencies associated with advanced technologies slow down the penetration of Cloud Computing. Because of underdeveloped ICT infrastructure, the Internet in many parts of Africa is generally of poor quality of service limiting the likelihood that Cloud Computing can thrive. It is posited that the limited penetration of Cloud Computing in the African polity is a source for concern as Africa misses out on exciting opportunities brought about by the digital age and therefore misses out on information globalisation currently being advocated for throughout the world.

There has been a growing trend in the capacity of African countries to offer reliable Cloud computing services. For example, there are over 112 data centres in Africa with South Africa taking the centre stage with 15 mostly operated by Teraco in Cape Town, Johannesburg and Durban. Having data centres locally in Africa will significantly reduce the cost of Cloud or Fog Computing services thereby allowing more people and organisations to have access to different innovations surrounding Fog/Cloud Computing.

Known as a continent stung by the lack of adequate resources to push the development agendas, Cloud and Fog Computing are good candidates for revamping the ICT experience and implementation in Africa owing to its cost-effectiveness. The cost-effectiveness is achieved by the fact that there is no need for expensive processing and storage systems as these tasks are done by virtual servers in the Cloud, no IT maintenance costs and expensive software, no need for dedicated experienced IT staff to drive technology innovation agenda as innovation can be accessed through shared platforms online, accessed through simple Web-based platforms, etc. Despite this being the case, there is need for adequate levels of Internet connectivity, virtualisation needs to be achieved at a higher level, and there should generally be higher trust in the security of the systems used [19].

In order to effectively tap into the different digital opportunities brought by Cloud and Fog Computing at a continent level, there is a need for Africa to pursue coming up with strategies that are going to be at the core of transition to Cloud or Fog Computing given its unique contextual nuances. This strategy is going to act as a blueprint for integrating and mainstreaming Cloud and Fog Computing in different business processes of both the public and private enterprises. One of the key roadblocks for the penetration of Cloud and Fog Computing in the different socio-economic frameworks of Africa has been the varying legislative and regulatory frameworks. Looking at the EU, there was recognition that the different frameworks at country level hampered the penetration of ICTs into the different socio-economic establishments and therefore moved to come up with harmonised and common ICT strategy for Europe. At the moment, ICT penetration in the EU is comparatively higher than many regions of the world and there is now a serious move to realise the benefits of cross-border ICT integration. Although Africa has autonomous regional groupings, it is encouraged that these need to talk to each other in order to come up with harmonised legal, regulatory and legislative frameworks to promote seamless penetration of emerging technology innovations. Another key requirement to realise the development of Cloud and Fog Computing in Africa is the need for attracting investment towards the development of the ICT infrastructure to support future technology innovations [19].

The emergence of Cloud Computing is going to create nearly 145,000 jobs in South Africa. Sikhosana [20] articulated the key benefits of Cloud Computing especially amongst the African business entities as it culminates into the reduction of the cost in the investment of the ICT infrastructure. Cloud Computing will further promote innovation in different socio-economic establishments by encouraging software development competence [21]. The Cloud readiness of South Africa has attracted many Cloud providers such as Google to the South African technology market [22]. Some of the common Cloud services in South Africa include email hosting and archiving, Web hosting, data backup and configuration, application development.

Because of the relatively advanced economy, South Africa has a comparatively advanced Cloud services infrastructure with many different Cloud services providers. For example, a data centre to support Cloud services was built in Johannesburg by the IBM in 2009. In the same year, Vmware launched vSphere as a Cloud operating system specified oriented to serve the South African market. Another Cloud provider very active on the South African landscape is Dimension Data which offers Cloud business services such as OpSource and *BlueFire to the Cloud consumers in South Africa. Dimension Data has developed capacity to handle any contemporary Cloud service demands given their collective experience with Managed Cloud Platforms (MCPs), CloudControl management system and private service offerings. Xi [23] investigated penetration of Cloud Computing in a provincial government in South Africa and found that Cloud Computing adoption was in its embryonic phase. Further penetration of Cloud services is accentuated by the continued improvement of bandwidth given the implementation of the undersea telecommunication cables. Despite this being the case, there are clear indications that South Africa presents a contextual terrain ready to adopt Cloud Computing. Most of all, the leadership seems ripe as there are a lot of strategic movements towards encouraging Cloud Computing development in South Africa [13].

A recent study done by the international data corporation (IDC) has opined that a majority of the South African companies are considering the implementation of Cloud Computing in one way or the other. Over 93% of the South African companies have or are in the process of developing Cloud Computing strategies. Other countries such as Kenya and Egypt have a relatively developed Cloud and Fog Computing infrastructure. Given the present state of affairs, the single-key limitation of Cloud service penetration in South Africa has been the limited understanding on the part of the individuals and business on what Cloud Computing entails and how to implement it. Strategic initiatives needed for development of Cloud Computing include the need to revise and strengthen most of the current legislation on the management of Cloud-hosted data geared towards preventing data lock-in, clarify the different roles and relations in the Cloud data management environments such as data centre managers, data protection. There is need to also put in place regulatory watch entities, training and awareness programmes, cross-border agencies, etc.

3.9 Conclusion

This chapter has articulated the development projectile of Cloud and Fog Computing. The first parts of the chapter were hinged on articulating the basic and fundamental concepts of Cloud and Fog Computing in the light of resource and information management. It has been posited that the emergence of Cloud and Fog Computing will change the computing paradigm towards pervasive information and networked resource management.

Although there a myriad of benefits that can be accessed by using Cloud and Fog Computing, there are a host of challenges that need to be overcome if Fog Computing were to be adopted on a large scale. This chapter has articulated the different issues and challenges that need to be addressed by yesterday. Currently, there are different Cloud architectures that are being proposed from different contextual settings. The key thinking is that there is need to collaboratively work towards designing open interfaces and systems that can go a long way in facilitating access to Cloud resources by different technology platforms from diverse IT vendors. One of the active players towards this effort is the OpenFog consortium which has proposed an architecture which can be used as reference when designing Fog applications.

Although other regions of the world over have shown increased adoption of Cloud and Fog Computing, it is clear that Africa is lagging behind in this regard. The challenges with regard to the penetration of Cloud and Fog Computing in Africa can be mitigated by considering rigorous capacity building and institutional training, amongst others. A lot of work needs to be done in order to achieve meaningful penetration of Cloud and Fog Computing into the different socio-economic establishments.

References

1. Schubert L, Jeffery K (2012) Advances in clouds expert group report public version 1.0 research in future cloud computing. In: Schubert L, Jeffery K (eds) Research in future cloud computing
2. Ukil A, Jana D, De Sarkar A (2013) A security framework in cloud computing infrastructure. Int J Netw Secur Appl (IJNSA) 5(5). https://doi.org/10.5121/ijnsa.2013.550211
3. Stanton B, Theofanos M, Joshi KP (2015) Framework for cloud usability NIST special publication 500–316. http://dx.doi.org/10.6028/NIST.SP.500-316. Accessed 3 May 2017
4. Yi S, Li C, Li Q (2012) A survey of fog computing: concepts, applications and issues. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.702.7773&rep=rep1&type=pdf. Accessed 26 Oct 2017
5. Badger L, Bohn R, Chu S, Hogan M, Liu F, Kaufmann, et al (2011) US government cloud computing technology roadmap Volume II release 1.0 (Draft). http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-293.pdf. Accessed 11 Nov 2017
6. Aus. Govt (2013) Cloud computing strategic direction paper: opportunities and applicability for use by the Australian Government April 2013. https://books.google.co.za/books?id=6V2inQAACAAJ&dq=CLOUD+COMPUTING+strategic+DIRECTION+PAPER:+Opportunities+and+applicability+for+use+by+the+Australian+Government&hl=en&sa=X&ved=0ahUKEwjq7ebhoLTYAhUjIcAKHU0XBE8Q6AEIJjAA. Accessed 26 Oct 2017
7. Gorlatova M (2017) Reference architecture. http://pages.di.unipi.it/throughtheFog/wp-content/uploads/sites/13/2017/02/gorlatova.pdf. Accessed 3 May 2017
8. ASF core (2009) Apache hadoop core. http://hadoop.apache.org/core. Accessed 26 Oct 2017
9. Budiu Isard M M et al (2007) Dryad: distributed data-parallel programs from sequential building blocks. SIGOPS Oper Syst Rev 41(3):59–72
10. ServePath (2009) Managing storage in the cloud. https://www.snia.org/sites/default/files/GoGrid_Cloud-storage-mgmt-deck_v1b.pdf. Accessed 26 Oct 2017
11. ElasticHosts (2009) How cloud computing should be done. Blog. https://www.elastichosts.com/blog/article-on-the-hot-aisle/. Accessed 3 May 2017
12. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. https://courses.washington.edu/info445/docs/p107-dean.pdf. Accessed 15 July 2017
13. Trope J (2014) Adoption of Cloud computing by South African firms: an institutional theory and diffusion of innovation theory perspective. MCOM (Information Systems) [Unpublished] University of the Witwatersrand
14. Foster Keahey K I et al (2005) Virtual workspaces: Achieving quality of service and quality of life in the Grid. Sci Program 13(4):265–275
15. Nurmi D, Wolski R et al (2009) The eucalyptus open-source cloud-computing system. In: Procceeding of 9th IEEE/ACM international symposium on cluster computing and the grid
16. Barham P, Dragovic B, Hand SK, Harris T, Ho A, Neugebauery R et al (2003) Xen and art of virtualisation. http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf. Accessed 26 Oct 2017
17. Vaquero LM, Rodero-Merino L (2014) Finding your way in the fog: towards a comprehensive definition of fog computing. ACM SIGCOMM Comput Commun Rev 44(5):27
18. Hong K, Lillethun D, Ramachandran U, Ottenwälder B, Koldehofe B (2013) Mobile fog: a programming model for large–scale applications on the internet of things. https://davelillethun.files.wordpress.com/2014/04/mcc03-hong.pdf. Accessed 3 May 2017
19. ITU (2012) Cloud computing in Africa: situation and perspectives. http://www.itu.int/pub/D-PREF-THEM.07-2012. Accessed 15 July 2017
20. Sikhosana BHS (2015) The impact of cloud computing on business operations. MA computer auditing [Unpublished]: University of Johannesburg. Retrieved from: http://ujdigispace.uj.ac.za. Accessed 3 Mar 2018

21. Haig-Smith T, Tanner M (2016) Cloud computing as an enabler of agile global software development. *12*
22. Olumide AA (2014) Assessment of cloud computing readiness of financial institutions in South Africa. MCOM (Information Systems), [Unpublished], University of Cape Town
23. Xi L (2014) Readiness assessment of cloud-computing adoption within a provincial government of South Africa. University of the Western Cape, Masters in Information Management

Chapter 4
Fog Computing in a Developing World Context: Jumping on the Bandwagon

Bvuma Stella and Kelvin Joseph Bwalya

Abstract The emergence of Fog Computing has culminated into a plethora of service options for computing users. Fog computing provides opportunities where a network of remote servers on the Internet can store and process data, rather than a local server or a personal computer as elaborated by the Compaq Computer in 1996. Defining policies for adopting Fog Computing from the perspectives of both the individual users and organisations has become essential for choosing the right Fog Computing services. The aim of this chapter is to articulate the vantage points of Fog Computing in the informational computational continuum and explore the factors affecting the adoption of Fog Computing by users or organisations with a special focus in the developing world. Amongst others, the chapter explores the contemporary privacy and security concerns and other issues of Fog Computing especially from the standpoint of service excellence. Further, the chapter explores contextual challenges that may be encountered when implementing Fog Computing in developing world contexts such as low latency and jitter, context awareness and mobile support due to underdeveloped ICT infrastructures. The chapter integrates the Technology, Organisation and Environment (TOE) framework which can be employed to understand the factors influencing the penetration of Fog Computing in different businesses in the developing world. The chapter also explores opportunities bordering on how Fog Computing can advance the agenda of service ubiquity and pervasiveness.

4.1 Introduction

Due to the importance of diverse emerging technologies in the different socio-economic establishments in the world, technology adoption in different contexts has been a hot research topic amongst researchers and practitioners. Fog

B. Stella (✉) · K. J. Bwalya
School of Computer Intelligence and Information Systems,
University of Johannesburg, Johannesburg, South Africa
e-mail: stellab@uj.ac.za

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_4

Computing is proving to be a paradigm shift in computational sciences and brings about ubiquity in the computation of diverse information resources and application execution raising its popularity in many organisations around the world [1]. Fog Computing brings about new computational capabilities in capturing, storage and processing of large quantities of both structured and unstructured data (Big Data) and enables application execution in remote sites using technology infrastructure belonging to other entities. The advancement of Internet of Things (IoT), Big Data and predictive analytics has made its demand around the world and in the developing world to go over the roof. In a nutshell, the emergence of Fog Computing on the information management landscape unlocks opportunities that were unheard of barely 20 years ago let alone perceived.

Fog or Edge Computing is a new computing paradigm, a concept proposed by research community [2], that has emerged and seen as a disruptive technology specifically on how humans utilise computers in their everyday lives. This paradigm allows the flexibility to compute and store resources at the Edge of the Internet, in close proximity to mobile devices, sensors, end-users and IoT devices [3]. The conceptualisation of Fog Computing is hinged on the understanding of contemporary and emerging information environments such as information ecosystems and the ability of technology devices automatically communicating and exchanging data using the machine-to-machine mode. In this scenario, the physical proximity of the technology devices improves latency, bandwidth, trust and survivability. Organisations aiming to adopt Fog Computing need to have a clear understanding of the factors affecting the adoption of emerging technology (ies). In any given context where an emerging technology is here marked for adoption, challenges and factors need to be outlined which in turn will assist the policy makers or decision makers in developing clear understanding of the potential benefits, as well as the challenges involved in the process of planning and implementation of the adoption.

Given the aforementioned, this chapter aims to explore the conceptual characteristics of Fog Computing within the realm of the information computational continuum and then explore the factors at the centre of adoption of Fog Computing analysed with the lens of a developing world context. The chapter explores how Fog Computing impacts on the informational computational continuum given the different computing paradigms utilised in the management of heterogeneous information resources. Out of these endeavours, the chapter proposes a conceptual framework that may guide investigations of factors influencing Fog Computing in developing world contexts.

This chapter is arranged as follows: the next section introduces the general characteristics of Fog Computing and articulates the general functional and non-functional characteristics of any Fog Computing environment, and the following main section explores the different factors influencing the adoption of Fog Computing. Section 4.4 explores the general factors influencing technology adoption and compares these with the factors influencing adoption of Fog Computing, thereby proposing a conceptual framework for Fog Computing adoption. Sections 4.5 and 4.6 explore and discuss technology adoption theories; and

Fog Computing measurement constructs in terms of organisational, technological and environmental factors. The chapter ends with a conclusion which gives a recap of major themes discussed and outlines the future works that need to be further explored to advance the vision of Fog Computing.

4.2 Fog Computing

Before the articulation of Fog Computing, it is important to understand its root by exploring the fundamental characteristics of Cloud Computing especially with the focus whence Computing emanates. Cloud Computing allows remote application or program execution, information storage and processing owing to the scarcity including higher cost of requisite and compete IT infrastructure which ensures that the cost for business process automation and digitisation is significantly reduced. The Cloud Computing paradigm has a couple of service models [4] which include software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). SaaS provides users with the capability to utilise software applications running on the providers' Cloud infrastructure and can be accessed using any device with a Web browser. The IaaS offers the fundamental computing resources as a capability, and IaaS allows the capability for users to manage processing power, storage, networks and other fundamental computing resources and forms the base on which PaaS and SaaS are supported. PaaS provides a development environment with capability for users to create and run their own custom-made applications using programming languages and tools supported by the Cloud service provider. While there are many concerns with regard to the safety of utilising the Fog Computing services, the main concerns are confidentiality and security, the available data and its integrity and authentication of available information. To date, majority of people are utilising multiple devices on a daily basis, with personal Cloud Computing as a major factor in storing data from multiple devices to the same location. With so many devices connected to the Internet, it is more necessary than ever to be able to access your files from every one of those devices: your phone to your laptop, your iPad to your smart TV, etc. An easily accessible, user-friendly personal Fog storage system that does not compromise on security will benefit and boost the user's confidence. Clients using either personal or public Cloud services may suffer anxiety due to a lack of control over such things as availability, portability, isolation, integrity, transparency and the confidentiality of their data. These issues need to be addressed if anticipated benefits of Fog Computing are to be harnessed and for universal adoption to be achieved.

There are many benefits associated with the use or adoption of Fog Computing. These benefits outweigh those provided by Cloud Computing and usher in a computing environment that has the capacity to reduce or remove cost strains and increase the flexibility and scalability for innovative computing operations. However, Fog Computing topology differs from that of Cloud Computing in that Cloud Computing allows the storage of data or content in a data centre including the offsite or Cloud environment, whereas Fog Computing points of contact are located on the edge of a network. In this regard, the use of the word "Fog" entails that the points of contact or data are spread out amongst their users. Simply put, instead of a remote site where computing activities are executed, a Fog Computing environment entails that the different devices in the vicinity of a computing device are able to exchange information and act of computing nodes.

The Cloud data centre is the offshore technology configuration that provides several databases that are used for massive parallel data storage and processing, and as a console for offshore programming execution. This enables Big Data mining, access and processing which is at the centre of contemporary dynamic and real-time information processing and management. The many databases in this centre enable machine learning and as an overall platform provide dynamic information management in the Fog environment. The Fog node provides an overall Internet gateway connecting the Cloud data centre to the communication highway to the different interaction platforms in the Fog environment. The Fog node allows real-time processing of the data, data caching and overall computation offloading. Figure 4.1 provides a schematic representation of how Fog Computing works.

Fog Computing allows information processing to happen in a data hub on a device on the edge of the network. For example, this processing can happen in a smart router or gateway device. The Fogging process has a greater advantage because it can generate huge amounts of data for processing and analysis. A great

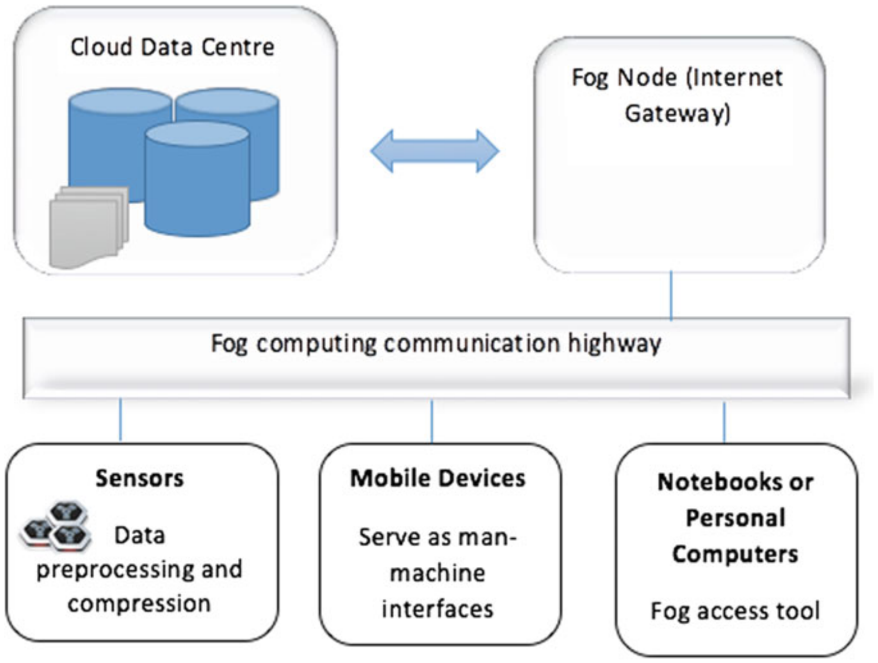

Cloud Data Centre
Fog Node (Internet Gateway)
Fog computing communication highway
Sensors
Data preprocessing and compression
Mobile Devices
Serve as man-machine interfaces
Notebooks or Personal Computers
Fog access tool

Fig. 4.1 How Fog Computing works (*Adapted from* [38])

deal of bandwidth is needed and the back-and-forth communication between the sensors and the Cloud can negatively impact the performance. The latency issue can result in negative impact in some cases such as gaming, but delays in data transmission might become life-threatening in case of vehicle-to-vehicle communication system or large-scale distributed control system for rail travel.

In summary, the special feature in Fog Computing is that it allows the moving of data interface to the close proximity of the user. The data is not forwarded in several network hops, to the central Cloud located on the other side of the globe, but in just one hop, directly to the closest network component. The component is in charge of processing and assessing the data, answering to it, if possible, or getting back to the Cloud. Fog Computing brings in the advantage of data on the edge near [4] the device which allows the data to be analysed at close proximity, produce data and act on the data. A differentiate factor with the Cloud Computing architecture is that it cannot conserve network bandwidth when moving all data to the edge of the data centre for processing and results in latency. Because the Cloud servers lack the ability to communicate with many protocols used by IoT devices, the regulations and privacy concerns do not allow offsite storage of other data types.

Although Cloud Computing is still relevant when it comes to IoT applications, it is quickly becoming obsolete. There is now a realisation that perhaps Fog Computing will soon take over and Cloud Computing pushed to the side lines with Fog Computing handling all the critical work. The current growth rate of the Internet of Things emphasises the need for special infrastructure base that can handle all its requirements, and at the present moment, Fog Computing seems to be the most viable option available.

With regard to many of the developing world countries, Cloud and Fog Computing paradigms present themselves as viable options given their lower cost. It cannot be overemphasised that many of the developing countries are faced with various challenges such as socio-economic and political problems that limit their ability to invest in an expensive information and communication technology (ICT) infrastructures. These various challenges which in turn cannot rule out the utilisation or adoption of technology are to be explored and managed effectively in order to address them. Fog Computing is one of the new technology disruptors that has emerged and made its entrances recently. Developing countries are to benefit from this new emerging technology when they adopt or utilise it. However, it is imperative that when adopting Fog Computing or any technology for that matter, an in-depth understanding of benefits and challenges faced must be deeply understood. Other than the aforementioned characteristics, the following section presents some of the characteristics of Fog Computing.

4.3 Characteristics of Fog Computing

4.3.1 Improved Quality of Service

The advantage of Fog Computing is the amount of time in providing response from the data centre. When the response is quicker in particular with devices that has 5G capability, the advantage that Fogging provides is the lower latency and jitter. Fog Computing can therefore provide better or improved results, increase service levels and safety in different environments such as transportation, mining and the public sectors. In developing countries such as South Africa (SA), the government has pronounced interest in providing accountable and transparent public service delivery where the utilisation of ICTs is highly encouraged. In the contemporary technology landscape, the use of emerging IT platforms and solutions such as IoT, Cloud or Fog Computing may hugely benefit the governance constituencies.

Organisations focusing and intending on reliable business agility can benefit with Fog Computing by managing the organisational applications effectively because Fog application program allows the device to function according to the user's needs. Fog Computing provides improved awareness and response to events by eliminating a round trip to the Cloud for analysis. It avoids the need for costly bandwidth additions by offloading gigabytes of network traffic from the core network. It also protects sensitive IoT data by analysing it inside company walls. Ultimately, organisations that adopt Fog Computing gain deeper and faster insights, leading to increased business agility, higher service levels and improved safety.

4.3.2 Reduction in Latency

The world Internet usage and population statistics tabled the current Internet usage at over 3 billion, and this is seen to be increasing. Meanwhile the report compiled by the ITU [5] indicates the increasing number from 738 million in 2000 to 3.2 billion in 2015. The increase of Internet users does not allow the sustainability to continue relying on Cloud Computing as a centralised server, because Cloud Computing allows access to online data and thus enabling users to access, share and store data in remote servers. The challenge of a centralised Cloud server cannot be dependable any longer, because they are commonly located not closer to the users which leads to high access latencies, hence the suggested option is to move towards a decentralised environment such as Fog Computing which is sometimes referred to as Edge Computing. Cloud Computing gained its popularity with regard to data processing and effective high computation power ability [6]. The challenges are experienced when there is a bottleneck of huge data, due to lack of bandwidth increase thus resulting in latency. One of the challenges with Cloud Computing is the cost to transfer data from the Cloud, such as downloading large amounts of data and storage costs, for example moving data takes time and is costly and poor unavailable capacity. The challenge of latency has been proven by Sarkar and Misra [7] where they indicated a system that resulted in Fog Computing being lower than the Cloud Computing using a theoretical model of Fog Computing architecture.

Fog Computing allows the decentralisation of servers, storage, services and application to increase the reliable service and better experience. The platforms such as the IoT or devices operating on a 5G model application which are commonly utilised on the Edge device may have devastating disadvantages.

Fog Computing has the capability to service users to proximity. They also have the capability to execute computational task and allow the process of communicating and storing users on the edge of the network. Fog Computing enables low latency and effectively allows the real-time interactions especially for latency-sensitive or time-sensitive applications [2]. According to Hu [8], when the Fog Computing model is utilised into the identification users and resolution field, its result is that the response time is less than the Cloud Computing response time.

4.3.3 *Support for Mobility*

With Fog Computing, multiple users, devices (vehicles, wearable gadgets, sensors, smart devices, etc.) and organisation can cooperate with one another using their own Fog infrastructures. The current paradigm of Big Data generated daily and the strong entrance of IoT are processed by devices such as mobile devices transmitted at the edge of the network or near terminal devices and allow local data to be accessible. Moreover, various mobile devices can also communicate directly to each other. The data does not have to be transmitted to the Cloud or even the base station.

The accessing of mobile devices is fundamentally on physical proximity by some communication technologies, such as Bluetooth, Near Field Communication (NFC) or Millimetre Wave communication. This way they can avoid the intermittent network connectivity caused by mobility [1]. Furthermore, by using the idea of "Data Sherpa", the data from a static sensor (Edge) can be transferred to a mobile smart phone (Edge) with Bluetooth technology when they are proximity, and then smart phone transmits the data to Fog or Cloud [9, 10]. Fog Computing architecture supports location-based mobility demands and enables administrators to control where users and mobile devices are coming in and how they access the information [11]. This improves the performance of system and quality of service.

4.3.4 *Enhanced Heterogeneity*

Fog nodes are highly dynamic and heterogeneous at different levels of networks hierarchy for low latency and scalability requirements. The Fog Computing environment is deployed in many different forms that can be virtual or physical in nature

[12]. The Fog Computing platforms such as routers on the Edge, access points and gateways have computational and storage capabilities that can run on multiple OS and process data application in different location. Azaan and Huh [13] agree that because Fog Computing is highly virtualised in nature "virtual computing nodes and virtual network nodes", these nodes are seen to be used as Fog nodes therefore becoming heterogeneous. Moreover, the network infrastructure of Fog Computing is also heterogeneous, which includes not only high-speed links connecting to data centre, but also wireless access technologies (e.g. WLAN, Wi-Fi, 3G, 4G, ZigBee) connecting to the Edge devices [14]. The network platform of Fog Computing is multi-tiered hierarchical architecture from the edge to the core where various IoT applications (including intelligent transportation, smart cars, smart cards), the resources and service interfaces of Fog nodes are highly dynamic and heterogeneous at different levels of architecture hierarchy to address the requirements of widely distributed applications that need low latency [15].

4.4 Factors Affecting the Adoption of Fog Computing

4.4.1 Security of the Environment

Popovic and Hocenski [16] agree that Fog Computing presents critical research areas for research. Users and organisations have similar concerns with regard to the security equations around the utilisation of Fog Computing. According to Hugos and Hulitzky [17], one of the concerns taking priority when considering the adoption of technologies such as Cloud Computing is security concerns. The concerns are typically the question around the originality and authenticity of the data, data transmission, data storage or third parties' access to an organisation's data. The risk associated with security concerns such as the authorisation and authenticity of data from each node connected over the network, because Fog Computing is an open network, there can be many users' organisations, and users should easily identify others connected or nodes that are connected should easily identify each other (the process of knowing "who is who" while connected). The focus should not undermine the planning and designing of how data centres are structured and how authorisation techniques in a multiple Fogging environment are implemented. Indeed, the planning and designing of Fog Computing in organisations are necessary to both their strategic feasibility and analysis of all possible benefits such as operational and vulnerabilities' factors.

The risk associated with the adoption of Fog Computing should be addressed, and guidelines should be clear when formulating policies around the adoption of Fog Computing. Linthicum [18] has pointed that many organisations are reluctant in adopting technologies because of the security concerns. Organisations concerned with regard to security, reliability are mainly due to unsecure connectivity in a networking environment. The connectivity of malicious applications that threaten the connected nodes can affect the downtime which organisations cannot afford to experience. Organisations are advised to enforce security measures when adopting the Fog Computing network. The surety measures are necessary especially noting that in Fog Computing you have multiple users connected at the edge of the network and there may be multiple distributions of resources.

4.4.2 Connectivity with Respect to Access to Data

In order for organisations to have an effective Fog Computing environment, the availability and access to data should avoid Internet interruptions or poor connectivity. Fog Computing is not excluded from the disadvantages of technology interruptions experienced by the traditional IT environment, indeed even in the traditional IT environment, when there are technological interruptions the users are affected negatively and in many occasions productivity levels drop. The challenges of technology interruptions are not new especially where two or more nodes share or access information or the networking environment. Fog Computing can be a beneficiary of these challenges and lead to the vulnerability of attacks such as denial of service (DoS) which is the process of an unavailable or interrupted server by its users.

Many organisations have suggested the solution of allocating various resources to the users of DoS attack with the intention to prevent users from experiencing interruptions by the attacks. Moreover, organisations are concerned with cost reduction particularly on the hardware; however, it must be noted that saving or reducing cost on hardware does not mean saving or reducing cost with bandwidth. In fact, organisations may find themselves spending more money on bandwidth because working with intense and complex data (Big Data) may require adequate and sufficient bandwidth.

Poor access to connectivity is not new to SA; this is because certain geographical areas have better coverage, while others are struggling, left behind in the “digital divide”.

4.4.3 Governance and Support

The importance of government support when adopting any technology cannot be emphasised enough. If the government including the policy developers do not have proper structures and planning in place when adopting Fog Computing, it can attribute to undesirable outcomes. Lam [19] indicates that there is a need to understand the reasons why there is usually resistance by users and the employees when new technologies are introduced in any organisation. One of the most likely general cause of hesitation is the desire to avoid the changing world modules and moving away from the comfort zones by employees. Therefore, in order to avert such scenarios, it is important for users to receive support from the start and during the implementation of any new technology [20, 21].

Generally, the challenges that the South African government is facing with regard to the digital dream include limited robust Internet infrastructure, limited ICT skills and high cost of access and/or low connectivity. The results from the ITU [22] research have tabled the factors showing that Africa is lagging behind with regard to Cloud Computing skills. Government and top management in organisations have an obligation to provide in-depth knowledge of Fog Computing, and they should deepen the understanding of legal issues, strategic alignment and factors that affect the adoption of Fog Computing in order to be more flexible when confronting new governance challenges in the Cloud Computing environment [22].

Users that have subscribed to mobile devices are deepening their cries when it comes to cost, bandwidth size, reliable connectivity and the security concerns with regard to adoption of Fog Computing. According to Kannabiran [23], lack of ICT infrastructure, poor communication infrastructure and ICT skills affects the adoption of technology.

In developing countries where ICT infrastructure is currently the main challenge, particularly because there are rural and urban locations such as in SA, the connectivity challenge is at the disposal which translates into certain regions not to fully benefit from the emerging technologies such as Fog Computing. For example, the banking industry may not operate in full functionality in rural areas where ICT infrastructure is still a nightmare to many users. In order to understand the factors that influence technology adoption at the individual level, it is important to explore the technology adoption theories and models and understand how each of the constructs influence technology adoption.

4.5 Technology Adoption Theories

There are various technology acceptance models used to explain individuals' adopting of technologies when they are introduced for the first time. Fog Computing paradigm is relatively new, and therefore, it can be posited that its adoption needs to be understood and relatively planned in detail before the actual deployment process. It can further be posited that understanding the factors influencing individual adoption of Fog Computing has not been researched to any appreciable extent in any context.

While there are many theories available with regard to technology adoption, there are others gaining popularity such as the technology acceptances models (TAM) developed by [24]. TAM has been reviewed by researchers as the most popular model that predicts the use and acceptance of technology systems by individual users; TAM uses two main variables which are the perceived ease of use and perceived usefulness, and these variables simply present the thought that when users are to use new technology they are usually influenced by how and when they will adopt or utilise the technology [24]. For example, Davis [24] defines the perceived usefulness as “the degree to which a person believes that using a particular system would enhance his or her job performance”. Users may adopt or utilise technologies such as Fog Computing or Cloud when they perceive them as useful and add value to them, the same can be said with organisations; they can only adopt if the new technologies are to add value to their organisations and even allow them to remain competitive. If the technology does not add any value to individuals or organisation, the level of adoption may be poor or lacking thereof. Meanwhile with perceived ease of use, this variable defines the perception of the user in terms of its complexity and usability; users may perceive that if the technology is effortless to use, they may adopt such technology. Figure 4.2 shows a schematic representation of the different constructs used as individual technology adoption entities in the TAM.

Other popular theories such as Unified Theory of Acceptance and Use of Technology (UTAUT) [25], diffusion of innovation (DOI) Rodger’s theory and the TOE framework [26] have been used extensively in measuring technology adoption in different contextual settings [27]. DOI and TOE are the only ones that view technology adoption at the organisational level. The TOE framework was found to be consistent with the DOI theory, in which Rogers [26] identified the internal and external characteristics of organisation and the individual characteristics of employees as drivers for organisational innovation.

Tornatzky and Fleischer [27] developed the TOE framework by identifying three context groups: technological, organisational and environmental. The technological context looks at the internal and external effects of a technology on the business. Organisational context looks at several attributes regarding the organisation that could affect the technology adoption. The Environmental context looks at the organisations sector, competition and any other external factors.

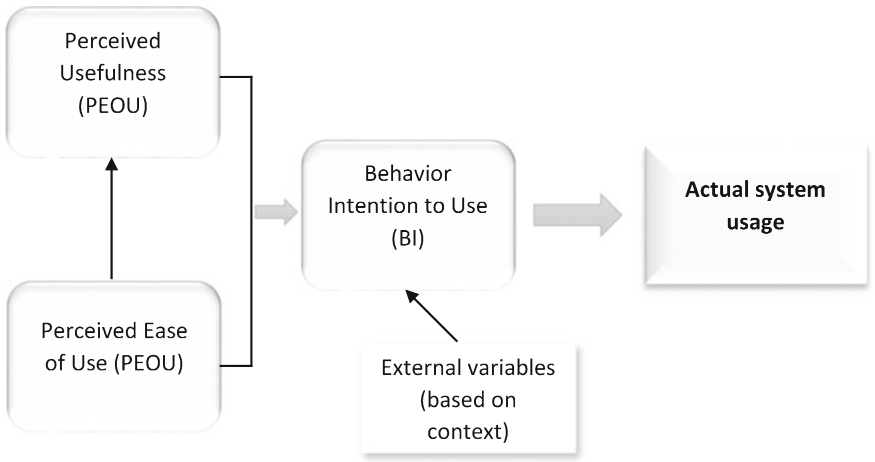

Perceived Usefulness (PEOU)
Perceived Ease of Use (PEOU)
Behavior Intention to Use (BI)
Actual system usage
External variables (based on context)

Fig. 4.2 Technology acceptance model (*Adapted from* [24])

The TOE framework looks at the same features in its technology and organisation contexts, but it also includes the environment context, which makes it the more appropriate choice to explain innovation adoption in an organisation [28]. Conceptualised upon Tornatzky and Fleischer [27] and Low et al. [29] TOE framework, this chapter conceptualises the conceptual usage of the TOE in Fog Computing environments. It is worth noting that the Fog Computing environment includes the need for a requisite legal and regulatory framework that can pave way for both the public and private sectors to engage in Fog Computing applications. The TOE framework in Fog Computing environments is shown in Fig. 4.3.

The TOE framework model is used to explore the technological, organisational and environmental factors that could influence the Fog adoption. Relative advantage, redundancy and performance are identified as the key technological factors, and complexity, compatibility and security are found to be the key technological challenges faced by the organisations. Management support, the size of the organisation and technological readiness are recognised as important organisational factors while competitive and trading partner pressure are identified as key environmental factors. The factors and challenges identified could be valuable for the planning stage of the adoption process and could provide guidance for organisations that are potential Fog Computing adopters.

Figure 4.4 articulates the proposed conceptual framework that can be utilised in measuring the degree of penetration of Cloud Computing at the organisational level. This framework is based on the TOE and can be used especially in public

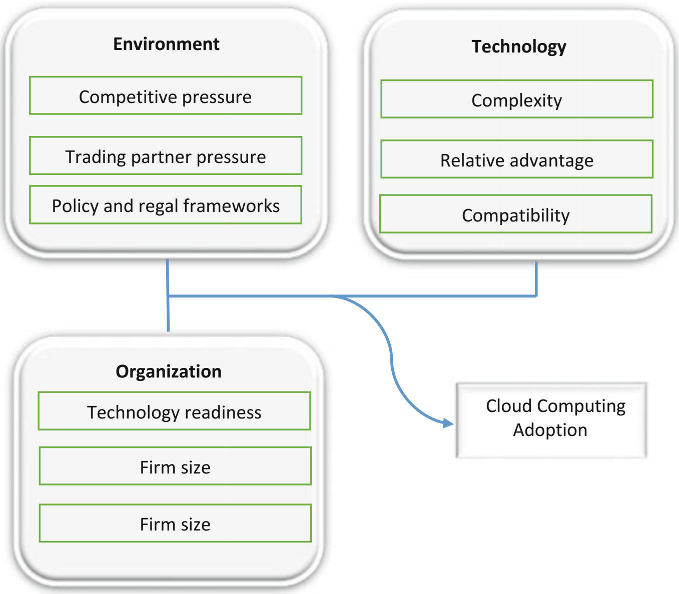


Fig. 4.3 Technology, organisation and environment framework for Fog Computing

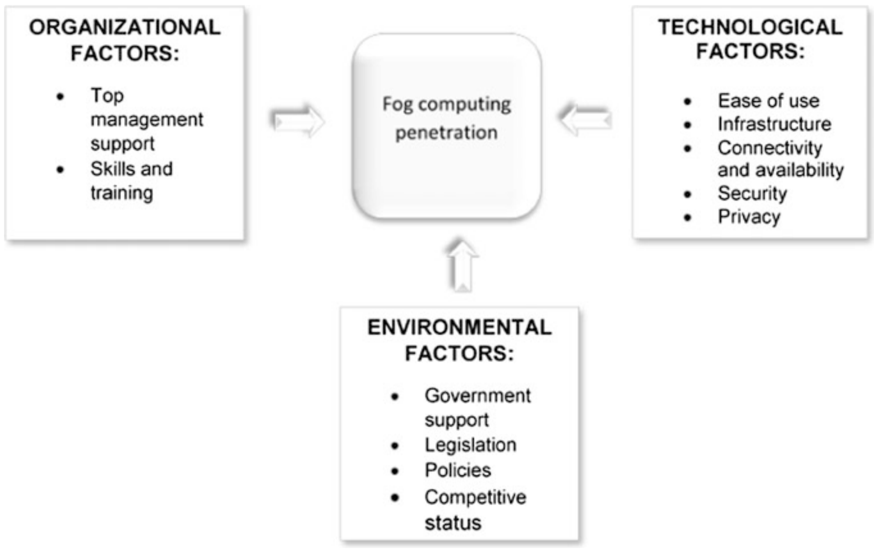


Fig. 4.4 Conceptual framework Fog Computing adoption model

sector organisations. The framework only presents the constructs that can be modified given the context in which it is used. The following section articulates each of the factors presented in the framework in Fig. 4.4.

4.6 Fog Computing Measurement Constructs

In understanding the different constructs at the centre of Fog Computing adoption, it is important to consider each of the different constructs espoused in the TOE framework as shown in Fig. 4.4.

4.6.1 Organisational Factors

4.6.1.1 Top Management Support

The organisational factor includes the planning and understanding of the adoption of technology. Top management and all decision makers must be involved when planning the adoption of the new technology. There is need for a deeper understanding of what is involed in the design and deployment of Cloud/Fog Computing before moving into the next phase of implementation. Top management must ensure that all factors that will affect the deployment of the technology are understood by all parties involved; this include the users of the technology in order to increase performance and provide productivity. This is a crucial phase because if there is level of uncertainties with the adoption of the technology, it can affect the organisation negatively [30].

Wang [31] agrees that the top management support plays a significant role when intending to adopt new innovative technology. He further alludes that it is critical to have top management support when deploying innovative, reengineering and change process.

4.6.1.2 Skills and Training

The skills required for the implementation and maintained of the Fog Computing environment must be in place. In order to achieve the full potential of any technology such as Fog Computing, it is imperative that correct skills and the employees with adequate skills are deployed to manage the environment. The intention is to place the right people at the right place. Unskilled and lack of readiness of users within an organisation may harbour negative results when implementing Fog Computing. Organisations must ensure that there is technology readiness of all members involved to ensure successful implementation.

4.6.2 Technological Factors

4.6.2.1 Reduced Complexity and Ease of Use

Technological factors play an important role in adopting Fog Computing, for example when users are to adopt any technology, there are several key factors to consider such as their perceived trust towards the technology. The complexity of the technology may reflect negatively from the user's perspective when considering utilisation of the technology. Other challenges to consider are the connectivity, data security and privacy of the technology. Poor technology infrastructure also hinders the adoption of technology.

4.6.2.2 *Infrastructure, Connectivity and Availability*

It is imperative to manage the challenges of infrastructure and connectivity. Infrastructure challenges are usually associated with cost factors which have massive influence when considering technology adoption. Sarkar and Misra [32] in their study presented evidence that the service latency for a system associated with Fog Computing was significantly lower than that with Cloud Computing by theoretical modelling of the Fog Computing architecture. Fog Computing operates on the platform that is decentralised, thus the importance of availability of data on the edge is influential factor when coming to adoption of Fog Computing. In Fog Computing, there is a need to be in good operation and effective functionality in order to provide end-users with reliable and accurate available data. Fog Computing environment must be reliable and provide instant response without interruptions thus ensuring that end-users are recipients of high-quality service, high transmission rate and recovery.

4.6.2.3 Security and Privacy

Organisations have the responsibility to analyse and understand the adoption of Fog Computing, provide strategies to address the security concerns such as privacy and protection of data, backup and recovery plans of data. The platform in which Fog Computing operates is on the edge meaning at close proximity with the end-user. Devices on the Fog Computing platform are not guaranteed privacy and security. Hu et al. [33] and Qui et al. [34] have found a common ground that they are vulnerable to encounter malicious attacks. However, encryption and decryption can be deployed to attend to the unauthorised access or attacks. Lee [35] alluded to the fact that the solution to attacks can be resolved when implementing lightweight encryption algorithms or masking techniques. Various concerns with regard to security and privacy are presented as a result of a multi-level collaboration on multiple devices connected over the network; these concerns according to Dsouza et al. [36] can be addressed using the proposal that state the importance of a policy-based resource management and access control in Fog ecosystem. This policy-based resource management support collaboration that is secured and interoperability between heterogeneous user-requested resources.

4.6.3 Environmental Factors

4.6.3.1 Government Legislation and Policies

According to Bernsmed [37], the legislation and policy concerns are the obstacles that affect the adoption of technology in the health services. Government should provide clear legislation and policies that do not hinder the adoption of technologies. These policies have an impact on implementation, security and privacy. Government needs to provide clear guidelines with regard to compliance with regulations, for example if the government does not have proper guidelines on compliance and regulations, then organisations will be reluctant to adopt technologies with the concern of breaching of data, access and sharing data.

4.6.3.2 Competitive Pressures

Organisations are now competing globally and focusing on how to remain competitive; it has become obvious that for organisation to remain competitive it is critical for them to adopt the "right" technologies. There is huge amount of pressure for the world to compete globally, and these competitive advantages are facilitated by deploying innovative technologies. Government should provide leadership that plans to achieve the benefit of gaining a competitive advantage and operate within the competitive pressures or defend them against the competitive threats. This includes having the ability to gather information about the competitive and technological environment in order to succeed, the possibilities of surviving the competitive pressures can be gained by flexible legislations and policies that do not hinder technology adoption.

4.6.3.3 Location

The location or geographical location is a factor when considering the adoption of technology such as Fog Computing. Government institutions such as in SA are faced with challenges of crafting policies and regulations that can protect the data shared on the edge on an international level. The challenge with not having international policies and regulations is that it creates an opportunity to poor privacy and security issues.

4.7 Conclusions

The emergency of Fog Computing has revitalised computing towards a more ubiquitous mode where information can be anywhere anytime and computer applications or solutions executed automatically and pervasively. This has metamorphosed user experience in the information computing environments. This chapter has articulated fundamental concepts in Fog Computing and briefly discussed some of the key factors that may influence individual's and organisation's adoption of the Fog Computing paradigm. The chapter has posited that there are acute issues that need to be overcome if the benefits promised by Fog Computing were to be amassed.

Cloud Computing has presented its own challenges such as bandwidth resulting from sending all its data over a Cloud channel. The other characteristics shown from the Cloud Computing environment is the inability to provide quicker response time and the scalability challenge that results from server dependence located in remote place. However, in the Fog Computing environment, data is presented in a decentralised format and operates on the edge of the network thus consuming less time. Fog Computing produces less demand of bandwidth because all its data are aggregated at specific access points rather than sending them over the Cloud platforms.

The entrance of Fog Computing does not overrule the challenges and security concerns. Noting that in Fog Computing the application to process data to and from different or multiple nodes introduces issues of data privacy its visibility for the third parties. However, it is worth noting that unlike in Cloud Computing, Fog Computing has much more improved security because data in encrypted form easily travels to and from the network core. The process is managed when the data is processed through firewalls and other security points which in turn will allow early detection of malicious programs.

References

1. Vaquero LM, Rodero-Merino L (2014) Finding your way in the Fog: towards a comprehensive definition of Fog Computing. SIGCOMM Comput Commun Rev 44(5):27–32. https://doi.org/10.1145/2677046.2677052. Accessed 26 Feb 2018
2. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog Computing and its role in the internet of things. In: Proceedings of the 1st edition of the MCC workshop on mobile cloud computing, 13–16. https://doi.org/10.1145/2342509.2342513
3. Yi S, Li C, Li Q (2015) A survey of Fog Computing: concepts, applications and issues. In: Proceedings of the 2015 workshop on mobile big data, ACM, 37–42
4. Kavis M (2014) Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS). John Wiley & Sons, Hoboken. https://doi.org/10.1002/9781118691779
5. ITU (2015) ICT Facts and Figures ITU. http://www.itu.int/en/ITU-D/statistics. Accessed 19 Feb 2018
6. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Future Gener Comput Syst 29(1):84106
7. Sarkar S, Misra S (2016) Theoretical modelling of Fog Computing: a green computing paradigm to support iot applications. IET Netw 5(2):23–29
8. Hu P, Ning H, Qiu T, Song H, Wang Y, Yao X (2017) Security and privacy preservation scheme of face identification and resolution framework using Fog Computing in internet of things. IEEE Internet Things J 4(5):1143–1155
9. Varshney P, Simmhan Y (2017) Demystifying Fog Computing: characterizing architectures. Applications and Abstractions, arXiv 1702(06331):1–23
10. Shi C, Lakafosis V, Ammar MH, Zegura EW (2012) Serendipity: enabling remote computing among intermittently connected mobile devices. In: ACM MOBIHOC, 145–154
11. Hassan MA, Xiao M, Wei Q, Chen S (2015) Help your mobile applications with Fog Computing. In: Proceedings of the IEEE international conference on sensing, communication, and networking—workshops, 1–6
12. Kang K, Wang C, Luo T (2016) Fog Computing for vehicular ad-hoc networks: paradigms, scenarios, and issues. J China Univ Posts and Telecommun 23(2):56–96
13. Aazam M, Huh EN (2016) Fog Computing: the Cloud-iot/ioe middleware paradigm. IEEE Potentials 35(3):40–44
14. Bonomi F, Milito R, Natarajan P, Zhu J (2014) Fog Computing: a platform for internet of things and analytics. In: Bessiss N, Dobre C (eds) Big data and internet of things: a roadmap for smart environments. Studies in Computational Intelligence, vol 546, pp 169–186. https://doi.org/10.1007/978-3-319-05029-4_7
15. Hong K, Lillethun D, Ramachandran U, Ottenwälder B, Koldehofe B (2013) Mobile Fog. In: Proceedings of the second ACM SIGCOMM workshop on mobile cloud computing—MCC, 13–15. https://doi.org/10.1145/2491266.2491270
16. Popović K, Hocenski Z (2015) Cloud computing security issues and challenges, 344–349

17. Hugos MH, Hulitzky D (2011) Business in the cloud: what every business needs to know about cloud computing. John Wiley & Sons, Hoboken
18. Linthicum D (2013) Cloud adoption's tipping point has arrived. http://www.infoworld.com/d/Cloudcomputing/Cloud-adoptions-tipping-point-has-arrived-221335
19. Lam R (2011) Organisational readiness and change management in the cloud age. Cloud computing: principles and paradigms, 549–572. http://pfakhri20.persiangig.com/document/mabahesvije/para22.pdf. Accessed 25 Feb 2018
20. Dhiman A (2010) Analysis of on-premise to cloud computing migration strategies for enterprises. Massachusetts Institute of Technology. http://dspace.mit.edu/handle/1721.1/70796. Accessed 26 Feb 2018
21. Swink M (2000) Technological innovativeness as a moderator of new product design integration and top management support
22. ITU (2012) Cloud computing in africa: situation and perspectives. ITU
23. Kannabiran G (2012) Enablers and inhibitors of advanced information technologies adoption by SMEs. J Enterp Inf Manage 25(2):186–209. https://doi.org/10.1108/17410391211204419
24. Davis FD, Bargozzi RP, Warshaw PR (1989) User acceptance of computer technology: a comparison of two theoretical models. Manage Sci 35:982–1003
25. Venkatesh V, Morris M, Davis G, Davis F (2003) User acceptance of information technology: toward a unified view. MIS Quart 27(3):425–478
26. Rogers EM (1995) Diffusion of innovations, 5th edn. The Free Press, New York
27. Tornatzky LG, Fleischer M, Chakrabarti AK (1990) Processes of technological innovation. MA Books, Lexington
28. Oliveira T, Martins MF (2011) Literature review of information technology adoption models at firm level. Electron J Inf Sys Eval 14(1):110–121
29. Low C, Chen Y, Wu M (2011) Understanding the determinants of cloud computing adoption. Ind Manage Data Syst 111(7):1006–1023
30. Thamhain Hans J (2005) Management of technology: managing effectively in technology—intensive organizations. John Wiley & Sons, Hoboken
31. Wang YM, Wang YS, Yang YF (2010) Understanding the determinants of RFID adoption in the manufacturing industry. Technol Forecast Soc Chang 77(5):803–815
32. Sarkar S, Misra S (2016) Theoretical modelling of Fog Computing: a green computing paradigm to support iot applications. IET Netw 5(2):23–29
33. Hu V et al (2014) Guide to attribute based access control (ABAC) definition and considerations, NIST
34. Qiu T, Zhao A, Xia F, Si, Wu DO (2017) Rose: robustness strategy for scale-free wireless sensor networks. IEEE/ACM Trans Netw, pp (99), 1–16. https://doi.org/10.1109/tnet.2017.2713530
35. Lee K, Kim D, Ha D, Rajput U (2015) On security and privacy issues of Fog Computing supported internet of things environment. In: Proceedings of the international conference on the network of the future, pp 1–3
36. Dsouza C, Ahn, GJ, Taguinod M (2015) Policy-driven security management for Fog Computing: preliminary framework and a case study. In: Proceedings of the IEEE international conference on information reuse and integration, pp 16–23
37. Bernsmed K, Rubsamen T, Onen M, Sudhol M (2014) Healthcare services in the cloud—obstacles to adoption, and a way forward. In: Proceedings of the 9th international conference on availability, reliability and security
38. Zao JK, Nguyen KT, Wang YH, Lin ACH, Wang BW, Liu JWS (2014) Trustworthy emergency information brokerage service (TIBS). WIT Trans Built Env 133:216–227. https://doi.org/10.2495/DMAN130221

Part II
Frameworks and Technologies

Chapter 5
Analyzing IoT, Fog and Cloud Environments Using Real Sensor Data

Tamas Pflanzner and Attila Kertesz

Abstract There is a growing number of communicating devices joining the Internet, and we will soon face a world of a distributed computing environment with interconnected *smart* devices. Cloud-based systems have also started to dominate the Internet space, with the emergence of the Internet of Things (IoT) paradigm. In spite of the huge developments in the connectivity of devices, there is still much to do in areas such as device connectivity, communication protocols, latency, Internet bandwidth, inter-operability. In this context, Fog Computing, the latest paradigm, can come to rescue to improve the service quality by keeping and processing the data close to the user. This suggests that IoT applications can be supported by Cloud and Fog technologies to aid in the data management tasks. Besides, in many real-world solutions, we need to use simulations to investigate the inner workings of complex ICT systems. In the current presentation, our goal is to provide means to develop efficient data management algorithms in a simulation environment. In this chapter, we first analyze sensor data formats in the context of smart cities and develop a data retrieval tool for gathering and filtering the considered open datasets. We analyze three smart city initiatives that met our criteria and publish open data produced by the IoT sensors and devices. Then, we discuss how IoT data could be made available via the use of this tool for simulation environments. We also exemplify its utilization in an open-source IoT device simulator.

5.1 Introduction

Internet of Things (IoT) solutions represent a new trend for forming a dynamic global network infrastructure with self-configuring capabilities [1]. In these networks, smart devices are interacting and communicating not just among themselves but also with the environment by exchanging data, and reacting autonomously to events by triggering actions with or without direct human intervention [2].

T. Pflanzner · A. Kertesz (✉)
Software Engineering Department, University of Szeged, Szeged, Hungary
e-mail: keratt@inf.u-szeged.hu

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_5

According to recent reports [3], there will be close to 30 billion devices always online and more than 200 billion devices discontinuously online by 2020. Such estimations call for smart solutions that provide means to interconnect and control these devices in a secure and efficient way. Cloud computing [4] has the potential to serve the IoT needs, since it enables flexible resource provisions to quickly respond to dynamic conditions, and it is able to store and process sensor data in a remote location accessed from anywhere.

In the past decade, we experienced an evolution in Cloud Computing. The first Clouds appeared in the form of single virtualized datacenters, and then these broadened into a larger system of interconnected, multiple datacenters. As the next step, Cloud bursting techniques were developed to share resources of different Clouds, and then Cloud federations were realized by interoperating formerly separate Cloud systems.

In the case of IoT systems, data management operations are better placed close to their origins, thus close to the users, which resulted in better exploiting the edge devices of the network. The group of such edge nodes formed what is termed as the *Fog* [5], where Cloud storage and computational services are performed at the edge of the network, closer to the user organisations. This new paradigm enables the execution of data processing and analytics applications in a distributed way, possibly utilizing both Cloud and nearby in-house resources. Fog Computing can be used to reduce service latency and to improve service quality by keeping and processing the data close to the user. As a result, Cloud and Fog technologies can be used to aid data management tasks, but their application gives rise to complex distributed systems that still need a significant amount of research. In this chapter, we:

- Analyze sensor data formats and datasets of three smart city projects,
- Propose a data retrieval service for gathering and filtering open datasets to provide means for gathering and filtering open IoT datasets, and
- Discuss how IoT data could be made available by this tool for simulating IoT–Fog–Cloud systems, and we also exemplify its utilization in an open-source IoT device simulator.

The remainder of this chapter is structured as follows: Sect. 5.2 discusses related approaches in this field, while Sect. 5.3 analyzes the available open datasets of three smart city initiatives. Section 5.4 introduces our proposed data gathering tool and exemplifies its utilization with an IoT device simulator called MobIoTSim. Finally, we conclude our presentation in Sect. 5.5.

5.2 Related Works

In the IoT–Fog–Cloud environments, data is generated by sensors, actuators, and other smart devices, and collected, stored, and processed by local or remote gateway services depending on the applications of Fog or Cloud services. The design and application of such gateway solutions is not trivial. Kang et al. [6] introduced the main types and features of such IoT gateways, representing the state-of-the-art, and research directions in this field. Besides in real-world solutions, in many cases, we need to use simulations to investigate the operation of such gateways and to understand the inner workings of complex IoT–Fog–Cloud systems. There are many simulators available to examine distributed, and specifically Cloud and IoT, systems.

Zeng et al. [7] proposed IOTSim that supports and enables simulation of big data processing in IoT systems limiting themselves to the MapReduce model. The iFogSim solution [8] can also be used to model IoT and Fog environments by investigating resource management techniques. They use an online gaming application to demonstrate the usability of their tool. The DISSECT-CF simulator [9] has also been extended with IoT features, and its operation is demonstrated with a meteorological case study.

Moving closer to real-world applications, we may choose to simulate a part of the ecosystem, e.g., the devices that produce the data to be processed. For such purpose, we may use the simple IoT Simulator [10], which is an IoT sensor or device simulator that is able to create test environments made up of thousands of sensors and gateways on a computer, supporting many of the common IoT protocols. Its drawback is that it needs a specific, RedHat Linux environment. The Atomiton simulator [11] manages virtual sensors, actuators, and devices with unique behaviors. With this tool complex, dynamic systems can be created for specific applications, but it is a commercial, Web-based environment with very limited documentation.

A more advanced and generic solution, developed by the authors of this chapter, is MobIoTSim [12] that can be used to simulate any smart device; it can also be connected to any Cloud. It has predefined device templates and a built-in connection with the IBM Cloud. We use this tool to demonstrate our proposed approach later in Sect. 5.4. When using MobIoTSim for simulating IoT systems, working with real data could result in more convincing and more reliable applications.

In research works addressing distributed, Grid and Cloud systems, two main sources are normally used: grid workloads archive [13] and the parallel workloads archive [14]. These archives contain trace files of real-world utilization. Our goal in this chapter is to show our first attempt to create a similar archive for the benefit of the IoT research community.

5.3 Analyzing Sensor Data Formats in the Context of Smart Cities

As a first step in this research, we searched the Internet for publicly available projects of various IoT application areas. We found three smart city initiatives that met our criteria that had published open data of IoT sensors and related smart devices. In this section, we analyze these projects and the data in detail.

5.3.1 Sensor Data in the SmartME Project

The SmartME project [15] was initiated in 2015 by a group of researchers in the Mobile and Distributed Systems Laboratory at the University of Messina. Their goal was to set up a crowdfunded system using the IoT paradigm in the municipality area of the city of Messina, Italy. During the planning phase of the system, they found Cloud services very useful to be integrated with the IoT components to enable a virtual infrastructure management framework. They used OpenStack to implement the required Cloud services. They named this approach a software-defined city, where techniques of computer science, networking, data storage and sensing can work and interact together. Their solution also served as a reference architecture for building up smart cities, which represents a unified environment of static sensor network components and dynamically connecting mobile devices and objects, where contextualization is an important enabler of geo-awareness.

The project has an illustrative Web site [16], where the installed sensors and devices can be overviewed in a map. The related IoT system is composed of low-cost microcontroller boards equipped with sensors and actuators and installed on buses, traffic lamps, and buildings over the municipality area of Messina. This sensor network can be used to collect data and information of the urban area to design advanced services for citizens. In the SmartME project, they use a collection of sensors to monitor environmental properties, traffic conditions, and energy utilization.

Some of the smart city data is published in an open data platform [17] with the use of an open-source data portal software called CKAN [18], to store and visualize the collected data in a standard open data format. It has a layered structure to categorize and store data, which are accessible in different formats through an API. To collect the sampling data from the installed sensors and devices and send them to the CKAN datastore, they developed specific plugins running asynchronously on the IoT devices. The plugins periodically check the voltage levels of the connected pins of the sensors, through which they obtain the environmental data to be sent to the datastore, together with a timestamp and the geographical position of the device, which are Arduino YUN boards. It is an embedded board based on Atmel and Atheros micro-controllers running Linino OS, a Linux OpenWrt distribution based on OpenWrt. The boards have built-in Ethernet and WiFi support, as well as micro-USB connections. To host a set of low-cost sensors, a Tinkerkit Shield is used attached to a YUN board. They use a Tinkerkit thermistor, Ldr and Mpl3115 sensors to monitor temperature, brightness, and pressure properties. A Honeywell HIH-4030 sensor is used to monitor humidity, and an Arduino KY38 to track the environmental noise level. Finally, a Groove MQ9 sensor is planned to be installed for monitoring air quality, but such data is not available by the time of writing.

CKAN provides a flexible way for collecting, storing and accessing data, but after analyzing the SmartME repository we found it hard to compose a group of sensor data to be used for a specific simulation. This strengthened our research hypothesis that there is need for a mediator service, which is able to gather and filter data automatically, tailored for a specific requirement.

Now, taking a closer look at the SmartME Web site [16], we can see (at the time of writing) that there are 21 sensors groups (or nodes) within the Messina region. Each group collect sensor data at a given location about the following environmental properties: temperature, brightness, humidity, pressure, and noise. Table 5.1 summarizes these properties and the sensor and device types used to monitor them. After analyzing the open datasets, we can state that property values are sampled every 10 min, which are varying based on the device location and the actual weather conditions. They were monitored between 2016 and 2017. In the following, we present the statistical result of our investigation, in which we analyzed five datasets for each sensor type.

To measure temperature values in the SmartME project, Tinkerkit Thermistor sensors are used. The voltage level of the module varies between 0 and 5 V, depending on the outdoor temperature. The sensed value range is between 0 and 1023, these are converted to C. Table 5.2 shows statistical summaries for the examined 5 datasets (denoted by a prefix of their original identifier).

A sample temperature measurement from these datasets in JSON format can be seen in Listing 5.1. The same temperature measurement in CSV format can be seen in Listing 5.2.

Table 5.1 Sensor types in the smart city of Messina

Property	Model	Unit	Device
Temperature	Thermistor	°C	TinkerKit
Brightness	LDR	Lux	TinkerKit
Humidity	HIH-4030	%	Honeywell
Pressure	Mpl3115	hPa	TinkerKit
Noise	KY-038	Amp	Keyes

Table 5.2 Temperature measurements in the smart city of Messina

Database ID	Interval	Freq. (s)	No. of sample	Values (in °C)
2f4bbebc–…	2017.05–06	605	117	28–32
cf5c179a–…	2016.04–2017.09	605	6548	6–29
6f88f1eb–…	2017.05–09	605	10,458	18–54
606903a8–…	2016.03–2017.09	605	46,120	3–41
fdcf57f7–…	2016.04–2017.09	605	63,515	5–33

Listing 5.1 Sample temperature sensor data of SmartME

```
  {
      "_id": 1,
      "Date":                     "2017-05-
19T10:51:03.278000",
      "Temperature": 28.9273120410964,
      "Altitude": 5.35,
      "Latitude": 38.19401,
      "Longitude": 38.19401
    }
```

Listing 5.2 Sample temperature sensor data of SmartME

```
      1,2017-05-19T10:51:03.278000,
   28.9273120410964,5.35,38.19401,38.19401
```

To measure brightness, a TinkerKit Light Dependant Resistor (LDR) module is used, its resistance is decreased with higher brightness (it outputs 5 V under no light, and 0 V for the highest brightness). Table 5.3 shows the summary of brightness data in SmartME.

A sample brightness measurement from these datasets in JSON format can be seen in Listing 5.3. The same brightness measurement in CSV format can be seen in Listing 5.4.

Table 5.3 Brightness measurements in the smart city of Messina

Database ID	Interval	Freq. (s)	No. of samples	Values
c54a6a42–…	2017.05–06	605	117	1–368
af811db2–…	2016.04–2017.09	605	6544	0–754
407eea34–…	2017.05–09	605	10,452	1–1231
84e60570–…	2016.03–2017.09	605	46,126	0–916
f8c80ed5–…	2016.04–2017.09	605	63,515	0–2562

Listing 5.3 Sample brightness sensor data of SmartME

```
{
    "_id": 1,
    "Date": "2017-05-22T11:40:11.539000",
    "Brightness": 200.7016666209489,
    "Altitude": 17.8,
    "Latitude": 38.16883,
    "Longitude": 15.54466
  }
```

Listing 5.4 Sample temperature sensor data of SmartME

```
    3,2017-05-
  21T16:45:12.134000,31.692271842701814,
  120.82,38.16087,15.52085
```

Table 5.4 shows statistics of humidity environmental properties for 5 datasets in the SmartME project. They were measured with Honeywell HIH-4030 sensors, which are ideal for battery powered applications.

A sample humidity measurement from these datasets in JSON format can be seen in Listing 5.2a. The same humidity measurement in CSV format can be seen in Listing 5.5.

Table 5.4 Humidity measurements in the smart city of Messina

Database ID	Interval	Freq. (s)	No. of samples	Values
9124f050–…	2017.05–06	605	117	31–56
ea82840b–…	2016.04–2017.09	605	6544	17–110
337adbaa–…	2017.05–09	605	10,449	18–113
bde446bc–…	2016.03–2017.09	605	46,125	10–95
2be73f19–…	2016.04–2017.09	605	63,518	10–90

Listing 5.2a A sample humidity measurement in JSON

```
{
    "_id": 1,
    "Date": "2017-05-19T10:51:03.278000",
    "Humidity": 43.07611202676012,
    "Altitude": 5.35,
    "Latitude": 38.19401,
    "Longitude": 38.19401
  }
```

Listing 5.5 Sample humidity sensor data of SmartME

```
 1,2017-05-
19T10:51:03.278000,43.07611202676012,5.35,38.1
9401,38.19401
```

To monitor air pressure, MPL3115 sensors are used with TinkerKit. They provide high resolution and support power saving mode, and their operating range is between 20 and 110 kPa. Table 5.5 provides pressure-related statistics.

A sample air pressure measurement from these datasets in JSON format can be seen in Listing 5.6. The same air pressure measurement in CSV format can be seen in Listing 5.7

Table 5.5 Pressure measurements in the smart city of Messina

Database ID	Interval	Freq. (s)	No. of samples	Values
e025857e–…	2017.05–06	605	117	1005–1010
c1f5e699–…	2016.04–2017.09	605	6573	946–1007
e42905aa–…	2017.05–09	605	10,457	1001–1024
4f185156–…	2016.03–2017.09	605	46,123	3–1033

Listing 5.6 Sample air pressure sensor data of SmartME

```
{
    "_id": 1,
    "Date": "2017-05-22T11:40:11.539000",
    "Pressure": 1007.1175,
    "Altitude": 17.8,
    "Latitude": 38.16883,
    "Longitude": 15.54466
  }
```

Listing 5.7 Sample air pressure sensor data of SmartME

```
  1,2017-05-
22T11:40:11.539000,1007.1175,17.80,38.16883,15.
54466
```

For sensing noise, they use Arduino KY-038 microphone sound sensor modules providing both analog and digital outputs (with adjustable sensitivity). Table 5.6 summarizes noise measurement statistics for five datasets.

A sample noise measurement from these datasets in JSON format can be seen in Listing 5.8. The same noise measurement in CSV format can be seen in Listing 5.9.

Table 5.6 Noise measurements in the smart city of Messina

Database ID	Interval	Freq. (s)	No. of samples	Values
a61d33f0–…	2017.05–06	605	124	0–3
24ef5d60–…	2016.04–2017.09	605	6546	0–65
356224e4–…	2017.05–09	605	10,451	0–73
9bf842ca–…	2016.03–2017.09	605	46,120	0–55
5c1c9944–…	2016.04–2017.09	605	63,518	0–151

Listing 5.8 Sample noise sensor data of SmartME

```
  {
      "_id": 1,
      "Date": "2017-05-22T11:40:11.539000",
      "Noise": 1.82608695652173 9,
      "Altitude": 17.8,
      "Latitude": 38.16883,
      "Longitude": 15.54466
    }
```

Listing 5.9 Sample noise sensor data of SmartME

```
  1,2017-05-
22T11:40:11.539000,1.82608695652173 9,17.80,
38.16883,15.54466
```

Besides environmental property monitoring, prototypes have been developed for smart energy, parking, and traffic management. According to the information provided in the project Web sites [16] and [17], these parts of the SmartME project are still under development and testing, therefore partial datasets are available, and no real-time measurements are performed currently. Nevertheless, we found it interesting and useful to examine the open data on traffic light lamps. The summary of these measurements published in a single dataset is shown in Table 5.7. In this set, ten lamps were monitored for 10 days, and their actual working hours and states were saved.

A sample traffic light measurement from this dataset in JSON format can be seen in Listing 5.10. The same noise measurement in CSV format can be seen in Listing 5.11.

Table 5.7 Traffic light measurements in the smart city of Messina

Database ID	Interval	No. of samples	Working hours	Values
54b1f2ee–…	2017.06.17–26	5713	0–3825	True/False

Listing 5.10 Sample traffic light sensor data of SmartME

```
{
   "on": true,
   "working_hours": "3404",
   "id_lamp": "lamp-001",
   "timestamp": "2017-06-17T23:45:12",
   "count_power_on": "440",
   "lat": "37.55560",
   "lng": "14.98149",
   "_id": 1
  }
```

Listing 5.11 Sample temperature sensor data of SmartME

```
3  lamp-003  37.55560  14.98149  True

   311  3741 2017-06-17T23:49:24
```

5.3.2 Sensor Data in the CityPulse Project

CityPulse was a former European FP7 project [19] run between 2014 and 2016. As stated on their Web site [20], its goal was to develop innovative, reliable, and real-time smart city applications by adopting and integrating the Internet of Things paradigm, knowledge-based computing, and reliability testing.

The project designed a flexible and extensible smart city data analytics framework. They developed software components for real-time data stream publication and annotation, installed at two locations: in Aarhus, Denmark, and in Brasov, Romania. The project team composed of citizens, stakeholders, and technical colleagues has developed over a hundred smart city scenarios that were published and made available online. The set of key datasets are published as CityPulse Reference Datasets that consists of over 180 GB of time series data and live data feeds according to [21].

The CityPulse Dataset Collection containing various semantically annotated datasets is available in [22]. In this subsection, we summarize the contents of these datasets archiving weather observations and parking-related monitoring information. The collection also includes datasets of specific events, which we found less relevant for further investigations and simulations, therefore, we excluded them from our work. Most measurements are available in raw JSON format and in annotated TTL formats. Some measurements are also available in CSV format.

In the CityPulse project, the following environmental properties were monitored both in Aarhus and Brasov: temperature and dew point (in °C), pressure (in mBar), humidity (in %), wind direction (in degree) and wind speed (in kph). We could not find detailed information on the exact sensors used to perform these measurements. By examining the datasets, we can see that there in no more real-time monitoring, and the publicly available data dates back to 2014, containing only some months of measurements. The data structure is different than the one used in the previously seen SmartME project. The frequency of sampling is also different: generally three measurements are performed in an hour (after 10, 20, and 30 min in a loop).

We examined in detail the two available datasets of drewpoint observations from the city of Aarhus. The summary of these measurements is given in Table 5.8.

A sample traffic light measurement from this dataset in JSON format can be seen in Listing 5.12.

Listing 5.12 Sample traffic light sensor data of SmartME

```
  {
  "2014-02-13T06:20:00": "2.0",
"2014-02-13T13:50:00": "1.0",
"2014-02-13T06:00:00": "1″
  }
```

Table 5.9 shows statistics of air pressure environmental properties at the same as the previous drewpoint measurements.

A sample air pressure measurement from this dataset in JSON format can be seen in Listing 5.13.

Table 5.8 Drewpoint measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (°C)
2014.02–06	600–1800	8352	2014.02–06
2014.08–09	600–1800	3782	2014.08–09

Table 5.9 Air pressure measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (mBar)
2014.02–06	600–1800	8353	986–1038
2014.08–09	600–1800	3782	992–1030

Listing 5.13 Sample air pressure sensor data of SmartME

```
  {
  "2014-08-01T01:00:00": "1012",
"2014-08-01T13:00:00": "1013",
"2014-08-01T01:20:00": "1012"
  }
```

Table 5.10 shows statistics of humidity environmental properties at the same as the previous drewpoint measurements.

A sample humidity measurement from this dataset in JSON format can be seen in Listing 5.14.

Listing 5.14 Sample humidity sensor data of SmartME

```
  {
  "2014-02-13T06:20:00": "93",
"2014-02-13T13:50:00": "66",
"2014-02-13T06:00:00": "91"
  }
```

Table 5.10 Humidity measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (%)
2014.02–06	600–1800	8353	15–94
2014.08–09	600–1800	3782	16–94

Table 5.11 Temperature measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (°C)
2014.02–06	600–1800	8353	−3 to 25
2014.08–09	600–1800	3782	2–27

Table 5.11 shows statistics of temperature environmental properties at the same as the previous drewpoint measurements.

A sample temperature measurement from this dataset in JSON format can be seen in Listing 5.15.

Listing 5.15 Sample temperature sensor data of SmartME

```
  {
  "2014-08-01T01:00:00"": "18",
"2014-08-01T13:00:00": "22",
"2014-08-01T01:20:00": "18"
  }
```

Table 5.12 shows statistics of wind direction environmental properties at the same as the previous drewpoint measurements.

A sample wind direction measurement from this dataset in JSON format can be seen in Listing 5.16.

Listing 5.16 Sample temperature sensor data of SmartME

```
  {
  "2014-08-01T15:50:00": "170",
"2014-08-01T02:20:00": "220",
"2014-08-01T23:50:00": "140"
  }
```

Table 5.12 Wind direction measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (°)
2014.02–06	600–1800	8353	0–360
2014.08–09	600–1800	3782	0–360

Table 5.13 Wind speed measurements in the smart city of Aarhus

Interval	Freq. (s)	No. of samples	Values (kph)
2014.02–06	600–1800	8353	0–63
2014.08–09	600–1800	3782	0–39

Table 5.13 shows statistics of wind direction environmental properties at the same as the previous drewpoint measurements.

A sample wind speed measurement from this dataset in JSON format can be seen in Listing 5.17.

Listing 5.17 Sample temperature sensor data of SmartME

```
  {
  "2014-02-16T03:00:00": "16.7",
"2014-02-16T15:00:00": "24.1",
"2014-02-16T03:20:00": "18.5"
  }
```

Besides weather observations, we can find a parking-related dataset as well. It contains sensed information of 8 different parking lots in the city of Aarhus. The published data covers a period of 6 months. The statistical summary of this dataset is provided in Table 5.14.

Finally, pollution monitoring was supposed to complement parking space monitoring in the CityPulse project. Since no such sensor infrastructure was installed within the project, they decided to simulate this data and to generate artificial datasets for pollution information. The dataset was generated randomly (within a predefined range) to cover 3 months with 449 observation points, including the following properties: ozone, particulate matter, carbon monoxide, sulfur dioxide, and nitrogen dioxide.

Table 5.14 Parking data in the smart city of Aarhus

Parking lot	Interval	Freq. (s)	No. of samples	Total spaces	No. of cars
8 places	2014.05–11	30	553,055,303	131k65–1240	0–1236
8 places	2014.02–10	30	55,303	65–1240	0–1236

5.3.3 Sensor Data in the Smart City of Surrey in Canada

The government of the city of Surrey, Canada, has defined its Open Data Program in 2014 [23]. Their original goal was to set up an open, transparent, and accessible government and to empower citizens and to help small businesses to create value. The open data published through its projects is freely available for everyone to use and republish according to the Open Government Licence of Surrey (details in [24]). By the time of writing, 359 datasets are available through its Web site [25].

It has many dataset categories, e.g., land use, infrastructure, transportation, environmental services, business, economy, and culture. One of the earliest datasets in this database is the Historical Climate Data, which contains environmental measurements sampled between 1929 and 2007. It contains information on measured minimum and maximum temperature values, and information on rainfall, snowfall and gust. Table 5.15 presents statistical information on this dataset.

There are also many water-related measurements. Many of these samples are viewable and searchable through a service called City of Surrey Mapping Online System (COSMOS) having a graphical interface [26]. They are able to monitor residential water consumption and have access to various valves, mains, fittings, and drainage monitoring devices. For example, water meters are specific devices used to measure the amount of water flowing through water connections. This dataset contains 65,798 samples with various parameters, such as identifier, location (street, number), status, and geolocation of the device. Water pipe underground are used to transfer drinking water. A dedicated dataset is used to monitor their availability and properties including 28,100 samples.

Another possibly reusable dataset is the container of water levels of the river Fraser that flows nearby Surrey. The set contains measurements for two months in 2015 at two locations. Statistical data on this dataset is depicted in Table 5.16, and detailed statistical sample values of this dataset are shown in Table 5.17.

Table 5.15 Historical climate data of surrey

Type	Interval	Freq.	No. of samples	Values
Temperature	1929.01–2007.02	Monthly	658	−9.7 to 25.2 °C
Rainfall	1929.01–2007.02	Monthly	493	0–333 mm
Snowfall	1929.01–2007.02	Monthly	496	0–92.7 mm

Table 5.16 Statistics on the river fraser water levels dataset

Name	Interval	Freq.	No. of samples	Sample size (kB)
Waterlevels-2015 March	2015.03	60	1488	45
Waterlevels-2015 April	2015.04	60	1433	43

Table 5.17 Statistics on the river fraser water level monitoring stations

Name	Station	Values (m)
Waterlevels 2015 March	Manson	−0.899 to 2.022
	192 ST	−0.649 to 2.062
Waterlevels 2015 April	Manson	−0.752 to 1.673
	192 ST	0.3288–1.8522

In the group of environmental datasets, we can find further information, e.g., on tree planting (Park Specimen Trees). This sample set contains information on trees living in the parks, along the major roads, or specific properties of Surrey. In the transportation category, we can find information related to cars, transit, routes and gas emissions. Using the real-time transit dataset, live transit information including stops, stop estimates, buses, routes, and statuses, which are also available in a map representation.

5.4 Filtered Datasets for Experimenting with IoT–Fog–Cloud Environments

In the previous sections, we introduced datasets of three smart cities. There are many other smart city projects in the world, and this number is expected to grow in the near future. Hopefully, more and more datasets will be accessible publicly containing useful sensor information. To efficiently and easily work with several datasets located at different sources, we need a service which gathers and filters such data. There are three main functions that should be implemented in such a service, via:

- List the datasets
- Show detailed informations about a specific dataset
- Query/filter a dataset.

For browsing and searching for the required sensor dataset for further reuse, a Webpage is the best way to provide these features. To help other applications to reuse the data and for filtering capabilities, an API is also preferable. The dynamic list of the available datasets can be collected in different ways, based on the data source. A datasource can be online accessible through an own API, or it can be downloadable stored in a file or in a local database. This list should contain basic information about the datasets, such as the name, source, categories and size. Even this list could be searched, filtered and ordered to help finding the right dataset for the user or application. After selecting a specific dataset, more details can be obtained, like the date interval of the data, field names, types, and ranges, with an example datasample. Even different statistics can be calculated and displayed in a table or with a chart. The query/filter should include at least the following three main parts:

- Opportunity to include or exclude fields from the results;
- Responsible for the data filtering based on the filed types. For example, if the field is an integer, the minimum and maximum values can be given, similar if it is a date type, just in a valid date format. In case, when the field contains a name or ID, the tool should display the possible distinct values. If the field type is a Boolean, the user could select a criterion to select only the true, false, or both values.
- Specification of the result format. It can be a preview or a downloadable file in JSON format or even a stream.

To implement the above-mentioned requirements, we propose a tool called SUMMON, which is an abbreviation for *smart city usage data management and monitoring*. Its main features and operations are depicted in Fig. 5.1. In the first prototype of the proposed tool, we implemented some basic features. It is a Web site implemented in nodeJS and uses the CKAN system API.[1] In case of the SmartME smartcity project, a base URL[2] for the API requests is also available. The CKAN system has a hierarchical structure, where the so organizations are on the top. It is possible to list all organizations with the http://smartme-data.unime.it/api/action/organization_list call, we can even request more information with the `all_fields = true` parameter, but unfortunately the datasets (or in CKAN dialect: packages) of an organization cannot be listed, and this is the reason why we used a different API call to query the datasets. We can get more detailed information about a dataset, like the license, maintainer, author.

SUMMON uses an API call to receive the schema of the dataset, so it can generate a form based on that. After the user finished with the form, the settings are used to generate an SQL statement, which is runned with the following call: http://smartme-data.unime.it/api/action/datastore_search_sql?sql = <SQL > . The results are displayed as a preview, embedded in an html code with the possibility to modify the form and see some example results, or in JSON format to be downloadable.

In order to demonstrate the applicability of SUMMON, we used a mobile IoT device simulator called MobIoTSim [12]. It was developed to target researchers and future IoT application users with the aim to exemplify IoT device handling without owning or buying real sensors. It can be used to test and demonstrate the inner workings of IoT Cloud systems utilizing multiple devices. It can be easily connected to a private gateway service (available and referred to in [27]) we formerly developed for the IBM Cloud platform. It is able to receive process sensor data coming from several devices simultaneously. It has a Web-based graphical interface to visualize sensor data coming from MobIoTSim. Messages (defined in JSON format) sent by the simulated devices are managed by an MQTT server. It can also be used to send responses (or notifications) back to the simulated IoT devices in MobIoTSim.

[1]http://docs.ckan.org/en/latest/api/index.html

[2]http://smartme-data.unime.it/api/action

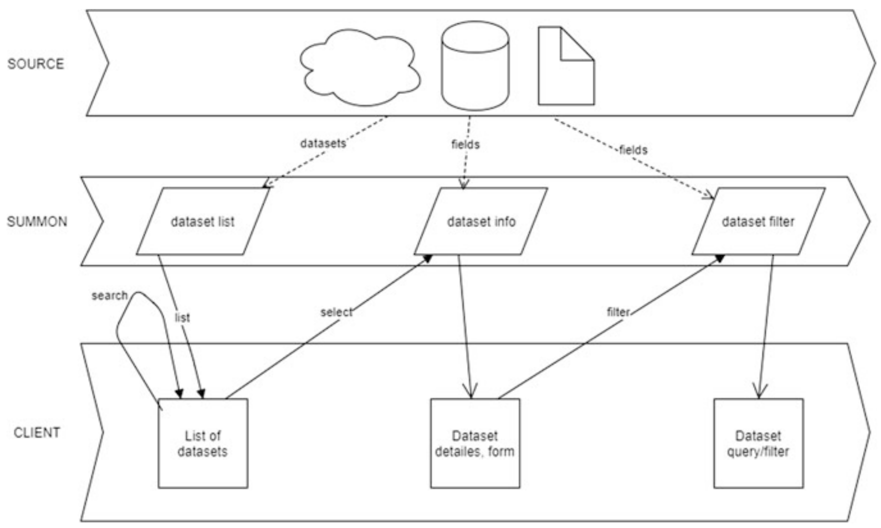


Fig. 5.1 Basic operations of SUMMON

As Fig. 5.2 shows the simulation of a whole IoT environment scenario starts with the filtered JSON data from SUMMON imported to the MobIoTSim simulator. When the user starts a simulation with MobIoTSim, the sensor data are sent in the form of messages to the MQTT Broker of the IBM Bluemix IoT Service [28]. After it receives these real-life values, the broker forwards them to a gateway application. The gateway application shows a real-time chart with the data, even from several devices simultaneously.

In cases, where we need just the specific values from the dataset, or just a specific number or value of records, SUMMON can generate a form, where the required criteria can be added, as shown in Fig. 5.3.

The downloaded JSON data from SUMMON can be imported to the MobIoTSim simulator. The process can be done similarly as creating a new device (as shown in Fig. 5.4), but instead of generating random data, we can select the downloaded file to read the values from it. In this way, the created simulated device can send real-life data to the Cloud. Even more simulated devices can be added, with the same or different datasource imported from SUMMON.

The IoT Service in the IBM Bluemix Cloud receives the data from the MobIoTSim with MQTT protocol, and forwards it to a gateway Web application. This application is capable of plotting a real-time chart with the received data, shown in Fig. 5.5.

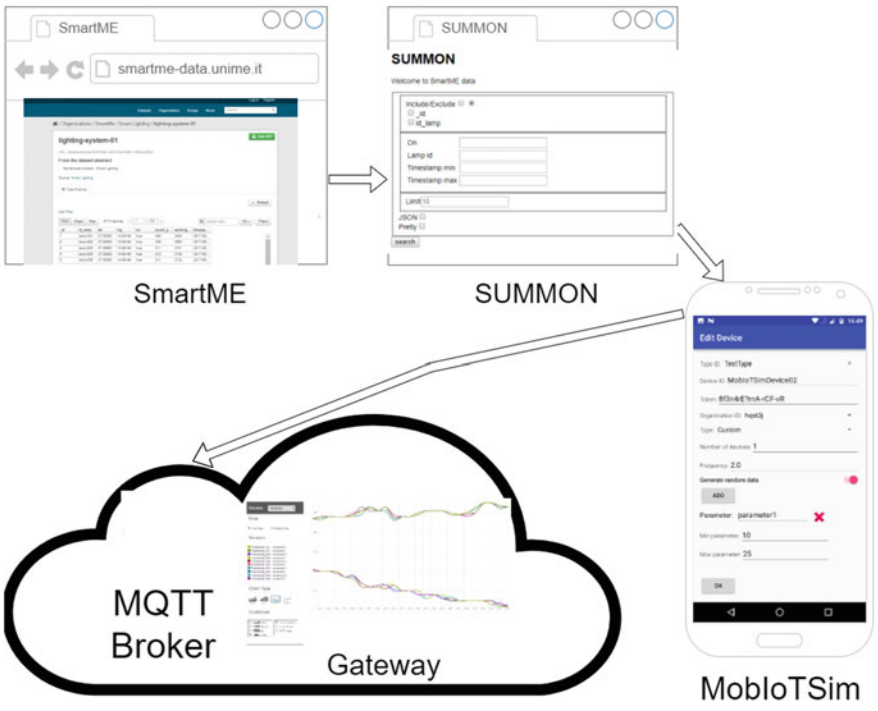
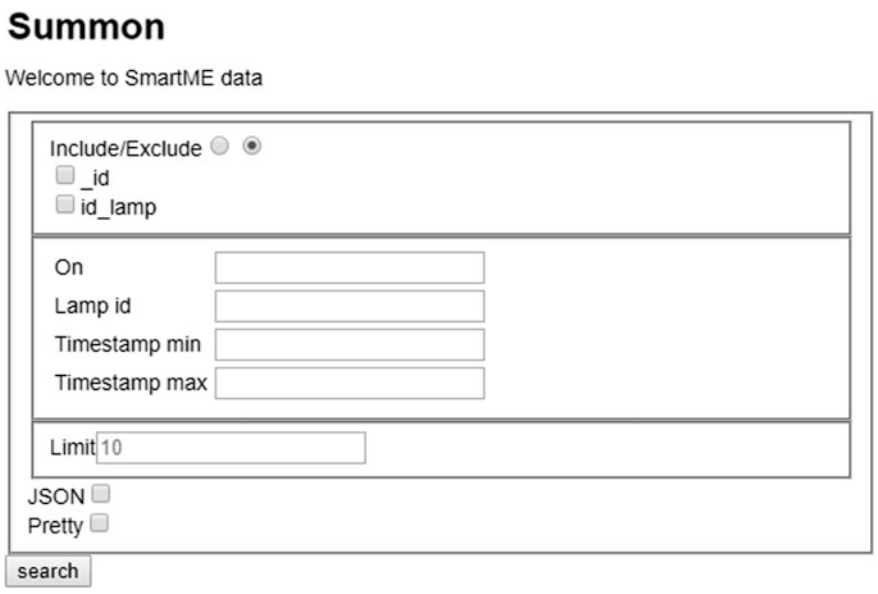


Fig. 5.2 IoT environment simulation with SUMMON and MobIoTSim



Summon

Welcome to SmartME data

Include/Exclude ○ ●
☐ _id
☐ id_lamp

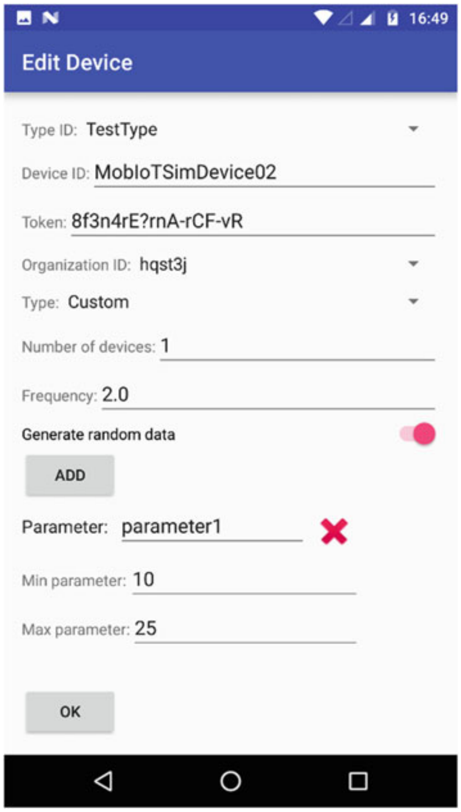
On
Lamp id
Timestamp min
Timestamp max

Limit 10

JSON ☐
Pretty ☐

search

Fig. 5.3 Form generated by SUMMON



16:49

Edit Device

Type ID: TestType

Device ID: MobIoTSimDevice02

Token: 8f3n4rE?rnA-rCF-vR

Organization ID: hqst3j

Type: Custom

Number of devices: 1

Frequency: 2.0

Generate random data

ADD

Parameter: parameter1

Min parameter: 10

Max parameter: 25

OK

Fig. 5.4 Import file to MobIoTSim

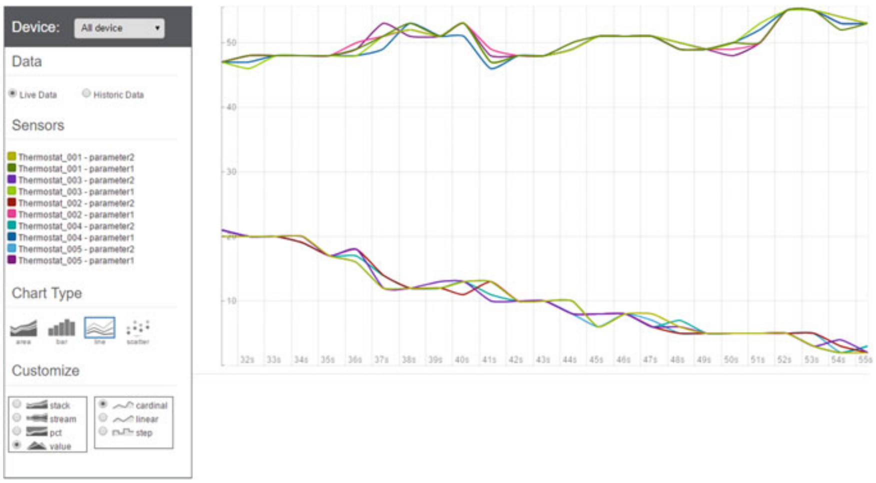

Device: All device
Data
Live Data
Historic Data
Sensors
🟨 Thermostat_001 - parameter2
🟩 Thermostat_001 - parameter1
🟪 Thermostat_003 - parameter2
🟩 Thermostat_003 - parameter1
🟥 Thermostat_002 - parameter2
🟥 Thermostat_002 - parameter1
🟦 Thermostat_004 - parameter2
🟦 Thermostat_004 - parameter1
🟦 Thermostat_005 - parameter2
🟪 Thermostat_005 - parameter1
Chart Type
area
bar
line
scatter
Customize
stack
stream
pct
value
cardinal
linear
step
50
40
30
20
10
32s
33s
34s
35s
36s
37s
38s
39s
40s
41s
42s
43s
44s
45s
46s
47s
48s
49s
50s
51s
52s
53s
54s
55s

Fig. 5.5 Gateway shows real-time chart in the IBM Bluemix Cloud

5.5 Conclusion

Following the recent technological trends in ICT, IoT environments generate unprecedented amounts of data that needs to be stored, processed, and analyzed. Cloud systems already started to dominate the Internet, and with the appearance of things of the IoT paradigm, IoT Cloud systems are formed. Fog Computing can be used to reduce service latency and to improve service quality by keeping and processing the data close to the user. IoT applications can also be supported by Fog technologies to achieve these goals, resulting in IoT–Fog–Cloud systems having many open issues.

In this chapter, we addressed some of these issues and analyzed sensor data formats and datasets of three smart cities projects. Based on the results of these investigations, we proposed a tool called SUMMON to provide means for gathering and filtering open datasets. We also discussed how filtered IoT data could be made available by this service to perform IoT–Fog–Cloud investigations in simulated environments. Finally, we demonstrated its utilization with an open-source IoT device simulator called MobIoTSim.

Our future work will address the extension of our proposed data retrieval service by supporting additional sensor datasets from different IoT application areas, and we also plan to enable additional APIs for communicating with our service.

Acknowledgements This research was supported by the Hungarian Government and the European Regional Development Fund under the grant number GINOP-2.3.2-15-2016-00037 (Internet of Living Things), and by the UNKP-17-4 New National Excellence Program of the Ministry of Human Capacities, Hungary.

References

1. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener Comput Syst 29(7):1645–1660
2. Sundmaeker H, Guillemin P, Friess P, Woelfﬂe S (2010) Vision and challenges for realising the internet of things. CERP IoT—cluster of european research projects on the internet of things, CN: KK-31-10-323-EN-C
3. Rossi B (2015) Gartner's internet of things predictions. Online: http://www.information-age.com/technology/mobile-and-networking/123458905/gartners-internet-things-predictions
4. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25:599–616
5. Dastjerdi V, Buyya R (2016) Fog computing: helping the internet of things realize its potential. Computer 49(8):112–116. https://doi.org/10.1109/MC.2016.245
6. Kang Kim D, Choo H (2017) Internet of everything: a large-scale autonomic IoT gateway. IEEE Trans Multi-Scale Comput Syst 3(3):206–214. https://doi.org/10.1109/TMSCS.2017.2705683
7. Zeng X, Garg SK, Strazdins P, Jayaraman PP, Georgakopoulos D, Ranjan R (2016) {IOTSim}: a simulator for analysing IoT applications. J Syst Arch 72:93–107. https://doi.org/10.1016/j.sysarc.2016.06.008

8. Gupta H, Dastjerdi AV, Ghosh SK, Buyya R (2016) iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. arXiv:1606.02007, CLOUDS-TR-2016-2
9. Markus A, Kertesz A, Kecskemeti G (2017) Cost-aware IoT extension of DISSECT-CF. Future Internet 9(3):47. https://doi.org/10.3390/fi9030047
10. SimpleSoft SimpleIoTSimulator. Online: http://www.smplsft.com/SimpleIoT\-Simulator.html . Accessed Dec 2017
11. Atomiton IoT Simulator. Online: http://atomiton.com/simulator.html. Accessed Dec 2017
12. Pflanzner T, Kertesz A, Spinnewyn B, Latre S (2016) MobIoTSim: towards a mobile IoT device simulator. In: IEEE 4th international conference on future internet of things and cloud workshops (FiCloudW), Vienna, pp 21–27. https://doi.org/10.1109/w-ficloud.2016.21
13. Iosup A, Li H, Jan M, Anoep S, Dumitrescu C, Wolters L, Epema DHJ (2008) The grid workloads archive. Future Gener Comput Syst 24(7):672–686. https://doi.org/10.1016/j.future.2008.02.003
14. Feitelson DG, Tsafrir D, Krakov D (2014) Experience with using the parallel workloads archive. J Parallel Distrib Comput 74(10):2967–2982. https://doi.org/10.1016/j.jpdc.2014.06.013
15. Bruneo D, Distefano S, Longo F, Merlino G, Puliafito A (2017) Turning Messina into a smart city: the #SmartME experience. In: Stratigea A, Kyriakides E, Nicolaides C (eds) Smart cities in the mediterranean. Springer, Cham. https://doi.org/10.1007/978-3-319-54558-5_6
16. #SmartME project website (2017) Online: http://smartme.unime.it/. Accessed Dec 2017
17. #SmartME (2017) SmartME project datasets. Online: http://smartme-data.unime.it/dataset. Accessed Dec 2017
18. CKAN (2017) CKAN website. Online: https://ckan.org/. Accessed Dec 2017
19. Bischof S, Karapantelakis A, Nechifor C, Sheth A, Mileo A, Barnaghi P (2014) Semantic modeling of smart city data. In: Proceedings of W3C workshop on the web of things: enablers and services for an open web of devices. Berlin, Germany
20. CityPulse (2017) CityPulse EU FP7 project. Online: http://www.ict-citypulse.eu. Accessed Dec 2017
21. CityPulse (2017) CityPulse project final report. Online: http://www.ict-citypulse.eu/page/sites/default/files/citypulse_d1.4_final_report_final.pdf. Accessed Dec 2017
22. CityPulse (2017) CityPulse dataset collection website. Online: http://iot.ee.surrey.ac.uk:8080/index.html. Accessed Dec 2017
23. Open data program of the city of surrey (2017) Online: http://data.surrey.ca/pages/about. Accessed Feb 2018
24. Open data documents city of surrey (2018) Online: http://data.surrey.ca/dataset/open-data-documents. Accessed Feb 2018
25. Open datasets of the city of surrey (2018) Online: http://data.surrey.ca/dataset. Accessed Feb 2018
26. COSMOS (2018) COSMOS website. Online: http://cosmos.surrey.ca/external/. Accessed Feb 2018
27. Github (2017) Private gateway service for MobIoTSim. Online: https://github.com/sed-szeged/MobIoTSimBluemixGateway. Accessed Dec 2017
28. IBM (2017) IBM bluemix website. Online: https://www.ibm.com/cloud/. Accessed Dec 2017

Chapter 6
Performance Enhancement of Fog Computing Using SDN and NFV Technologies

J. Pushpa and Pethuru Raj

Abstract The concept of Edge or Fog Computing is fast emerging as a powerful Computing paradigm for real-time data capture, processing, decision-making and actuation. The accumulation of Edge devices in different forms in multiple environments is the key motivation for the surging popularity of Fog Computing. In this chapter, we concentrate on identifying the important parameter of device performance in Fog Computing environments. This measurement metric helps to analyse the Quality of Service (QoS) attributes for Fog or Edge applications. The Fog device connectivity guarantees the goal of low latency for fulfilling time-sensitive applications and use cases. However, there are issues with the participating devices and the connectivity between devices and the intermediaries such as the IoT gateway, broker or the middleware. The primary communication happens in wireless mode, and hence, there are several variables that can intermittently go wrong. The purpose of this chapter is to expose the various performance issues being associated with the Fog Computing model and to convey the ways and means of establishing and sustaining the goal of high performance.

6.1 Introduction

The faster maturity and stability of Edge technologies has blossomed into a big factor in realizing scores of digitized elements, smart objects and sentient materials out of common and casual items in our midst. These empowered entities are data generating, data capturing, buffering and data transmitting devices. That is, tangible things are peppered with intelligence and prepared for the future to ease human existence. These are mostly resource-constrained, and this latest phenomenon is

J. Pushpa (✉)
Visvesvaraya Technological University, Belgaum, Karnataka, India
e-mail: pushpaj.mca07@gmail.com

P. Raj
Site Reliability Engineering (SRE) Division, Reliance Jio Infocomm Ltd.,
Bangalore, Karnataka, India

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_6

called the Internet of Things (IoT). Furthermore, a wider variety of gadgets and gizmos in our working, walking and wandering locations are futuristically instrumented to be spontaneously interconnected and exceptionally intelligent in their behaviours. Thus, we hear, read and even feel connected with the cognitive devices and machines in our everyday life.

Once upon a time, all our personal computers were connected via networks (LAN and WAN), and nowadays, our personal and professional devices (fixed, portables, mobiles, wearables, handhelds, phablets, etc.) are increasingly interconnected (through BAN, PAN, CAN, LAN. MAN and WAN) to exhibit a kind of intelligent behaviour. This extreme connectivity and service-enablement of our everyday devices go to the level of getting seamlessly integrated with off-premise, online, and on-demand cloud-based applications, services, data sources and content. This cloud-enablement is capable of making ordinary devices into extraordinary ones. However, most of the well-known and widely used embedded devices individually do not have sufficient computation power, battery, storage and I/O bandwidth to host and manage the complex IoT applications and services. Hence, performing data analytics on individual devices is a little difficult.

As we all know, smart sensors and actuators are being randomly deployed in any significant environments such as homes, hospitals, hotels in order to minutely monitor, precisely measure and insightfully manage the various parameters of the environments. Further on, powerful sensors are embedded and etched on different physical, mechanical, electrical and electronic systems in our everyday environments in order to empower them to join in the mainstream Computing. Thus, not only environments but also all tangible things in those environments are also smartly sensor-enabled with a tactic as well as the strategic goal of making them distinctly sensitive and responsive in their operations, offerings and outputs. Sensors are sweetly turning out to be the inseparable eyes and ears of any important thing in near future. This systematic sensor-enablement of ordinary things not only makes them extraordinary but also lays out a stimulating and sparkling foundation for generating a lot of usable and time-critical data. Typically, sensors and sensor-attached assets capture or generate and transmit all kinds of data to the faraway cloud environments (public, private and hybrid) through a host of standard-compliant sensor gateway devices. Precisely speaking, cloud environments represent the dynamic combination of several powerful server machines, storage appliances, and network solutions and are capable of processing tremendous amounts of multi-structured data to spit out actionable insights.

However, there is another side to this remote integration and data processing. For certain requirements, the local or proximate processing of data is mandated. That is, capturing sensor and device data and transmitting them to the faraway cloud environments is not going to be beneficial for time-critical applications. Thereby the concept of Edge or Fog Computing has emerged and is evolving fast these days with the concerted efforts of academic as well as corporate people. The reasonably powerful devices such as smartphones, sensor and IoT gateways, consumer electronics, set-top boxes, smart TVs, Web-enabled refrigerators, Wi-Fi routers are classified as Fog or Edge devices to form Edge or Fog clouds to do the much-needed local processing quickly and easily to arrive and articulate any hidden knowledge, thus making those devices powerful and eminent by improving their performance. Achieving the performance of Fog devices is wrapped up with many parameters such as low latency, suitable protocols, upstream and downstream the data, node discovery algorithm, load balancing, resilient and data processing. In this context, the performance parameters are an indefinite set whose size varies with nature of application and environment.

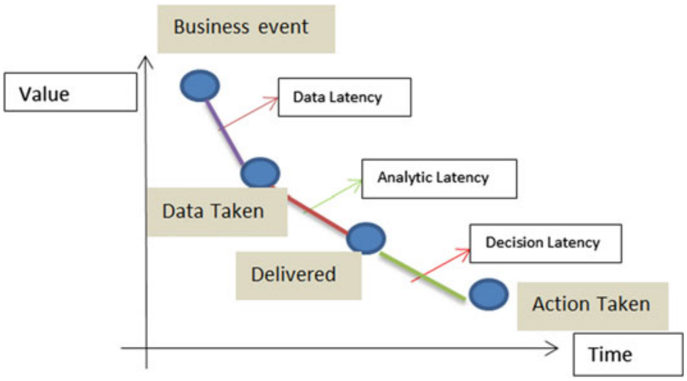
This chapter is organized as follows. In Sect. 6.2, we articulate the Fog Computing paradigm comparing with cloud Computing and in the Sects. 6.3–6.5, Fog Computing paradigm and Fog nodes with their functionality and behaviour are discussed. Section 6.6 accumulates the inputs from the previous discussion and develops an algorithm for Fog Computing working strategy. Sections 6.7 and 6.8 are the crucial part of our chapter for understanding the parameter to measure the Fog Computing and its performance improvement.

6.2 The Paradigm of Fog/Edge Computing

Traditional networks, which feed data from devices or transactions to a central storage hub (data warehouses and data marts), cannot keep up with the data volume and velocity created by interconnected IoT devices, nor can the data warehouse model meet the low latency response times that users demand. The Hadoop platform in the cloud was supposed to be an answer. But sending the data to the cloud for analysis also poses a risk of data bottlenecks as well as security concerns. New business models, however, need data analytics in a minute or less. The problem of data congestion will only get worse as IoT applications and devices continue to proliferate.

There are certain interesting use cases such as rich connectivity and interactions among vehicles (V2V) and infrastructures (V2I). IoT provides solution for domains like entertainment, education, and information, public safety, real-time traffic analysis and information, support for high mobility, context awareness and so forth. Such things see the light only if the infotainment systems within vehicles have to identify and interact with one another dynamically and also with wireless communication infrastructures made available on the road, with remote traffic servers and FM stations, etc. The infotainment system is emerging as the highly synchronized gateway for vehicles on the road. Local devices need to interact themselves to collect data from vehicles and roads/expressways/tunnels/bridges to process them instantaneously to spit out useful intelligence. This is the salivating and sparkling foundation for Fog/Edge Computing.

The value of the data decreases as the time goes. That is, the timeliness and the trustworthiness of data are very important for extracting actionable insights. The moment the data gets generated and captured, it has to be subjected to processing. That is, it is all about real-time capture. Also it is all about gaining real-time insights through rule/policy-based data filtering, enrichment, pattern searching, aggregation,



Graph 6.1 Slow performance

knowledge discovery, etc., to take a real-time decision and to build real-time applications. Graph 6.1 clearly articulates how the delay in capturing and analysing data costs a lot in terms of business, technical and user values.

The latest trend of Computing paradigm is to push the storage, networking and computation to Edge/Fog devices for availing certain critical services. As devices are interconnected and integrated with the Internet, their computational capabilities and competencies are uniquely being leveraged in order to lessen the increasing load on cloud infrastructures. Edge devices are adequately instrumented at the design stage itself to interconnect with nearby devices automatically so that multiple devices dynamically can be found, bound and composed for creating powerful and special-purpose Edge clouds. Thus, the concept of Fog or Edge Computing is blooming and gaining much attention.

The essence and gist of Fog Computing are to keep data and computation close to end-users at the Edge of the network, and this arrangement has the added tendency of producing a new class of applications and services to end-users with low latency, high bandwidth and context awareness. Fog is invariably closer to humans rather than clouds, and hence, the name 'Fog Computing' is overwhelmingly accepted across. As indicated and illustrated above, Fog devices are typically resource-intensive Edge devices. Fog Computing is usually touted as the supplement and complement to the popular cloud Computing. Students, scholars and scientists are keen towards unearthing a number of convincing and sellable business and technical cases for Fog Computing. Being closer to people, the revitalized Fog or Edge Computing is to be extremely fruitful and fabulous in conceptualizing and concretizing a litany of people-centric software applications. Finally, in the era of big, fast, streaming and IoT data, Fog/Edge Computing can facilitate Edge analytics. Edge devices can filter out redundant, repetitive and routine data to reduce the precious network bandwidth and the data loads on clouds. Figure 6.1 vividly illustrates the fast-emerging three-tier architecture for futuristic Computing.

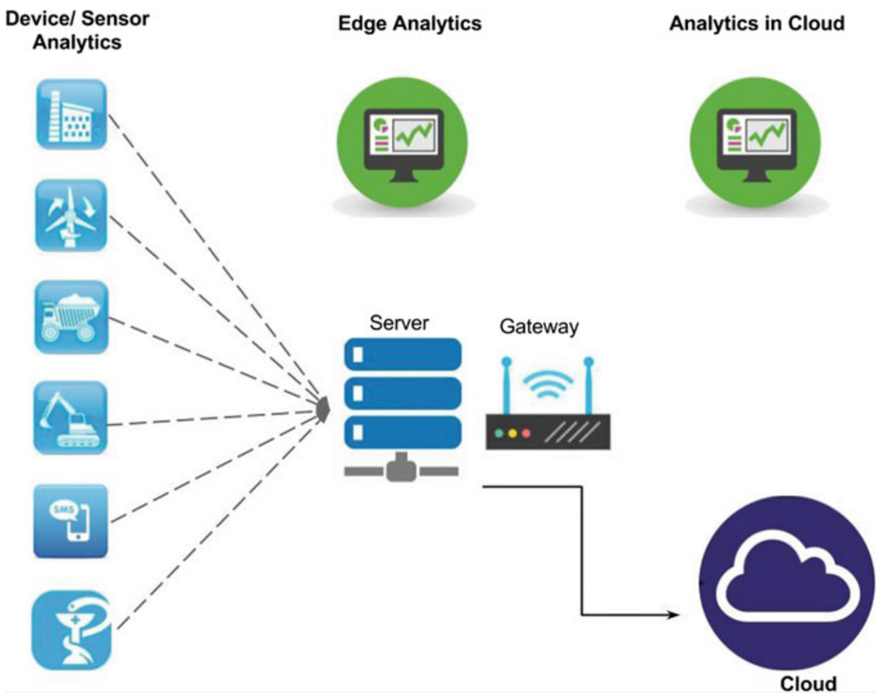


Fig. 6.1 End-to-end Fog–cloud integration architecture

The digitized objects (sensors, beacons, etc.) at the lowest level are generating and capturing poly-structured data in big quantities. The Fog devices (gateways, controllers, etc.) at the second level are reasonably blessed with computational, communication and storage power in order to mix, mingle and merge with other Fog devices in the environment to ingest and accomplish the local or proximate data processing to emit viable and value-adding insights in time. The third and final level is the faraway cloud centres. This introduction of Fog devices in between clouds and digitized elements is the new twist brought in towards the ensuing era of knowledge-filled services. Fog devices act as intelligent intermediaries between cloud-based cyber/virtual applications and sensor/actuator data at the ground level. Here is another representation of Fog Computing as articulated in Fig. 6.2.

6.3 Defining Fog Computing

Technology is accelerating rapidly with nature of work gets done is smarter and challenging. This can be achieved by making your device computationally capable and intelligent to take appropriate decision in IoT. IoT is a great opportunity for making cities as smart cities with a network enabling for the transformation of given

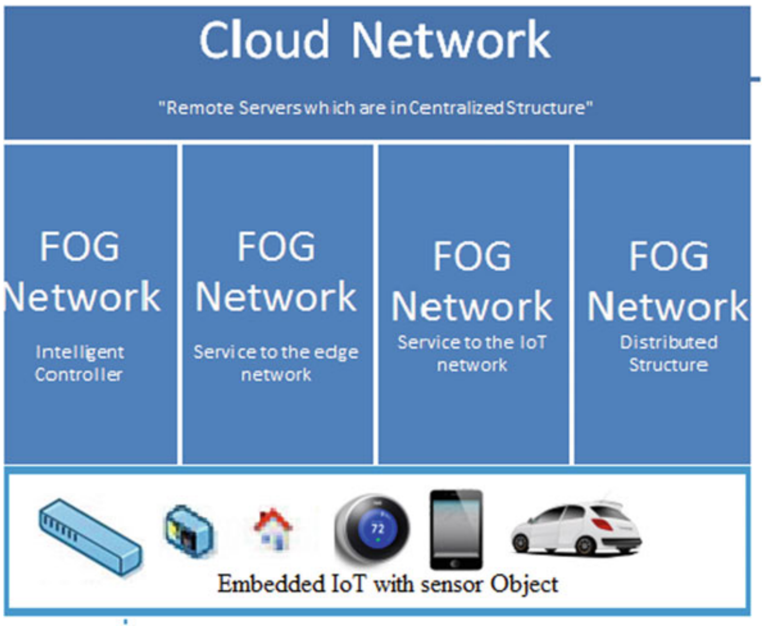

Cloud Network
"Remote Servers which are in Centralized Structure"
FOG Network
Intelligent Controller
FOG Network
Service to the edge network
FOG Network
Service to the IoT network
FOG Network
Distributed Structure
72
Embedded IoT with sensor Object

Fig. 6.2 Fog as the intermediary between physical and cyber worlds

data. The demand for IoT devices with high response and minimal latency in Edge network is increasing rapidly. As per research reported in [1], by 2020, smart devices in use will touch around 15 billion units, providing the service to all these devices is one of the biggest challenges. Cloud service will support IoT by Scaling up and controlling the data generated by IoT. The biggest challenge in cloud technology arises when an IoT environment requires low latency for mission-critical data which require results in a mini- or microsecond. This necessitates the Edge network in the cloud to be located closer to IoT devices. But, providing such big cloud infrastructure nearing smart device is expensive and unrealistic.

Here, a new idea emerges, i.e. making the networking device smarter makes the job simpler for the above problem associated with cloud environments. This mechanism is rising on the concept of the self-driven car which requires massive data of road history and quick response for immediate action. Hence, Edge Computing has come into the picture with tremendous change in the industry but as we know that each advantage has its own disadvantage. Edge Computing brings computation power into networking device but fails when devices fail or sabotage (single point of failure). The fault or drawback of Edge Computing is overcome by a new concept called mini-cloud or Fog Computing in networking Edge of IoT. A small data set or small cloud service has to be located near to IoT, and these small clouds create a Fog node. The concept of Fog Technology was introduced by Cisco in January 2014 as Fog Computing to address the demand for cloud service in IoT. Fog Computing is a networking hub of Fog nodes which perform application processing or providing services to the IoT in a distributed system with minimal latency. Fog Computing plays a major role in fields such as vehicle-to-vehicle communication, Smart Light monitoring system, diagnosing the patient for treatment, chemical industry, oil industry, which expects agility in a special situation. Melioration of Fog nodes in term of performance can be achieved by encapsulating objects such as Scalability, Security, Hierarchy and Agility is a research topic to the world. Further, we will discuss Fog Computing and mainly focusing on the performance which is one of the pillars of Fog node.

The concept of Fog Computing is to bring the process closer to the Edge of the network for analysing and processing the data generated by IoT. It acts as a middle layer between cloud services and IoT devices [2]. Ken Hosac, VP of Business Development at Cradlepoint, defines the concept of Fog Computing as an 'extension of cloud Computing to the utmost Edge of the Network'. Matt Newton, director of technical marketing at Opto 22, suggests that Fog Computing pushes intelligence down to the local area network level of network architecture, processing data in a Fog node or IoT gateway. According to Cisco, the Fog paradigm extends the cloud to be closer to the things that produce and act on IoT data. Each view is different, but conceptually Fog works like a cloud but is not a replacement for cloud.

Cumulating the above definition, one can realize that Fog Computing is a virtual or physical distributed paradigm of resources with computational, analytical, storage and network connectivity which look like a mini-cloud. In Fig. 6.3, Fog Computing is depicted as a class with a collection of functions and parameters. An instance of the class is called object, and Fog node is an object of a class Fog. A Fog node behaves like a mini-cloud and interoperates with various Edge/IoT devices.

In Sect. 6.4, we discuss the criteria, functionality and behaviour of Fog nodes.

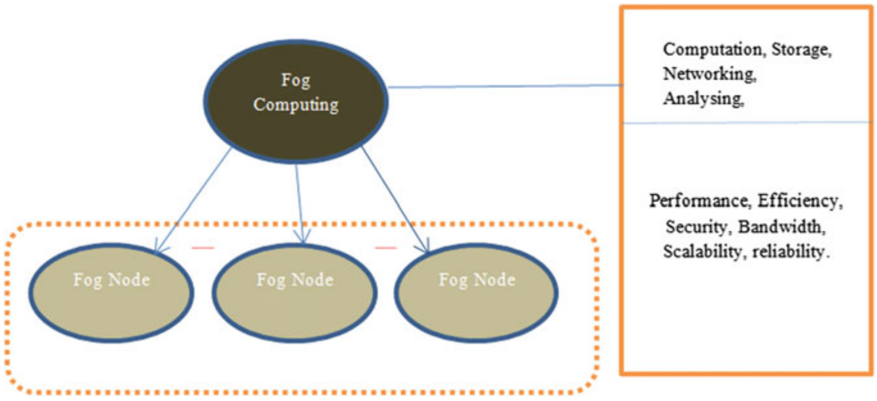

Fog Computing
Computation, Storage, Networking, Analysing,
Performance, Efficiency, Security, Bandwidth, Scalability, reliability.
Fog Node
Fog Node
Fog Node

Fig. 6.3 Fog Computing

6.4 What Are Fog Nodes?

A Fog node is defined as an element for processing the computation by filtering the data which is sent by the Edge or sensor devices. Fog node can be either hardware component or software component which is arranged in a hierarchical structure. Fog nodes play major role in Fog Computing, and the function of the node is defined in an intelligent way which should satisfy the following criteria:

- High response time (RT) in terms of milliseconds.
- Low latency.
- Decision-making (data analysis).
- Storing the critical data.
- Optimal bandwidth.

Fog node are the building block of Fog Computing. Cisco attempts to define Fog node as providing infrastructure to the IoT devices by receiving the data which require RT in a millisecond and forward the remaining data to cloud service for further processing using some protocol such as DDS/RTPs, where Distributed Data Store (DDS) is used for replication in case of failure of any node and Real-Time Transport Protocol is used to transfer the real-time data of any type of stream. Fog node receives the data and performs analysis on streams of data for its time sensitivity. Fog node also provides transient storage for 1–2 h by sending the periodic summary to cloud. Fog Computing by Marin-Tordera [1] discuss different angles such as the producer–consumer layer, embedded system, Computing layer and a system with CPU and storage device.

So, Fog Computing leverages system with collections nodes to achieve the desired result by IoT. Nodes of Fog are also known as Fog nodes. Fog nodes can be any router, wireless access points, switches, host, router, video surveillance camera, Cisco Unified Computing System (UCS). As Bonomi et al. [3] said, the Fog nodes concept is not defined; hence, it opens interpretation with various angles. Fog nodes according to OpenFog [4] are embedded with intelligent algorithms for Computing, storing the critical data and forwarding the data between Edge device to Fog and Fog to cloud with the smart network as depicted in Fig. 6.4. Here,

- *Computation* is the process of Computing the data for the desired result; it encapsulates many operations such as technique/model for offloading data from the smart device, high-performance IO devices and multi-tenant instantiation.
- With respect to *Storage*, accessing the data also impacts on the performance of Fog. Hence, storage tier should accomplish reliability, robust, low power and cost-sensitive devices.
- An *Accelerator* such as Graphics Processing Unit (GPU), Digital Signal Processor (DSP) and Field Programmable Gate Arrays is the units for providing the additional power of processor if required for the critical task.
- In Fog Computing, a *Network* of Fog node should qualify scalability, guarantee delivery and latency sensitive.

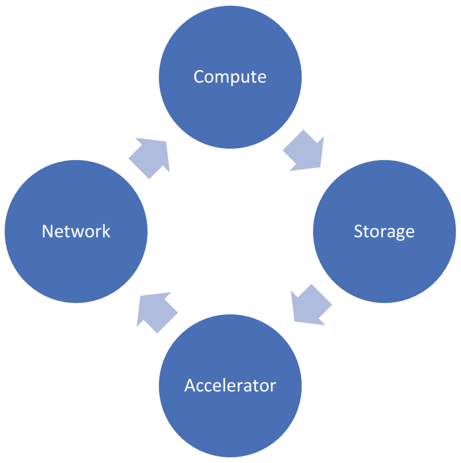


Fig. 6.4 Member function of fog node

Distinct Features of Fog nodes

A cluster of Fog nodes makes Fog to be closer to Edge devices for providing better computation and viable to ease the execution of IoT applications. Its main goal is to improve the performance of the network with low latency (discussed in Sect. 6.6). Considering the application of IoT such as Smart cities, smart vehicle, traffic control, smart building and much more are growing drastically by generating a gigantic quantity of data in terms of a trillion. As we discussed earlier, a Fog node is a small cloud which cannot process the big volume of data. Edge Computing devices or smart network devices filter the data at the receiving level and send quick response expected data to Fog. Fog nodes have the image of IoT application to analyse and compute the data accordingly. When traffic increases towards the Fog, Fog instantiation happens dynamically by a hypervisor. Hence, we can conclude that a Fog node is not the only physical device that can be a virtual node. Fog is composed of the following features to provide better QoS to the client [4], as discussed below.

- Monitoring the emergency requirement: Fog stores the image of the respective IoT which is collaborated to it for a quick response.
- Satisfying the response time criteria: Prioritizing the critical time constraint request.
- Computation in a robust manner: Fog is located in LAN; hence, time delay is comparatively less and the CPU is robust for computation.
- Providing the security wall: Data Protection Mechanism is enabled for the ingress.
- Supporting heterogeneous Edge devices: Fog Computation is distributed in nature; hence, it follows a hybrid structure which gives better efficiency compared to the tree structure.

- Regular interaction with the cloud: Fog updates and monitors the data periodically from the cloud for new images.

A Fog node can be a single processing node or it can be a cluster of nodes which are distributed and follow the decentralized or hybrid structure. In hybrid structure, some of the Fog nodes are interconnected with each other to establish peer-to-peer connectivity, and in distributed structure, each node is a monolithic processor which share distributed database and establish the connectivity. Connectivity between Fog nodes also influences on the QoS of Fog Computing. Without connectivity study, the chapter of Fog Computing will be incomplete. In the next section, we discuss device connectivity and the communication bandwidth.

6.5 Connectivity Technologies

Network connectivity refers to the level and variety of connections that are established between Fog nodes and IoT devices. Connectivity can impact high on the performance since it is intermittent in the rural area or in a remote place. Hence, opting the best Technology for connectivity is also one of the major parameters. Network connectivity of Fog devices can be wired and wireless connectivity. Wired connectivity is typically Ethernet connectivity supporting speed from 10 Mbps to 100 Gbps; links may be copper or fibre links which support different wavelengths and transmission mode. The speed of the network connectivity is specified in Table 6.1.

Here, we are focusing more on wireless technology than wired since wired connection already proved by itself as reliable connectivity but distance, cost, maintenance and natural disaster are some of the weakest points in it. Virtualization and mobile network are the emerging transmission models of data, which manifest new technology to achieve time-sensitive network. Wireless technologies are more flexible and provide mobility compared to a wired network as shown in Fig. 6.5.

The underlying wireless connectivity approaches relating to IoT are now discussed below [5]:

Table 6.1 Rate of data transfer

Speed	Rate of data transfer
ZigBee	250 kbps
Wi-Fi	20–100 Mbps
Bluetooth	25 Mbps
3G	20 Mbps
4G	50 Mbps
5G	20 Gbps
LPWAN	800 Mbps

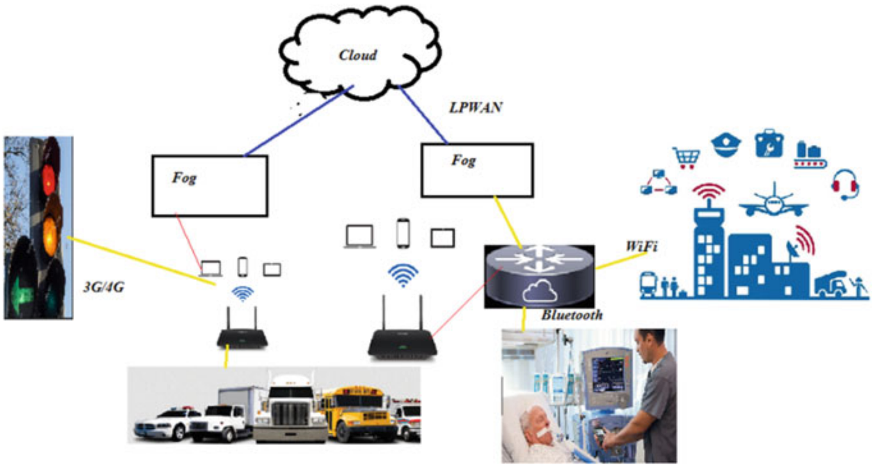


Fig. 6.5 Connectivity technology

a. 3G/4G: These are based on cellular technology which is based on a high speed of around 1 Gbps [4] and lower latency. 3G networks come with anywhere–anytime communication by providing high data rate within Radio Access Service. It paved for video conference, broadband and smartphone. Long-term evolution (LTE) and eNBs make differences in 3G for controlling the radio resources [6]. 4G network is an extension of 3G by establishing high-speed data packet for uplink/downlink which signifies speed of 4G is 10 times greater than 3G. Upcoming network connectivity expected in 2020 is 5G mobile wireless technology which is more suitable for IoT devices. It is most power-efficient technology than any other. It supports superfast download to IoT for small data requirement and crowned with the low latency of <1 ms as quoted by matt's in Qualcomm.
b. Bluetooth: It is a wireless technology specially designed for short-range communication between the devices. New Bluetooth mesh technology extends its service by improving scalability, security, reliability and also support broadcast topology. The strength of Bluetooth is less energy consumption which helps to avoid battery drain. This technology is appropriate for small business such as tracking oil, monitor construction equipment or home appliances. At present, it covers the radius about 100 m and the data as to pass through a gateway for communication.
c. NFC: Near field communication enables peer-to-peer connectivity which requires 0.2 s for establishing a connection is used for social networking. Its usage is basically to communicate between smart devices and share the data which is being close to each other with a distance in terms of inches. Its ability is to provide the connectivity for unconnected objects. IoT can leverage the NFC in smart grid for providing the interface between the nodes. Comparatively, it is cost effective with minor security threats.

d. ZigBee and Z-Wave: They are WPAN technologies which are low power consumption and low cost; the area ranges up to 100 metres and controls mostly used in home automation. Both are IEEE 802.15.4 and mesh network technology, suitable for Personal Area Network (PAN). With respect to throughput, ZigBee is in the race but falls back in energy consumption comparatively more than Z-Wave.
e. Wi-Fi: It is a WLAN technology that covers smaller geographical areas local to campus or building depends on a number of the access point. The rate of data transfer ranges from few megabyte to multiple gigabytes. The most common standards are IEEE 802.11a, b, g, n and ac. IEEE802.11ac is appropriate for IoT scenario which requires low power consumption; due to the larger wavelength of the 5 GHz band it can penetrate through any obstacles results in minute signal loss and speed up to 500 Mbps.
f. LPWAN: Low Power Wide Area Network (LPWAN) Low data transfer, low cost and greater power efficiency to cover the large area. LPWAN provides the longer connectivity with low power consumption for a longer period of time. There are various LPWAN technologies available such as M2M, weightless, LoRA/LoRaWan and Ingenu. Those technologies are varied with respect to speed, scalability, low cost, the rate of data transfer. LPWAN has emerged with varying feature varying with uplink, downlink and speed with following technologies:

   - When the system requires connectivity with the limited downlink of small data, then choose *SigFox*.
   - The connectivity between sensor and gateway with prime uplink then *LoRa* will be suitable for the system.
   - *RPMA* is suitable for high-speed connection with superior uplink and downlink.

Choosing the suitable network connectivity also plays a major role to achieve the goal towards better performance.

6.6 Structure and Behaviour of Fog Computing

In the previous section, we discussed Fog Computing, Fog nodes and their connectivity. Combining those concept helps to design the structure and write the algorithm which is discussed as a workflow in stepwise in this section. Fog is a distributed Computing system which is located in the middle layer between IoT and cloud. Edge–Fog cloud architecture [7] proposed by Nitinder Mohan is almost nearing the concept proposed by Cisco [4] regarding the structure of Fog environment. Considering their inputs, Fog Computing as depicted in Fig. 6.6 suggests that Fog Computing is a layered architecture following a hierarchy. The top layer is cloud system which provides the service application for computation, resource sharing, storing the data as ubiquitously and connected to the high-speed network.

It is a data centre where data is handled by Big Data; cloud follows centralized infrastructure which supports complex topology and scalability but fails in easy access in terms of quick response due to locality. This drawback is overcome by Edge Computing. Edge devices directly connected to IoT device as networking devices provide services with low latency since it is directly connected to smart IoT. It may be a switch, mobile, system and gateway with Computing and storage services. Since it is a hardware device, it fails to support heterogeneity which makes collaboration complex. Fog Computing overcomes the drawback of cloud and Edge Computing to make the IoT system robust. Fog Computing is looking like a reflection of the cloud as shown in Fig. 6.6 whose performance can be brightened with new emerging technologies like SDN, NFV and management service (host/ Node, resource, structural arrangement, data analysis).

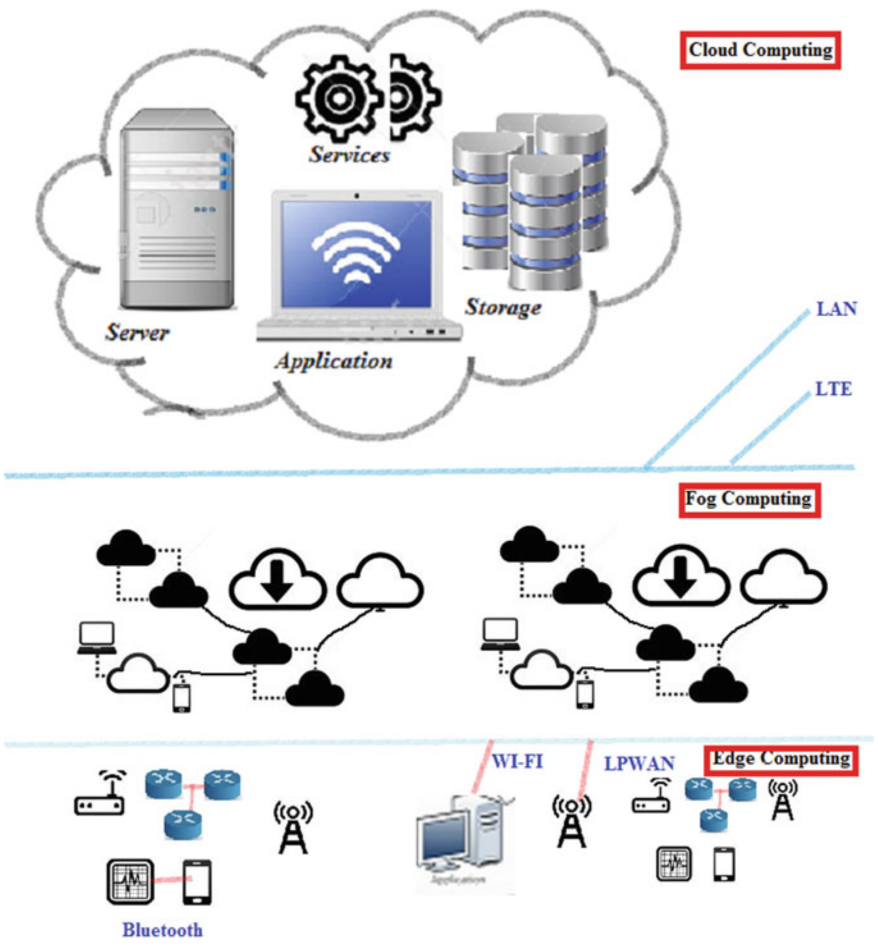


Fig. 6.6 Architecture of Fog Computing

Algorithm

Step 1: Role of Edge Device

- A smart device will send a request to Edge devices. The Edge device is a gateway between cloud and IoT devices.
- Edge device decides whether to send data to cloud or Fog using data analysis algorithm.
- Filtration model or logic resides in Edge device to make such decision and forward the data to appropriate environment.
- If the expected time duration is short in terms of seconds, it forwards to Fog else to cloud.

Step 2: Role of Fog node

- If the data request is sent to Fog, it receives the request by the Edge device.
- IoT Application is deployed on Fog. Fog nodeprocesses the request and replies to a smart device.
- The content table which contains the IoT Application information is updated.
- Fog sends the data summary to the cloud and updates its table content.

Step 3: Role of Cloud

- If the data request is sent by Edge/Fog device, then cloud processes the data and sends the response to the smart device.
- Cloud also analyses the request sent by Fog for periodic updating.
- Cloud also sends rules to Fog if any changes to be done in Fog.

The working scenario is good enough for a limited transaction. Assume if the number of smart devices increases, then we require a cluster of virtual nodes to handle the scenario and a controller to monitor them. Virtualization will introduce by itself for the proliferation of smart device and Fog nodes to utilize the resources efficiently. In our chapter, we also propose about interconnected Fog. Interconnected Fog, communicates with each other for exchanging the data if the round trip is less than cloud round trip. We assume that round trip between Fogs is lesser since the traffic congestion is minimum compared to the cloud since Fog is closer to smart devices than cloud. We will discuss the connected Fog with an example in a case study.

Our discussion encourages to adopt the Fog paradigm in the IoT scenario but some question arises. What happens when connected Fog node fails? How does Fog handle traffic congestion? Is Fog the ultimate solution to IoT? After a survey, we found the following answers to those questions.

1. What happens when connected Fog node fails?
   Solution: Resilience, it can be achieved by hybrid topology or by duplication of the node or by virtualization.
2. How does Fog handle traffic congestion?
   Solution: Distribute Computing or Multi-tenant Environment or Instantiation.

3. Which parameter decides the sensitivity?
   Solution: Parsing the metadata by the Fog node provides the sensitive information in term of response time.
4. Which is the best topology does Fog follows?
   Solution: Hierarchical or hybrid structure of node is the best topology which provides better connectivity between the Fog nodes.
5. Is Fog the ultimate solution to IoT?
   Solution: Agility and Fog sever with the control system (support hardware and software).

The above questions are the considerable inputs for *performance*, and the solutions are *characteristic* of adopting the technique for Fog Computing. These two concepts are discussed in Sects. 6.7 and 6.8, respectively.

6.7 Analytic and Performance Parameters for Fog/Edge Nodes

In Fog Computing, Fog nodes are the devices of Fog which process the requests. Thus, Fog nodes can be routers, switches, gateways and any smart devices which compute, store and transmit data. Improving their efficiency improves performance; the real opportunity lies in configuring the nodes and optimizing their performance.

Performance, efficiency, and latency are the key factors for success of Fog Computing. Performance of Fog Computing is achieved conceptually as shown in Table 6.2. As we discussed in Sect. 6.4, Fog node is a device which is a composition of computation, storage and networking service. Time complexity is influencing all those services such as data accessing and manipulation in the storage system, the network speed and time to process the data by computation service. In Fog, time complexity is known as latency. Latency is the time taken to transfer the data between the nodes (Smart device, Fog node, Edge node, networking devices), and the latency influenced by parameters such as bandwidth, delay, the rate of data transferred, throughput and delay. Table 6.2 exhibits the parameter to measure performance in Fog Computing conceptually.

Performance measurement and related parameters are big ocean of which we cover only a part of it by focusing on traffic flow, response time, low latency and protocols as depicted in Fig. 6.7. Cost, energy consumption, scheduling and storage are some of the parameters which present some of the added challenges to measure the performance of Fog Computing.

Table 6.2 Conceptual performance parameter

Entity	Characteristics/attributes
Computation	Data analysis, compression, scheduling, filtration
Storage	Data transaction, query language, updating, recovery management
Network	Traffic management, topology management, load balancing, connectivity, conservation of bandwidth

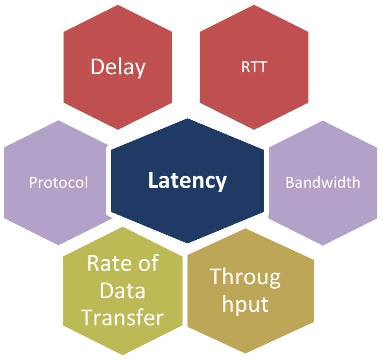


Fig. 6.7 Performance parameters

Considering the parameters for better performance, Fog nodes should possess the above attributes. For this, numerous approaches have been proposed, e.g. in [4, 8–10]. Many associated discussions focus on virtualization with intelligent programming to make Fog nodes more efficient and robust to handle the growing IoT situations. Some relevant factors are discussed below:

- Latency: Low execution time of a task is called latency; this can be achieved with a way of handling packet as in GTP gateway [11] and employing new paradigms such as SDN or NFV.
- Response time: The duration of response received and request sending is response time (RRT); bandwidth plays a major role for quick response time [12].
- Bandwidth: The volume of data carried is bandwidth. Edge analytics, stream mining [12] and 5 g connectivity are used to improve the bandwidth.
- Protocols: Protocol is a prominent role to provide state of information, queuing system, management of devices, gathering data and security (XMPP, AMQP, SNMP, CoAP).
- The rate of data transfer: The time taken by the data packet to travel a given destination.
- Delay: This is the time taken by a process while waiting in a queue.
- Throughput: This refers to amount of data processed for a given time.

6.8 Performance Enhancement Techniques

It is generally accepted that Fog nodes should adopt the following strategies for higher performance:

- Virtualization techniques
- Hierarchical models
- Software defining.

In Network Function Virtualization (NFV), hypervisor creates multiple virtual Fog nodes which help to use resources and handle the traffic. The virtual Fog nodes share the resource in an efficient way and can process the different request parallel or can distribute the computation among them to achieve high response time. The way of communication or interaction is defined with the help of a hierarchical model. A hierarchical model may be tree topology or star topology or Fog-interconnected topology or maybe the combination of differing structure which helps to define our paradigm in an easily approachable way. An important technology for networking is SDN which makes Fog nodes more intelligent, analytical and efficient by programming them as custom based. SDN decouples the forwarding and control plane of the device which also inclusive of management plane in the device. Control plane is intermediary between data plane and forwarding plane; this concept helps to write the decision-making algorithm, management policy, route discovery, etc., in the data plane. Many frameworks have also emerged that are mainly concentrated towards managing the network by programming model and resilience of the infrastructure using virtualization. Virtualization is a concept used for creating multiple virtual instances of Fog. This can be accomplished by using Network Virtualization Function (NVF) and creating a programmable network using SDN by managing the nodes by structural arrangement using hierarchical model. The following sections present a discussion on the three technologies listed above.

6.8.1 Network Function Virtualization (NFV)

This refers to creating a new instance of an object with the properties of class virtually. Network Function Virtualization (NFV) brings benefits to enterprises by creating virtual storage, computation and networking to reduce the expenditure by providing scalability. Deploying NFV helps to achieve load balancing, scaling, network upgrading, optimize power consumption and reduce the cost in terms of Capex and Opex. NFV is a software application which creates the virtual systems by cloning system behaviour for different computation. Provision for multi-tenant from a different vendor which provides the openness may need to integrate with Fog server, hypervisor and virtual appliances. NFV relates with SDN to orchestrate the network system. Fog server is a centralized system which conjugates with distributed Fog nodes. Each Fog node works independently with instruction provided by Fog server. NFV also facilitates resource sharing between the nodes, orchestrating between Fog nodes and restructuring when nodes shrink and grow. It not only creates the computation class but also virtualizes the storage for resource accessibility and routing devices such as switches. Hence, virtualization mends the performance of latency and throughput towards progress.

The roles of NFV technology in Fog Computing are:

- Sharing resources: Consolidating the resources or distributing the resources can be achieved by NVF.
- Reducing power consumption: Location optimization, storage structure and workload consolidation are the features of power management.
- Replacing traditional appliances with hardware: NFV supports hybrid network with hardware and a virtual network appliance. It also manifests by moving from error-prone structure to stable structure that is achieved by the feature of resilience.

6.8.2 *Software Defining*

NFV virtualizes the hardware, whereas SDN controls the services offered by the programs. SDN is an intelligent technology in the network that controls the forwarding devices and manages the networking policy dynamically. SDN is designed with four parts and three layers. It follows the layered architecture as shown in Fig. 6.8. The top layer of SDN contains application services call it as Northbound (NB) layer which contains security, flow rules, intelligent action services, a middle layer of SDN will orchestrate the networking operation, control and manage the data is known as the controller. There are many controllers which are classified

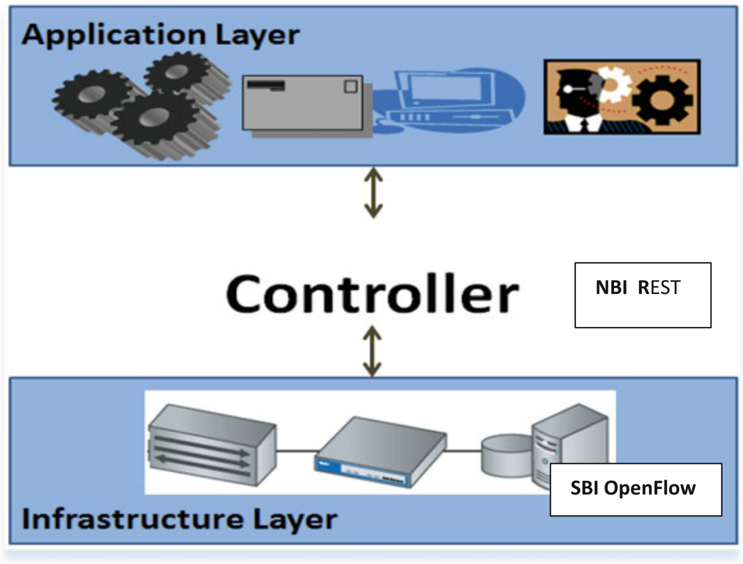


Fig. 6.8 SDN layers

based on their efficiency in terms of throughput, round trip and bandwidth consumption; some of the controllers are ONOS, ODL, Floodlight and many more. The controller controls the routing devices through OpenFlow channels. OpenFlow helps to transmit commands, request, status and statistic and filters which helps to filter the data to protect against attacks such as MitM attack. Broom Filter, proposed in [13], performs filtration to avoid the malicious data in the packets.

SDN leverages the integration, aggregation and compression by incorporating NB service in Edge Computing. The smart way of processing the data requires for fussing and diffusing the information. It enables the module to integrate with any updates to the existing base code. Load balancing, partitioning, topology discovery, host remote access are some of the features of NB; it also expands the module with user required service module.

Southbound (SB) layer prime operation directs the networking device to forward the packet. Here, Edge devices such as gateway, switches, hub and router can perform computation by filtering the data in the metadata which is available in the packet and make a routing decision. If networking device fails to perform the action on arrived packed, then controller commands the switches for further action and update the routing table.

The roles of SDN controller in fast-expanding Fog Computing approach are as follows:

- Fog Orchestration: SDN controller will perform Fog orchestration for monitoring and controlling the data stream in Fog. The Controller is the brain of SDN which performs network operating system. Orchestration can help to allocate the resources, scalable and balance the nodes by structural arrangement.
- Traffic Management: Load balancing and route optimization can be achieved in traffic module in NB of SDN. When traffic congestion in networks occurs, then offloading the loaded nodes or diverting the packet to optimize the route.
- Resilience: Fog node can create a mesh by linking the nodes for intercommunication laterally within Fog tree. When any node fails then hypervisor provisions the virtual node for achieving fault tolerance.
- Proactive model for time sensitivity: SDN supports two models, proactive and reactive models. Fog fits into a proactive model which stores the node location and routing decision in the node (statistical report) which helps to compute quick support time sensitivity.
- Dynamic policy-based management: An incoming service request policy from the repository for managing the services varies based on constraints or customer requirement. SDN manages repository and integrates new policy for suitable action.

6.8.3 Hierarchical Models

The structural arrangement of nodes varies with a parameter of priority, efficiency, complexity and capacity. This model helps to aggregate or distribute the data. The hierarchical structure not only benefits Fog node management and associated database but also helps to direct the traffic to control and reduce the congestion.

In this context, peer-to-peer communication requires limited bandwidth comparatively but increases the delay and traffic towards Fog. Hence, Fog Computing follows the hierarchical model which supports quick response, sharing a resource, distributed Computing, hazardous detection and scheduling.

The main purpose of hierarchical models is represented in Fig. 6.9; the four components are briefly elaborated as follows:

- Resource Allocation: The resource allocation is done at the top level of structure based on inputs such as priority or critical time.
- Load Balancing: A job broker in Fog node allocates the jobs to the node which may be the root of the cluster or individual node. Root node will distribute the job to Edge devices either at the smart device and organized in a hierarchical structure.
- Quick Response: Responsiveness depends on aggregation of computation time, traffic, data offloading, connectivity and structure of node association. In the hierarchical structure, the speed is high due to its distributive service helps to achieve a quick response [14].
- Collaboration: Task distribution, resource sharing and resilience help to achieve collaboration. In a hierarchical model, security management can be achieved with a cluster of Fog node and the scan process detects the hazards in the network [15].

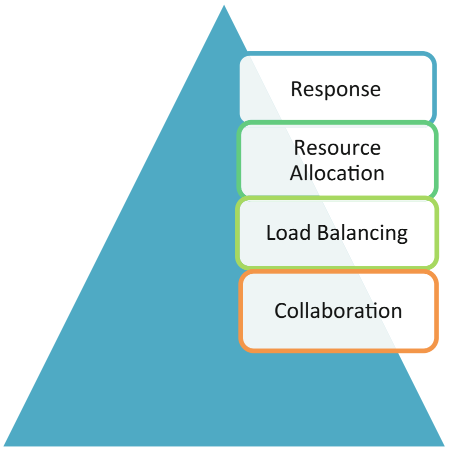


Fig. 6.9 Characteristics of the proven and potential hierarchical model

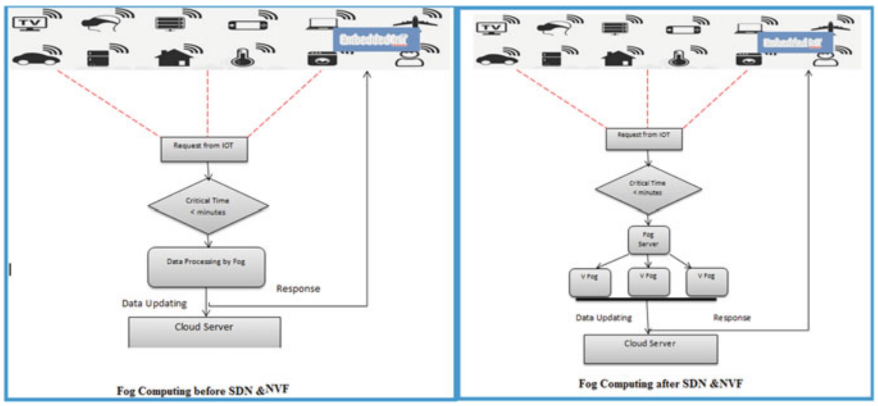


Fig. 6.10 Software defines Fog Computing

Figure 6.10 shows the difference between Fog Computing before and after deployment of SDN and NFV. It addresses the performance issue by spanning virtual switches and Fog nodes and also centralizes and programming the forwarding control from network model by deploying SDN control on Fog server which increases the rate of data transfer, throughput and minimizes the delay.

6.9 Fog Computing Use Cases

We consider a scenario of Smart City as IoT environment with sensors for traffic detection (TD), IoT for health care (HC), VANET and mobile communication (MC). Each data is processed by different Fog nodes where appropriate IoT deploys respectively. In this situation, when a vehicle has an accident and an occupant gets hurt, the patient can get treatment with minimal time as VANET sends critical data to Fog such as nature of accident and location where it happened. This data also reaches the TD system that helps to direct traffic on different roads and send messages to HC for treatment through Fog-interconnected nodes. Let us discuss some more case studies such as video surveillance and vehicle detection.

Video Surveillance

Security threats arise in all situations, and monitoring security manually is an unachievable goal. IoT provides a solution with the help of closed-circuit television (CCTV) in which real-time data is sent to the network. Consider a situation where rubbery takes place in a locked house pictured in Fig. 6.11. If there is a constant image, then data is moved to cloud else if any alert message passes, then data is moved to Fog. Fog Computing contains smart router or hub which speed up to computation by the root node. SDN plays a crucial role in time sensitivity situation by establishing the route to specify the destination using proactive model, here to

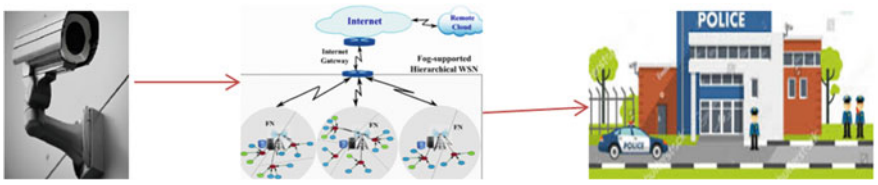


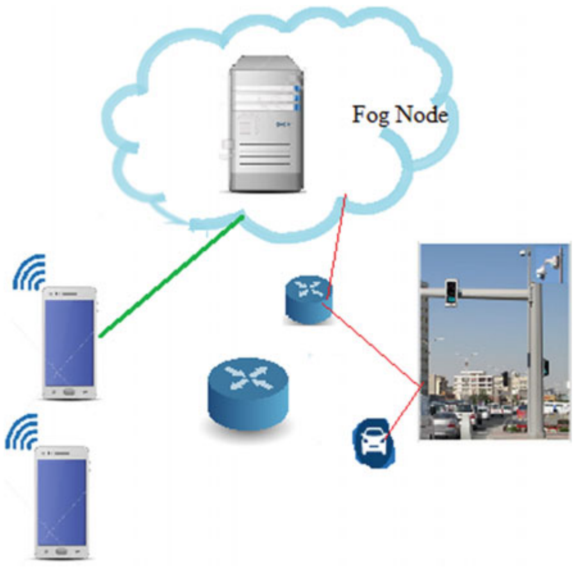
Fig. 6.11 Video surveillance

the nearest police station or to the owner if he is nearby using decision-making algorithm. Through SDN, resource sharing and virtualization are achieved by NFS. Edge devices connected to the IoT devices which send the data from the sensor have been processed by Edge node or Fog node. If the data volume is big, then the hierarchical model of distribution is adopted. In this example, heterogeneous nodes are communicating which requires more bandwidth than homogenous nodes.

Vehicle Detection

In this scenario, hierarchical model approach plays a vital role, by detecting the missing vehicle. The scenario is summarized in Fig. 6.12. The root device sends data to its child node to sense or capture the missing vehicle with few details like the image, type and vehicle number. For example a missing car is indicated by the red line and after detecting by Fog node sends a signal to owner represent in green line in the Fig. 6.12. Edge device filters the data by using the parameter sent by the root node. Once it is identified by an Edge node, its sends to its hierarchy and the SDN controller updates the flow table. The pictures and detailed information of the

Fig. 6.12 Vehicle detection



vehicle are sent to the nearest station for tracing the vehicle. The IoT devices such as traffic light, CCTV and smart vehicles act as a sensor and send the signals to the Fog nodes (marked red lines are sensor devices), and if data is found the credential, then the authority sends the response to the centre (Fog updates the historical data to central data centre cloud).

6.10 Summary

With billions of connected devices joining in mainstream Computing, the future of Edge or Fog Computing is scintillating indeed. There are a number of industry use cases coming up for Fog Computing and analytics. All kinds of real-time analytics and applications are to be defined and designed through Fog Computing. With more number of devices in a Fog Computing environment, the operational complexity, the performance issues and the security problem are bound to escalate. There are enabling architectural patterns, simplifying frameworks, best practices, automated tools, accelerators and other contributing things in order to make Fog Computing and analytics pervasive and persuasive. IT professionals are working overtime in order to identify the barriers and surmount them in order to make the Fog idea visible and viable.

The traditional, closed, inflexible and expensive network infrastructures are transitioning towards open, flexible, virtualized, containerized and affordable network infrastructures. With the faster adoption and adaption of software-defined networking (SDN) and Network Function Virtualization (NFV) technologies and the associated tools, the network phenomenon is bound to see a series of paradigm shifts in the years ahead. Multiple and heterogeneous devices ought to be found and connected to capture, cleanse, store and crunch data being emitted by millions of digitized items and implantable sensors, fixed, portable and nomadic actuators in our personal, social and professional environments, the SDN and NFC-sponsored Edge or Fog Computing is the key requirement. However, with such massive number of connected devices and digitized assets in place, the performance Fog Computing and analytics have to be substantially improved in order to be people-centric.

In this chapter, we discussed the QoS parameter of Fog Computing and tuned our discussion to learn more about performance measurement of Fog Devices and connectivity. Performance of Fog is composed of many key values such as latency, bandwidth, connectivity, storage security. Approaching the new technology such as SDN and NFV is the more suitable solution for Fog sustainability. We also discussed some of the protocol and algorithm which serve as chain service to Fog Computing.

References

1. Tordera EM, Masip X, Garcia Alminana J, Farre J (2016) What is a fog node? a tutorial on current concepts towards a common definition, networking and internet architecture (cs.NI). Cornell University Library, cite as: arXiv:1611.09193v1
2. Iorga M, Feldman L, Barton R, Martin MJ, Goren N, Mahmoudi C (2017) The NIST definition of Fog Computing, NIST SP 800-191 (DRAFT). www.csrc.nist.gov/csrc/media/publications/sp/800-191/draft/
3. Bonomi F, Milito R, Zhu J, Addepall S (2012) Fog Computing and its role in the internet of things. In: Proceedings of MCC'12, Helsinki, Finland, pp 13–16, 17 Aug 2012
4. OpenFog Consortium (2017) OpenFog reference architecture for Fog Computing, While paper published USA OPFRA001.020817, Feb 2017
5. i-Scoop (2017) The internet of things in 2017: trends, technologies, market data and evolutions. www.i-scoop.eu/internet-of-things-guide/connectivity-networks-Fog-Computing-internet-of-things/. Accessed 22 Dec 2017
6. García-Pérez CA, Pedro M (2017) Experimental evaluation of Fog Computing techniques to reduce latency in LTE networks. Emerg Telecommun Technol (inpress). http://dx.doi.org/10.2759/54788, http://dx.doi.org/10.1002/ett.3201
7. Mohan N, Kangasharju J (2017) Edge-Fog cloud: a distributed cloud for the internet of things computations. In: IEEE Cloudification of the Internet of Things, arXiv:1702.06335v2
8. Westbase.io (2016) Fog Computing versus cloud Computing what's the difference, www.westbaseuk.com/news/Fog-Computing-vs-cloud-Computing-whats-thedifference/. Accessed 30 Nov 2017
9. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh S, Buyya R (2016) In: Buyya R, Dastjerdi AV (eds) Fog Computing: principles, architectures, and applications, pp 61–75. Morgan Kaufmann, New York. Cite as arXiv:1601.02752
10. Ai Y, Peng M, Zhang K (2017) Edge cloud Computing technologies for the internet of things: a primer. Digital Commun Netw, Chongqing University of Posts and Telecommuniocations. Production and hosting by Elsevier B.V, July 2017
11. Yi S, Hao Z, Qin Z, Li Q (2015) Fog Computing: platform and applications. In: Third IEEE workshop on hot topics in web systems and technologies (HotWeb), pp 73–78
12. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for vm-based cloudlets in mobile Computing. IEEE Pervasive Comput 8(4):14–23
13. Li C, Qin Z, Novak E, Li Q, Senior Member (2017) Securing SDN infrastructure of IoT–fog networks from MitM attacks. In: ICSEA 2017: The twelfth international conference on software engineering advances. ISBN: 978-1-61208-590-6
14. Jiang Q, Tang R, Liu P, Qiu Y, Xu H (2014) Research on dynamic data fusion algorithm based on context awareness. In: Proceedings of the 2014 IEEE international conference on progress in informatics and computing. IEEE, pp 529–534. https://doi.org/10.1109/pic.2014.6972391
15. Shah-Mansouri H, Wong VWS (2017) Hierarchical fog-cloud computing for IoT systems: a computation offloading game. Cite as: arXiv:1710.06089v1 [cs.DC] Oct 2017
16. Hirve R (2017) What is Fog Computing? why Fog Computing trending now? https://www.linkedin.com/today/author/rhirve
17. Byers CC, Wetterwald P (2015) Fog Computing distributing data and intelligence for resiliency and scale necessary for IoT: the Internet of Things. In: Symposium Ubiquity, pp 1–12. https://doi.org/10.1145/2822875
18. Peng G (2003) CDN: Content Distribution Network, Technical Report TR-125. Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY. http://citeseer.ist.psu.edu/peng03cdn.html

Chapter 7
Mechanisms Towards Enhanced Quality of Experience (QoE) in Fog Computing Environments

P. Beaulah Soundarabai and Pethuru Raj Chellaiah

Abstract The Fog or Edge computing emerges as one of the important paradigms for setting up and sustaining smarter environments across industry verticals. Our everyday environments are being meticulously advanced and accentuated through a bevy of edge and digital technologies and tools in order to be situation-aware and sophisticated. On the other side, we have powerful Cloud environments contributing as the one-stop IT solutions not only for business automation but also for people empowerment. Compared to the number of prospective Cloud environments, the number of Fog environments is going to be quite large with the availability of billions of connected devices. The scope of Fog environments, which are being touted as the most crucial for empowering people and in their everyday activities, is bound to escalate in the days to unfurl. The immediate challenge for Fog environments is to drastically enhance the quality of experience (QoE) for users. Academic professors and industry professionals have come out with a number of solution approaches and algorithms. This chapter is being specially prepared and presented in this book to tell all about the role and responsibility of Fog computing environments, the unique use cases and the various challenges, etc. Furthermore, the significance of QoE is described in detail and how that requirement can be attained by smartly applying various proven and potential technologies. This chapter also aims to motivate the readers and researchers to dig deep into this new critical requirement to unearth pioneering solutions towards enhanced QoE.

P. B. Soundarabai (✉)
CHRIST Deemed to be University, Bangalore, India
e-mail: peterindia@gmail.com

P. R. Chellaiah
Reliance Jio Cloud Services (JCS), Bangalore, India

© Springer International Publishing AG, part of Springer Nature 2018
Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_7

7.1 Introduction

Applications of the Internet of Things (IoT) are rapidly scaling up in almost all major domains of life due to the rapid inventions and development of networking technologies. Dynamic data communication in the real-time and delayed service defies the IoT applications to be placed on Cloud servers. Fog computing is an evolution of Cloud computing that extends the Cloud services to the IoT devices at the edge of networking. We have introduced IoT devices in each and every aspect of our lives, and our day-to-day activities are monitored by the relevant technologies; Cloud-based utilities take an enormous time to store and analyse this huge volume of data. IoT Cloud-centric architectures have only two layers such as the sensor and the Cloud layers. These two layers require higher sensors-to-Cloud bandwidth and complete and continuous connectivity. Device-to-Cloud communication is not much faster when comparing the essential business requirements and connectivity costs spent [1].

Several researchers have proposed an intermediate layer between the sensing resources and the Cloud-IoT architectures and named this middle layer as Fog computing. To analyse and act on this diversified volume of data, Fog nodes are used, in the Fog layer, that are generally network routers, switches and gateway servers. Fog computing is also known as Edge computing with the following advantages:

- It reduces the network congestion and load by ensuring the on-time delivery and services.
- It takes decisions and controls the IoT devices based on the action policies.
- It stores the data in the Cloud servers for later usage and for further big data analysis.

One difference between the Cloud and the Fog paradigm is that Cloud data stores are generally geographically far away from the IoT devices, whereas the Fog nodes are geographically distributed and available very close to these devices. This arrangement helps to reduce the round-trip time and increase the data processing by increasing the various storage and analytical resources availability.

Fog computing handles the variety, volume and velocity of data, generated by the IoT devices, with the new computing paradigm. It majorly reduces network bandwidth by reducing the amount of data travelling between and through the Cloud networks. It also avoids the cascading failure of the Cloud system by analysing the data close to the sensing devices themselves. Security measures of data are taken care of very well by monitoring and protecting them from unauthorized access and analysis.

Fog computing is well placed and geographically distributed gateway nodes near the edge of the network of dense IoT sensors and actuators. But the application-specific and design-related issues have to be addressed to get the best of this Fog decisions to the real-time usages. Integration of Fog and Cloud is also in the research field. There are different policies required to obtain the expected outcome in Fog; Quality of Service (QoS), situation and resource-aware application are also proposed [2]. As much as QoS is reviewed and revised, there should be researches on Quality of Service (QoS) also. QoE is the measurement of the services offered by the Fog nodes, and it is purely customer-centric measurement. There will be a vast difference of opinion and satisfaction level in the real-time environment, due to heterogeneous users of different services of the Fog computing. QoE is the overall customer satisfaction with the products or the services, and it's purely user dependent and in Fog computing. Although QoE improves the data processing time, quality of the network, proper resource utilization, etc., measuring QoE with the user's interests with various services varies from one to another and so developing QoE-aware policy models for Fog is a big challenge.

After deploying the system in place, if the QoE is evaluated on the application, it would create a number of complications that will lead to modifications in the system based on the feedback of QoE. On the other hand, it is also very difficult to evaluate the application and then deploying it, which satisfy the needs of user's interests. User feedback and the QoE of the specific application should continuously be monitored to have a better QoE ratio.

This chapter analyses several user's interests and expectations which can drastically impact the QoE. Required resources, data accessing, processing time and quality of analysis are some of the major User Expectation Metrics (UEM). Prioritizing a user application is based on this metric. Initially, though, this chapter briefly looks at the characteristics and some general applications of Fog computing, for the sake of completeness.

7.2 Key Characteristics of Fog Computing

Fog computing is being touted as the next-generation computing paradigm to solve some of the unique problems that are not being attended by Cloud computing. This section illustrates some of the distinct competencies of Fog computing.

- **Heterogeneity—**The device ecosystem is growing fast. It is anticipated that there may be billions of connected smart devices in the years ahead. The participating and contributing devices, machines, instruments, equipment, utensils, robots, consumer electronics, etc., for the Fog conundrum are going to be heavily different, distributed and decentralized. Also, these participants are being fitted with increasing compute, storage and networking power. Therefore, Fog computing is a highly virtualized and shared platform that provides compute, storage and networking capabilities. The services being provided by Fog environment devices are going to be disparate.
- **Edge location**—The Fog devices are being found at the edge of the network. That is, we have self-surroundings and situation-aware devices to form smart Fog environments. Devices are being networked in an ad hoc manner to form a

dynamic cluster of devices to work out bigger and better things and to provide real-time processing and decisions.

- **Location and Context Awareness**—As indicated above, the devices in the Fog environment know their environment details. Also, the activities going on, the presence of people, and their timely and spatial needs, etc., are being captured by Fog devices in order to come out with various unique services, including monitoring and reacting to the ambient environment.
- **Low latency**—There are several new applications and services emerging and evolving fast and that is the main reason for the surging popularity of Fog computing. Especially real-time applications are the most mandated ones for the leverage of Fog computing. That is, Fog environments are highly attractive for low-latency and time-sensitive applications. The widely used applications include distributed gaming, video streaming, and virtual and augmented reality.
- **Geographical distribution**—As we know, Cloud computing environments are typically centralized, consolidated, virtualized, containerized, automated and shared environments. On the other hand, Fog computing environments are distributed, many, environment-specific. Homes, hospitals, hotels, airports and other important junctions are being empowered with Fog devices to form competent Fog environments for devising and delivering new-generation applications. Inside advanced smart and connected cars, a kind of miniaturized Fog computing environments is being worked out in order to sharply enhance the experience of car occupants and driver.

As indicated elsewhere, sensors and actuators are the most prominent and dominant entities and elements in our everyday environments. Sensor and actuator networks are monitoring various environments, and the physical, mechanical and electrical systems and assets are being stuffed with sensors in order to participate in the mainstream computing. Furthermore, as far as the number of devices participating in a typical Fog environment is concerned, it is going to be huge. Fog devices are also seamlessly getting connected with mobile devices in order to be comprehensive in their actions and reactions.

In short, Fog computing is an attractive model for real-time data capture, storage, communication, processing and mining, knowledge discovery and dissemination, and actuation.

7.3 Fog Computing Applications

Having understood the unique properties of Fog computing, industries and organizations are unearthing a variety of applications—some of these are explained below.

7.3.1 Smart Grids

Smart grid is one of the prime applications where Fog computing has been fully utilized. Depending on varying demands for energy, low-cost and its achievability, smart grids can easily switch to other energy supplies such as winds and solar. The Edge or Fog devices collect all the local information and collectively take the real-time decision in order to enable smart grids to switch over to alternative energy sources. Also, Fog environments are being fitted with dashboards to give 360° view of what is happening in the environments.

7.3.2 Smart Traffic Lights

Fog devices smartly enable traffic signals to open lanes on observing flashing lights of the ambulance. It determines the existence of bikers and pedestrian and evaluates the distance and speed of the nearest vehicles. On detecting any kind of movements, cameras and sensors immediately turn on to capture and transmit. Smart lights support as Fog devices synchronize to forward warning signals to the nearby vehicles. The communication between the vehicle and access points are being improved with the arrival of 3G and 4G, powerful Wi-Fi, roadside units and smart traffic lights.

7.3.3 Self-maintaining Trains

A train ball-bearing monitoring sensor observes the changes in the temperature level, and any disorder automatically alerts the train operator and builds maintenance departments, thus helping to avoid dangerous disasters.

7.3.4 Wireless Sensor and Actuator Networks (WSAN)

The WSNs were originally intended to increase battery life by consuming less power. Actuators work as Fog devices to manage the measurement mechanism. Furthermore, the Fog devices capture the consistency and the oscillatory nature by generating a closed-loop system. For instance, in the life-saving air vents, sensors on vents monitor air situations flowing in and out of mines and automatically change airflow if conditions become harmful to miners. Most of these WSNs require less energy, less bandwidth, very low processing power and operating as a sink in a unidirectional manner.

7.3.5 Decentralized Smart Building Control

In decentralized smart building control, wireless sensors are set up to evacuate humidity, temperature or levels of several gaseous components in the building environment. Hence, information can be exchanged between all sensors in the floor and the reading can be mixed to form reliable evaluations. Utilizing distributed decision building, the Fog devices react to data. The system prepares to work together to decrease the input fresh air, temperature and output moisture from the air or increase humidity. Sensors reply to the movements by switching off or on the lights. The Fog computing is used for smart buildings which can manage basic requirements of conserving internal and external energy.

7.3.6 IoT and Cyber-Physical Systems (CPSs)

Fog computing has an important role in CPSs and IoT environments. IoT is a network that can interlink normal physical objects with identified address employing telecommunication and Internet. CPSs refer to the integration of systems' physical and computational elements. The combination of IoT and CPSs will change the world with communication and computer-based control systems, physical reality and engineered systems. Fog computing is made on the embedded system concept in which computers and software programs embedded.

The prime examples are linked vehicles, medical devices, etc. The objective is to combine the precision and concept of software and networking with the uncertain and vibrant environment. With the increasing cyber-physical systems, we will be capable to develop smart buildings, intelligent medical devices, agricultural and robotic systems.

7.3.7 Software-Defined Networks (SDN)

SDN is an increasing networking and computing concept. SDN concept integrated with Fog computing will eliminate the main problems in vehicular networks irregular collisions, connectivity and high packet loss rate. SDN provides support to vehicle-to-vehicle with vehicle-to-infrastructure communications and main control. It separates control and communication layers; control is performed by central server and server selects the communication route for nodes.

Researchers and IT professionals are constantly coming out with different industry cases for sustaining the Fog paradigm. There are several research papers exclusively pointing out next-generation.

7.3.8 Demystifying the Fog Computing Paradigm

Fog computing is a term originally introduced by Cisco. With the growing adoption and adaptation of the Internet of Things (IoT) technologies and tools, the Fog computing started to acquire a lot of attention. The main idea of Fog computing architecture is to distribute data processing and operation procedures over various devices that are connected with one another as well as with the faraway Cloud applications, services, platforms and databases.

One of the key features of Fog computing is a vertical distribution of functions by layers that extend from sensors, to the Fog, and, finally, to the Cloud depending on the processing latency. This architecture implements high-latency (days to months) enterprise operations in the Cloud, whereas technical operations with low latency (milliseconds to hours), starting from high speed to transactional analytics, are realized in Fog nodes. Patterns and rules for machine learning algorithms are formed in the Cloud; then they moved to the Fog for quick implementation.

7.4 Challenges of Fog Environments

With the growing number of connected and embedded devices, the device Cloud comprising several disparate devices is being formed in order to do local data capture, crunching and visualization. There are multiple device-to-device (D2D) and device-to-Cloud (D2C) communication protocols and channels. There are also a number of device data formats. The seamless connectivity and service integration are required for the successful implementation of Fog devices. Clouds are getting complicated with the entering of multiple devices in any environment (personal, professional and social). Data and protocol translations involve a lot of smart works, and there came numerous connectors, drivers, adapters, etc.

The other crucial challenge ahead is how to enhance the user experience while receiving the various services of Fog computing environments. There are techniques, technologies and tips being circulated in order to substantially increase the Fog environment experience. This chapter is being specifically prepared and incorporated in this book in order to tell all about the ways and means of enhancing the user experience.

7.4.1 Need for Quality of Experience (QoE)

Defining QoE for the Fog-based IoT applications is complicated and diversified, and to explain it, QoE can be compared with QoS [3]. QoS always checks the overall services and functionalities of the system to meet up the requirements and needs of the end-user, which is based on the definition of QoS by International

Telecommunication Union (ITU). According to ITU, QoS is defined as the complete feature of system services, and thus, meeting up the needs of the end-users. But ITU defines QoE as end-users' total acceptability of system services and thus assuring the meeting up of the specified needs of the end-users. QoE checks on user requirements and needs but QoS continuously monitors the attributes such as throughput, cost, service delivery deadline, packet loss ratio, throughput.

From the definitions of QoE and QoS, one can understand that fundamentally they both are different aspects but, in a few perspectives, they look the same. For instance, the users' expectation analysis for QoE will end up improving the QoS and thus will achieve QoE at a later stage. When users require clarity in the images, the systems model of image rendering changes by enhancing the algorithms which are improving the QoS, which will further enhance the QoE. To improve QoE, QoE-aware application placement has to be done wisely deciding which application is best suitable for which the customer's QoE gets improved along with system QoS improvement in the areas such as throughput, packet loss ratio, lower latency and meeting up the deadlines given by an authorized user.

To enhance the user experience of Fog computing, the applications need to be placed in a proper instance of Fog so that it becomes a QoE-aware app placement that will ensure that users get improved QoS with respect to service, latency, throughput and packet loss rate and even the service cost. While the mapping of applications to the Fog instances benefits in all the way, the task of mapping should not consume much time and processing power which might become a contradicting overhead to the system [3].

IoT applications in the Fog computing are divided into a number of interconnected application modules; but in general, there are two important modules:

- The first module, running at the users' smart devices such as mobile phones, bedside monitors, desktops and wearable devices, television sets, provides an interface for sensing data, data aggregation, authenticating the users, data storage in the sensing devices. This module is called client module and runs on the users' devices and senses the contextual data, aggregates and sends the data to the second application module and receives the output from the application module for the users.
- The second is the application module that performs the data operations. Here, the result of the processing is the end product of the Fog-based IoT applications. The application module is responsible for the data cleaning and filtration analysis and decision-making. The decision made by this module is sent back to the client module as notification and the data storage instructions.

Cloud data centres are the supreme computational location, and IoT devices are merely the data generators. IoT devices normally don't process the data which is sensed by them for the resource and energy utilization constraints. The Fog module comes in between these two modules to provide the better performance and response rates. Fog majorly operates between the Cloud and the user devices so as to reduce the latency and to obtain better user experience.

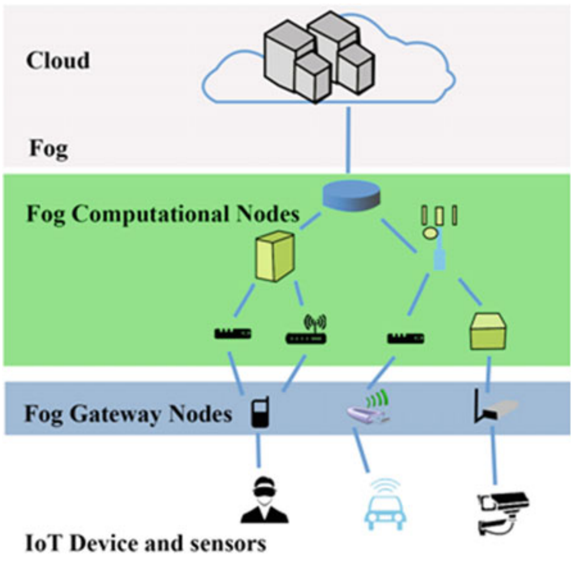


Fig. 7.1 Network model of FOG computing

The Fog layer itself is logically divided into two sublevels such as Fog gateway nodes (FGN) and Fog computing nodes (FCNs). FGNs are responsible for interaction between IoT devices and processing devices. FCNs are the upper level for nodes, which provide the resources and interface for the processing and analysis of data signals. Open Fog Consortium [4] says that the capacity and the intensity of computational processing and resource utilization for the gateway nodes and the computational nodes are completely different. Each Fog node can provide the interface for the application programs (API) so as to query the node. A hierarchical network model of Fog computing is provided in Fig. 7.1. It originates from the user IoT devices, reaches FGN through FCN and finally reaches the Cloud servers for bigger processing and storages purposes.

Invariably, all the Fog nodes will provide dynamic and faster communication with its other accessible nodes through powerful protocols [5] like CoAP and SNMP. There is a varying capability for the resource accessibility and the processing capacity between the lower-level and the higher-level Fog nodes; even the Fog nodes in the same level also might have distinguished privileges.

7.5 5G Technologies and the Fog Architecture

Fog and Cloud computing are the two powerful technologies that enable the 5G customers to have a computing paradigm which will help the requirements of the real-time application and also the low-latency applications running in the IoT network edge. The support for complex analysis and durable data storage in the 5G core are also the preferred qualities. The shift from Cloud to Fog in 5G, as presented in [6], discusses the key challenges and runtime adaptation of Fog services which is essential for IoT application.

In the next ten years, massive amount of objects of our day-to-day life will interact with each other to create a smarter living environment. This will significantly increase the user experience, and most of the objects will be very independent and create a proactive lifestyle. The sensors in the smart things are connected to the Internet services automatically sensing the required data at the right time by becoming context-aware and provide experiential information to the customers. In this way, they help the humans or the machines to make apt decisions. With the right way of processing the relevant sensed data, and the right connections between the devices and processing Fog nodes, IoE becomes more reliable and preferred by the customers.

Thus, the sensed data becomes information and will turn into actions automatically creating new capabilities with wide experience and unprecedented economic opportunity for an individual human being, business organizations, or even to the nations. By integrating Fog and Cloud services in 5G, it would minimize the network response time.

7.5.1 Fog Radio Access Network (FRAN)

FRAN is a potential paradigm for 5G mobile networking systems. It makes the best of Edge networking by integrating the Fog computing capabilities with RAN. In comparison with the conventional RAN, FRAN is slightly different with respect to the storage, computing and networking. Earlier, the centralized Cloud RAN (CRAN) was a popular technology and due to its monolithic and centralized nature, the increasing number of base stations, unpredictable user mobility and the huge volume of data, it has shown its limitations [7]. Later, heterogeneous CRAN has also been proposed which shows economic efficiency than CRAN, but all the data are exchanged through BaseBand Unit (BBU) pool that causes additional burden on the wireless links. Both CRAN and HCRAN have utterly failed due to the increasing data generated through social media communication channels.

FRAN was introduced by CISCO, to perform distributed signal processing and computing rather than by the centralized BBU pools. It also offers the storage of local data in the access points and user equipment instead of the Cloud data centres. It also performs collaborative radio signal processing and maximizes the Edge device usages.

7.5.2 FRAN Architecture

In the FRAN architecture, macro and small remote radio heads and access points of Fog computing connect to the BBU pool. Fronthaul is a connection between a new network architecture RAN consisting of centralized baseband controllers and remote stand-alone radio heads installed at remote cell sites located a few metres to tens of kilometres away.

The access points are connected using fronthaul links. So, the access points can implement radio signal processing locally and collaboratively with the help of their computing skills and can also manage the caching flexibly. Access points of Fog (FAPs) integrate the fronthaul RF and the functionalities of physical layer processing to the upper layers of the architecture.

Both the radio heads and FAPS are capable of caching but their cached data are completely different; FAPs fill their cache with locally most accessed data which are not cached by radio heads. The Edge devices are equipped with distributed storage, processing and control along with the effective resource management. Figure 7.2 depicts the FRAN architecture.

To have a better QoE, FRAN is adapted with four transmission modes such as the following:

- Device-to-device relay mode
- Distributed (local) coordination mode
- Centralized (global) mode
- Macro Remote Radio Heads (MRRH) mode [8]

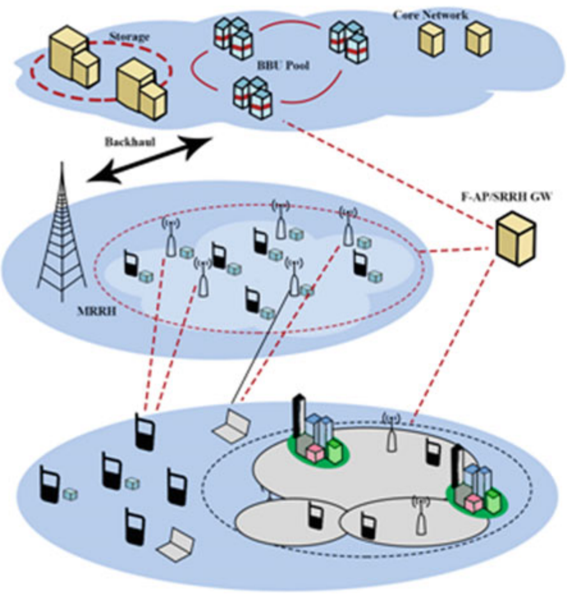


Fig. 7.2 FRANarchitecture

The Fog user devices dynamically decide the mode of transmission based on the communication location, the distance between the communicating devices, computing and caching capacities and the Quality of Service (QoS).

7.5.3 Handover Administration in FRAN

Administering the handover in any mobile communication is the most important but critical technique to guarantee an unfailing network experience. There are broadly two types of devices based on the mobility in mobile communications, viz the high-speed Fran User Equipment (FUE) and the low-mobility Fran User Equipment. High-speed FUEs need to use MRRH mode that has large geo coverage and reliable network, whereas the low-mobility FUEs can use a small-range radio head. Radio link failures and unwanted handovers are always possible in mobile networks due to a small area of access. Fronthaul gets overloaded due to frequent and rapid handovers affecting the whole network communication. So, the access points and radio heads can collaborate with FUEs in the decision-making of the transmission modes.

One of the main characteristics of 5G technology which yields to QoE is its ability to serve heterogeneous and diversified applications using the same communication platform. Due to the heterogeneous nature, there are many kinds of packets travelling in the network each with different traffic requirements, for instance, the vehicular network requires a very minimal latency. Context-aware applications are able to serve the types of traffic requirements. Fog computing is possible in any of the Edge nodes, core Cloud network or in third-party data centres where the local baseband units can become the computing servers. Computations happen in two ways such as communicational ones or the traffic flow ones.

FRAN attempts to take complete advantages of the Cloud computing model, diversified networks and Fog computing capabilities. The four types of Cloud are:

a. Global centralized communication and storage Cloud which is the same as of the centralized Cloud
b. Centralized control Cloud that is used for processing the high-power node functions
c. Distributed logical communication Cloud that is available in FAPs and FUEs, which are responsible for communication in local collaborate radio signal processing
d. Distributed logical storage Cloud that is responsible for the caching and local storage of data in the Edge equipment.

FRAN's performance gets affected due to the severe interference since all the four modes of transmission use the same radio resources for the shared FUEs. Two interference suppression techniques are proposed, namely coordinated precoding and coordinated scheduling. Coordinate precoding model is used to reduce the interference in the physical layer and the coordinated scheduling is for reducing the interference in the medium access control layer (MAC layer).

Though FRAN addresses the disadvantages faced in Cloud RANs and heterogeneous Cloud RANs to a certain extent, there are still issues like caching in the Edge devices, network function virtualization (NFV) and software-defined networking (SDN) which need to be addressed technically.

7.5.4 Caching in Edge Devices

Caching in Edge devices of FRAN reduces the latency in the content delivery and also provides better performance and customer experience by the flexibility in the content-aware techniques. The space for caching is rather very small as compared to the centralized Cloud caching mechanism; the hit rate is also less in Edge caching. So effective utilization of this Edge caching space is very essential. Policies such as what data to cache and when to release the caches require protocols to enhance the caching policies. The caching algorithms like most recently used, round robin, least recently used, first in first out, last in first out models need to be studied, and appropriate policy has to be used in FRANs.

7.6 Network Function Virtualization (NFV) and Software-Defined Networking (SDN)

The NFV deals with the functionality transfer from hardware devices to the software applications in the telecommunication firm by delinking the network functions from its underlying hardware. Virtualizing the SDN controllers in FRAN and how NFVs can be used to virtualize them are still a major challenge.

Edge computing platforms are compatible with proprietary gateways which hardly allow the external third-party services to deploy in them. Participatory Edge computing is proposed for home gateways which act as an open environment for deploying the external services in them. This creates flexibility in the platform for enabling local services to work on them [9].

There are many open Fog platforms with an objective of allowing the users to run their services in their local home computing devices. Sandstrom, Yunohost and Flockport are just to name a few. Sandstrom and Yunohost provide graphical user interfaces and allow them to create private Cloud platforms at home with few clicks. The Flockport platform runs on Linux through Linux containers to provide applications, but the scope of those applications is limited only to the availability in the Flockport application store. Figure 7.3 depicts the open platform Edge Cloud known as Guifi.net community that consists of heterogeneous distributed devices. These computing devices use Cloudy software to run their common services.

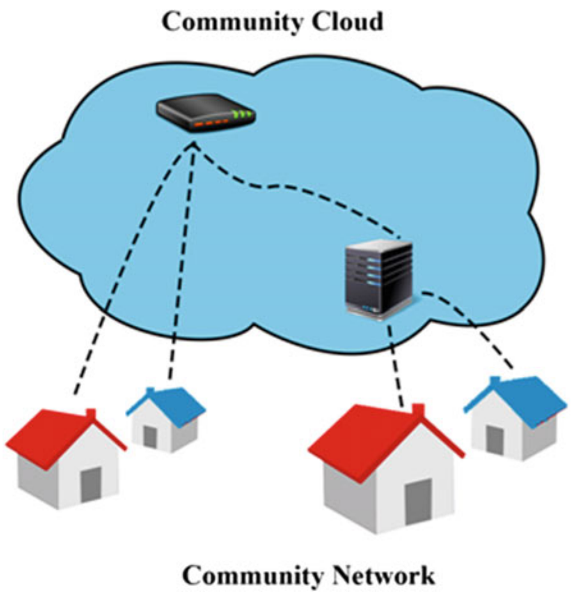


Fig. 7.3 Guifi.net's Cloud community network

Heterogeneous hardware is used due to the contribution of many types of devices for the community Cloud, and most of these devices are cheaper ones such as mini PCs; infrastructure is also distributed as the community devices are spread inside the home premises of the user.

Cloud nodes include different hardware devices and its distributed application is installed on every device connecting them to the community Cloud. These Cloud nodes are distributed over the Fog network. Mini PCs and smartphones are used in the user premises, and the desktop PCs are majorly available in the municipality controls creating the community Cloud. Both Cloud infrastructure and the application services are scalable and so they allow new devices and new services to be added easily. Initially, the community Cloud has the preset of installed services, and it is also open for the new services inclusion. This system is not only flexible but also user-friendly due to the availability of graphical interface for the installation and services. Security and commercial application services would still enhance the system for an economy generation with a secure way of communication and interaction.

7.7 Challenges of Fog Computing

As Fog computing environment is spread over the mobile devices of the user, and even in the backend Cloud server, application process scheduling in Edge computing is a tricky one compared to the Cloud computing situation. Decision-making as to where to do the computation is also difficult. Unpredictable latency and round-trip time, low bandwidth decision-making of the Fog node for computation, low-level hardware are very common in Fog. Minimum network utilization compared to Cloud computing, caching in the client's data are the advantageous benefits of Fog computing. The major threat would be on security as the Edge computing giving room for the processing to happen in the clients' devices there can be malicious nodes joining the Fog acting like a right user and would give serious security issues. To achieve preferred user experience, effective way of scheduling, deployment of authenticating the nodes and other security measures and decision-making of where to process the data are inevitable.

In all the ways other than the latency issues and network utilization, Cloud computing is highly preferable to the centralized control of processing; and security is provided by the Cloud servers. Since the independent agents are contributing towards Fog, the trust for Fog nodes and data security is to be standardized to get the best of Fog computing.

Though Cloud computing supports heterogeneous data processing and application services, it has dedicated platforms to take care of hardware and software management, resource sharing and scheduling based on the type of data and hardware and software resources compatibility with the incoming data. Fog computing can transparently move the data and computation from one device to the other based on the resource and energy availability information. There might be many users who connect to the network very briefly say maybe for a few minutes while they are in a moving state; there may be other users who move from one Fog node to another Fog node due to the mobility. Smooth handover policies are very much essential for the better user experience, and the Fog users should not feel the delay due to unpredictable delays and scheduling of the processes in continuously different devices.

Data management in Fog computing is also another challenge where the Cloud's gigantic "centralized storage" concept is overridden by the Fog's distributed storage into many nodes. Pedagogies to efficiently store and access the data, fetch the data with low latency, caching the encrypted data and still achieve energy consumption are the recent research issues in Fog. Achieving data consistency in the Fog environment, while caching and fetching data, is also a challenge and an open question for the researchers in this field. Arbitrary devices in the Fog architecture find it very difficult to exchange data due to the encryption and strict policies on privacy. Authentication protocols and measures for distributed denial of service attacks (DDoS) and other unauthorized activities should be dealt with very seriousness as sensitive data might be in the Fog environment which requires a high degree of security. The WMFOG, proposed by Hao et al. [10], takes the advantage of Fog computing model and achieves flexibility and performance improvement. It customizes policies to use the hardware resources connected with it.

Day by day, there is an exponential increase capability and the volume of sensor data, but the network performance is not scaling up to that level leading to a limited performance in IoT sensor network. Multipath transmission protocol for TCP is proposed with retransmission or acknowledgement under transmission rate network conditions, and it ensures the lower overhead [11]. But when the network is poor, this multipath transmission shows drastic difference due to the many numbers of retransmissions, packet loss, jitter and traffic congestion.

In the research reported in [12], a variety of multimedia-based services are offered for the smartphone and mobile device users through various wireless networks. The forecasts for the year 2020 predict that the 90% of global data traffic would have multimedia data and over 50% of the overall traffic would be video delivery data in wireless mobile devices. To cope with this multimedia era, powerful next-generation technologies like 5G network and mobile Edge computing, SDN and CRAN are already available.

Interactive 3D, high-definition (HD), ultra HD streaming of video, video on demand are the emerging multimedia applications which require unprecedented high speed and access along with very little latency. The user's expectation with respect to video data is to get videos on demand anywhere and anytime with a full satisfaction which defines the QoE. ABR and caching technologies for Fog nodes would facilitate this kind of QoE of providing video on demand.

ABR schemes such as Apple live streaming and Dynamic Adaptive streaming are also used over HTTP to provide such quality video streaming data without delay. Group of users in hot spots, from the same geographical area, might share the same preferences and they try to search for the same videos which are on hit for the day causing too many duplicate downloads. From the studies, we come to know that reduced traffic loads are because of downloading the same popular contents in the network. 10% of the top news or breaking news of the present day's video is watched by almost 80% of viewers. The cache schemes available for Fog computing to avoid redundant content transmission by making its users access from the caches in the Fog network. Relying on the Cloud for UHD video files will not provide the required QoE, rather it will end up having data tsunami that will saturate the network by increasing the delay. Fog supplements the Cloud by performing the data pushing, processing and decision-making in its Edge devices. A single Fog node can provide computing services to many mobile devices through virtualization paradigm. It creates many instances of the service to be virtualized and performs simultaneous tasks of different application services in them, which is a powerful feature of Edge computing and virtualization. So, part of Cloud's application services migrate to the Edge nodes and do the user requirements with QoE. Rosario et al. [13] have proposed an architecture for providing service migration, which is composed of Cloud computing tier in the first layer, with multiple tiers of Fog nodes (tiers 2, 3, 4, etc.). Video dissemination is provided by video services migration that can be triggered by energy conservation, user's movement inside the network, or traffic load increment or reduction.

Fog storage uses Rados and Rados as an object distributed storage model; it uses the CRUSH algorithm to locate the data in the network without any remote communication. Rados have two kinds of nodes such as the monitors and the Object Storage Daemons (OSD). Here, the object storage daemon nodes are used to store data. One of the monitors is chosen as the coordinator node that is responsible for up-to-date view of the clustermap. Initially, each client gets the clustermap details from the monitor and using crush algorithm locates the object. Hashing technique is used to locate the object. The replications of the same objects are placed in the same device. CRUSH algorithm is run to identify the object's location. Major design issue of such a model is how to deploy a Fog computing application node in the Edge network [13] which should continue to cooperate with the Cloud constantly and the methodology to provide network resources to such hybrid network which comprises of many layers or tires.

7.8 Future of Fog Computing

Fog computing provides ample opportunities for creating new applications and services that cannot be easily supported by the currently available computing and communication models. The Cloud computing paradigm is maturing and stabilizing fast. The next and intense focus is going to be on Fog computing. There are a few critical use cases that demand the Fog capability. The real-time applications, analytics and services demand matured and stabilized Fog environments. The Quality of Service (QoS) and experience (QoE) attributes need to be fulfilled in order to take the Fog concept to the next level. The faster proliferation of Fog environments is the need of the hour. The fulfilling technologies are being given extra thrust in order to simplify and streamline the set-up and sustenance of Fog computing environments, which are the most sought-after solution for producing and delivering people-centric, context-aware, real-time, service-oriented, event-driven, Cloud-hosted, insights-driven services. Scores of new Fog-based security services will be able to help address many challenges we are facing in helping to secure the Internet of Things.

The other reason why Fog will be shining in the days ahead is that data loses its value when it cannot be analysed fast enough. Fog computing infrastructure can capture any data immediately, subject them to a variety of investigations in order to extract and expose actionable insights. The Fog analytics is the greater enabler of many use cases across industry verticals such as oil and gas production to banks and retailers.

Security cameras, smartphones, sensor-attached physical assets, digitized entities, drones, robots, thermostats, advanced cars and IP televisions are being extensively used in our everyday environments. These generate a lot of usable and useful data to work on. Industrial environments are stuffed with a variety of sensors and actuators. Retail stores, warehouses, oil wells, refineries, pipelines and processing plants are being empowered with a bunch of sensors. Thus, the amount of IoT data gets bigger as days go by. Having understood that data is a strategic asset and we cannot throw out data. Instead, we need to cautiously collect all kinds of internal as well as external data, stock them in hot and cold data storages, process them through integrated analytics platforms to made sense out of data. Also, data monetization is another prospective domain being explored by various industry verticals. The fast-changing size, scope, speed and structure of data bring forth immense opportunities and possibilities for the technical professionals.

Organizations are beginning to look at Fog computing as the answer. Fog computing consists of putting microdata centres or even small, purpose-built high-performance data analytics machines in remote offices and locations in order to gain real-time insights from the data collected or to promote data thinning at the edge, by dramatically reducing the amount of data that needs to be transmitted to a central data centre. Without having to move unnecessary data to a central data centre, analytics at the Edge can simplify and drastically speed analysis while also cutting costs.

Organizations are currently reliant on large and complex clusters for data analytics, and these clusters are rife with bottlenecks including data transport, indexing and extract, as well as transform and load processes. While centralized infrastructures work for analyses that rely on static or historical data, it is critical for many of today's organizations to have fast and actionable insight by correlating newly obtained information with legacy information in order to gain and maintain a strong competitive advantage.

Data is priceless right at the beginning, in the seconds after it has been collected—consider the instance of a fraudster or hacker accessing bank accounts—but it loses all value during the time it takes to move it to the centralized data centre infrastructure or when it is upload to the Cloud environment. Losing value from that data due to slow transmission or decisions is not acceptable, especially when an edge computing platform that eliminates moving data provides the near-instant intelligence needed. Organizations cannot afford to wait days, weeks or even months for insights from data. With data analytics at the edge, users do not have to wait longer; besides possibility of data loss or theft is reduced.

7.9 Fog Data Analytics and Use Cases

The compelling use cases remain the key differentiator and motivator for the unprecedented success of the Fog computing model. The success of Fog computing is to do a lot of good things for the faster spread of the flourishing IoT concept. With IoT spreading its wings far and wide, the resiliency of Fog analytics of IoT data (big data, fast and streaming data) is being looked with aspirations. IoT evangelists, exponents and experts are articulating the need for the success of the Fog computing paradigm in taking the IoT discipline to the next level. Furthermore, there are cognitive analytics methods through the machine and deep learning algorithms, computer vision, natural language processing and neural networks. These advancements ultimately benefit the Fog analytics requirements to extricate actionable intelligence out of IoT data heaps.

7.9.1 IoT Sensor Data Monitoring and Analysis

IoT sensors are already creating massive amounts of data, and with the number of sensors collecting data growing, data volume is set to continue growing exponentially. Moving data analytics to the Edge in the Fog environment, with a platform that can analyse batch and streaming data simultaneously, enables organizations to speed and simplify analytics to get the insights they need, right where they need them.

7.9.2 Retail Customer Behaviour Analysis

Brick-and-mortar stores are looking for any competitive advantage they can get over Web-based retailers, and near-instant Edge/Fog analytics—where sales data, images, coupons used, traffic patterns and videos are created—provides unprecedented insights into consumer behaviour. This intelligence can help retailers better target merchandise, sales and promotions and help redesign store layouts and product placement to improve the customer experience. One way that this may be accomplished is through the use of Edge/Fog devices such as beacons, which can collect information such as transaction history from a customer's smartphone, then target promotions and sales items as customers walk through the store.

7.9.3 Mobile Data Thinning

Mobile data, much like the IoT sensor data, is being created with increasing rapidity but the downside of all of this data is that some, or even most of it, is not needed to answer data analysis queries. How can a company hone in quickly on which ad is having the most impact, while eliminating the "noise" of other social activities happening around it? Having edge/Fog analytics enables companies to better understand their data and analytics needs to thin big data and only process and analyse the information necessary to the query.

7.9.4 Compliance Analysis at Financial Branch Locations

One major function of data analysis at financial institutions is to find, and stop, non-compliant transactions. When organizations have to take the time to move data back to the central data centre or upload it to a Cloud environment for processing and analysis, the lag time decreases the value of this data. Using microdata centres in financial institution branches enables analytics to happen in real-time, meaning that non-compliant transactions are caught and stopped much more quickly, which can have a real and positive impact on the bottom line.

7.9.5 Remote Monitoring and Analysis of Oil and Gas Operations

Edge computing for manufacturing and oil and gas operations can mean the difference between normal operations and a disaster. Today's traditional centralized data analytics infrastructures can tell you what caused downtime, or in the very real case of these types of operations, an explosion—but only after the fact. Having near-instant analysis at the site as the data is being created can help these organizations see the signs of a disaster and take measures to prevent a catastrophe before it starts.

Thus, there are a number of newer capabilities being brought in by the Fog computing model. There are new deployment models, premium services, solving some of the toughest problems, real-time analytics, etc. These innovations, disruptions and transformations pitch for the Fog computing model.

7.10 Conclusion

Enhancing the quality of experience (QoS) is vital for the survival of any technology-inspired services. As there are more Fog environments popping up across the world, the important parameter and metric to be considered is the quality of user experience, QoE. We have discussed the importance of QoE and its enhancement techniques in this chapter in some detail. We have also described why QoS matters for the intended success of Fog computing. We have also written about the various benefits and real-world use cases relevant to the Fog computing paradigm. This computing model is to flourish in the days ahead as it is showing a lot of positive vibes and promises for the business worldwide as well as for the general public in the distributed IoT environments. We have also listed the key inherent challenges and how they are being addressed through a host of collaborative research endeavours, around the world.

References

1. Bellavista P, Alessandro Z (2017) Feasibility of fog computing deployment based on docker containerization over RaspberryPi. In: Proceedings of the 18th international conference on distributed computing and networking, ICDCN '17, ACM, 2017
2. Saurez E, Hong K, Lillethun D, Ramachandran U, Ottenw¨alder B (2016) Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In: Proceedings of the 10th ACM international conferenceon distributed and event-based systems, DEBS'16, ACM, 2016, pp 258–269
3. Mahmud R, Srirama SN, Ramamohanarao K, Buyya R (2017) Quality of experience (QoE)-aware placement of applications in fog computing environments. J LATEX

4. OpenFog Consortium Architecture Working Group (2017) OpenFog reference architecture for fog computing. OPFRA001 20817, p 162
5. Slabicki M, Grochla K (2016) Performance evaluation of CoAP, SNMP and NETCONF protocols in fog computing architecture In: Proceedings of the IEEE/IFIP network operations and management symposium, NOMS '16, IEEE, 2016, pp 1315–1319
6. Kitanov S, Janevski T (2016) State of the art: fog computing for 5G networks, 24th telecommunications forum TELFOR, 2016
7. Haijun Zhang Yu, Qiu Xiaoli Chu, Long Keping, Leung Victor CM (2017) Fog radio access networks: mobility management. Interference Mitigation and Resource Optimization, IEEE, p 2017
8. Peng M, Yan S, Zhang K, Wang C (2016) Fog-computing-based radio access networks: issues and challenges, IEEE, 2016
9. Khana AM, Freitag F (2017) On participatory service provision at the network edge with community home gateways, Procedia Comput Sci 109C (2017), Science Direct, 2017, pp 311–318
10. Hao Z, Novak E, Yi S, Li Q (2017) Challenges and software architecture for fog computing. IEEE Internet Comput 21(2):44–53
11. Xu Z, Xing K (2017) Coordinationless coordinated fastlane network service in wireless multimedia sensor networks, 2017 IEEE
12. Lu R, Heung K, Lashkari AH, Ghorbani AA (2016) A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT, 2016 IEEE
13. Rosário D, Schimuneck M, Camargo J, Nobre J, Both C, Rochol J, Gerla M (2018) Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support, Sensors, 2018

Chapter 8
Specifying Software Services for Fog Computing Architectures Using Recursive Model Transformations

Nuno Santos, Helena Rodrigues, Jaime Pereira, Francisco Morais, Raquel Martins, Nuno Ferreira, Ricardo Abreu and Ricardo J. Machado

Abstract Due to massive amounts of data transfer between smart devices, the adoption of mobile Internet and Internet of Things (IoT) within Cloud Computing applications has resulted in numerous issues including data decentralizing challenges. As a resolution, a new service-oriented approach called Fog Computing has appeared to resolve at least some of these. However, the design of Fog Computing architectures also lacks a systematized approach for using models aiming to abstract the fog environments' services specification. In this context, this chapter proposes the use of a set of software engineering approaches for Fog-based architecture design, centered in UML artifacts and executing the *four-step-rule-set* (4SRS) method. Here, the Fog Computing microservices are modeled in SoaML's Service Participant, Capabilities, Service Interface, and Service Architecture diagrams. The approach is demonstrated within a research project that aims to develop a set of services for Cloud, Fog, and IoT paradigms in a distributed industrial environment.

N. Santos (✉) · H. Rodrigues · J. Pereira · F. Morais · R. Martins · R. J. Machado
CCG/ZGDV Institute, Guimarães, Portugal
e-mail: Nuno.Santos@ccg.pt

N. Santos · H. Rodrigues · J. Pereira · F. Morais · R. Martins · N. Ferreira ·
R. J. Machado
ALGORITMI Centre, School of Engineering, University of Minho,
Guimarães, Portugal

N. Ferreira
i2S—Insurance Knowledge, S.A., Porto, Portugal

R. Abreu
Cachapuz Bilanciai Group, Braga, Portugal

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_8

8.1 Introduction

Nowadays, the service-oriented (SO) paradigm is heavily adopted in the software industry generally in every business domains, mainly due to the widespread use of Cloud Computing. The other driving forces for adoption are the increasing popularity of other technology enablers such as the Fog paradigm and microservices architectures. Service Orientation allows software solutions to invoke functionalities, in the form of *software services*, which may be deployed within external and distributed platforms. Service specification has gained special attention within the industrial domain, where the emergence of topics such as Industry 4.0 and Industrial Internet of Things (IoT) has been advocating strongly interoperable systems, promoting the use of technologies like Fog and Cloud Computing, IoT, cyber-physical systems, and machine-to-machine communications. However, requirements elicitation faces many obstacles, due the heavy reliance on interoperability (at system level, enterprise level, data acquisition level, etc.) and a perceived complexity of the underlying architectures. There is not yet a context for upfront eliciting requirements that specifically relates to technical, infrastructure, and deployment decisions to Cloud, Fog or, IoT software services.

The proliferation of Cloud Computing and the IoT technologies was in the genesis of a service-oriented approach for integrating data between the two, called Fog Computing. Fog Computing is defined as *a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so* [1]. It is used for middleware purposes, but also includes its own virtualization, computation, and persistence needs. Thus, its related architecture requires well-defined design approaches for handling the complexity. The design approaches for Fog Computing architectures also lack the systematized use of models for supporting the services specification.

This chapter proposes an approach for supporting requirements elicitation and architecture design based on the *four-step-rule-set* (4SRS) method [2] that is able to include Fog Computing-related implications regarding architectural models and service conceptualizations. Our approach starts by eliciting software user requirements and deriving the software component-based logical architecture by performing the 4SRS method. The 4SRS method takes as input a set of UML Use Cases, describing the user requirements for tackling the initial problem, allowing to derive a logical architecture representation of the system using UML Components diagram. Specification refinement through model transformations [3] is an approach that promotes an overview of the overall system for afterward having a context to elicit the technical requirements.

As already mentioned, this chapter attempts to use requirements engineering approach, in particular the 4SRS approach, to deal with the inherent complexity of a Fog-based architecture design. The approach addresses upfront the identification and organization of the required features that respond to elicited requirements. Additionally, the approach allows addressing issues related to communications and data flow between resources (software systems, IoT, CPS, etc.), integration and interoperability, multi-tenancy, software deployment, among others. The software logical architecture is partitioned into sub-systems, where each of them will ultimately derive a service-oriented logical architecture. For deriving the services architecture, we propose an SOA variant of 4SRS [4] using SoaML [5] notation. This variant of 4SRS is improved with data inputs for specifying services, like API, ports and communication formats, required consuming services. The approach is validated using a real research project for a Fog Computing solution, called Unified Hub for Smart Plants (UH4SP). It is used as a case study since it is composed with software services to address data flow from Cloud-based platforms, industrial systems, brokers, and IoT systems.

The UH4SP project aims to develop a platform for integrating data from distributed industrial unit plants supported by software applications and services deployed in a Cloud platform. It uses the data acquired from IoT systems for enterprise-level production management and collaborative processes between plants, suppliers, forwarders, and clients. The UH4SP services were validated within the project by performing a set of pilot scenarios, mostly related to the cement domain. The project started by eliciting user requirements for tackling the business processes, without refining technical requirements whether regarding Cloud, Fog, and IoT systems. The use of the recursive model transformationsallowed addressing iteratively the service specifications of the UH4SP system in a microservice architecture paradigm [6], ultimately regarding to the composing layers of the platform related to Cloud, Fog, and IoT. For the validation, it is also included in this chapter the definition of a set of scenarios related to the specified services, which supported the project's pilot scenarios.

This chapter is structured as follows: Sect. 8.2 presents the UH4SP project requirements, the Cloud and Fog imperatives, and the main concepts within Fog Computing architectures; Sect. 8.3 briefly presents the process of eliciting user requirements and its refinement for the services specification; Sect. 8.4 describes the derivation of the software logical architecture based on the user requirements; Sect. 8.5 describes the model recursive transformations for designing a service-oriented architecture for Fog contexts; Sect. 8.6 demonstrates the approach by presenting the Fog scenarios within the UH4SP project; Sect. 8.7 presents the conclusions.

8.2 Background

8.2.1 Review of Fog Computing Architectures and Modeling

The rapid development of mobile Internet and the Internet of Things (IoT) applications has presented severe challenges for Cloud Computing applications. These challenges are mainly due to massive amounts of data transfers toward the Cloud and the inability to achieve responsiveness in some application fields that demands rapid reaction to events [7]. In response, a paradigm for an architecture is based on the edge of the network emerged: called Edge Computing. In Edge Computing, the massive data generated by different kinds of IoT devices can be processed at the network edge instead of transmitting it to the centralized Cloud infrastructure due to bandwidth and energy consumption concerns [8].

Fog Computing [9] is a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers [10]. Its benefits relate to reducing the need for bandwidth by not sending every bit of information over Cloud channels and instead aggregating it at certain access points [11].

Large technological enterprises, e.g., IBM and Cisco, have fostered the linking between Fog Computing and IoT, promoting the emergence of services and applications in domains such as connected vehicles, smart grids, smart cities, health care and wireless sensors and actuators networks. In this context, Fog Computing extends the Cloud Computing paradigm to the edge of the network to address applications and services that do not fit the paradigm of the Cloud due to technical and infrastructure limitation including [11]:

- Applications that require very low and predictable latency
- Geographically distributed applications
- Fast mobile applications
- Large-scale distributed control systems

This paradigm is composed mainly of three technologies [8]: Cloudlets, Mobile Edge Computing (MEC), and Fog Computing. The applications located on the edge of the network, called Cloudlets [12], were proposed for responding to low latency in machine communications. A Cloudlet is a trusted, resource-rich computer—or cluster of computers—that is well connected to the Internet and available for use by nearby mobile devices. MEC is a technology in the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers [13].

Fog architectures may be deployed with several tiers (N-tiers) of nodes between the Cloud on top and the IoT devices at the bottom: (i) some nodes located at the edge (typically focused on sensor data acquisition/collection, data normalization, and command/control of sensors and actuators); (ii) some nodes located in a higher (middle) tier (focused on data filtering, compression and transformation, and some edge analytics, including machine and system learning capabilities, required for critical real-time or near-real-time processing); and (iii) some located at the higher tiers or near the backend Cloud (typically focused on aggregating data and turning the data into knowledge). Edge routers each implement distributed capabilities, namely regarding computation, networking, and storage. Fog nodes in this network cooperate to implement distributed IoT applications [14].

Fog Computing performs computation, networking, and storage at intermediate levels, between the Cloud and the IoT. The research in [14] lists key requirements for a Fog architecture, namely: low latency, reduced network bandwidth, enhanced security and privacy, geographic locality of control, data-rich mobility, reliability and robustness, advanced analytics and automation, hierarchical organization, energy efficiency, environmental constraints, multiple-level programmability, multi-tenancy, virtualization, orchestration and management, scalability, agility, modularity, and openness.

A Fog infrastructure ranges from data centers, core of the network, edge of the network, and end points. The Fog architecture enables distributed deployment of applications requiring computing, storage, and networking resources spanning across these different players. Similar to Cloud, Fog architecture supports multi-tenancy pattern, where each tenant perceives its resources as dedicated, and defines its own topology [10].

The Fog-as-a-Service (FaaS) relates to the Fog-based services that perform between Cloud and the IoT environments. Here, a Fog service provider owns an array of Fog nodes. The software architecture related to the FaaS paradigm includes the following [14]:

- Fog hardware layer for networking, compute, and storage elements and their device drivers
- Fog infrastructure software layer including a versatile execution environment. Also, one or more operating systems, and functions to optimize virtualization and the use of containers
- Fog management software layer to support operational functions like life cycle management and orchestration, as well as rich application programming interfaces (APIs) for supporting the development of applications, and software development kits (SDKs) for supporting programmers to develop in the programming environment
- Cloud infrastructure software layer, composed of software supporting the Fog that runs on the Cloud, providing portals, services, resources, and tools
- Business/user applications layer that runs at the top of the stack; policy/security and analytics/data model blocks that span the vertical hierarchy, providing their capabilities at all layers.

The software architecture for a Fog platform is composed of the following [10]:

- Heterogeneous Physical Resources, i.e., Fog nodes (ranging from high-end servers, edge routers, access points, set-top boxes, and even end devices such as vehicles, sensors, mobile phones etc.)
- Fog Abstraction Layer that provides generic APIs for monitoring, provisioning, and controlling physical resources

- A service orchestration layer
- Northbound APIs.

Additionally, it is possible to integrate this mode, with an M2 M architecture [15]. In terms of technology, containers and clusters are used for supporting microservices in Fog settings [16], also using Topology and Orchestration Specification for Cloud Applications (TOSCA) [17] framework for designing the orchestration of the containers and clusters.

The OpenFog Reference Architecture (OpenFog RA) [9] is the de facto standard for Fog Computing architectures, which refers to hardware, software, and system elements for Fog settings. It is developed by the OpenFog Consortium, which was founded by the collaboration of ARM, Cisco, Dell, Intel, Microsoft, and Princeton University. The OpenFog RA is complementary to other IoT and industry standards, namely Industrial Internet Consortium (IIC), ETSI-MEC (Mobile Edge Computing), OPC-UA, Open Connectivity Foundation (OCF), OpenNFV, among others. The OpenFog RA enables Fog–Cloud and Fog–Fog interfaces. It defines the required infrastructure to enable building FaaS, which includes Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), and many service constructs specific to Fog.

To the best of the authors' knowledge, there is not much research in the literature on modeling of Fog architectures using UML. In comparison with Cloud applications, the Cloud Application Modeling Language (CAML) approach [18], the MultiCloud Migratable Applications (MULTICLAPP) framework [19], and the Cloud Modeling Language (CloudML) approach [20] refer to UML profiles in order to model applications for provisioning and deployment in Cloud platforms. The research in [21, 22] proposes an approach for deriving functional requirements and modeling them in UML Use Cases of Cloud applications.

8.2.2 Cloud- and Fog-Related Imperatives of UH4SP

In this subsection, the industrial context of the UH4SP project is briefly described, as well as the common domain production management difficulties, motivating the need for solutions as the ones the UH4SP project proposes. Additionally, main arguments as to why the data management using Cloud Computing and IoT systems suggested the development of an intermediate Fog layer and its related architectural imperatives are also presented.

The UH4SP project emerged as a solution for the cement production industry. In this domain, an industrial unit plant is typically composed of a set of: (1) fabric silos, responsible for storing bulk and bagged materials, which may contain grain, coal, cement, carbon black, woodchips, food products, and sawdust; (2) logistics circuits, where trucks follow a path for loading or unloading material; and (3) other points for transformation activities, where some industrial activities typically end in storing a good in a silo or warehouse.

A production unit typically relates to a local brand unit, i.e., an industrial plant of a given industrial group (typically multinational corporations). While an industrial unit plant manages only their local productions, the industrial group is responsible for the business management of all group's plants. Industrial unit plants may also have an associated end customer, which is an entity that belongs to the industrial group but is responsible for retailing the produced goods. Customers and suppliers, respectively, purchase and provide material to the industrial unit plants. Finally, a forwarder is responsible for the transport of material from/to the plant. They may relate to a contracted service, or a given client may have their own forwarder. Some industrial groups additionally have intermediary warehouses. It is a warehouse from a given industrial group, responsible for storing loaded and unloaded goods. These warehouses typically perform as intermediaries, facilitating long-distance displacement of goods as they are usually located between routes from the industrial group's plants. Finally, production management may also include external services, i.e., services provided to an industrial unit plant, e.g., equipment (kiosks, weighbridges, etc.) maintenance, software providers.

The UH4SP project focused at the industrial unit plants where the production management systems were developed by Cachapuz Bilanciai Group, located in Braga, Portugal, since Cachapuz was in fact the leader entity of the UH4SP project consortium. Cachapuz's production system is responsible for performing the control of trucks arrival/exit as well as the loaded/unloaded activities, and for communicating with the plant's ERP and the industrial hardware. The current solution is deployed on-premises. It is designed for process automation and operations improvement within an industrial unit. It has a considerable scaling and complexity, which made it not adequate and flexible to enable the development and deployment of Cloud services based on modules and external access. The on-premises deployment is difficult for promoting a corporate-level management, since in order to the industrial group manager to have an integrated analysis of the group's plants, he was only able to access each plant's ERP one at a time using a remote virtualized environment (Fig. 8.1). The remote business analysis was also impossible to perform in some contexts, namely within plants located in low-connectivity spaces. The current solution did not enable the incorporation of remote technical interventions. Finally, the current solution was not able to respond to a previous need to enable third-party access (e.g., forwarders, customers, suppliers) to allow the inclusion of collaborative tools in process execution and analysis.

To overcome these issues, the settled UH4SP project objectives were: (1) to define an approach for a unified view at the corporate (group of units) level; (2) to develop tools for third-party entities; (3) ensure in-plant optimization; and (4) ensure system reliability. The strategy was based on the adoption of Industry 4.0 and Industrial Internet of Things paradigms, namely:

- To evolve the information systems architecture for a Cloud-based model, namely to enable the applications availability in a Software-as-a-Service (SaaS) model

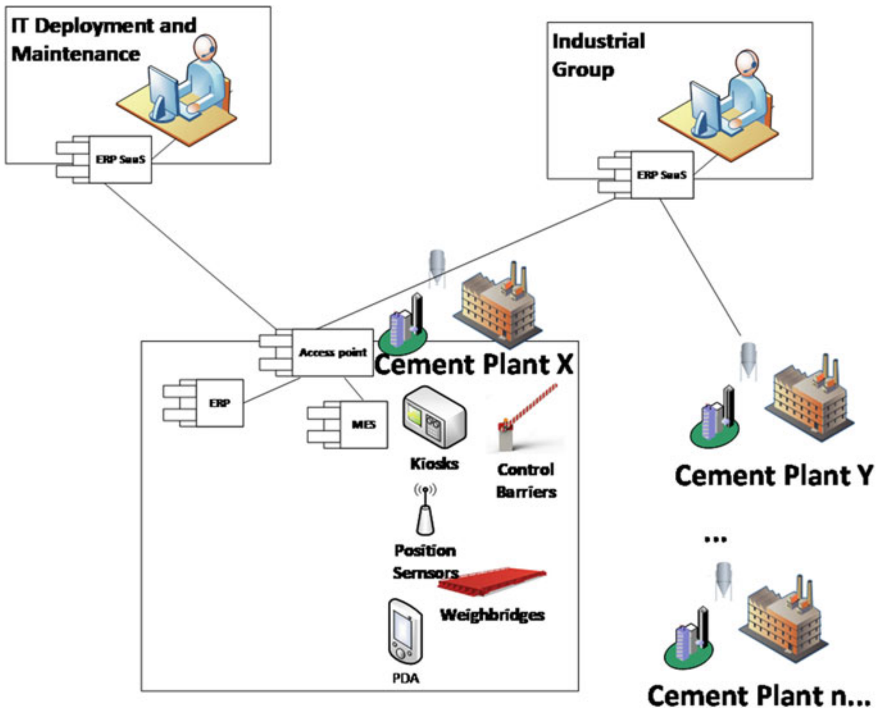


Fig. 8.1 Problem overview

- To provide a corporate, aggregate, and integrated vision of the operations carried out in several industrial unit plants, regardless of their geographic location
- Replace the hardware installation (service kiosks, information panels, etc.) at the level of the industrial unit plant by the use of intelligent sensors, advanced identification systems, and mobile devices
- Provide robust Disaster Recover and Business Continuity mechanisms required by the high availability and criticality of this type of solutions.

The UH4SP project has included the development of a Cloud Computing and service-based architecture, IoT systems and desktop and mobile Web-based applications. Early within the requirements elicitations and architecture design phase of the project, the consortium identified latency issues that the solution would encounter once implemented and in execution. Remote business analysis and Disaster Recover/Business Continuity would face some issues that would not be assured by a centralized Cloud platform. Namely, there were some predicted high-volume of uploaded production data, as well as some identified scenarios with low-connectivity problems. Hence, a Fog Computing layer was proposed.

Thus, the project's architecture design included the design of a Fog layer. Specifically relating to Fog Computing architectures, the UH4SP project aimed the following imperatives:

- To optimize the upload of production data to the Cloud, spread through the edges
- To optimize Disaster Recover/Business Continuity mechanisms with redundant data throughout the edges, so production data can be accessed even when connectivity problems with the pre-defined edge occur (at least, the neighbor edge)
- To ensure a gateway for local services, for a correct orchestration of the edges of each factory.

These imperatives are mainly related to middleware services and are part of the project's user requirements, along with additional requirements for handling the data within the enterprise-level production and collaborative tools. All user requirements were elicited and included in the software logical architecture. The refinement of the Fog services specification was only possible after the recursive model transformations.

8.3 Overview of Service Specification Approach

Using a UML Use Case-based approach, very common in software engineering, for designing a service-oriented architecture, allows identifying some of these elements; nevertheless, much information is lost as UML Use Cases relate to user requirements and these are far from the desired architecture elements. Thus, by using a software engineering method as the 4SRS [2], the derived information will now relate to the system requirements. For Cloud applications, modeling includes the desired software features and also the provisioning and deployment in Cloud platforms [18]. Fog applications modeling faces similar needs. Specification refinement through model transformations [3] is an approach that promotes an overview of the overall system for afterward having a context to elicit the technical requirements.

In the present work, we propose a model-based approach for a Fog-based architecture design, using requirements engineering approaches to design service-oriented architectures that respond to elicited requirements. This approach firstly uses typical gathered user requirements, namely functionally decomposed UML Use Cases and a *Domain-driven Design* (DDD) [23] approach, which are input for the 4SRS method that allows modeling of the entire Fog Computing architecture in a logical architecture diagram (using UML Component notation). Afterward, we propose to refine sub-systems of the architecture iteratively, in order to identify, model, and specify a set of software services in SoaML diagrams, such as Service Participants, Service Interface, Capabilities, Service Data, Service Architecture, Service Contracts, until all logical architectural elements are supported by software services.

The 4SRS method takes as input a set of UML Use Cases describing the user requirements and derives a software logical architecture using UML Components.

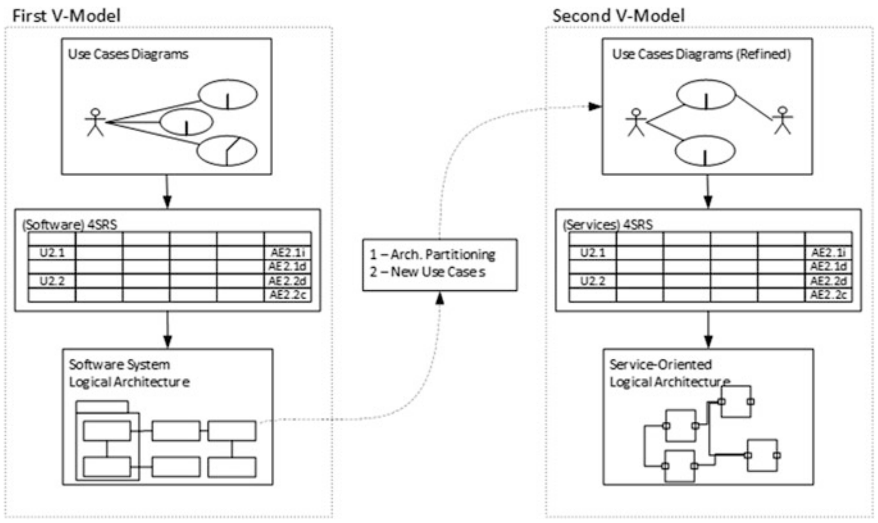


Fig. 8.2 Recursive architectural model transformations for Fog-based service design

The logical architecture is then refined trough successive 4SRS iterations (by recurring to tabular transformations), producing progressively more detailed requirements and design specifications. An overview of the approach is depicted in Fig. 8.2.

We propose the architecture design to be performed and refined iteratively, where each iteration regards the refinement of a given subset of the architecture. Afterward, the approach provides the tools to derive the services for that subset of the architecture and the required inputs for specifying the services. The result of this approach is the services specification related to all software features in SoaML diagrams for a Fog Computing solution. Additionally, the use of the 4SRS in this approach also allows for new features and services to be added.

For understandability purposes regarding the used architectural views—the software logical and the refined services logical—they are contextualized regarding the abstraction level, purpose, and associated phase (Fig. 8.3). For the abstraction, architectures are contextualized under the levels of OMG's model-driven architecture [24], where both are classified as Platform-Independent Model (PIM). Kruchten's 4+1 model [25] allows to contextualize the views, where the software logical architecture relates to the Logical view and the refined service logical architecture relates to the Development view. Finally, both architectures are used, mainly, within the design phase of software development.

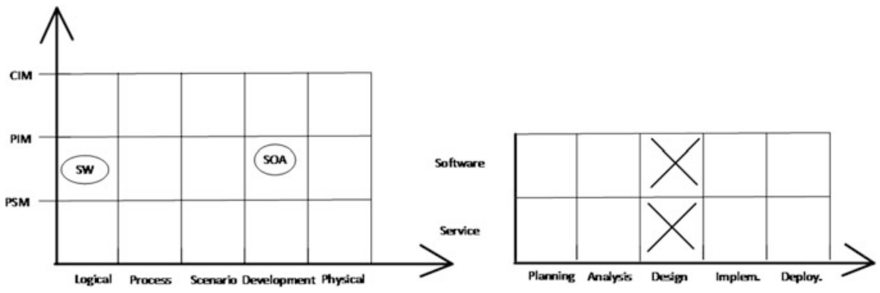


Fig. 8.3 Contextualization of the software and the refined services logical architecture by **a** the abstraction level and Kruchten's 4+1 model (on the left) and **b** the software development phase (on the right)

8.4 From Software System Logical Architecture Design to Services' Identification

This section describes the initial phase of the previously described approach. This initial phase starts by gathering high-level requirements and designing the logical architecture of the entire software system. Requirements are modeled in UML Use Case diagrams. The logical architecture is derived by performing the 4SRS method and modeled in UML Components diagram. Afterward, the architecture is partitioned so that only Fog functionalities are identified, creating context for further Fog's microservices specification.

8.4.1 Designing the Software System Logical Architectural View

This subsection relates to the first 4SRS method execution, where the goal is to design a software system logical architecture of the overall project.

At the inception stage of the requirements elicitation project, the business information that serves as input for requirements is gathered, e.g., business needs, project goals, vision document. Techniques like interviews, questionnaires, workshops are additional and complemental approaches of the aforementioned document analysis for gathering inputs on requirements. Reference models are also inputs for the requirements. Cloud Computing references such as NIST [26] are inputs for Cloud-related functionalities [21]. The Fog Computing issues are based on the architectural layers [14], e.g., business applications, Cloud management, Fog management, Fog infrastructure, and IoT systems.

The goal is to model requirements in UML Use Case diagrams, following functional decomposition techniques. The use of UML Use Case diagrams for specifying requirements is mandatory, since the 4SRS method for deriving the

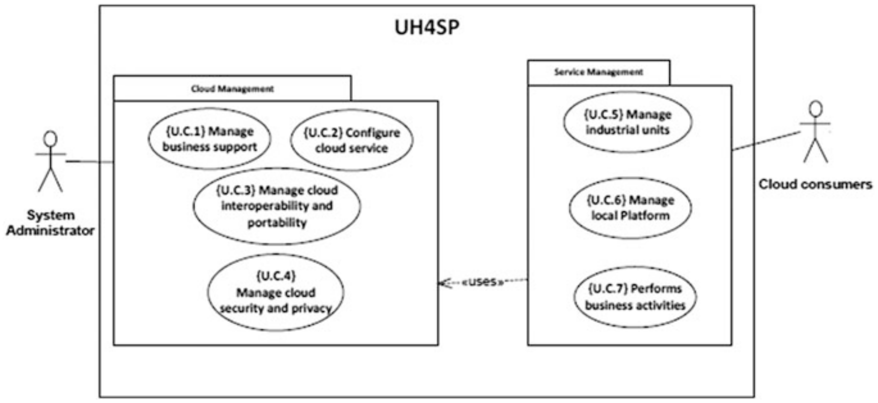

UH4SP
Cloud Management
{U.C.1} Manage business support
{U.C.2} Configure cloud service
{U.C.3} Manage cloud interoperability and portability
{U.C.4} Manage cloud security and privacy
Service Management
{U.C.5} Manage industrial units
{U.C.6} Manage local Platform
{U.C.7} Performs business activities
«uses»
System Administrator
Cloud consumers

Fig. 8.4 Recursive architectural model transformations for Fog-based service design

logical architecture uses Use Cases as input. Figure 8.4 depicts the first-level Use Cases, as follows:

- Use case {*UC.1*} *Manage business support*—relates to the management of Cloud services that support the UH4SP processes, *e.g.*, user accounts and profile configurations, customer licenses configurations, etc.
- Use case {*UC.2*} *Configure Cloud service*—relates to service configuration and monitoring, e.g., manage Cloud services, monitor platform, measure services utilization, configure deployment models and define service level agreements.
- Use case {*UC.3*} *Manage Cloud interoperability and portability*—relates to system interoperability and portability, where the required configurations are assured to synchronize data and integrate local (industrial units) information systems with the Fog and, afterward, with the Cloud.
- Use case {*UC.4*} *Manage Cloud security and privacy*—relates to Cloud security and privacy, e.g., manage backups, monitor activities, and configure data access control.
- Use case {*UC.5*} *Manage industrial units*—relates to the management of industrial unit information and configurations.
- Use case {*UC.6*} *Manage local Platform*—relates to the management of Fog environments, e.g., system admin assistance, IT resources management.
- Use case {*UC.7*} *Performs business activities*—relates to a set of functionalities performed within the industrial units, involving acquisition from equipment, augmented reality technology, and other technological support within the business processes.

At the end, the Use Case model is composed by 37 Use Cases after the decomposition and encompasses functional requirements related to the local IoT communications, Fog-level management, Cloud-level management, and business (Web and mobile) apps. A subset of the refined Use Cases is depicted in Fig. 8.5.

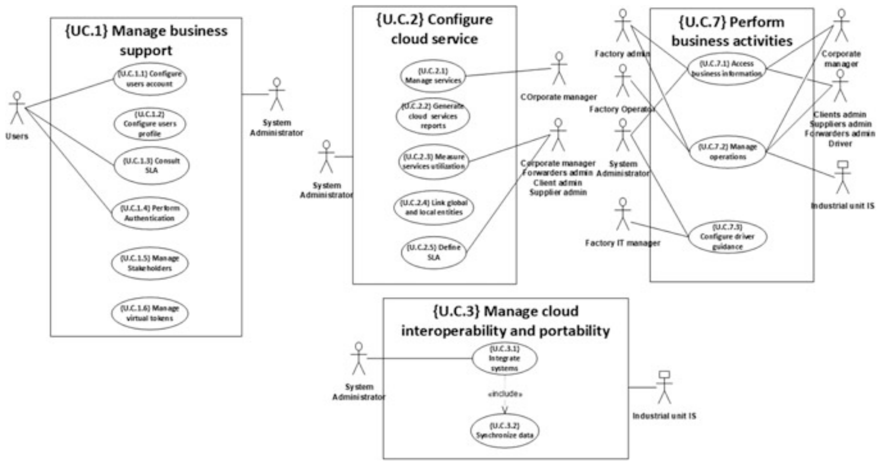
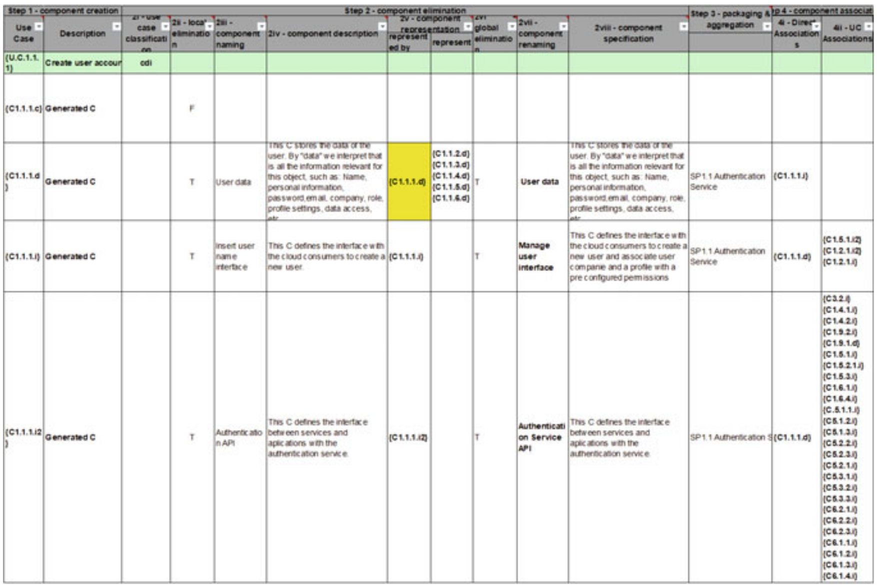

{UC.1} Manage business support
{U.C.1.1} Configure users account
{U.C.1.2} Configure users profile
{U.C.1.3} Consult SLA
{U.C.1.4} Perform Authentication
{U.C.1.5} Manage Stakeholders
{U.C.1.6} Manage virtual tokens
Users
System Administrator
{U.C.2} Configure cloud service
{U.C.2.1} Manage services
{U.C.2.2} Generate cloud services reports
{U.C.2.3} Measure services utilization
{U.C.2.4} Link global and local entities
{U.C.2.5} Define SLA
COrporate manager
Corporate manager Forwarders admin Client admin Supplier admin
{U.C.7} Perform business activities
{U.C.7.1} Access business information
{U.C.7.2} Manage operations
{U.C.7.3} Configure driver guidance
Factory admin
Factory Operator
System Administrator
Factory IT manager
Corporate manager
Clients admin Suppliers admin Forwarders admin Driver
Industrial unit IS
{U.C.3} Manage cloud interoperability and portability
{U.C.3.1} Integrate systems
«include»
{U.C.3.2} Synchronize data
System Administrator
Industrial unit IS

Fig. 8.5 Recursive architectural model transformations for Fog-based service design

Having all the requirements elicited, gathered, modeled, and validated, these Use Cases can now be used as input for the logical architecture derivation. The 4SRS method supports the functional decomposition of requirements and derives a logical architecture composed by UML Components that relate to each functionality [2, 27].

The execution of the 4SRS transformation steps can be supported in tabular representations. Moreover, the usage of tables is a tool so that the transformations can be partially automated. These tabular representations constitute the main mechanism to automate a set of decision-assisted model transformation steps. The cells are filled with the set of decisions that were taken and made possible the derivation of a logical architecture for the Cloud design. A small part of the 4SRS method execution table is represented in Fig. 8.6. Each column of the table concerns a step/micro-step of the method execution.

The first step regards the creation of software architectural components. At this point, the method execution is agnostic if the components are to be executed within the business apps, Cloud, Fog, or local communication level. It is agnostic since at this point the requirements relate to user needs and only after the refinement the technical aspects regarding Cloud, Fog, IoT, and apps are addressed. The 4SRS method associates, to each component found in analysis, a given category [2] referring to interface, data, or control. Interface components refer to interfaces with users, software, or other entities (e.g., devices); data components refer to generic repositories (data), typically containing the type of information to be stored in a database; control components refer to a control component, where the business logic and programmatic processing are included. This categorization makes the architectures derived by the 4SRS to be compliant with architectures from object-oriented programming or by Model-View-Controller (MVC) and Entity-Boundary-Controller (EBC) patterns.



Step 1 - component creation		Step 2 - component elimination									Step 3 - packaging & aggregation	Step 4 - component association	
Use Case	Description	2i - use case classification	2ii - local elimination	2iii - component naming	2iv - component description	2v - component representation		2vi global elimination	2vii - component renaming	2viii - component specification		4i - Direct Associations	4ii - UC Associations
						represented by	represent						
{U.C.1.1.1}	Create user accour	cdi											
{C1.1.1.c}	Generated C		F										
{C1.1.1.d}	Generated C		T	User data	This C stores the data of the user. By "data" we interpret that is all the information relevant for this object, such as: Name, personal information, password,email, company, role, profile settings, data access, etc	{C1.1.1.d}	{C1.1.2.d} {C1.1.3.d} {C1.1.4.d} {C1.1.5.d} {C1.1.6.d}	T	User data	This C stores the data of the user. By "data" we interpret that is all the information relevant for this object, such as: Name, personal information, password,email, company, role, profile settings, data access, etc	SP1.1 Authentication Service	{C1.1.1.i}	
{C1.1.1.i}	Generated C		T	Insert user name interface	This C defines the interface with the cloud consumers to create a new user.	{C1.1.1.i}		T	Manage user interface	This C defines the interface with the cloud consumers to create a new user and associate user companie and a profile with a pre configured permissions	SP1.1 Authentication Service	{C1.1.1.d}	{C1.5.1.i2} {C1.2.1.i2} {C1.2.1.i}
{C1.1.1.i2}	Generated C		T	Authentication API	This C defines the interface between services and aplications with the authentication service	{C1.1.1.i2}		T	Authentication Service API	This C defines the interface between services and aplications with the authentication service	SP1.1 Authentication S	{C1.1.1.d}	{C3.2.i} {C1.4.1.i} {C1.4.2.i} {C1.9.2.i} {C1.9.1.d} {C1.5.1.i} {C1.5.2.1.i} {C1.5.3.i} {C1.6.1.i} {C1.6.4.i} {C.5.1.1.i} {C5.1.2.i} {C5.1.3.i} {C5.2.2.i} {C5.2.3.i} {C5.2.1.i} {C5.3.1.i} {C5.3.2.i} {C5.3.3.i} {C6.2.1.i} {C6.2.2.i} {C6.2.3.i} {C6.1.1.i} {C6.1.2.i} {C6.1.3.i} {C6.1.4.i}

Fig. 8.6 4SRS method execution using tabular transformations

In the second step, components are submitted to elimination tasks according to pre-defined rules. At this moment, the system architect decides which of the original three components (i, c, d) are maintained or eliminated taking into account the entire system. The components are first compared for its representation in the context of a Use Case and later compared for redundancy within the entire system. Additionally, each maintained component is named and textually described relating to its behavior.

In the third step, the remaining components (those that were maintained after executing step 2), for which there is an advantage in being treated in a unified process, should give the origin to aggregations or packages of semantically consistent components.

The final step refers to the associations between components. The method provides steps for identifying such associations based on descriptions from Use Cases, as well as from the own components during the components creation.

The UH4SP logical architecture had, as input, 37 Use Cases and, after executing 4SRS method, was derived with 77 architectural components that compose it (Fig. 8.7). Architectural elements were grouped into five major packages, namely:

- {*P1*} *Configurations*—related with system configurations, that is accounts configurations, services configurations, security configurations, and tasks configurations;
- {*P2*} *Monitoring*—related with system monitoring, for example measure services utilization, consult SLA, monitor activities, and others;

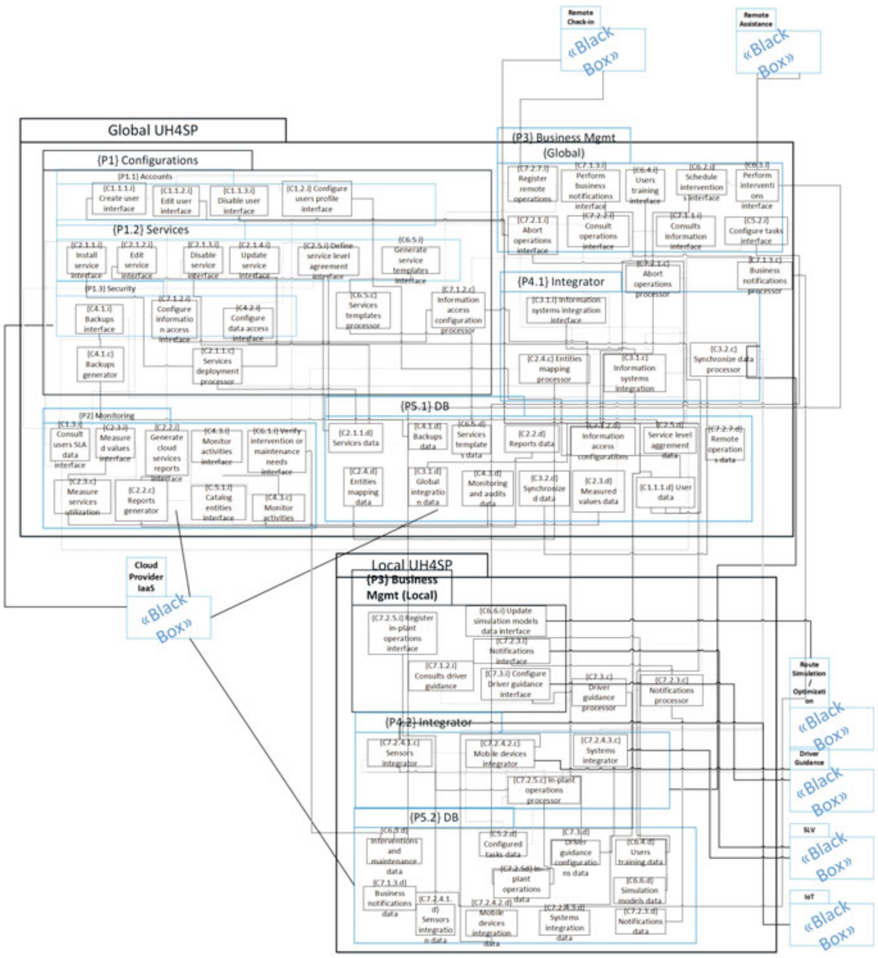


Fig. 8.7 UH4SP logical architecture derived after 4SRS execution

- {*P3*} *Business management*—related with logistic operations, information consults, simulation models, interventions and maintenance needs, and others;
- {*P4*} *UH4SP integration*—related with system integration between IoT systems and the Fog; this package is composed mostly by control components that integrate the various information systems, sensors, and mobile devices;
- {*P5*} *UH4SP Fog data*—related with temporarily stored data in Fog databases, storing all industrial local data from systems, sensors, mobile devices, and business processes, which are then synchronized to the Cloud and available to authorized stakeholders. It is not the purpose of this diagram in this paper to analyze its components, but rather to have a representation to follow the incremental design; thus, the figure is not zoomed.

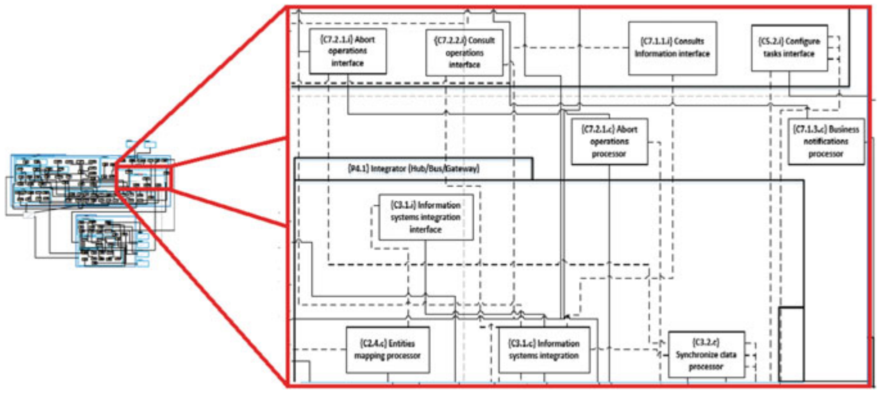

{C7.2.1.i} Abort operations interface
{C7.2.2.i} Consult operations interface
{C7.1.1.i} Consults Information interface
{C5.2.i} Configure tasks interface
{C7.2.1.c} Abort operations processor
{C7.1.3.c} Business notifications processor
{P4.1} Integrator (Hub/Bus/Gateway)
{C3.1.i} Information systems integration interface
{C2.4.c} Entities mapping processor
{C3.1.c} Information systems integration
{C3.2.c} Synchronize data processor

Fig. 8.8 Package {*P4*} UH4SP integration of the logical architecture, which refers to Fog management layer

The logical architecture is composed of components that can be related to the typical FaaS layers presented in [14], namely apps layer, Cloud services layer, Fog services layer, and IoT layer. Functional specification for components related to business apps is included in {*P3*} *Business management*. The Cloud services layer is specified by the components included in {*P1*} *Configurations*. Fog management layer is included in {*P4*} *UH4SP integration* (Fig. 8.8), which is also responsible for the integration with the IoT layer. Fog infrastructure services layer is included in {*P2*} *Monitoring*. {*P5*} *UH4SP Fog data* refers to the Fog hardware management.

8.4.2 Defining Cloud/Fog Applications and IoT Layers Within the Architecture

The partition initiates with the identification of architecture subsets (or modules). The identification of architecture subsets is dependent on the identification of software *concerns* [28]. If the concerns relate to layers, like in the case of Fog architectures [14], one possible way to define the architecture subsets is to base them in those layers, like business apps, Cloud management, Fog management, Fog infrastructure, and local IoT. Figure 8.8 depicts the architecture partition for the Fog layer.

In the case of the UH4SP project, it is composed of different entities for software development, where each entity was included in the project due to different expertise. The project team was composed as follows: (i) Cloud and Fog infrastructure experts, (ii) Cloud and Fog architecture experts, (iii) industrial software service development experts, (iv) mobile, image recognition, 3D, and augmented reality experts, and finally (v) industrial process simulation experts. Thus, the first

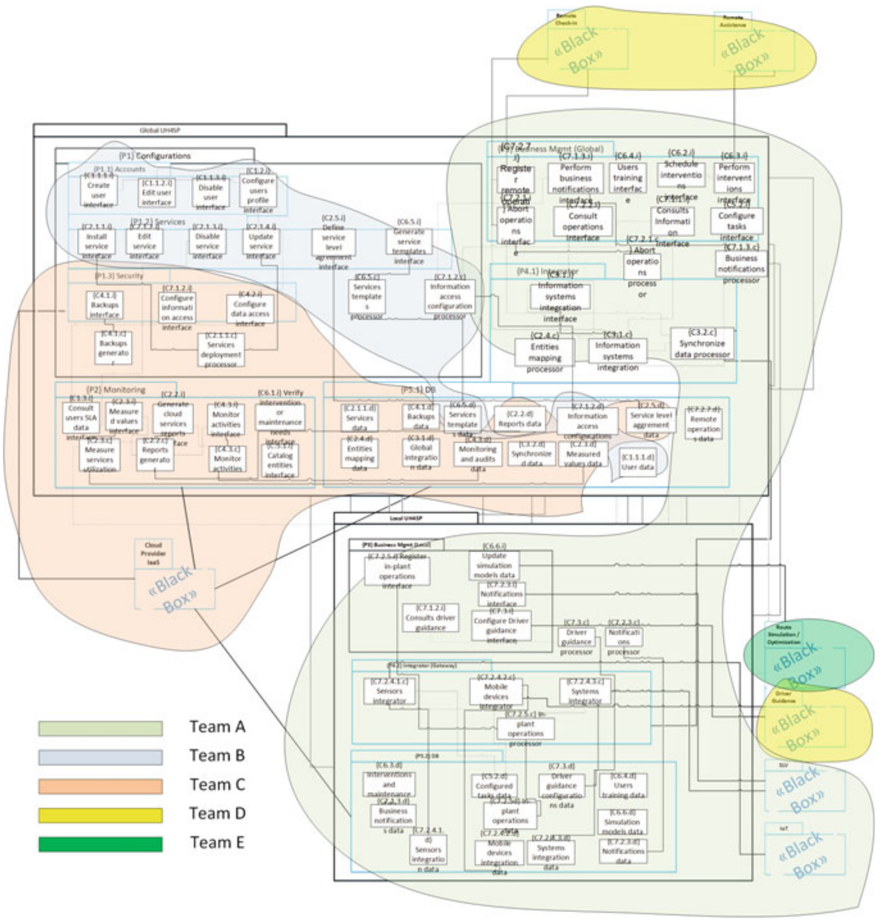


Fig. 8.9 Architecture modularization

concern in modularizing the architecture is in identifying architecture modules by the entities' core competencies (Fig. 8.9).

Each modularization may be refined. Flow between components (including components from different modules) is validated by modeling some processes. Dependencies can be depicted using, e.g., sequence diagrams as in [29], namely some functionalities that must be implemented and executable in order for other functionalities to properly execute. In fact, the variant sequence diagrams presented in [29] are powerful tools for bordering the modules, as well as validating (not just the modules but as well the whole) architecture.

It must be assured that a module has composing software components that, together, deliver working software. Only if the set of components are able to deliver

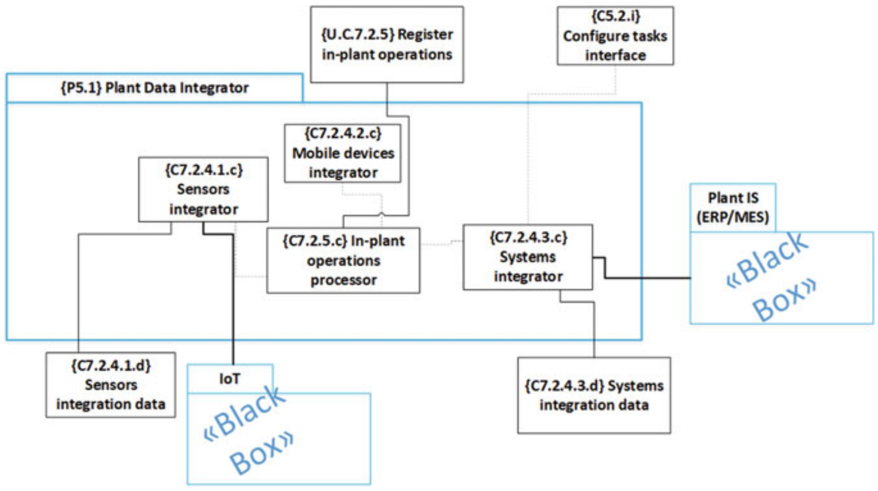

{U.C.7.2.5} Register in-plant operations
{C5.2.i} Configure tasks interface
{P5.1} Plant Data Integrator
{C7.2.4.2.c} Mobile devices integrator
{C7.2.4.1.c} Sensors integrator
{C7.2.5.c} In-plant operations processor
{C7.2.4.3.c} Systems integrator
Plant IS (ERP/MES)
«Black Box»
{C7.2.4.1.d} Sensors integration data
IoT
«Black Box»
{C7.2.4.3.d} Systems integration data

Fig. 8.10 UH4SP FOG-related sub-system

working software, it is also possible to perform acceptance testing and, afterward, integration testing within the code deployment.

We applied filtering and collapsing techniques as in [27, 28] to the border that relates to the architectural elements within the module. These techniques redefine the system boundaries, which now regards only the given module as a sub-system for design. During the filtering process, all entities not directly connected to the module must be removed from the resulting filtered diagram.

Inside the system border defined for the given module, through the respective coverage, the architectural elements were maintained as originally characterized. The architectural elements with direct connections to the module are maintained, and the ones without direct connection are removed. Figure 8.10 shows one of the UH4SP sub-systems, namely the one composed of Fog-related functionalities.

The transition from the software system logical architectural diagram to the service Use Case diagram is performed by applying defined rules [27]. An inbound (software) component is transformed into a (service) Use Case of the same type. By inbound, we mean that the element belongs to the partition under analysis. On the other hand, an outbound (software) component is transformed into an actor, representing an external software component that interacts with the (service) Use Case, for instance, through messaging or APIs. The service Use Case diagram referring to Fog management is depicted in Fig. 8.11.

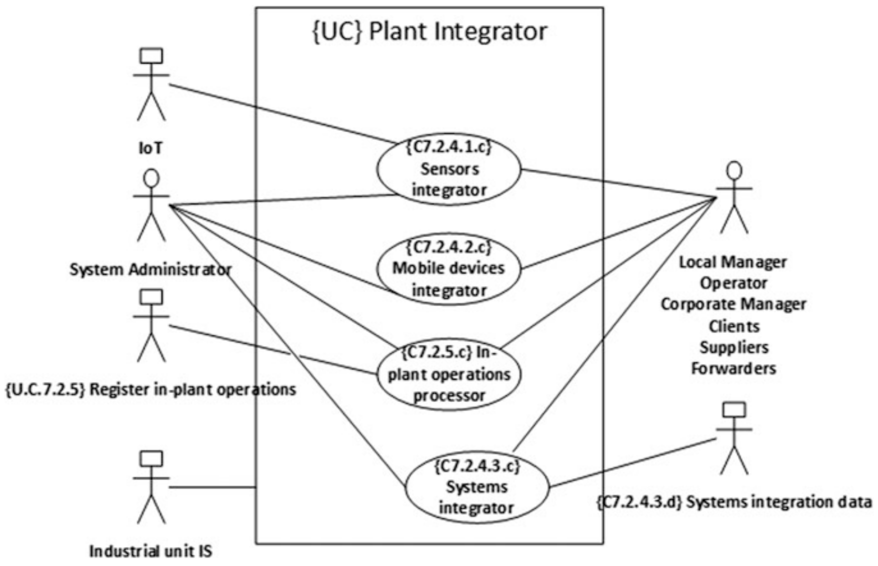

{UC} Plant Integrator
IoT
System Administrator
{U.C.7.2.5} Register in-plant operations
Industrial unit IS
{C7.2.4.1.c} Sensors integrator
{C7.2.4.2.c} Mobile devices integrator
{C7.2.5.c} In-plant operations processor
{C7.2.4.3.c} Systems integrator
Local Manager
Operator
Corporate Manager
Clients
Suppliers
Forwarders
{C7.2.4.3.d} Systems integration data

Fig. 8.11 Refined Use Cases resulting from the model transformation

8.5 Software Microservice Design for Fog Computing Architectures

In the previous section, the 4SRS method execution was agnostic. The second 4SRS method execution relates to specific contexts if the architecture is composed of different layers, like this case. If the architecture subset is related to business apps, the 4SRS as suggested by [2] is used for the Cloud management layer and the 4SRS method for Cloud design [29]. In this section, we present a 4SRS method execution for designing Fog Computing architectures. For modeling purposes, we propose the use of SoaML diagrams.

8.5.1 Specifying Services for the Fog Environment Using 4SRS

The Use Cases from Fig. 8.11 are then used as input for a recursive execution of the 4SRS [27]. The recursive execution of the 4SRS method is composed with the same steps as the previous execution, with the difference that at this point the addressed requirements relate to refined functionalities. This section describes how to use the 4SRS within the design of the microservices architecture for managing the Fog. With that purpose, the 4SRS is used to derive services based on Use Cases, similarly to [4]. The front-end apps used within the monitor and configuration of the

Fog infrastructure, namely its computation and storage, are not described since it is out of the presented partition. In this case, the 4SRS must derive typical software components rather than software services.

The 4SRS method associates, to each component found in analysis, a given category (interface, data, control). Even though a service is composed of interface, control, and data layers (if within a MVC pattern), the category relates to the main logic within the service. Like in the 4SRS presented in the previous section, services are submitted to elimination tasks according to pre-defined rules. The 4SRS method includes a micro-step related to the service behavior description. It includes a general purpose, the HTTP verbs under which the microservice is exposed to other services as a REST API, the invocation of HTTP methods required to consume services that are necessary in order to fulfill its purpose, and the properties that compose the dedicated database of the microservice. At the end of the 4SRS method execution, the output is a set of microservices.

8.5.2 Representation Using SoaML Diagrams

In the previous section, microservices were identified based on refined functionalities from recursive model transformations. In this section, we provide guidelines for using service behavior from the 4SRS method execution in SoaML diagrams, namely: Service Participants, Service Capabilities, Service Architecture, and Service Interfaces. The following inputs for SoaML diagrams are available: Participants, namely the identification of «request» (i.e., the invoked services) and «service» (i.e., the exposed REST API) ports and the required Service Interfaces; Capabilities, namely the included invoked methods required for providing the services and the classes that refer to the resources of the REST API; and Service Data, namely as it is able to provide the composition of the tables regarding the dedicated database of the service. The Service Data diagram was not addressed in this contribution; however, it is planned to be included in future research. The used inputs are the following:

- HTTP verbs (from the service description micro-step) relate to the API specification, to be included in the *Participant* and in *Service Channels* diagrams as a service port (i.e., service output) and finally used in order to define the "*provider*" role in *Service Interface* diagrams and *Service Architecture* diagrams.
- HTTP methods (from the service description micro-step) list of the methods called in order to fulfill the services purpose, included in the Service Participant and in the Service Capabilities.
- Properties (from the service description micro-step): data attributes related to the (shared or dedicated) microservice database, within the *Service Participant.*
- Associations (step 4 of 4SRS): service requests to other *Service Participants* within the microservices architecture, to be included as input port in the Participant and in *Service Channels* diagrams and finally used in order to define the microservice as a "*consumer*" role in *Service Interface* diagrams and *Service Architecture* diagrams.

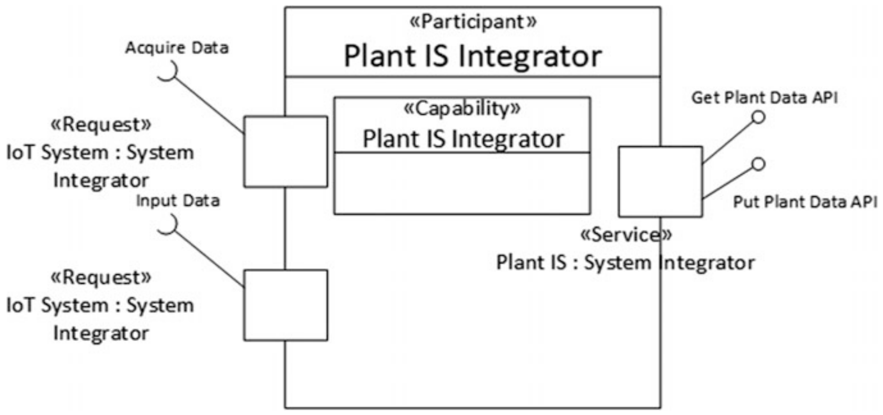


Fig. 8.12 Participant with ports, interfaces, and capabilities (methods/properties)

Each microservice identified within the 4SRS method execution is represented as a Service Participant. Thus, the set of Service Participants compose the microservices architecture. The required invocations for the Participant (Fig. 8.12) were identified based on the Use Case description, where the interactions with other use cases were previously described. Additionally, the same interactions allowed identifying the need for methods that call those services and the properties (data) within the Capabilities.

The Capabilities of a given service are applicable for a database per service scenario. On the other hand, Capabilities of services that include representation (micro-step 2.v) as well as the services with association (further in Step 4) are applicable for a shared database scenario.

A given ServiceChannel is related to a service association from the 4SRS (step 4). They allow that services can consume or provide other services. Within SoaML diagrams, these associations provide input for the Service Interface or Contracts between services, and a Service Architecture with the definition of provided and consumed services (Fig. 8.13).

The Service Architecture includes the other services that are consumed. Additionally, microservices are consumed by exposing an API. Hence, it is not specified in this diagram any particular service but rather the «*Microservice API*» stereotype of service interface (the specification of such extension to the metamodel is out of the scope of this paper).

This API is also referred in the Participant diagram, which supports the service ports *Get Plant Data API* and *Put Plant Data API*. The remaining consumed services are represented in the request port.

The Service Interface diagram (Fig. 8.14) refers to one of the interfaces included in the Service Architecture. It should be noted that the use of tools such as API

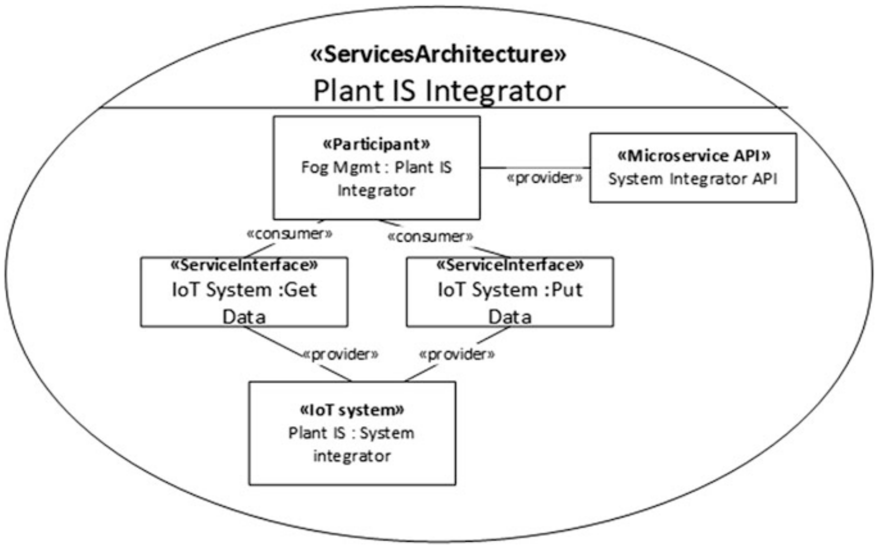


Fig. 8.13 Service architecture

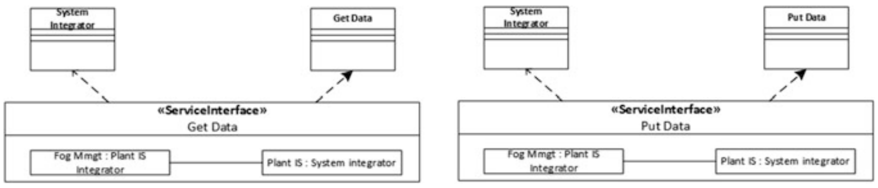

System Integrator
Get Data
«ServiceInterface»
Get Data
Fog Mmgt : Plant IS Integrator
Plant IS : System integrator
System Integrator
Put Data
«ServiceInterface»
Put Data
Fog Mmgt : Plant IS Integrator
Plant IS : System integrator

Fig. 8.14 Service Interface

Gateway or Enterprise Service Bus and the representation of the associations between services in such scenarios are out of the scope of this chapter.

8.6 UH4SP Fog Scenarios and System Overview

In terms of modeling, the services specification ends with the definition of a set of scenarios. By modeling a variant version of sequence diagrams as the ones presented in [29], which are composed by the architecture components rather than the classes, allows to validate the information flows defined by the services interfaces.

The sequence diagrams, in this section, refer to pilot scenarios performed within the UH4SP project, namely: (1) remote check-in of truck drivers via a mobile app; (2) remote business plant key performance indicators analysis; and (3) remote monitoring of plant equipment.

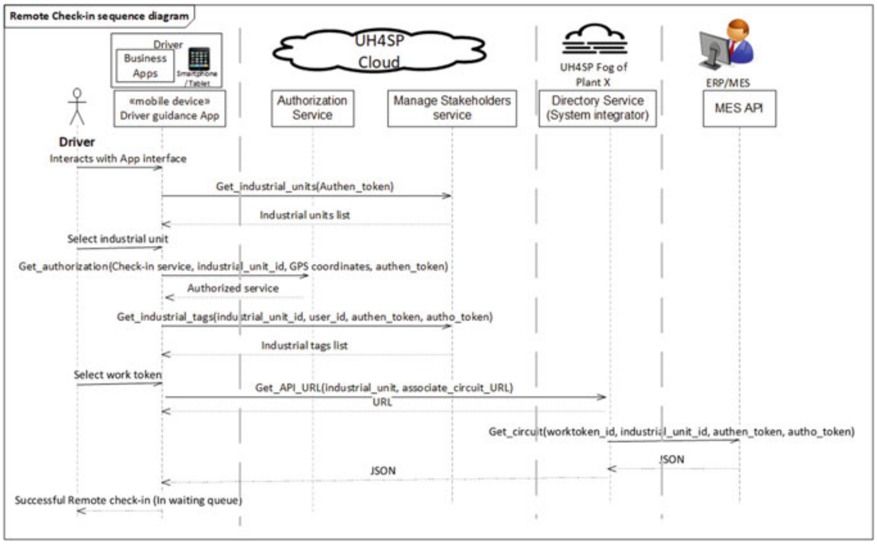
Additionally, after modeling efforts output components structuring, i.e., architecture and behavior, i.e., sequence diagrams, allowed creating context for depicting the system for the UH4SP project, namely business apps, Cloud, Fog, and IoT. A detailed discussion is presented at the end of this section.

8.6.1 Remote Check-In

The truck driver, in Fig. 8.15, accesses a mobile device and wishes to perform a remote check-in in a given industrial unit plant. In order to access the application, he has to introduce his user credentials (username and password). Afterward, the user chooses the plant that has to go to load or unload. However, this action requires validations from the local systems; e.g., it may be necessary for the ERP or MES to validate data of the driver, truck, company, or cargo (using an API). The driver waits until receiving notification that can be to enter in the plant, which is sent to the mobile app in a JSON format.

8.6.2 Remote Plant Business Analysis

A given stakeholder, as shown in Fig. 8.16, accesses the platform, via the Web or mobile app, and consults the business indicators. This consultation includes the



Remote Check-in sequence diagram
Driver
Business Apps
Smartphone /Tablet
UH4SP Cloud
UH4SP Fog of Plant X
ERP/MES
«mobile device» Driver guidance App
Authorization Service
Manage Stakeholders service
Directory Service (System integrator)
MES API
Driver
Interacts with App interface
Get_industrial_units(Authen_token)
Industrial units list
Select industrial unit
Get_authorization(Check-in service, industrial_unit_id, GPS coordinates, authen_token)
Authorized service
Get_industrial_tags(industrial_unit_id, user_id, authen_token, autho_token)
Industrial tags list
Select work token
Get_API_URL(industrial_unit, associate_circuit_URL)
URL
Get_circuit(worktoken_id, industrial_unit_id, authen_token, autho_token)
JSON
JSON
Successful Remote check-in (In waiting queue)

Fig. 8.15 Remote check-in sequence diagram

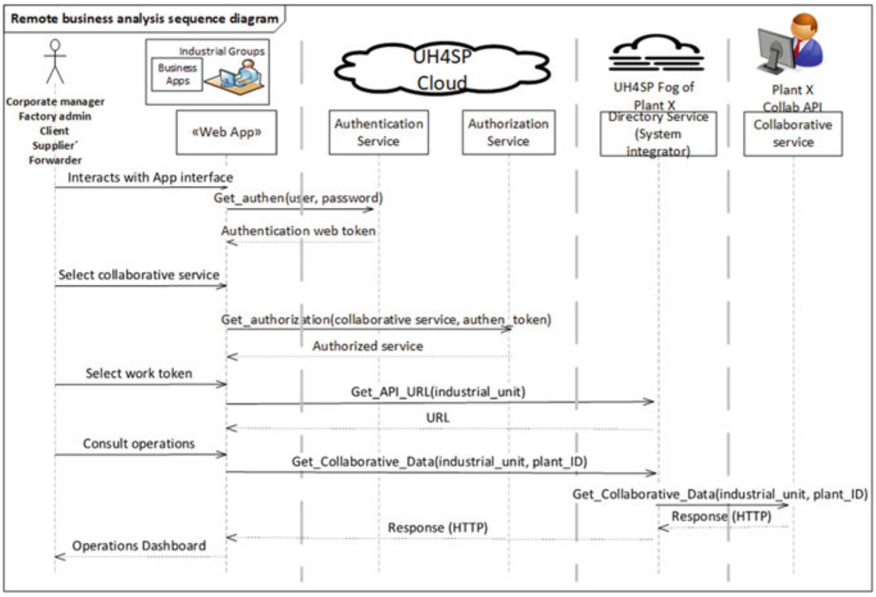

Remote business analysis sequence diagram
Corporate manager
Factory admin
Client
Supplier
Forwarder
Industrial Groups
Business Apps
«Web App»
UH4SP Cloud
Authentication Service
Authorization Service
UH4SP Fog of Plant X
Directory Service (System integrator)
Plant X Collab API
Collaborative service
Interacts with App interface
Get_authen(user, password)
Authentication web token
Select collaborative service
Get_authorization(collaborative service, authen_token)
Authorized service
Select work token
Get_API_URL(industrial_unit)
URL
Consult operations
Get_Collaborative_Data(industrial_unit, plant_ID)
Get_Collaborative_Data(industrial_unit, plant_ID)
Response (HTTP)
Response (HTTP)
Operations Dashboard

Fig. 8.16 Remote business analysis sequence diagram

visualization of waiting times/operations and deviations by trucks, product, local operations, etc. The visualized data depends on the user permissions.

A corporate manager consults the aggregate data about the corporator's plants. The plant admin only consults data about his plant. The forwarder consults data about his trucks and drivers operations. The suppliers and clients, represented by forwarders, consult the operations.

8.6.3 Remote Equipment Monitoring

In Fig. 8.17, the plant IT manager accesses the mobile or Web app and wishes to use the "equipment management service." After that, the user will have at his disposal all the equipment to which he has access to and that can be consulted.

8.6.4 Discussion on the Overall System

The deployment scenario is depicted in Fig. 8.18. Here, the Fog management layer-related services are deployed between the services referring to the local plant "things" layer and the Cloud services. Some business apps may communicate directly with the Fog layer.

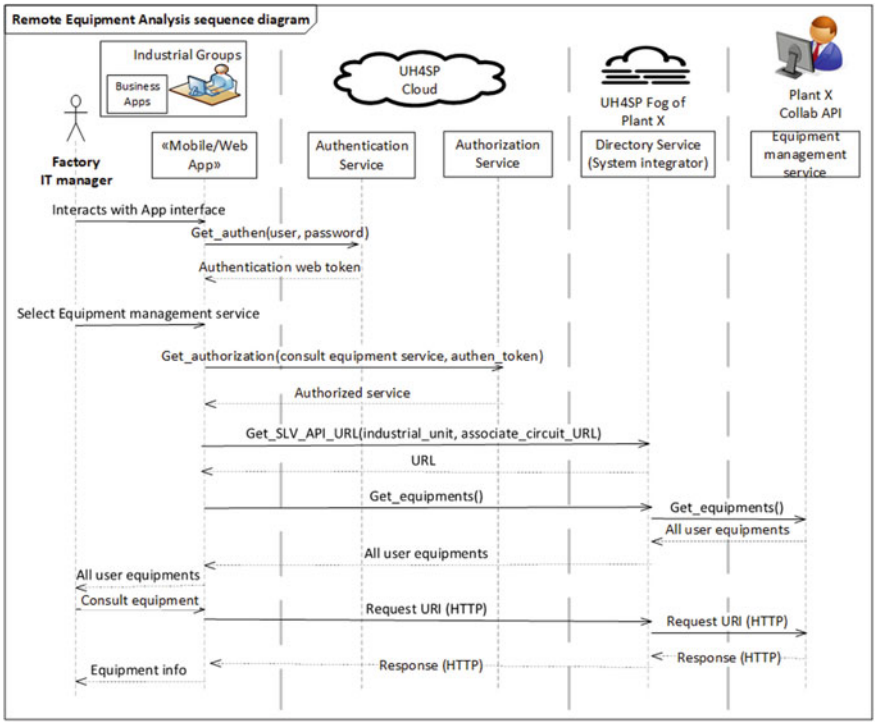


Fig. 8.17 Remote equipment analysis sequence diagram

The UH4SP business apps, either desktop Web apps (e.g., remote plant business analysis, remote equipment analysis) or mobile Web apps (e.g., remote check-in), are the main interfaces with human actors. For supporting the business processes, the apps perform invocations to microservices from the UH4SP Cloud mostly regarding required validations for performing those processes (e.g., authentication, tenant authorizations). After these validations, the apps perform invocations to Fog microservices from the intended plant. In the UH4SP project, the Fog services relate to middleware services such as integration, data synchronization, and service orchestration. Finally, these Fog services interoperate directly with the plant's IoT systems to support the remote executions, as described within the scenarios in the previous sections.

Finally, the UH4SP pictorial architecture is depicted in Fig. 8.19. Represented on the top are the business apps for the actors like the IT supplier, corporate manager, and third parties (e.g., forwarders, drivers, suppliers, clients) to perform business processes. These apps use Cloud Computing services.

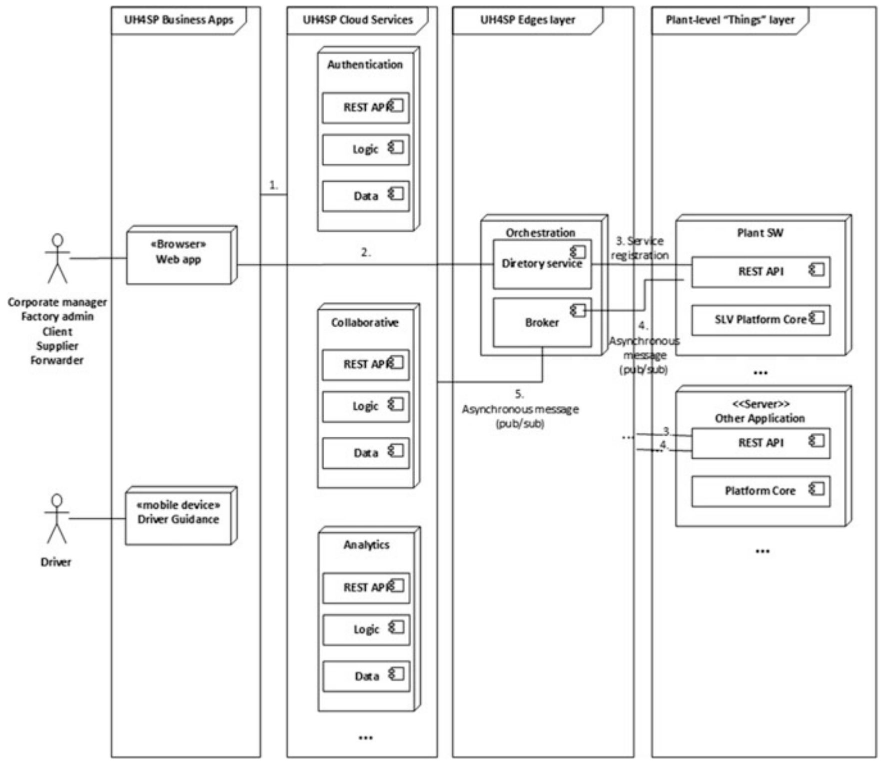


Fig. 8.18 Overview of the deployment

Specifically relating to the Fog Computing issues, this layer is composed of middleware microservices that enable the integration, synchronization, and orchestration of production data from the plant's IoT systems.

The Fog services are specified in order to be used within plants (mostly cement plants), whether the plant interoperates with Fog using IoT ("smart") systems and devices or just plant's software systems through APIs. They are also applicable if the data relates to production in cement plants or in intermediary warehouses (specially dedicated to logistics processes).

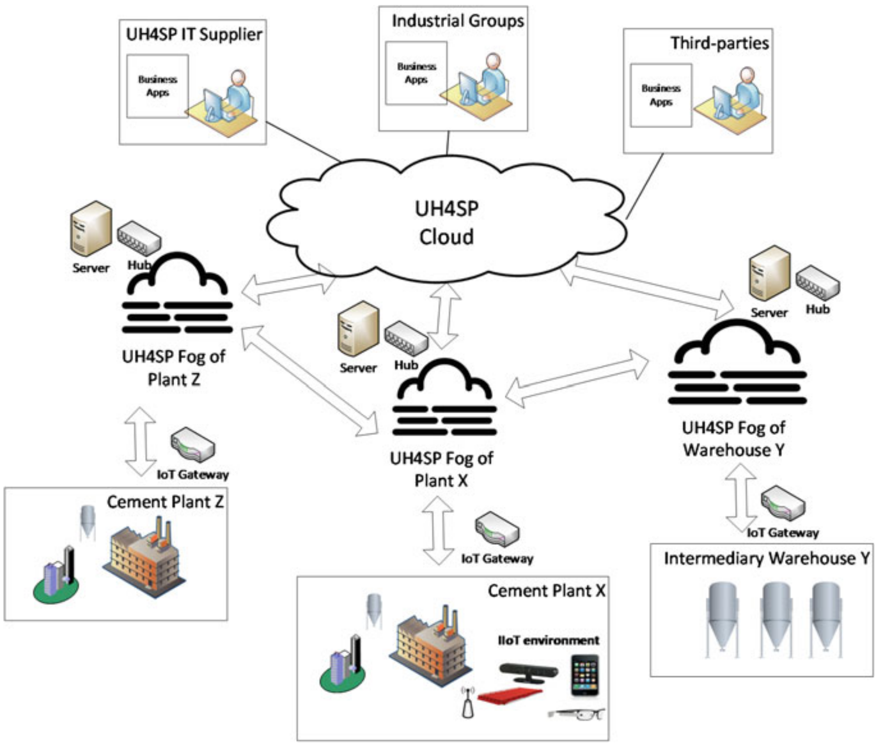


Fig. 8.19 Pictorial representation of the UH4SP Fog context

8.7 Conclusion

Fog Computing architecture design relies very heavily on interoperability, as this technology executes as intermediary integration between the Cloud Computing and the IoT systems. Although complex, Fog architecture design lacks of systematized support of model-driven approaches.

In this chapter, we proposed an approach for using models in designing Fog Computing architectures, starting with UML Use Cases for user requirements, using the 4SRS method to derive a software logical architecture in UML Components and a recursive model transformations based again on the 4SRS method to refine the service specification and its modeling in SoaML diagrams.

The first execution of the 4SRS produces a software logical architecture based on the elicited user requirements, without having concerns of refining technical requirements for executing in the Cloud or with Fog technologies. By using recursive model transformations, the software components are input for identifying service Use Cases which, after a second execution of the 4SRS, derives logical architecture composed with refined information, more suitable for specifying services. The Fog layer within this approach is based on a microservice architecture, modeled as SoaML diagrams.

In the case of the UH4SP project, the Fog Computing mainly relates to middleware services. It was addressed in this chapter in a microservices approach. In the future, computation and virtualization components from the Fog layer will be addressed in UML, SoaML, or any other modeling language that is suitable.

The SoaML diagrams are used as complimentary views for modeling the Fog applications, software services, communications, and data flows between the software, the IoT environment, and the CPS systems. Additionally, the use of models allows addressing interoperability at an abstract level, allowing clearly identifying the needs.

Acknowledgements This work was developed within the project UH4SP: Unified Hub for Smart Plants (Project ID 017871), under Portuguese National Grants Program for R&D projects (P2020—SI IDT), COMPETE: POCI-01-0145-FEDER-007043 and FCT—Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

1. Vaquero LM, Rodero-Merino L, Gradiant Vigo L (2014) Finding your way in the fog: towards a comprehensive definition of fog computing. ACM SIGCOMM Comput Commun Rev 44:27–32
2. Machado RJ, Fernandes, JM, Monteiro P, Rodrigues H (2005) Transformation of UML models for service-oriented software architectures
3. Santos N, Ferreira N, Machado RJ (2017) Transition from information systems to service-oriented logical architectures: formalizing steps and rules with QVT. In: Ramachandran M, Mahmood Z (eds) Requirements engineering for service and cloud computing. Springer, Cham, pp 247–270
4. Salgado CE, Teixeira J, Santos N, Machado RJ, Maciel RSP (2015) A SoaML approach for derivation of a process-oriented logical architecture from use cases. Explor Serv Sci 201:80–94
5. OMG (2012) Service oriented architecture Modeling Language (SoaML) Specification
6. Lewis J, Fowler M (2014) Microservices: a definition of this new architectural term. https://martinfowler.com/articles/microservices.html
7. de Brito MS, Hoque S, Magedanz T, Steinke R, Willner A, Nehls D, Keils O, Schreiner F (2017) A service orchestration architecture for Fog-enabled infrastructures. In: 2017 second international IEEE conference on fog and mobile edge computing (FMEC), pp 127–132
8. Ai Y, Peng M, Zhang K (2017) Edge computing technologies for internet of things: a primer. Digit Commun Netw
9. OpenFog Consortium Architecture Working Group (2017) OpenFog reference architecture for fog computing
10. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile Cloud computing—MCC '12. p 13. ACM Press, New York, NY, USA
11. Banafa A (2014) What is fog computing? https://www.ibm.com/blogs/Cloud-computing/2014/08/fog-computing/
12. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput 8:14–23

13. Patel M, Hu Y, Hédé IBM P, Joubert J, Thornton C, Naughton B, Roldan Ramos J, Chan C, Young V, Jin Tan S, Lynch D, Docomo N, Abeta S, Chen L, Shimizu Vodafone K, Neal A, Cosimini P, Pollard A, Klas G (2014) Mobile-edge computing—Introductory Technical White Paper. https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_introductory_technical_white_paper_v118-09-14.pdf
14. Byers CC (2017) Architectural imperatives for fog computing: use cases, requirements, and architectural techniques for fog-enabled IoT networks. IEEE Commun Mag 55:14–20
15. Datta SK, Bonnet C, Haerri J (2015) Fog computing architecture to enable consumer centric Internet of Things services. In: 2015 international symposium on consumer electronics (ISCE), IEEE, pp 1–2
16. Pahl C, Lee B (2015) Containers and clusters for edge cloud architectures—a technology review. In: 2015 3rd international conference on future internet of things and cloud, IEEE, pp 379–386
17. Oasis (2013) Topology and orchestration specification for cloud applications (TOSCA)
18. Bergmayr A, Troya J, Neubauer P, Wimmer M (2014) UML-based cloud application modeling with libraries, profiles, and templates. In: CloudMDE@ MoDELS, pp 56–65
19. Guillén J, Miranda J, Murillo JM, Canal C (2013) A UML profile for modeling MultiCloud applications. In: European conference on service-oriented and cloud computing, Springer, Berlin, Heidelberg, pp 180–187
20. Brandtzæg E, Mosser S, Mohagheghi P (2012) Towards CloudML, a model-based approach to provision resources in the Clouds. In: 8th European conference on modelling foundations and applications (ECMFA), pp 18–27
21. Pereira A, Machado RJ, Fernandes JE, Teixeira J, Santos N, Lima A (2014) Using the NIST reference model for refining logical architectures. In: International conference on computational science and its applications, Springer, Berlin, Heidelberg, pp 185–199
22. Teixeira J, Salgado C, Machado RJ (2016) Modeling an IaaS broker based on two cloud computing reference models. In: IEEE international conference on cloud engineering workshop (IC2EW), IEEE, pp 166–171
23. Evans E (2004) Domain-driven design : tackling complexity in the heart of software. Addison-Wesley, Boston
24. OMG (2003) MDA guide version 1.0.1
25. Kruchten P (1995) The 4+1 view model of architecture. IEEE Softw 12:42–50
26. Mell P, Grance T (2009) The NIST definition of cloud computing
27. Machado RJ, Fernandes J, Monteiro P, Rodrigues H (2006) Refinement of software architectures by recursive model transformations. http://dx.doi.org/10.1007/11767718_38
28. Ferreira N, Santos N, Machado RJ (2014) Modularization of logical software architectures for implementation with multiple teams. In: 2014 14th international conference on computational science and its applications, IEEE, pp. 1–11
29. Ferreira N, Santos N, Machado RJ, Fernandes JE, Gasevic D (2014) A V-model approach for business process requirements elicitation in cloud design. Adv Web Serv, 551–578

Chapter 9
A Data Utility Model for Data-Intensive Applications in Fog Computing Environments

Cinzia Cappiello, Pierluigi Plebani and Monica Vitali

Abstract Sensors, smart devices, and wearables have been widely adopted in recent years, bringing to the production of a vast amount of data which can be shared among several applications as input for their analysis. Data-intensive applications can benefit from these data but only if data are reliable and timely, and if they fit the requirements of the application. Designing data-intensive applications requires a trade-off between the value obtained by the analysis of the data, which is affected by their quality and volume, and the performance of the analysis that can be affected by delays in accessing the data and availability of the data source. In this chapter, we present a *Data Utility model* to assess the fitness of a data source with respect to its usage in a data-intensive application running in a Fog Computing environment. In this context, data are provided using a Data-as-a-Service (DaaS) approach, and both data storage and data processing can be placed in a cloud resource as well as in an edge device. The placement of a resource affects the quality of the service and the data quality as well. On this basis, the Data Utility model provides a support for making decisions on the deployment of data-intensive applications according to the impact of the task location, and on the selection of proper data sources as input for the application according to the application requirements, taking into consideration that both tasks and data can be moved from the edge to the cloud, and vice versa, to improve the efficiency and the effectiveness of applications.

9.1 Introduction

Due to the huge amount of data that are continuously produced by the smart devices in the Internet of things (IoT) environment, more and more data-intensive applications are required to properly manage these data. Indeed, data need to be stored,

C. Cappiello · P. Plebani · M. Vitali (✉)
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy
e-mail: monica.vitali@polimi.it

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_9

retrieved, processed, and provided in an efficient way. This means that the data-intensive applications need to be scalable, reliable, widely accessible, also considering possible privacy restrictions.

Currently, several tools focusing on different aspects have been proposed, for instance, HDFS [1] to provide a distributed file system, NoSQL databases [2] (e.g. MongoDB [3], Cassandra [4]) to ensure more flexibility and reliability, new programming models (e.g. MapReduce [5]), as well as new architectures (e.g. Lambda Architecture [6]) to improve the scalability. It is worth noticing that most of the approaches proposed so far go in the direction of making the data processing more efficient. For this reason, most of the solutions are conceived to run on cloud resources as they provide an environment ensuring scalability, reliability, and security.

Nevertheless, if we consider not only the data processing but the entire data management life cycle, it is clear how the cloud environment cannot be considered the only option. In fact, especially in IoT (Internet of Things) scenarios, data are produced on the edge (e.g. machineries, deployed sensors), then moved to the cloud where they are stored, processed, and, finally, sent back to the same place where they are generated to, for instance, better configure the machinery that has been monitored, or to create reports used by the operators of those machines. This simple example highlights how data continuously move from the edge to the cloud and vice versa, and, due to the network infrastructure, a significant latency could be introduced.

A reaction to this situation has been to leave the data processing to where the data are produced. This is the Edge Computing [7], and it works perfectly when the data processing does not require a significant amount of resources, as they need to leave only on the edge of the network. At the same time, Fog Computing has been arising as a platform able "*to provide compute, storage, and networking services between cloud data centres and devices at the edge of the network*" [8]. Similarly, as proposed by the OpenFog Architecture [9], we can consider Fog Computing as the useful combination of Cloud and Edge Computing where these two paradigms seamlessly interoperate [10].

In such a scenario, the approach proposed in this chapter relies on the Service-Oriented Computing paradigm where data sets are exposed as Data as a Service (DaaS). This requires a proper description of the data to the final user where, in addition to the functional aspects, it includes the non-functional aspects concerning the location in which the data are stored, the format used, and the quality of such data. Since the user will base the decision of using or not this data source according to her/his needs, the more dimensions we put in the description the more the variable the user has to consider; thus, the less usable will be the approach. For this reason, we introduce the notion of Data Utility, defined as the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics. The developers' requirements are expressed in terms of functional (e.g. the application need to extract the data about the exams of the patients performed in the last year) and non-functional (e.g. latency, data completeness) aspects as well as constraints (e.g. data must be stored encrypted). The model of Data Utility has been discussed in [11].

If, on the one hand, users can take advantage of this concise description, on the other hand, a higher burden is left to the providers. Indeed, the heterogeneity of the devices living on the edge or on the cloud, the influence of the network, the inherent Data Utility of the data sources are dependent to each other, and the goal of this chapter is to model such dependencies.

The chapter is organized as follows. Section 9.2 gives an overview of the state of the art in this field. To better clarify the need and the usage of the Data Utility model, we use the running example introduced in Sect. 9.3. Section 9.4 discusses the conceptual model of data-intensive applications running in a Fog environment. Section 9.5 introduces Data Utility and the concepts related to Data Utility definition and evaluation. Section 9.6 presents the life cycle of the data sources and their Data Utility from the application developer and the data owner perspectives. Section 9.7 discusses the implication of Data Utility during the application deployment. Section 9.8 concludes the chapter outlining possible future directions.

9.2 Related Work

The concept of Data Utility has been used in several contexts. Various definitions have been provided in the literature. For the general IT context, Data Utility has been defined by Kock [12] as "*The relevance of a piece of information to the context it refers to and how much it differs from other similar pieces of information and contributes to reduce uncertainty*". The value of data has been also of interest in the business scenario, where Data Utility has been defined as "*business value attributed to data within specific usage contexts*" [13]. A more complex definition has been given by Hundepool et al. [14] for Data Utility in the statistics context: "*A summary term describing the value of a given data release as an analytical resource. This comprises the data's analytical completeness and its analytical validity*".

The common factor between all these definitions is that the utility of a data set cannot be considered independently from the context in which data are used. Defining the context is not trivial, and several elements can be included in its definition. Moreover, the concept of context can change dynamically, including new elements that were not relevant before. For instance, the concept of Data Utility has been applied in its early stage in the information economics area. As discussed in [15], the economic factors considered relevant for the assessment of Data Utility are the predicted benefits and the costs of: (i) the analysis of the dataset, (ii) using the results of the analysis, and (iii) building the algorithms for executing the analysis of the dataset.

An important characteristic of Data Utility relates to its variation with respect to the specific goal of the data analysis. As an example, in [16], Data Utility is analysed for data mining applications while in [17] the focus is on users' requirements. Moreover, Data Utility might be influenced by the quality of service and the quality of data. For instance, the relation between Data Utility and quality of service has been investigated in [18], which discusses Data Utility with a focus on energy efficiency of mobile devices in a mobile cloud-oriented environment. The issue of energy efficiency for discovering interrelations between the evaluation of the data value and the effectiveness of run-time adaptation strategies has been discussed in [19]. Similarly, the influence of data quality on Data Utility is considered in [20], where relevant quality dimensions (e.g. accuracy and completeness) are considered in relations with Data Utility.

Finally, Data Utility has also been analysed with respect to the relation between IT and business, and this has paved the way for associating Data Utility to business processes. In this context, Data Utility is defined as a measurement of the gain obtained by using a dataset inside an organization [21]. Moreover, [22] discusses which are the information quality requirements in order to obtain reliable results from the execution of business processes.

The aim of this chapter starts from the results available in the state of the art for providing a definition of Data Utility comprehensive of all the relevant aspects emerged by this analysis: the context in which data are used, the content of the data, and the execution environment. We also evaluate how different configurations in terms of resource and data placement can affect Data Utility by using the concept of data movement introduced in [23] as a strategy for improving Data Utility.

9.3 Running Example

Being Data Utility a context-dependent aspect, a proper discussion about it requires the definition of a running example. In particular, this chapter relies on a scenario related to smart buildings. In more detail, we consider a building in which each room is monitored with sensors which allow a computing infrastructure to collect information about humidity, brightness, and temperature. Sensors store this information at the edge of the network and make it available to several applications, which want to manage the smart building. Additional data sources can be provided as DaaS and integrated with the information collected by the sensors. As an example, weather data provided by external services can be used to support some analyses.

In this running example, we focus on a specific data-intensive application which uses as input all the illustrated data sources. The goal of the application is to analyse and improve the comfort in the building. To reach this goal, the application executes several tasks:

- *Task A—Ambient Sensing Alignment*: this task collects data coming from the temperature, humidity, and brightness sensors located in the rooms of the monitored building. These data are pre-processed to prepare them for further analysis (e.g. timestamps alignment and data cleaning).

- *Task B—Ambient Sensing Aggregation*: the pre-processed data obtained in the previous step are analysed to obtain some statistical information like maximum, minimum, and average value for each sensor. Also, data of similar sensors are compared to obtain a typical behaviour for each kind of sensor (e.g. average temperature in the rooms of the building). This statistical information is stored to be used in further steps of the application.
- *Task C—Data Enrichment and Prediction*: this task performs predictions about the status of the rooms in the building by using the data generated in the previous step integrated with external information about the weather of the zone or city in which the building is located.
- *Task D—Visualization Preparation*: this task is in charge of presenting the results of the analysis performed in the previous tasks to the final user through visualization tools.

The structure of the application and the data sources involved in the analyses performed by its tasks are shown in Fig. 9.1. Tasks are represented as boxes labelled with the names of the tasks. The order in which the tasks are executed is modelled as a flow, represented through arrows connecting the different tasks. Data sources (DS) are modelled as inputs for the tasks. Two data sources are depicted: sensors of the building (labelled as DS_B, where B stands for building sensors) representing streaming data collected from sensors, and databases containing weather data, labelled in the figure as DS_W, where W stands for weather data. It is worth noticing that the output of a task can be considered as an input for the following task. This input is depicted in the flow between task t_i to task t_j as $E_{i,j}$; i.e., the information flow between *Task A* and *Task B* is labelled as $E_{A,B}$. In Fig. 9.1, this exchange of information is depicted as a data object attached to the flow between two tasks with a dashed line and labelled as described.

Other relevant information about the input of the tasks is related to the owner of the data source: data can be generated by sensors controlled by the application administrator (e.g. sensors in the building are the input of *Task A*, referred as DS_B) or by an external DaaS (e.g. weather data are the input of *Task B*, referred as DS_W).

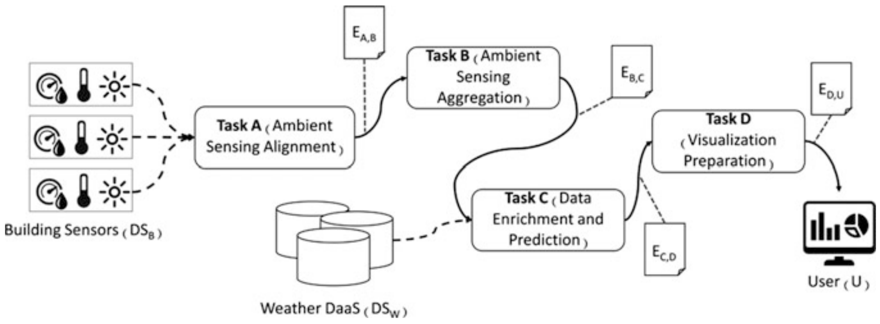

$E_{A,B}$
Task B (Ambient Sensing Aggregation)
$E_{B,C}$
$E_{D,U}$
Task A (Ambient Sensing Alignment)
Task D (Visualization Preparation)
Task C (Data Enrichment and Prediction)
$E_{C,D}$
Building Sensors (DS_B)
User (U)
Weather DaaS (DS_W)

Fig. 9.1 Example of data-intensive application in ambient intelligence domain (adapted from [11])

Moreover, the location of the data source is another element which affects the application (e.g. sensor data might be stored in an edge resource or in a cloud storage).

While for *Task A,* only a specific data source can be selected and identified in the sensors of a specific building, in other cases, alternative data sources might be associated with a task. This is the case of *Task C*, where the input is weather information, which can be retrieved by different DaaS providers fitting the requirements of the application. In this context, selecting the data source that better fits the requirements might be not trivial. In fact, several variables influence the efficiency and effectiveness of the task. One variable is related to the relative location of the task and the data source. As an example, placing *Task A* at the edge of the network might improve the execution of the task reducing the amount of data collected by the sensors in the building that need to travel from the edge to the cloud. Similarly, also *Task B* might be conveniently located in the edge. Another variable to consider is the type of resources available in the edge and in the cloud. In fact, computation in the edge is usually executed on devices with limited capabilities and performance, and this might have a negative impact on the application, creating a bottleneck. Finally, in case of alternative data sources, we might take into consideration that different data sources provide data with different quality of data (e.g. the different weather data sources could have different quality levels). The selection of a provider instead of another one eventually affects the quality of the analysis performed by the whole application.

All the described variables might be considered during the deployment of the application, making the work of the application developer really hard. To support someone in this job, Data Utility provides a complex metric reflecting to which extent a data source satisfies the requirements of an application. The introduction of Data Utility can reduce the effort of the application developer in the selection of the data sources and the tasks location.

9.4 A Model for Data-Intensive Applications in Fog Environment

Data-intensive applications are applications whose main goal is to manage a significant amount of information. They can be therefore defined by the data that they receive as an input, the processing steps executed to get results from them, and where the data are initially stored as well as where they are stored during the processing steps. According to this, data-intensive applications are usually modelled as a data flow between several processing steps. In this context, it is important to capture the relation existing between the data and the tasks operating on them. In [24], this relation is represented through a UML profile. A more refined attempt of modelling data-intensive applications is presented in [25], where they are modelled from different perspectives (from business to technical aspects).

With respect to the existing work, a peculiar aspect of the approach discussed in this chapter relies on the adoption of the Fog Computing paradigm for designing and running the application. As a consequence, the model adopted to define a data-intensive application mainly goes in the direction of specifying how Fog Computing may affect the design and the execution of this type of applications.

Fog Computing gets advantage of the capabilities of cloud extending them towards the edge of the network. In this way, the computation and data storage can be moved nearer to the final user, improving response time and reducing latency. The tools supporting Fog Computing hide the complexity and heterogeneity of the environment and data-intensive applications can consider cloud and edge devices as a continuum. From a designer standpoint, instead of specifying a precise deployment plan, it is more interesting to specify which are the characteristics that a node should, or must, have to run a task or to store data. At design time, movement of data and tasks between devices in the cloud or in the edge can be enacted to improve the efficiency and effectiveness of applications during their execution.

Based on these assumptions, and focusing only on the design standpoint that reflects the main goal of this chapter, we can model a data-intensive application as composed of three main elements: *resources*, *data sources*, and *tasks*. Figure 9.2 reports some of the characteristics of these elements with an emphasis on the relationships among them using the UML notation for class diagrams. In the model, a data-intensive application is represented as a composition of tasks and its related to resources and data sources. The three main elements are now described in details in the following paragraphs.

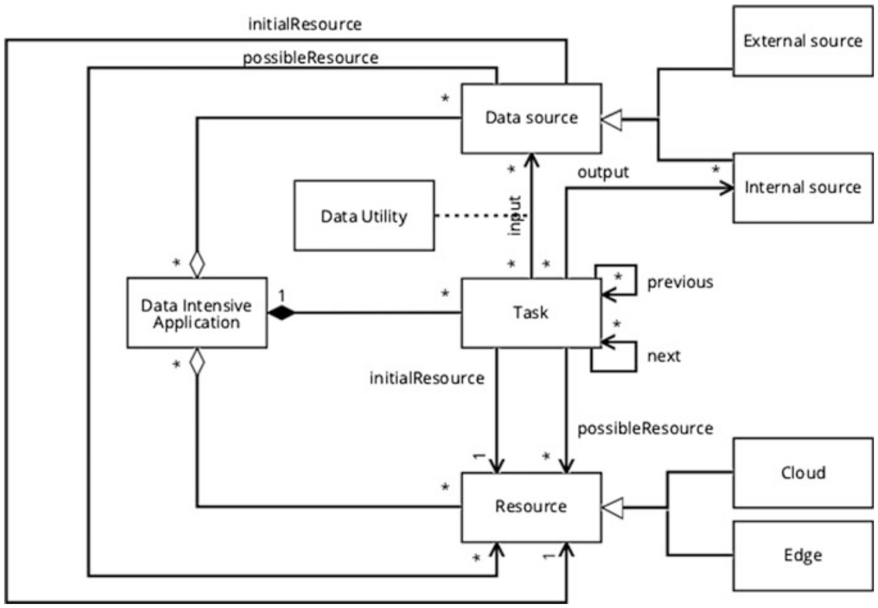


Fig. 9.2 Data-intensive application model (adapted from [11])

9.4.1 Resources

Resources are the nodes in which the application can be deployed and in which data can be stored. According to the Fog paradigm, resources can be placed either in the cloud or in the edge on heterogeneous devices, including virtual machines, sensors, laptops, and smart devices. The computational and storage capabilities of these nodes are of interest since they might affect the execution of the application and its performance. Knowing this information about a resource makes it possible to understand if it is suitable for hosting the application or the data consumed during its execution. According to this, when describing a resource, we need to define the computational capabilities in terms of number and frequency of CPUs, and the storage capabilities in terms of both RAM and disk capacity. The exhaustive description of the hardware features of a node is not of interest in this chapter, but standard approaches as the DMTF-CIM [26] can be adopted.

In Fig. 9.2, resources are classified in cloud and edge using the generalization notation of UML. In fact, this distinction is not only due to the location of the resource, but also to its features. First of all, one difference is related to the ownership of the resource. Usually, a cloud infrastructure is not managed by the application developer but belongs to an external entity, which provides only a limited set of information about, for instance, the location and features of the resource itself. For example, when deploying a VM on a cloud infrastructure, the cloud provider usually shares the information on the site in which the VM has been deployed, but it is not possible to access any information on the specific physical machine hosting it. Yet, the application developer has no control on migration of the VM from one server to another since this decision is managed by the cloud provider. In the case of an edge resource, we assume that the application developer can use only resources that s/he owns. According to this, the developer has a full knowledge and control on these resources and can reconfigure and adapt them during the execution. As an example, we assume that it is possible to adjust the sampling rate of a sensor for making it in line with the needs of the application.

9.4.2 Data Source

The second element of data-intensive applications that we want to analyse is the data source. The data source indicates the input of the application, including all the data required for executing the analysis and the relevant information written during the execution. Adopting the Service-Oriented Computing paradigm, data sources are exposed as DaaS; thus, we model them accordingly.

From a functional perspective, APIs can be used to define which are the type of data that the data source makes available [27], as well as information is related to how data can be accessed (i.e., endpoints). Being independent from a specific implementation, with APIs, it is possible to abstract from specific resources, and this is an advantage when dealing with heterogeneous environments as in the case of Fog Computing. In this way, the movement of a data source does not affect how the tasks access the content of the data source. Due to the different nature of data sources available, also different interactions are possible. We distinguish between conventional interactions, where data are accessed through queries, and stream interactions, which better reflect sources as sensors and monitoring systems, where data are continuously produced and updated.

A data source is stored in one of the resources described in the previous paragraph (either on the edge or the cloud), but it can be moved in other resources taking advantage of data movement in Fog environments. Following the approach in [24], data sources are classified into *internal sources* containing data that can be managed by the application designer and *external sources,* which are data provided by an external service. The former class includes also the data produced by the application during its execution and exchanged between its tasks, as well as data stored in edge and cloud devices but owned and/or managed by the application developer. In Fig. 9.1, examples of this kind of sources are the data produced by sensors (DS_B) and data exchanged among tasks ($E_{i,j}$). External sources are managed externally from the application, e.g. the weather data DS_W in the example.

Finally, a data source is associated with the resources. With *initial resource*, we indicate the resource in which a data source is deployed at the moment of the Data Utility evaluation. With *possible resources*, we indicate the set of storage nodes satisfying functional constraints of the data source (e.g. storage size and file system) in which the data source can be moved.

A data source described in this way can be associated with a Data Utility value expressing its relevance in a given context. Using the model for data-intensive applications introduced in this paragraph, Data Utility characterizes the association between a task and a data source providing the data. It depends both on the content of the data source and its location. As regards the content, it can be evaluated by considering the classical data quality dimensions, therefore, Data Utility is a multi-dimensional evaluation including attributes like accuracy, reliability, timeliness, precision, completeness [28]. As regards the location, Data Utility is affected by the resources involved in the deployment for both hosting the tasks and storing the data sources, and their location. For this perspective, Data Utility can be evaluated using classical quality of service dimensions (e.g., response time, availability, and throughput).

9.4.3 Tasks

A data-intensive application can be seen as a sequence of tasks, each one with inputs and outputs, each of them cooperating in the analysis of the data set for extracting the desired knowledge. As discussed in [25], a task is a unit of work with a duration in time and, in the context of data-intensive applications, we can identify a specific set of tasks, e.g. data cleaning, data integration, compression, and encryption. Typical tasks are associated with typical algorithms, but can be personalized and integrated with custom scripts. A data flow model can be used to express how data are acquired, transformed, and returned by the tasks composing the data-intensive application [29]. To represent the order in which tasks are executed, the flow of tasks is expressed using the *next* and *previous* attributes. Tasks are also associated with the data sources which contain their *inputs* and host their *outputs*. An output is always connected to a data source which is an internal source, since the data produced by the task are managed internally. On the contrary, an input can be stored both in an internal source (in case of data exchanged between tasks or data source managed by the application developer) or an external source (in case of a data source managed externally from the application).

Tasks are deployed on computational resources available in the described Fog environment. Similarly, to the data sources, a task is associated with the resource in which it is initially deployed (*initialResource*) and with a list of computational resources which are able to host the task (*possibleResources*).

9.5 Data Utility in Fog Computing Environments

As discussed in Sect. 9.2, Data Utility has been defined in different ways in the literature. All these definitions agree on its dependency on the context in which data are used. Therefore, the assessment of Data Utility is a complex issue since context can be composed of several elements, and it usually changes over time. Moreover, Data Utility aims to provide an indication of the relevance of data for the usage context that is defined in terms of system characteristics and application designer's goals. Therefore, before discussing Data Utility in detail, we present in the following the components of the usage context.

9.5.1 Usage Context

As stated above, the usage context depends on the status of the system and the application designer requirements. According to the proposed model introduced in Sect. 9.4, Data Utility is defined as an association class among tasks and data sources.

Starting from the available data sources, for each data source, it is important to know the data source schema (S_j), the initial resource $\left(\mathrm{ir}_j\right)$ on which it is currently deployed, and the set of possible resources on which it could be deployed $\left(\mathrm{PR}_j\right)$. Thus, the set of all the available data sources DS can be represented as:

$$\mathrm{DS} = \left\{\mathrm{ds}_j\right\} = \left\{\left\langle S_j, \mathrm{ir}_j, \mathrm{PR}_j \right\rangle\right\}$$

where ds_j is the single data source and each data source is defined by a data source schema S_j, an initial resource ir_j, and a set of potential resources PR_j.

Moving to the tasks that compose the data-intensive application, each task should be described by the application developer in terms of the following:

- a description of the operations performed by the task D_i (e.g. aggregation, filtering, clustering, association);
- the set of inputs and outputs IN_i and OUT_i;
- the position of the task within the data flow in terms of the set of tasks that precede and follow (P_i and N_i);
- the current resource ir_i on which the task is deployed and the resources on which it can be potentially deployed PR_j the analysed task in the data flow process.

Formally, we assume that each task t_i is defined by:

$$t_i = \langle D_i, \mathrm{IN}_i, \mathrm{OUT}_i, P_i, N_i, \mathrm{ir}_i, \mathrm{PR}_i \rangle$$

It is worth noticing that, as also depicted in Sect. 9.3, tasks may gather inputs from:

1. a specific data source (i.e. *Task A*);
2. a previous task (i.e. *Task B*);
3. a data source that should be selected from a set of candidate sources (i.e. *Task C*).

We can see the first and the second cases as situations in which the task relies on a single and known source. Conversely, the last case concerns the situation in which the application developer could need support in the selection of the sources.

To better distinguish among these situations and to also consider that a task may have several inputs, we can model the k-th input of the i-th task (IN_{ik}) as:

$$\mathrm{IN}_{ik} = \langle A_{ik}, \mathrm{CDS}_{ik} \rangle$$

where A_{ik} is the set of the attributes of the data source required by the task (e.g. temperature, humidity), and CDS_{ik} the set of candidate data sources from which data have to be extracted. Since candidate data sources are a subset of the data sources available according to the needs of the designer, we can say that $\mathrm{CDS}_{ik} \subseteq \mathrm{DS}$. Such set can include both internal and external sources. If the cardinality of this set is more than one, it means that the designer would like to be supported in the identification of the most suitable source among the ones available. In this case, for triggering the selection procedure a data request R_{ik} is created for capturing the application designer's goals, i.e. the elements that can affect the source selection. The data request R_{ik} is composed of three elements as follows:

$$R_{ik} = \left\langle \mathrm{IN}_{ik}, f^*_{ik}, \mathrm{NF}^*_{ik} \right\rangle$$

where

- IN_{ik} is the *input definition*: a task may require several inputs. For each input, the application designer has to specify the list of attributes that the task needs and the set of candidate data sources from which data have to be extracted;
- f_{ik}^* is the list of *functional r equirements*: the * indicates that it is an optional parameter used to express functional requirements. It can be seen as a predicate, composed of atoms linked by traditional logical operators (i.e. AND, OR) that allows the designers to specify restrictions over the allowed values in order to better drive the source selection (e.g. city = "Milan" AND Temp > 23);
- NF_{ik}^* is the list of *non-functional requirements*: also optional, they contain requests related to data quality, QoS, and security aspects.

Note that all the non-functional requirements can be expressed by the developers or automatically gathered considering the kind of task. Each non-functional requirement is an expression with which the related constraint to a DQ or QoS dimension is specified (e.g. data source completeness >0.9 or latency <10 s).

Quality requirements include:

- Data quality requirements: they focus on the quality of the content provided by the source;
- QoS requirements: they focus on non-functional properties such as availability, latency, or cost. Since the assessment of such properties mainly depends on the resources on which the task or the data are deployed/stored, the evaluation whether QoS requirements are met has to consider the placement of the tasks and data sources.

Therefore, for example, if an application wants to analyse the data related to the temperature values collected in a specific room of the building in the month of August, the data request can be the following:

- Date, time, temperature;
- Date BETWEEN 01/08/2017 AND 31/08/2017;
- latency <60 s AND accuracy >99% AND completeness >98%.

It is possible to enrich the request with additional non-functional constraints that can be derived by the type of task. For example, a task that performs data mining operations requires a high amount of data and high completeness. If the developer has not specified requirements on such dimensions, the system, by analysing the type of task, will add these constraints to improve the effectiveness of the source selection.

9.5.2 Data Utility

The fitness of a data source to the application developer's requirements is expressed through the evaluation of the Data Utility. This evaluation is led by the request R_{ik}

provided by the users together with the characteristics of the data source and of the task that requests it. For each data source provided by a DaaS, the Data Utility evaluation is performed to assess which is, among the alternative sources, the one better fitting the application developer's requirements. In particular, Data Utility evaluation has to take into account the capability of the source to satisfy the functional requirements of the task: the degree with which the source contains the data requested by the application:

- the capability of the source to satisfy non-functional requirements of the task: the degree with which the source is able to satisfy quality requirements;
- the reputation of the source: the degree with which the source is trustworthy.

More formally, we assume that the data sources are associated with a set of metadata that reveal the *Potential Data Utility* (PDU) that summarizes the capabilities of the data source and can be periodically evaluated independently of the context. The PDU is calculated by looking at the data and the characteristics of the data source. It is derived from a *data quality* and a *reputation* assessment. From a data quality perspective, it is important to highlight that errors, missing data, or updated data might significantly affect the usage and potential benefits of data. The assessment of data quality dimensions may contribute to the understanding of the potential value of the data. The list of dimensions (e.g. accuracy, consistency, completeness, timeliness) and the assessment metrics depend on the type of data contained in the source. For example, in case of data collected from sensors, precision and data stability constitute the two most relevant dimensions to take care of. Generally speaking, we assume that each source is associated with a set of data quality dimensions and related values. Besides the content, also, the usage of the source is considered and defined as reputation index. This index depends on the frequency with which the source has been successfully used and on the scope of data (e.g. generic or specific, be integrated with other sources, used with other sources).

A data source has to be evaluated by considering the context that in our scenario is composed of a data-intensive application and the available resources. QoS capabilities have to be evaluated by considering all the available options that the Fog environment offers. Thus, both tasks and data can be moved: (i) from edge to cloud, (ii) from cloud to edge, (iii) from edge to edge, and finally (iv) from cloud to cloud. Variation of the placement of a task or a data source on a specific resource has surely an impact on the QoS: in fact, the computational cost for obtaining data and the latency changes on the basis of the chosen location. For instance, it is reasonable to assume that the ambient sensing alignment task can be more efficient if it is executed closer to the sensor data to be aligned. As we assume that both data and task can be moved, we calculate the QoS dimensions for each possible configuration defined in terms of task placement associating a task t_i to the resource r_x in which it is deployed $(\langle t_i, r_x \rangle)$ and data placement associating a data source ds_j to the resource r_y where it is stored $\left(\langle \mathrm{ds}_j, r_y \rangle\right)$.

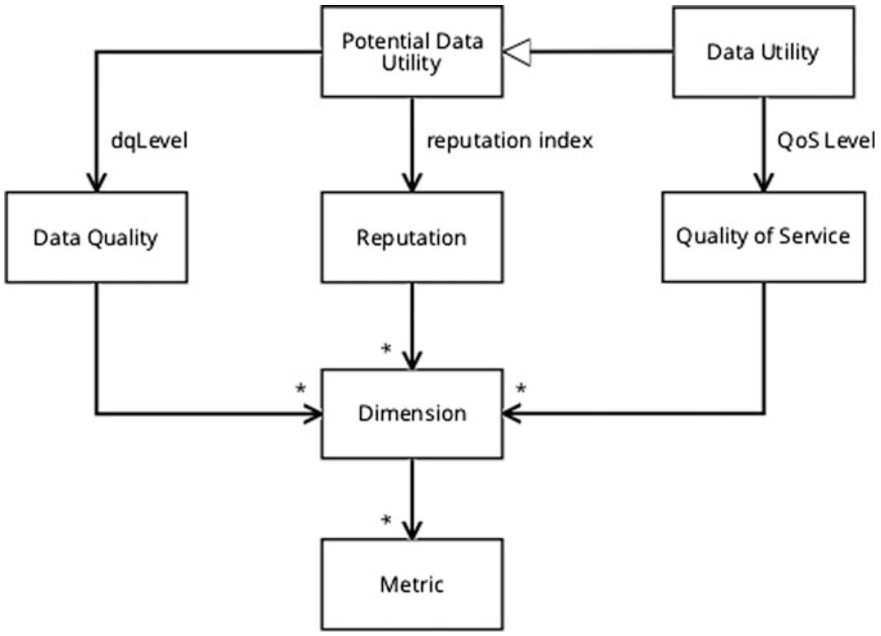


Fig. 9.3 Model of the utility components (adapted from [11])

Data Utility of a data source ds_j for a task t_i is defined as:

$$\mathrm{DU}_{ixjy} = f\left(\langle t_i, r_x\rangle, \langle \mathrm{ds}_j, r_y\rangle\right)$$

Both tasks and data sources, according to our data-intensive application model, can be placed on different resources belonging to PR_i and PR_j. Data Utility depends also on the task placement $(\langle t_i, r_x\rangle)$ and data sources placement $\left(\langle \mathrm{ds}_j, r_y\rangle\right)$ where $r_x \in \mathrm{PR}_i$ and $r_y \in \mathrm{PR}_j$.

In summary, as shown in Fig. 9.3, Data Utility can be assessed by considering three main aspects: data quality, reputation, and quality of service. Each of them is evaluated by means of *dimensions*, each one associated with different *metrics* (more than one assessment function might be available for a single dimension).

9.6 Data Life cycle

As Data Utility evaluation is dynamic and changes over time according to the variation in terms of its components (e.g. data quality, quality of service, and reputation), we can describe the life cycle of a data source from two perspectives: the DaaS perspective and the application developer's perspective. The former considers the data source in the traditional life cycle phases: from the data source creation to the disposal. The latter considers the data source as an object to access and manipulate.

Focusing on the DaaS perspective, the data life cycle is represented in Fig. 9.4. Firstly, the data source, in order to be managed by the platform, has to be registered, and the registration requires the definition of metadata that describe the source and its contents. Once that the source is registered, the Potential Data Utility can be evaluated as all the quality dimensions independent from specific context in which data are used (e.g. accuracy, completeness, consistency) can be assessed. In this way, the source is enriched with PDU metadata that are one of the main drivers for the selection of data with respect to the applications requests. After this phase, the source is available for use. Periodically, the PDU is re-evaluated and corresponding metadata are updated. All the criteria of the Data Utility are instead assessed when an application sends a request. Data Utility evaluation considers the different variants of the context on the basis of the location of the application and the data source. This means that, for each request, several Data Utility vectors are calculated and a ranking can be defined.

Moving to the application developer's perspective, the data life cycle starts with the submission of the data request in which the application developer specifies the data sources, and the functional and non-functional requirements (see Fig. 9.5). Once the set of valid data sources is identified on the basis of the functional requirements, each source is enriched with the utility scores that are evaluated considering the application and data source status. On the basis of these data, the

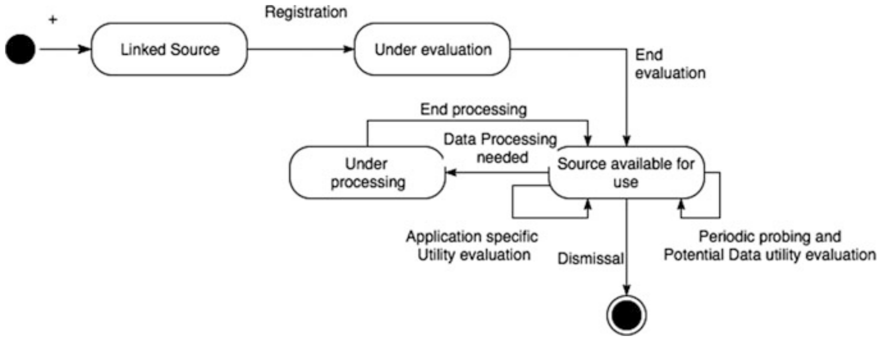


Fig. 9.4 Data life cycle from the DaaS perspective

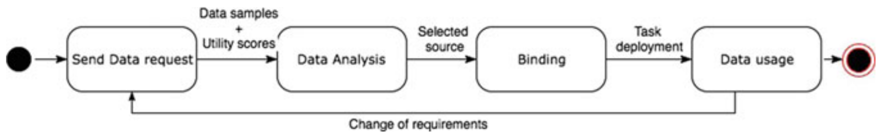


Fig. 9.5 Data life cycle from an application perspective

application developer makes the final decision and the selected data source is instantiated and bind to the application.

9.7 Using the Data Utility Model

Given a task and a data source composing the data-intensive application, the evaluation of the Data Utility considers all the possible configurations, in the edge or in the cloud, of both the data source and the task. At this stage, the Data Utility model returns an evaluation based on three main dimensions, i.e. reputation, data quality, and QoS. Thus, a tool supporting the designer by adopting the proposed Data Utility model can show, for each task and each configuration, where more than a data source is available (e.g. *Task C*), and how these three dimensions vary (see Fig. 9.6). In case a ranking of the different alternatives is required, different methods for identifying the best data source are available. They range from a simple aggregation (e.g. average or weighted average) of the different dimensions to more advanced techniques for multiple criteria decision analysis (i.e. MCDA methods).

Although the proposed model provides a significant and useful tool for data-intensive application designers to understand the impact of moving data and tasks with respect to the Data Utility, this evaluation focuses only on a task level analysis. For this reason, we advocate the need of a global Data Utility model that is able to capture the Data Utility of the entire application based on the selection of the different sources. At this stage, the definition of a Global Data Utility measure, that is the utility of the results provided to the final user, is under investigation, and here, we would like to outline which are the main elements to be considered.

First of all, we want to highlight that an increasing number of tasks and possible data sources imply an exponential increasing of the Data Utility evaluations required. This increment can be mitigated by the number of constraints that the designer can put to the requests in terms of allowed data movements. For instance, *Task A* and source DS_B are only considered in the edge because of a constraint forbidding to move the data produced by the sensors. For simplification, we assume

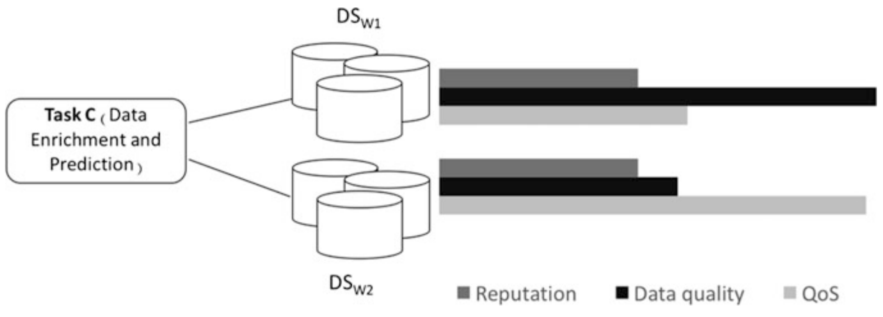


Fig. 9.6 Using Data Utility model to evaluate alternative data sources (adapted from [11])

that each task can be moved in the edge only if its predecessor is already in the edge (otherwise moving it does not provide any advantage since communication is not improved). This is true for our example, but not generally true when several tasks use data produced by edge devices.

To give an idea, referring to the example discussed in Sect. 9.3, the different possible configurations are summarized in Table 9.1, where only two weather data providers are considered: $\mathrm{DS_{W1}}$ and $\mathrm{DS_{W2}}$. The table is horizontally divided into four sections, each one representing a possible task deployment configuration, where on the right all the combinations about data source selection and placement is considered.

A second aspect to be considered concerns the mutual influences between tasks, making the data source selection and movement a complex decision. Two factors should be considered in the global Data Utility computation:

- *The Data Utility of a task t_i influences the Data Utility of a task t_j with t_i successor of t_j*. It means that the selection of a data source for task t_i has an effect on t_j which uses the output provided by t_i, both directly and indirectly. In the example depicted in Fig. 9.1, selecting a data source $\mathrm{DS}_{\mathrm{W}x}$ optimizing the Data Utility for *Task C* impacts on the Data Utility of *Task B* in which different requirements could make a data source $\mathrm{DS}_{\mathrm{W}y}$ more convenient for the global Data Utility. On the computation movement perspective, this dependency is also relevant. As an example, moving *Task B* to the edge may improve its Data Utility by allowing a faster access to its input (generated by *Task A*), but affects the performance of *Task C* for which a higher latency in data retrieval must be considered.

Table 9.1 Deployment alternatives in a Fog environment

	Task A	Task B	Task C	Task D	$\mathrm{DS_B}$	$\mathrm{DS_{W1}}$	$\mathrm{DS_{W2}}$
Depl. 1	Edge	Edge	Edge	Edge	Edge	Edge	–
Depl. 2	Edge	Edge	Edge	Edge	Edge	Cloud	–
Depl. 3	Edge	Edge	Edge	Edge	Edge	–	Edge
Depl. 4	Edge	Edge	Edge	Edge	Edge	–	Cloud
Depl. 5	Edge	Edge	Edge	Cloud	Edge	Edge	–
Depl. 6	Edge	Edge	Edge	Cloud	Edge	Cloud	–
Depl. 7	Edge	Edge	Edge	Cloud	Edge	–	Edge
Depl. 8	Edge	Edge	Edge	Cloud	Edge	–	Cloud
Depl. 9	Edge	Edge	Cloud	Cloud	Edge	Edge	–
Depl. 10	Edge	Edge	Cloud	Cloud	Edge	Cloud	–
Depl. 11	Edge	Edge	Cloud	Cloud	Edge	–	Edge
Depl. 12	Edge	Edge	Cloud	Cloud	Edge	–	Cloud
Depl. 13	Edge	Cloud	Cloud	Cloud	Edge	Edge	–
Depl. 14	Edge	Cloud	Cloud	Cloud	Edge	Cloud	–
Depl. 15	Edge	Cloud	Cloud	Cloud	Edge	–	Edge
Depl. 16	Edge	Cloud	Cloud	Cloud	Edge	–	Cloud

- *The Data Utility of a task t_j influences the Data Utility of a task t_i with t_i predecessor of t_j.* Maximizing the Data Utility means selecting the best coupling between a task and a data source according to the task requirements, while maximizing each components of the Data Utility model. As an example, focusing on source selection, if both $\mathrm{DS}_{\mathrm{W}x}$ and $\mathrm{DS}_{\mathrm{W}y}$ satisfy the task requirements, $\mathrm{DS}_{\mathrm{W}x}$ could be selected because of a better timeliness if compared with $\mathrm{DS}_{\mathrm{W}y}$. However, t_j requirements could be violated by $\mathrm{DS}_{\mathrm{W}x}$, and in this case, $\mathrm{DS}_{\mathrm{W}y}$ would be preferable.

Finally, providing techniques and heuristics for selecting the deployment configuration maximizing the global Data Utility for the user of the application is out of scope, but is an interesting challenge that might be considered in the future.

9.8 Concluding Remarks

Nowadays, the amount of available data sources is continuously increasing. This is mainly due to the fact that new technologies and applications allow us to transform many aspects of our life into digital data. Data-intensive applications analyse such data in order to understand them and support decision-making processes, the design of advanced services, or the optimization of existing processes.

However, having a high quantity of accessible data sources is not always an advantage for the application developers. In fact, they may experience some difficulties in selecting the appropriate source for their goals. For this reason, in this chapter, we have introduced the Data Utility concept that is able to evaluate the relevance of a data source along the usage context. The context is defined in terms of both the developers' requirements and the status of the system in which data sources and applications are stored and deployed. In particular, we define our approach by considering data-intensive applications running on a Fog environment. In such scenario, the location of both tasks and data sources can be changed by using cloud or edge resources; therefore, the influence of data and computation movement is also considered in the Data Utility definition. Currently, the presented contribution supports the Data Utility driven selection of the data sources at the task level. However, the evaluation of the optimal utility for an application as a whole is under investigation to consider the influences among tasks, as well as the constraints over the deployment.

Acknowledgements This research has been developed in the framework of the DITAS project. DITAS project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement RIA 731945.

References

1. Borthakur D (2008) HDFS architecture guide. Hadoop Apache Proj
2. Pokorny J (2013) NoSQL databases: a step to database scalability in web environment. Int J Web Inform Syst 9(1):69–82 (Mar 29)
3. Chodorow K (2013) MongoDB: the definitive guide: powerful and scalable data storage. O'Reilly Media, Inc
4. Cassandra A (2018) http://cassandra.apache.org (last accessed 26 Jan 2018)
5. Bhandarkar M (2010) MapReduce programming with apache Hadoop. In: IEEE international symposium on Parallel and distributed processing (IPDPS), 2010 19 Apr 2010, pp 1–1
6. AWS Lambda (2018) https://aws.amazon.com/lambda/ (last accessed 26 Jan 2018)
7. Shi W, Dustdar S (2016) The promise of edge computing. Computer 49(5):78–81
8. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on mobile cloud computing. MCC '12, pp 13–16
9. OpenFog Consortium Architecture Working Group, OpenFog Architecture Overview (February 2016). http://www.openfogconsortium.org/ra
10. Plebani P, Garcia-Perez D, Anderson M, Bermbach D, Cappiello C, Kat RI, Pallas F, Pernici B, Tai S, Vitali M (2017) Information logistics and fog computing: The DITAS approach. In Proceedings of the forum and doctoral consortium papers presented at the 29th international conference on advanced information systems engineering, CAISE 2017, vol-1848. Essen, Germany. CEUR, pp 129–136
11. Cappiello C, Pernici B, Plebani P, Vitali M (2017) Utility-driven data management for data-intensive applications in fog environments. In: International conference on conceptual modeling. Springer, Cham, pp 216–226
12. Kock N (2007) Encyclopedia of E-collaboration. Imprint of IGI Publishing, Hershey, PA, Information Science Reference
13. Syed MR, Syed SN (2008) Handbook of research on modern systems analysis and design technologies and applications. Imprint of IGI Publishing, Hershey, PA, Information Science Reference
14. Hundepool A, Domingo-Ferrer J, Franconi L, Giessing S, Nordholt ES, Spicer K, de Wolf PP (2012) Statistical disclosure control. Wiley
15. Weiss GM, Zadrozny B, Saar-Tsechansky M (2008) Guest editorial: special issue on utility-based data mining. Data Min Knowl Discov 17(2):129–135
16. Lin YC, Wu CW, TsengVS (2015) Mining high utility itemsets in big data. Springer International Publishing, Cham, pp 649–661
17. Ives B, Olson MH, Baroudi JJ (1983) The measurement of user information satisfaction. Commun ACM 26(10):785–793
18. Wang RY, Strong DM (1996) Beyond accuracy: what data quality means to data consumers. J of Manage Inf Syst 12(4):5–33
19. Ho TTN, Pernici B (2015) A data-value-driven adaptation framework for energy efficiency for data intensive applications in clouds. In: 2015 IEEE conference on technologies for sustainability (SusTech), pp 47–52
20. Moody D, Walsh P (1999) Measuring the value of information: an asset valuation approach. In: European conference on information systems
21. Even A, Shankaranarayanan G, Berger PD (2010) Inequality in the utility of customer data: implications for data management and usage. J Database Mark Customer Strategy Manage 17 (1):19–35
22. Gharib M, Giorgini P, Mylopoulos J (2016) Analysis of information quality requirements in business processes, revisited. Requirements Eng 1–23
23. D'Andria F, Field D, Kopaneli A, Kousiouris G, Garcia-Perez D, Pernici B, Plebani P (2015) Data movement in the internet of things domain. In Proceedings of European conference on service oriented and cloud computing, ESOCC 2015. pp 243–252

24. Gomez A, Merseguer J, Di Nitto E, Tamburri DA (2016) Towards a UML profile for data intensive applications. In: Proceedings of the 1st international workshop on quality-aware DevOps. Saarbrücken, Germany, pp 18–23
25. Nalchigar S, Yu E, Ramani R (2016) A conceptual modeling framework for business analytics. Springer International Publishing, Cham, pp. 35–49
26. Distributed Management Task Force Inc. Common Information Model (DMTF-CIM). https://www.dmtf.org/standards/cim
27. Cleve A, Brogneaux AF, Hainaut JL (2010) A conceptual approach to database applications evolution. Springer Berlin Heidelberg
28. Batini C, Scannapieco M (2016) Data and information quality-dimensions. Principles and Techniques, Data-Centric Systems and Applications, Springer
29. Garijo D, Alper P, Belhajjame K, Corcho O, Gil Y, Goble CA (2014) Common motifs in scientific workflows: an empirical analysis. Future Generation Comput Syst 36:338–351

Part III
Advanced Topics and Future Directions

Chapter 10
Context-Aware Intelligent Systems for Fog Computing Environments for Cyber-Threat Intelligence

Arif Sari

Abstract The capture and deployment of intelligence to build strong defense is more holistic and a more focused approach to security for private and government organizations, due to diversified activities of threat actors and malicious users. It is essential for the private sector and government organizations to use technologies such as Cloud, Fog and Edge Computing that provide storing, processing, and transferring vast amounts of public and private data that can be used to generate classified and unclassified threat intelligence. Fog Computing is a platform that uses computation, storage, and application services similar to the Cloud Computing paradigm but fundamentally different due to its decentralized architecture. It is an emerging technology that gives the Cloud a companion to handle the exabytes of data generated daily from the devices in the Internet of things (IoT) environment. In this scenario, the context-aware intelligent systems of Fog Computing that analyze the IoT data become crucial weapons to detect, mitigate, and prevent any possible cyber-threats, much earlier. The IoT environments generate vast amount of data that requires different context-aware intelligent systems to analyze this volumetric and complex data for security proposes to provide cyber-threat intelligence (CTI). The CTI is a collection of intelligence combining open-source intelligence (OSINT), social media intelligence (SOCMINT), measurement and signature intelligence (MASINT), human intelligence (HUMINT), and technical intelligence (TECHINT) from deep and dark Web. All these intelligence systems have functions that are based on specific context-aware intelligence mechanisms. In this chapter, sensitive data collection and analysis methods and techniques are elaborated through several context-aware intelligence systems to expose the use of Fog Computing for analysis of edge networks data for cyber-threat intelligence.

A. Sari (✉)
Department of Management Information Systems, School of Applied Sciences,
Girne American University, Canterbury, UK
e-mail: arifsarii@gmail.com

© Springer International Publishing AG, part of Springer Nature 2018
Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_10

10.1 Introduction

The incident detection and response becomes one of the major challenges for any organizations, since threats are relentless and come in all shapes and sizes due to rapid development and diversification of cyber-threats. Thus, the use of cyber-threat intelligence becomes an essential component of an organization's security policy since the significant increase in frequency and sophistication of cyber-attacks by different threat actors that compromise systems, disrupt services, commit financial fraud, and expose or steal intellectual property and other sensitive information.

Traditional security approaches and deployment of traditional incident response methodologies are no longer sufficient in the face of the widening gap between offensive and defensive capabilities and prediction of diversified and sophisticated techniques of adversaries to prevent attacks. Cyber-threat intelligence (CTI) provides information that can help public or private organizations to identify, assess, monitor, and respond to cyber-threats. The cyber-threat information produced by organizations covers tactics, techniques or procedures (TTPs) about threat actor(s), security alerts, state-of-the-art system, and recommended security system configurations and many details related with the threat depending on intelligence system used to [1].

In order to develop a proactive methodology to fight against cyber-threats, it is necessary to constantly harvest and process knowledge about the corresponding threat actors but not just the specific incidents. Collection of in-depth information about adversaries' actions and knowing the "who," "what," "where," "how," and "when" these actions are taking place are the key factors to establish security and decrease the possibility of adversaries' success [1].

Development of a threat intelligence platform (TIP) is an inevitable action for the organizations to examine current systems and adjust security policies and configurations to defend systems against adversaries' sophisticated and evolving attacks. The threat landscape is already large, and it is growing, becoming more complex and getting more efficient as time passes. The TIP is used to manage the flood of data since volume of intelligence is so massive and conduct in-depth analysis to organize tactical, operational, and strategic reporting in one platform. It is well known that public, private, and even governmental organizations operate in Cloud, Fog, and Edge Computing environments. Even the smallest unit of data which is stored in the Cloud, Fog, or Edge that will be collected, analyzed, and processed into information is crucial for threat intelligence to protect systems against adversary's actions [2].

The Cloud paradigm has provided vast amount of opportunities for enterprises to take an advantage of big data analytics because of its available features such as distributed data management scheme and scalability. But both scalability and processing power have been affected due to a burden of exploding amount of big data. Apart from this, low latency at the edge of network and additional needs of services such as demand of location awareness becomes significant challenges for Cloud Computing.

The emergence of Fog Computing paradigm was introduced as an extension to Cloud Computing at the edge of machine-to-machine (M2M) network to support the Internet of things (IoT) vision. Variety of advancements resulting due to the Fog Computing paradigm includes location awareness, mobility, low latency, and possibility of geographically widespread distribution [2].

The data that is stored or transferred through edge (infrastructure) networks, as well as the Fog and Cloud systems, is the main source for CTI either classified as public or private data resources. CTI aggregates, analyzes, and acts based on this volumetric flow of data and generates tactical, operational, or strategic intelligence reports.

In the rest of this chapter, Sect. 10.2 elaborates the technologies used by private companies and government such as Cloud, Fog, and Edge Computing including architectures and working mechanisms; Sect. 10.3 presents the classification of cyber-threat intelligence components, methods, and techniques such as OSINT, SOCMINT, MASINT, HUMINT, and TECHINT while Sect. 10.4 discusses methods to use Fog and Edge Computing technologies for context-aware intelligence. Section 10.5 presents a brief summary with conclusions.

10.2 Cloud, Fog, and Edge Computing Paradigms

Fog Computing has an independent management computing structure that is processed and stored among the real resources and the Cloud infrastructure [3]. This develops data processing performance by reduction of the data transfer costs and consequently reduction of the need for processing and storage of unnecessary data in the Cloud platforms. The Fog Computing paradigm leads to a steady rise in the Internet of things' (IoT) devices to a major extent, giving a constantly rising quantity of devices (in terms of capacity, diversity, and speed) [3]. In the IoT environment, in order to process the obtained data, there is a need for computing resources that are necessary to sustain a best level of functionality. This can raise availability, scalability, and credibility problems in client/server paradigm where data is detected by the client and processed by the server. When traditional client/server architecture encounters an overcharge, this overcharge leads many connected nodes and devices to become inactive.

The goal of the Fog paradigm is to develop a scalable solution in a decentralized architecture. This is achieved by constituting a hierarchically local platform based on the Cloud system and end user devices. Figure 10.1 illustrates Cloud, Fog, and Sensor layers and communication pattern between these layers [4].

The analyzing, processing, collecting, filtering, and transmitting data over this platform save time and increase efficiency. It is Cisco who initially presented the idea for a new paradigm, called Fog Computing [5].

Figure 10.2 illustrates the connections and relationship between the Cloud, Fog, and Edge networks. As shown in the figure, the Cloud platform solutions (such as data centers) have direct connections with Fog nodes, where these Fog nodes

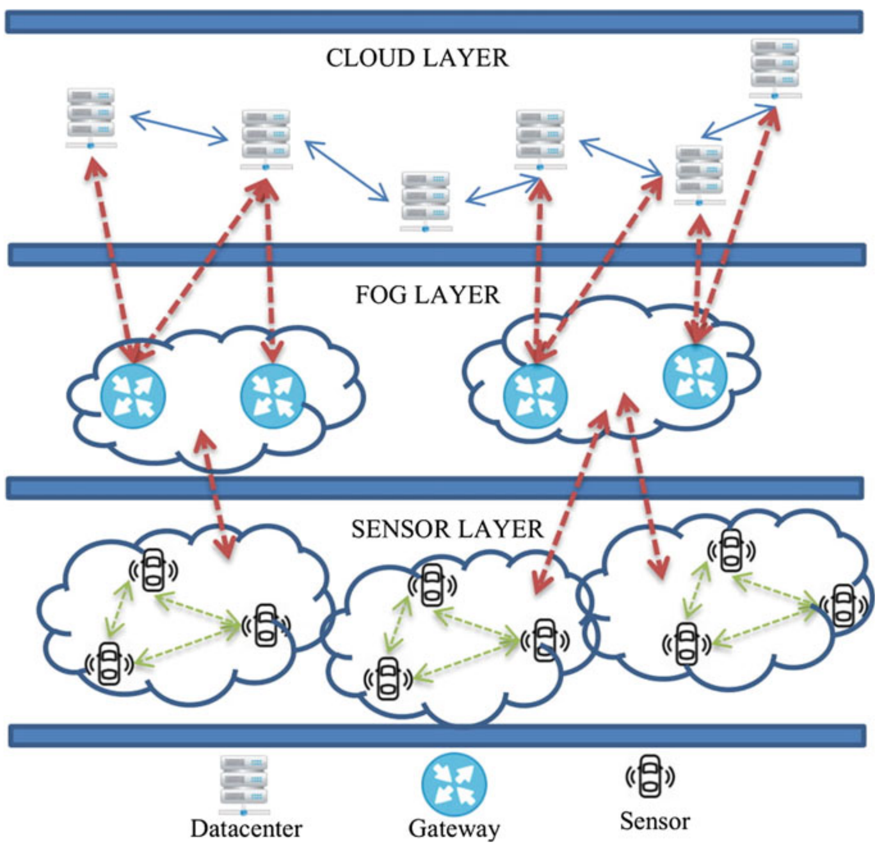


Fig. 10.1 Communication between Cloud–Fog–Sensor layers

receive data from edge devices of edge network. There are many Cloud platforms that are used for commercial solutions. On the other hand, there are some issues with this platform such as lag, portability, or intelligent networks, and networks that are already connected to devices. Latency increases with distance, the velocity of the Internet connection, and with the resource contention on virtual machines [6].

Additionally, software applications tend to process a variety of data at high speed, and by the time the data in the Cloud system is analyzed, it may be too late to respond to the relevant IoT devices. Consider an example from the medical field. Here, the IoT device's delayed effect with sensitive data may well become life threatening. Based on the aforementioned problems, Cisco provides and expands the Cloud platform closer to end user's devices and proposed solution by way of Fog Computing architecture [7].

Fog Computing distributes the computing tasks throughout the network that contains geographically widespread nodes to minimize the communication overhead and achieve high-performance computing (HPC) [3]. One of the main

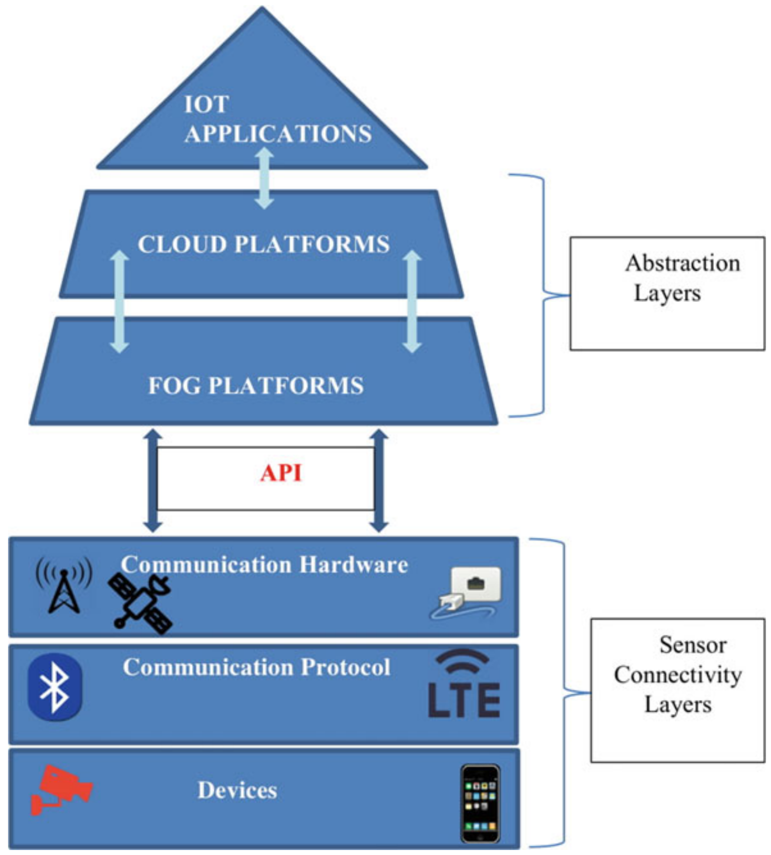


Fig. 10.2 Cloud, Fog, and Edge network classification

characteristic of Fog network is that it contains separate communication layers combined with very small scale of networks to decrease latency. Figure 10.3 presents the separated layers of Fog Computing with edge infrastructure.

The Edge Computing is a new model for optimizing Cloud Computing systems through performing data processing at the edge of the network, near the source of the data. The edge nodes collect data to be analyzed through specific algorithms and prepare meaningful information through visualizations and reporting facilities [4, 6, 7]. In a later section of this chapter, data collection and analysis of data through edge networks are explained in more detail.

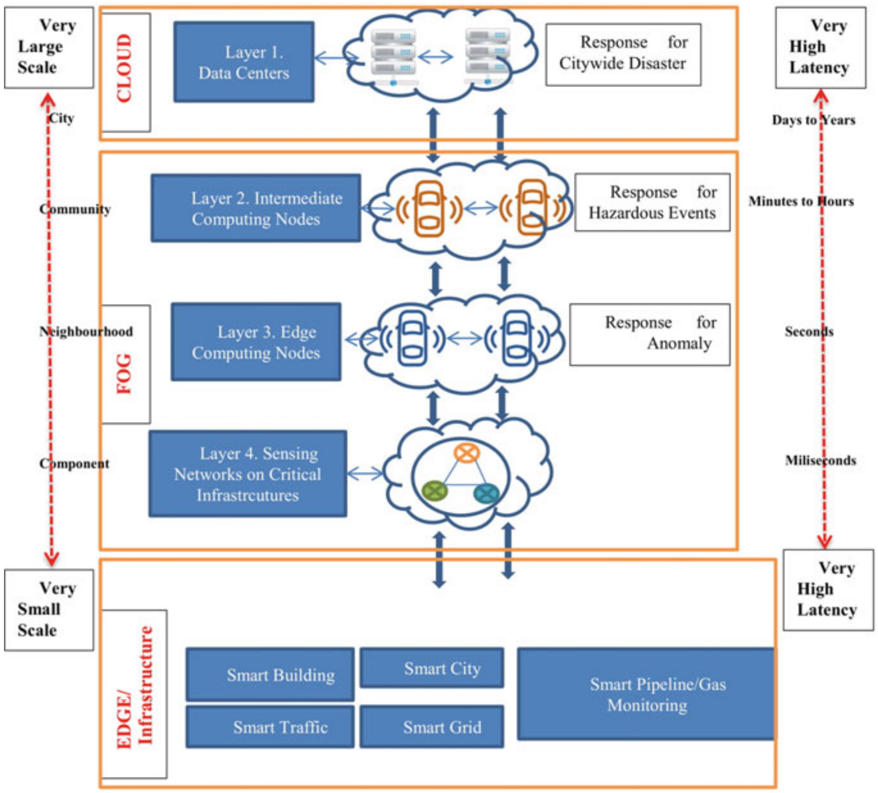


Fig. 10.3 Fog Computing architecture with edge infrastructure

10.3 Cyber-Threat Intelligence (CTI)

The cyber-threat intelligence can be obtained from internal or external sources. This intelligence is the knowledge of a threat's capabilities, infrastructure, motives, goals, and resources. The information assists in the operational, tactical, or strategic defense of network-based assets of public, private, or governmental organizations [8]. The internal resources such as internal network event data, log files (DNS logs, firewall logs, event logs), alerts, and past incident response reports provide information to recognize adversaries and strop threats. Apart from this, retaining the malware previously used for attack purposes, relevant packet capture and net flows are some of the invaluable internal resources to recognize and stop threats [8]. The external resources are the "open-source" intelligence-classified resources such as structured or unstructured reports, emails of sharing groups and blogs of particular groups or government, which are available to public in some extent and provide indicators based on relevance for detection and recognition of the attack [9]. An effective and efficient CTI relies on the following characteristics [10]:

- Timely: The intelligence must be delivered rapidly and on time, with minimal delay and latency, and provide sufficient information related to threat actors and prepare a suitable action against it.
- Relevant: The collected intelligence must have applicability with an operating environment of the recipient, address threats that corresponding organization may likely to face, and provide insight related to degree of risk associated with particular threat.
- Accurate: The collected intelligence must be correct, complete, and consistent. There must not be any ambiguity related to collected intelligence which may lead to false positive or false actions to be taken that may result in an inappropriate response or false sense of security.
- Specific: The collected intelligence must have concrete and significant details related to threat actors and provide details about how threat or threat actors can affect the system or organization.
- Actionable: The collected intelligence should identify the actions that can be taken as a suitable response to threat in order to prevent threat actors to damage or mitigate their attacks.

As mentioned previously, CTI is a collection of intelligence from open-source intelligence (OSINT), social media intelligence (SOCMINT), measurement and signature intelligence (MASINT), human intelligence (HUMINT), and technical intelligence (TECHINT) from deep and dark Web. All these intelligence systems have functions based on specific context-aware intelligence mechanisms [10]. The following subsections elaborate each of these intelligence systems in detail.

10.3.1 Open-Source Intelligence (OSINT)

The OSINT is intelligence derived from publicly available data which can be obtained legally and ethically from the public domain [11]. One of the main advantages of OSINT is that it is available at low cost and cannot be ignored. The sources can be classified as not just publicly available data but also unpublished materials including electronic information and human knowledge which is accessible legally and ethically. Since data and information sources are crucial for intelligence, one of the major difficulties of using OSINT is identifying relevant and reliable sources from the vast amount of publicly available information.

The private sector is one of the main reliable sources for cyber-threat intelligence due to technologies used in private sector such as Cloud, Fog, and Edge Computing provides storing, processing, or transferring vast amount of public or private data that can be used to generate classified or unclassified threat intelligence.

There are different ways to expose the importance of private sector for collecting intelligence and its potential contribution to cyber-threat intelligence needs. The different ranges of information services offered in private sector to satisfy the demand of intelligence in the market such as direct observation, document

Table 10.1 Private sectors that feeds intelligence

Schools	Universities	Libraries
Business	Private investigators and information brokers	Defense
Media	Government	Intelligence

acquisition, telephone surveys, market research, broadcast translation, experts on-demand, multi-expert research, recruited agents and industrial espionage (offered by competent organizations), commercial online searching are another evidence of robust intelligence capability potential of private sector [11, 12].

Table 10.1 shows the private sectors that store, process, or transfer data which is not available through commercial online services.

Schools provide intelligence about individuals who receive education pertaining specific languages in specific area. This information is effective intelligence for agencies to contact individuals and offer part-time or full-time translation jobs. Apart from this, schools that provide technical- or certification-based special trainings and qualifications are the key sources to form technical support team for customized tasks and operations. Libraries store vast amount of data that servers as a repository for archiving historical, political-military, geographical or economic data pertaining to specific regions or countries.

Businesses and organizations are also unprecedented sources of intelligence that collects information through primary or secondary research such as communication infrastructure and logistics, political conditions, economic and financial state, rate of corruption, climate conditions, demographic structure and culture about countries and regions around the world that are targeted to make an investment. This type of information is crucial for the business since it affects the operational efficiency and is quite useful source of intelligence.

Countries or regions where it is not possible to explore through communication infrastructures such as satellites or other different intelligence channels require external intelligence resources for intelligence exploitation. Here, information brokers or field specialists are the key sources of intelligence in case of a lack of intelligence sources related to the targeted country or region. A leading businessman, leading scholar, or an information broker specializing in targeted region or country can quickly identify or provide information as intelligence [11, 12].

Governments have experts in different fields with specified areas of operations. Bringing those experts who serve in different fields together can help to achieve extraordinary intelligence since most of these experts do not fully communicate what they know during their regular submission of reports.

Other intelligence resources such as "intelligence organizations" operating within targeted country are also repository of information and data relating to targeted region or country. Private businesses and governments use a variety of technological infrastructures such as Cloud or Fog Computing to collect, process, analyze, and transfer intelligence pertaining to targeted market country or region.

10.3.2 Social Media Intelligence (SOCMINT)

Social media is a popular form of virtual communication platform designed to bring people together for different forms of communication. It is also the collection of online communication channels (forums, social networking, micro-blogging, Wikis,) dedicated to categorized inputs of different community's interaction, content-sharing, and collaboration activities. The Web sites, Web portals, or applications designed and dedicated to different types of social media channels include Facebook, Google, YouTube, Twitter, Vimeo, Flicker, Snapchat, Instagram, Wikipedia, LinkedIn, Reddit, Swarm, and Pinterest [13].

The volume and variety of data produced through intensive use of social media channels for different purposes is increasing at an exponential rate. The data gathered from different social media channels is analyzed through social media analytics that produce information useful for making strategic business decisions. Intelligence agencies around the world are demanding to know more about threat actors, targets, their identities, behaviors, locations, financing and intentions, life styles, relationships, and similar content which is useful for cyber-threat intelligence [13].

Social media applications proposed to share information between members of dedicated social media groups which are categorized based on the content such as video, text, news, pictures. The Facebook, LinkedIn, or Instagram, used for sharing information, posts, or pictures between members of group, and Twitter are used to broadcast information to entire platform. Also, YouTube–Vimeo serve as platforms where videos and other media files can be shared easily with public or dedicated groups [13].

Context is a most important factor for social media analysis and for all semantics. The interpretation of data may be difficult due to lack of observation of formal grammar and syntax usage by the peers or users. Social media interpretation is not literally minded since users and peers in the system rely on their past knowledge and conversation experience of each other to arrive at specific and correct interpretation. The monitoring of social media offers a variety of precious and useful information; however, it requires management, analysis, and processing of vast amount of data produced through usage of social media. The mapping of social graphs of relationships between individuals exposes their links and collaboration on events, cases, people, or incidents.

Application programming interfaces (APIs) provide context-aware analysis of Facebook or Twitter that allows external analyzers to gain access to original online service. Using the relevant APIs of Twitter, the developer or analyzer can conduct a context-aware search based on keywords, hash-tags, or other filtering parameters [13, 14]. The Twitter API collaborates with integrated Twitter deck to conduct analysis of very large number of topics.

There are also numerous different analysis types, used for context-aware intelligence systems, dedicated to different intelligence platforms. Some of these are briefly discussed below.

The sentiment analysis (SA) provides opportunity for all tweets to be classified semantically whether corresponding or targeted tweets expressing different emotions such as like–dislike, fear–anger, hostility–passivity. Such analysis is potentially useful source of intelligence for the armed forces or for the police to forecast the mood of a crowd or demonstration. This is important and an efficient way to measure public reaction against specific events such as economic crises or political elections in advance. The sentiment analysis is based on specific algorithms designed to correlate statistically with the expression of some emotional indicators and classify the text accordingly. Most of these algorithms are supported by machine learning (ML) or artificial intelligence (AI) algorithms in order to model the behavior from the analysis of these texts. The sufficient volume of the material is critical for an appropriate, reliable, and validity of the analysis [13, 14]. This system is deployed as “All-Source Intelligence Hub” and used by metropolitan police to provide operational intelligence to the armed forces and local police departments. Interpreting and examining the volumetric social media posts’ data is inevitable in order to cope with public events and forecast the likelihood of criminal violence or disturbance and adjust the volume of intervention with an appropriate type of armed forces to enhance public safety.

Apart from the issues of SOCMINT, already discussed, videos posted on YouTube or Vimeo contains terrorist propaganda and instructional material such as building explosives, traps, bombs, or similar violence-supported videos are traced by the intelligence services. Some of the videos posted by the terrorist groups or supporters include their contact details where intelligence services trace these links to collect in-depth information about the source of the propaganda [15]. Even though the users sign a privacy agreement during their registration progress with social media platforms such as Twitter, Facebook, and Instagram, there are reports that social media platforms share personal details with third parties and therefore violate the privacy agreements—as these disclose the categorized information and private data of social media users to others.

10.3.3 Technical Intelligence (TECHINT)

Technical intelligence is an intelligence type used by military or governmental officers about weapons, equipment, tools, and technologies of the foreign nations. The main theme of using TECHINT is to provide in-depth information about the targeted threat factor to expose the basic engineering techniques of developed weapons, equipment, or tools and technologies of foreign nations. The collected information provides insight about basic technical characteristics, capabilities and limitations, production methods, applied research and applied engineering of the targeted weapon systems, communication systems and related infrastructures. The collected scientific information is crucial to expose current state of the targeted nations and threat actors in world market [16].

Table 10.2 TECHINT report production

TECHINT report components
Collection of material
Exploitation and testing
Production of finished intelligence

The TECHINT considers three main steps to provide intelligence report about targeted threat actors. These are shown in Table 10.2.

The collection of materials (weapon, system sample, tool, equipment, and component) is one of the important phases of TECHINT. The material captured during battlefield or obtained through spying activities provides opportunities to exploitation and in-depth investigation of materials to expose details about corresponding threat actor technologies as stated above. The exploitation phases cover a variety of in-depth operational and technical tests. The in-depth exploitation of material or item provides details for production of finished intelligence [16].

10.3.4 Measurement and Signature Intelligence (MASINT)

Measurement and signature intelligence (MASINT) is where technical intelligence obtained by quantitative and qualitative analysis of measurement data is collected through different communication channels [17, 18].

Due to its useful context with a requirement to detect, track, identify, and describe the signatures (distinctive characteristics) of fixed or dynamic target sources, the accurate MASINT provides a strategic advantage in the planning of military operations and deployments. Table 10.3 indicates the data used in MASINT and analysis of data collected from different objects. In order to operate in

Table 10.3 MASINT data analysis

Data	Comments/relevant questions
Thermal Infrared (JR) heat imaging	Where are the hot spots and how does the target look when in operation?
Near IR imaging	Just past visible light; used by night vision goggles; camouflage detection and use in night operations
Acoustic signatures	What does the system sound like, or where is a target such as a sniper
Seismic data	How much does the ground shake when the target roles past
Imaging radar data	Seeing through smoke, haze, foliage, and even water to try to detect and identify a target
Laser imaging and detection	Use of laser imaging systems for target and chemical/biological agent detection; detection and classification of emitters
Dimension and feature profiling	Deriving measurements on dimensions, weight, capacity, color, etc. to support modeling, simulation and algorithm development

Table 10.4 Components of MASINT

Radar Intelligence (RADINT)	Acoustic Intelligence (ACOUSTINT)	Nuclear Intelligence (NUCINT)	Radio Frequency/ Electromagnetic Pulse Intelligence (RF/ EMPINT)	Laser Intelligence (LASINT)
Infrared Intelligence (IRINT)	Unintentional Radiation Intelligence (RINT)	Chemical and Biological Intelligence (CBINT)	Electro-optical Intelligence (ELECTRO-OPTINT)	Directed Energy Weapons Intelligence (DEWINT)
	Spectroscopic intelligence	Materials intelligence	Effluent/debris collection	

functional operations and data collection, MASINT has diversity of component systems. MASINT includes all the components shown in Table 10.4.

The measurement capabilities of MASINT identify the battle space entities via multiple means that are difficult to spoof, and it provides intelligence that confirms the more traditional sources. The MASINT has to infer things about an object that it can only sense remotely [14, 18, 19].

10.3.5 *Human Intelligence (HUMINT)*

Human intelligence (HUMINT) is a category of intelligence derived from human sources such as their activities, interrogations, and conversations [14]. Human interrogation collectors are responsible for supervising and performing information collection progress and operations. The information that is necessary for intelligence can be collected in different ways that may be categorized as "special reconnaissance," "espionage," or "clandestine." Table 10.5 shows the categories of human intelligence collection methods used by different authorities.

As it is shown in Table 10.5, the categorization elaborates different types of methods to collection corresponding intelligence. The main aim of clandestine intelligence is to observe and report enemy positions (e.g., artillery, air base) against any possible attack [20]. The special reconnaissance is a method used by military personnel or variety of highly trained government and quasi-government officials. The other two are elaborated below.

Clandestine HUMINT

This refers to an intelligence collection through recruitment of human agents for specific tasks known as "spies" who work for specific organizations, governments, or authorities within a host country. Clandestine cell system is another method of organizing group of people in particular organization, government, or in bodies of the government in order to take specific actions and create resistance against penetration and opposing organization or government. The cell structure is a kind of hidden structure to maintain security. In this hidden structure, each small group of

Table 10.5 HUMINT collection methods

Human Intelligence Collection Methods	
Special reconnaissance	
Clandestine	(a) Asset recruiting (b) Cell system (c) Covert action (d) Direct action (e) Operational techniques
Espionage	(a) Agents (field-handling) (b) Asset black operation (black bag) (c) Concealment device (d) Cryptography (e) Cutout (f) Dead drop (g) Denial and deception (h) Eavesdropping (i) False flag (j) Industrial espionage (k) Intelligence assessment (l) Interrogation (m) Numbers station (n) Official cover (Non-official) (o) One-way voice link (p) Resident spy (r) Steganography (t) Surveillance

people in the cell knows the identities of only the people in their cell; the other cells and members, activities, or responsibilities are not visible to these cell members. Thus, a cell member who is apprehended and interrogated will not know the identities of other cells, their activities, or the higher-ranking individuals in the organization. The direct action is used by the military or special forces classified as short-duration strikes or small-scale offensive actions (seize, capture, exploit, destroy, recover, or damage) that use clandestine approach to target with high level of secrecy and exfiltration as soon as objective is completed [20]. The covert action is a method of clandestine HUMINT that provides remote actions with high level of secrecy such as financing local participants or specific groups in the different countries for specific events to lead specific actions such as sabotage, explosion, rebellion, assassination.

Espionage HUMINT

This has a variety of different subsequent methods to collect information from people either in a position of trust for the enemy, or with access to people with such access. This method is also known as “agent handling.” Communication between these agents and headquarter proceeds through different communication channels such as direct communication, radio signals, or by conventional communication method such as leaving a message in a hard-to-find place. Another commonly used method is “dead drop” which is widely used by intelligence services to exchange data during espionage activities [17]. For instance, USB devices are widely used for public dead drop where the corresponding intelligence data or information stored inside this item is passed on using a secret location between two individuals or agents. The way of using such methods do not require both parties to meet directly or physically thus maintaining operational security [20]. However, using this method is not a fail-proof, since exposure of one of the parties may lead to revealing the location of that specific drop. This may result for the counterintelligence to feed deceptive information to their enemy through this channel or expose and identify the other operative using it.

10.4 Using Fog/Edge Computing for Context-Aware Intelligence

Due to production of vast amount of data around the world and exposure of big data technologies, Fog Computing is adopted in industrial Internet of things (IIoT) with cooperation of Mobile/Edge Computing [21].

Cloud technology is too heavy to process data and respond in real time, and IoT nodes do not have computing and storage resources to perform analytics and machine learning tasks. As mentioned in a previous section of this chapter, the Fog technology is placed at an appropriate position (at the edge of the network) that has sufficient computation, storage, and networking functions to mimic Cloud capabilities and provide local data intake with low latency and holistic intelligence simultaneously. The efficiency of Fog Computing technology conducts short-term analytics at the edge that reduces the amount of data that needs to be sent to the Cloud for processing [21].

The successful adoption of IoT requires deployment of Fog and Cloud Computing technologies together since Cloud Computing is centralized and highly efficient computing infrastructure that can perform analysis of large volume of data from largely dispersed sources while Fog technology provides processing, transferring, or analysis of specific data (nearer to the devices that generate such data) with accuracy and low latency.

The power of Fog Computing provides a variety of advantages for those companies that analyze and process vast amounts of data during their business operations. Moving computation from Cloud to Fog networks provides ability to generate faster data analysis time to produce actionable insights in real time and gain significant business benefits [22] such as quicker strategic decisions. In fact, with Fog Computing shouldering the burden of short-term analytics at the edge, Cloud resources will be freed to take on the heavier tasks.

It is not necessary for all raw applications’ data to be processed entirely in the Cloud infrastructure. It is more suitable to do a pre-processing, classification, or analysis of the data at the edge and then send only the pertinent information to a

Table 10.6 Fog Computing advantages and characteristics of intelligence

Intelligence characteristics	Fog Computing advantages
Timely	Low latency for processing data and very low delay jitters
Relevant	Categorization and analysis of specific data at the fog layer
Accurate	Geographically distributed with location awareness ability
Specific	Provides proximity to its end users or targeted systems
Actionable	Provides real-time instructions to take action

centralized data center or the Cloud. The complete application and processing of data are composed by Edge and Cloud Computing.

The main characteristics of intelligence explained previously in this chapter are intersecting with advantages of Fog Computing technology which are shown in Table 10.6.

The advantages of Fog Computing provide timely, relevant, accurate, specific, and actionable analysis results which completely intersect with Fog Computing technology. Since Edge and Fog paradigms complement each other, Fig. 10.4 illustrates the cooperative functionality of Fog Computing with edge technology [22].

As shown in Fig. 10.4, Fog Computing performs data processing with low latency, performs responses at the edge of the network, manages, orchestrates, organizes, and filters these results and transfers the information to the data center. The relevant data determined at the edge networks is analyzed which satisfy the requirement of relevancy of intelligence. There are differences between Fog and Edge Computing, especially the Mobile-Edge Computing (MEC) [22] as classified in Table 10.7.

Ambient-assisted living (AAL) is a novel discipline proposed for real-time processing of data collected from sensors to elevate senior citizens' life style and independence [23]. However, delay-sensitive applications were not convenient

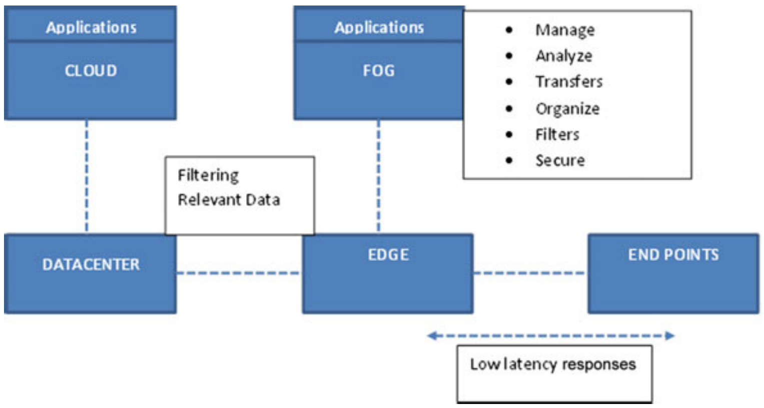


Fig. 10.4 Functionality of Fog and Edge Computing

Table 10.7 Classification of Fog and Edge Computing implementations

	Fog Computing (FC)	Mobile-Edge Computing (MEC)
Node devices	Routers, switches, access points, gateways	Servers running in base stations
Node location	Varying between end devices and Cloud	Radio network controller/macro base station
Software architecture	Fog abstraction layer based	Mobile orchestrator based
Context awareness	Medium	High
Proximity	One or multiple hops	One hop
Access mechanisms	Bluetooth, Wi-Fi, mobile networks	Mobile networks
Internode communication	Supported	Partial
Bandwidth utilization	Low	Very low
Server overhead	Low	Very low
Scalability	High	High
QoS and quality of experience	High	High
Energy consumption	Low	Low
Storage	Low	Low
Storage duration	Data can be stored for hours up to days	Temporary storage, does not support huge data collection
Network congestion	Low	Low
Energy consumption	Low	Low

because of requirement of real-time processing with lowest latency. The Fog technology becomes an inevitable tool to cope with latency and delay issues for delay-sensitive applications. The context awareness is also important key parameter for intelligence when processing vast amount of data while latency, accuracy and relevance are critical parameters. It is also crucial for the applications and use cases where information about corresponding networks and surrounding devices are exposed to the Edge nodes.

In this context, the Mobile/Edge Computing servers prove to be advantageous since they are placed in the radio network controllers. The servers receive information about the device location, load on the network and also the capacity of the network. However, since the nodes for Fog Computing are usually devices with a narrower view of the network, like routers or switches, the context awareness is less than that of Mobile/Edge Computing. However, the ability to communicate among the nodes themselves mitigates this issue to some extent [24].

The conceptual architecture of Context-Aware Intelligent system of Fog Computing is shown in Fig. 10.5, where the real-time data processing is available at the edge level which provides real-time data analytics. This opportunity provides

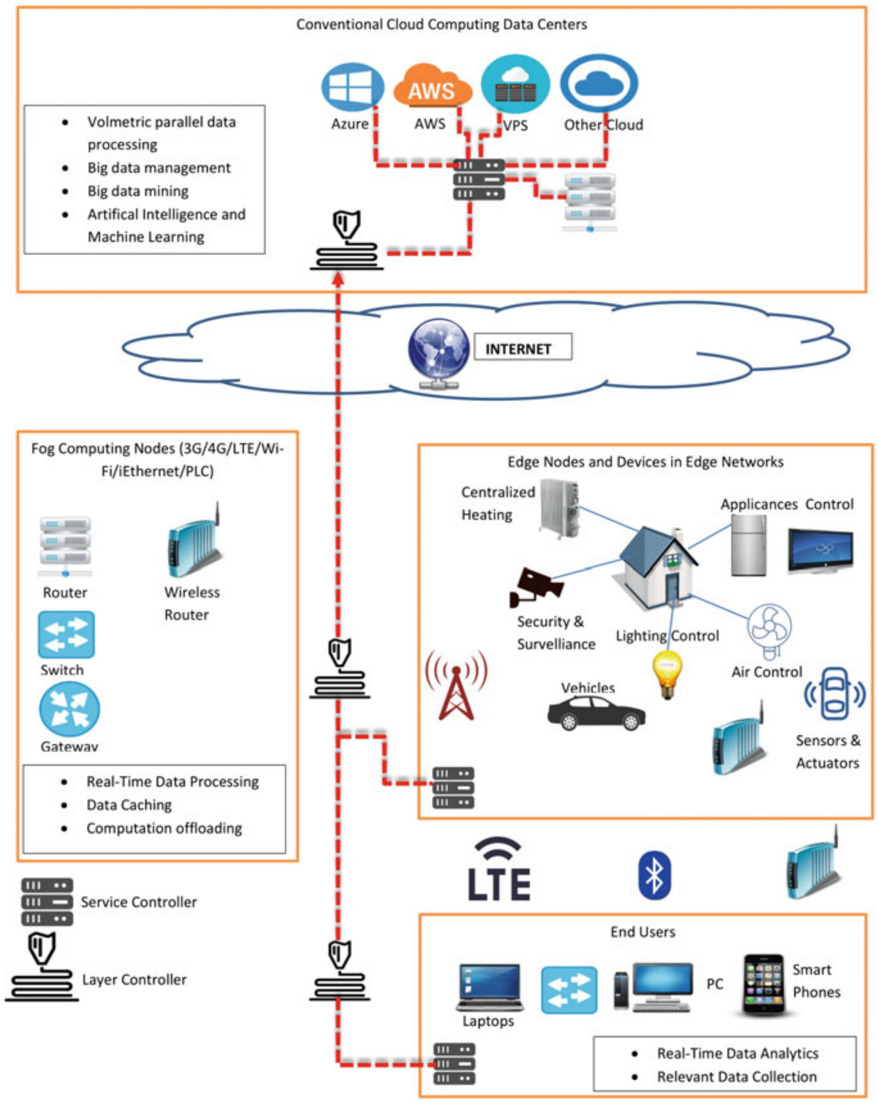


Fig. 10.5 Conceptual model of context-aware Fog Computing

timely intelligence related to threat actors. Data caching is also provided which leads vast amount of data collection opportunity with computation offloading.

Real-time information sharing, filtration, analysis, processing and real-time data extraction, real-time visual assistance and face recognition, video surveillance, smart healthcare, and ambient-assisted living are some of the key beneficiary areas for using Fog model to analyze data at the edge for intelligence [24–26]. Intelligence at the edge can also play a key role in reducing network traffic and saving bandwidth by deploying different analytical algorithms such as anomaly detection from crowd behavior.

The edge nodes collect data which are analyzed through specific algorithms and prepare meaningful information through visualizations and reporting facilities. Refer to Fig. 10.6 that illustrates the progress of data collection and processing through Fog nodes in Edge Computing through deploying algorithms on Edge nodes and data aggregators [27–30].

As presented in Fig. 10.6, collection of intelligence progresses through Fog/ Edge nodes and completed through exploratory analysis. The Edge nodes receive data from the public and transfer it through data communication channels to data aggregators. Exploitation of the communication devices such as Fog nodes can provide low latency, fast-track data analysis opportunity for targeted data or threat

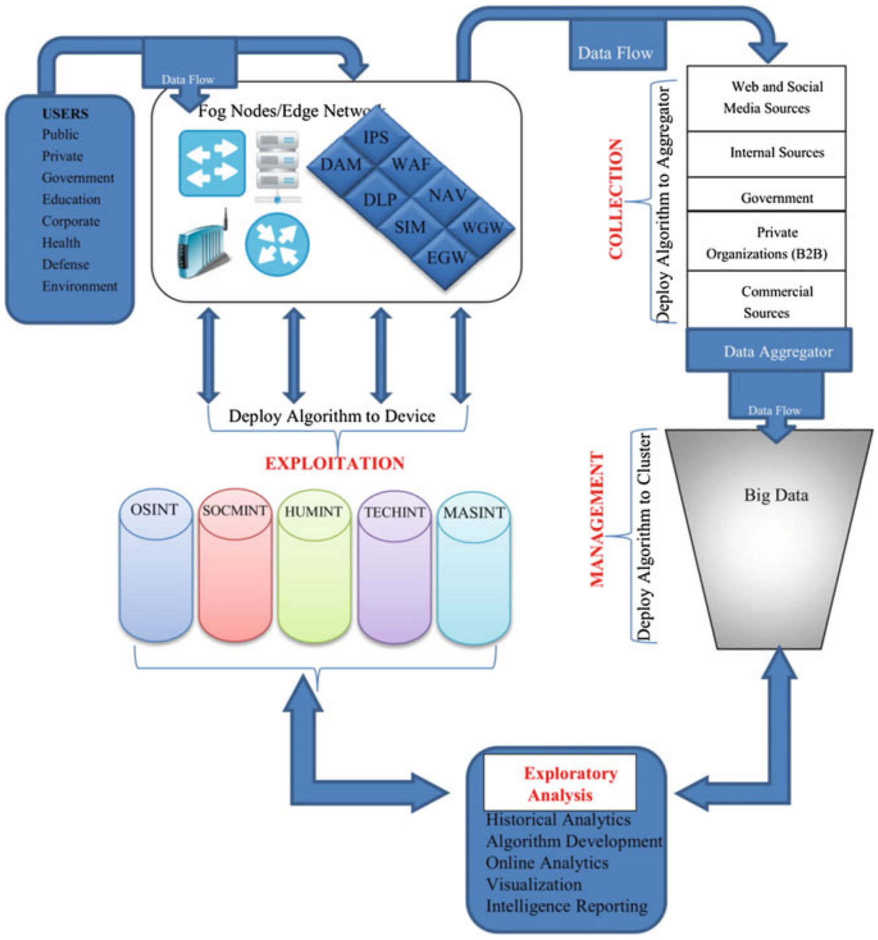


Fig. 10.6 Collection of intelligence progress through Fog/Edge nodes

actors. The algorithms that are deployed to conduct in-depth analysis in Fog nodes are used by exploratory analysis phase separately based on all-source intelligence hubs (OSINT, SOCMINT, HUMINT, TECHINT, and MASINT). This provides a great challenge for data reduction at analysis stage, decreases latency and increases reliability since source of data is closer to the targeted threat actors. Collection, management, and exploitation are shown as separate steps in Fig. 10.6. The collection phase indicates the public and private users communicating through Fog nodes for active participation in online services and transfers data to data aggregator. The management phase contains classification of collected volumetric data since incoming network traffic comes from different resources through Edge networks, to form big data. The deployment of algorithms on data aggregator for classification eliminates complexity of targeted data analysis for intelligence reporting and data visualization [30, 31].

The algorithms can support devices such as surveillance cameras to locally process video images and only dispatch relevant information based on contextual requirements. For example, analytical algorithms can detect anomalies locally after processing the video streams and can reconfigure the camera settings (video quality, camera angle, operational configurations, and so on) and/or trigger actuations to switch between analytical algorithms deployed at the edge or the Cloud.

10.5 Conclusion

This chapter highlighted the importance of Fog Computing technology and its widespread use for cyber intelligence. The Fog networks become a bridge between Edge networks and Cloud architectures which expose a new field of study and expertise that offers a lot of research challenges, but the main goal of this chapter is to make sense of data in any Fog environment. This newly proposed Fog networks can provide intelligent connection of objects, people, processes, data, and things in hierarchical IoT networks. The Fog architectures and their deployment exposed significant improvements of performance, networking, and enhanced computation power that affected latency, bandwidth, reliability, security, and overall Internet of Edge network performance for cyber intelligence.

Since effective threat intelligence management is an ongoing effort, researches have focused on how to exchange cyber-threat intelligence (CTI) and critical information through a secured shared platform among organizations to expand level of protection against adversaries. However, a significant gap has existed in terms of availability of real-time cyber-threat intelligence systems and development of cyber-threat intelligence frameworks.

The collection of mechanisms and methods that currently exist for cyber-threat intelligence such as OSINT, SOCMINT, MASINT, HUMINT and TECHINT, gather intelligence based on data stored and transferred through Cloud, Fog, or Edge networks and provide specific reports or information about threat actors.

These reports or information about threat actors which are not shared with other public or private authorities creates a vulnerability for other chains in the market.

The threat landscape is already large, and it is growing at a fast rate. It is becoming more complex and getting more efficient as time passes due to variety of methods, techniques, and platforms used by threat actors. Public and private authorities, governments, corporate organizations, and all other digital market participants should examine their defensive positions and adjust their operations with new strategies and policies against evolving technologies, methods, and platforms of threat actors that endanger themselves.

References

1. Mutemwa M, Mtsweni J, Mkhonto N (2017) Developing a cyber threat intelligence sharing platform for South African organisations. In: 2017 Conference on Information Communication Technology and Society (ICTAS), Umhlanga, pp 1–6. https://doi.org/10.1109/ictas.2017.7920657
2. El-Sayed H et al (2018) Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing environment. In: IEEE Access, vol. 6, pp 1706–1717, 2018 https://doi.org/10.1109/access.2017.2780087
3. Bonomi F, Milito R, Natarajan P, Zhu J (2014) Fog Computing: a platform for internet of things and analytics. Springer International Publishing, Cham, pp 169–186
4. Aazam M, Huh EN (2014) Fog Computing and smart gateway based communication for cloud of things. In: International conference on future internet of things and cloud
5. Dastjerdi AV, Buyya R (2016) Fog Computing: helping the internet of things realize its potential. Computer 49(8):112–116
6. Luan TH, Gao L, Li Z, Xiang Y, Sun L (2015) Fog Computing: focusing on mobile users at the edge. CoRR, vol. abs/1502.01815
7. Al-Shuwaili A, Simeone O (2017) Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. IEEE Wireless Commun Lett 6(3):398–401
8. Deliu I, Leichter C, Franke K (2017) Extracting cyber threat intelligence from hacker forums: support vector machines versus convolutional neural networks. In: 2017 IEEE international conference on big data (Big Data), Boston, MA, pp 3648–3656. https://doi.org/10.1109/bigdata.2017.8258359
9. Guido D (2011) A case study of intelligence-driven defense. In: IEEE security & privacy, vol. 9, no. 6, pp 67–70. https://doi.org/10.1109/msp.2011.158
10. Oman D (2017) Social media intelligence. In: Dover R et al (eds) The Palgrave handbook of security, risk and intelligence, pp 355–371. https://doi.org/10.1057/978-1-137-53675-4_20
11. Steele RD (1997) Open source intelligence, what is it? why it is important to the military? Open Source Solutions Inc. International Public Information Clearing House, pp 329–341
12. Omand D (2010) Securing the state. Hurst, London
13. Omand D, Bartlett J, Miller C (2012) Introducing Social Media Intelligence (SOCMINT). Intell Nat Secur 27(6):801–823. https://doi.org/10.1080/02684527.2012.716965
14. Omand D (2015) Understanding digital intelligence and the norms that might govern it. CIGI, Ottawa and Chatham House, London
15. Omand D Bartlett J, Miller C (2012) Introducing Social Media Intelligence (SOCMINT). Intell Nat Secur 27(6): 801–823
16. Butler R (2004) Review of intelligence on weapons of mass destruction. UK House of Commons, HC 898, July 14

17. Obama B (2014) Remarks by the President on review of signals intelligence. Department of Justice, Washington, DC, 17 Jan. www.whitehouse.gov/the-press-office/2014/01/17/remarks-president-review-signals-intelligence
18. Dudczyk J, Kawalec A, Cyrek J (2008) Applying the distance and similarity functions to radar signals identification. In: 2008 International radar symposium, Wroclaw, pp 1–4. https://doi.org/10.1109/irs.2008.4585771
19. Sene DE, Caldwell WT, Grigsby JA, George JD, Evans HE, Emmerich CJ (1999) Calibration stars. In: 1999 IEEE aerospace conference. Proceedings (Cat. No.99TH8403), vol 4, Snowmass at Aspen, CO, pp 297–306. https://doi.org/10.1109/aero.1999.792098
20. Omand D (2006) Ethical guidelines in using secret intelligence for public security. Cambridge Rev Int Aff 19(4):613–628
21. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R (2016) Fog Computing: principles, architectures, and applications. CoRR, vol.abs/1601.02752
22. Stojmenovic I, Wen S (2014) The Fog Computing paradigm: scenarios and security issues. In: 2014 federated conference on FedCSIS. IEEE
23. Zao JK, Gan TT, You CK, Méndez SJR, Chung CE, Te Wang Y, Mullen T, Jung TP (2014) Augmented brain computer interaction based on Fog Computing and linked data. In: 2014 international conference on Intelligent Environments (IE). IEEE, pp 374–377
24. Loke SW (2015) The internet of flying-things: opportunities and challenges with airborne Fog Computing and mobile cloud in the clouds. CoRR, vol. abs/1507.04492
25. Misra P, Simmhan Y, Warrior J (2015) Towards a practical architecture for india centric internet of things: an India-centric view. IEEE IoT Newslett
26. Al-Badarneh J, Jararweh Y, Al-Ayyoub M, Al-Smadi M, Fontes R (2017) Software defined storage for cooperative mobile Edge Computing systems. In: Proceedings of 4th international conference on Software Defined Systems (SDS), pp 174–179
27. Vallati C, Virdis A, Mingozzi E, Stea G (2016) Mobile-edge computing come home connecting things in future smart homes using LTE device to-device communications. IEEE Consum Electron Magn 5(4):77–83
28. Ahmed A, Ahmed E (2016) A survey on mobile edge computing. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO). IEEE, pp 1–8
29. Shahid MA, Sharif M (2015) Cloud computing security models, architectures, issues and challenges: a survey. Smart Comput Rev 5:602–616
30. Stantchev V, Barnawi A, Ghulam S, Schubert J, Tamm G (2015) Smart items, fog and cloud computing as enablers of servitization in healthcare. Sens Transducers 185(2):121
31. Amin SM, Wollenberg BF (2005) Toward a smart grid: power delivery for the 21st century. IEEE Power Energ Magn 3(5):34–41

Chapter 11
SEF-SCC: Software Engineering Framework for Service and Cloud Computing

Muthu Ramachandran

Abstract Service computing and cloud computing have emerged to address the need for more flexible and cost-efficient computing systems where software is delivered as a service. To make this more resilient and reliable, we need to adopt software engineering (SE) principles and best practices that have existed for the last 40 years or so. Therefore, this chapter proposes a Software Engineering Framework for Service and Cloud Computing (SEF-SCC) to address the need for a systematic approach to design and develop robust, resilient, and reusable services. This chapter presents SEF-SCC methods, techniques, and a systematic engineering process supporting the development of service-oriented software systems and software as a service paradigms. SEF-SCC has been successfully validated for the past 10 years based on a large-scale case study on British Energy Power and Energy Trading (BEPET). Ideas and concepts suggested in this chapter are equally applicable to all distributed computing environments including Fog and Edge Computing paradigms.

11.1 Introduction

Service-oriented software engineering and cloud software engineering have emerged to address software as a set of services which deal with user needs. At the same time, we have seen a large number of downfalls due to software failures. This was in part due to the lack of adopting well-designed software engineering practices. Now, there is a need to revise these practices given the emergence of service and cloud computing, which will revolutionise the next generation of software engineering.

Software engineering reinforces the application of engineering principles to software development. The major difference between software engineering and

M. Ramachandran (✉)
School of Computing, Creative Technologies, and Engineering,
Leeds Beckett University, Leeds LS6 3QS, UK
e-mail: M.Ramachandran@leedsbeckett.ac.uk

© Springer International Publishing AG, part of Springer Nature 2018
Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_11

other branches of engineering is that software is intangible. However, if there is any malfunction in a software system, it can have tangible effects. In recent years, with the use of software systems and/or software as a service (SaaS), especially for high integrity applications such as airborne, for financial clouds to medical IoTs, which lead to a very high possibility of a software failure and cyber-attack causing loss of life and financial instability. These software systems are used to conduct our daily routine activities as well as run business organisations. With the possibility of a system overload and hackers, existing systems are far from being secure and safe. Therefore, it is important to build software which is reliable and trustworthy. Some of the reasons why systems and software fail are:

- Increasing complexity of software systems that are distributed in nature,
- Failure to use software engineering methods, techniques, and processes,
- Lack of adopting and implementing building security, and
- Lack of adopting secure software engineering practices.

Appropriate software engineering techniques help us to build larger and more complex systems that demand changes rapidly and on-the-fly. Systems have to be built and delivered more quickly; their complexity has to be managed. Thus, new systems need to have new capabilities that were previously thought to be impossible.

Software can be written without using software engineering methods and techniques. However, studies show that many software failures and breach of security have happened in recent years. Many companies have drifted into adopting software development practices as their products and services have evolved. This is mainly because they have not used software engineering methods during the development phases. Consequently, their software is often more expensive and less reliable than it should be.

The following are some of the important reasons why software engineering principles need to be used in service and distributed computing scenarios including cloud and Fog computing, viz:

- Good software engineering software design improves software reliability, extensibility, reusability, quality, and maintainability.
- Technical debt in a project should not be accumulated.
- Growing complexity of information systems.
- Systems and software complexity in terms of design and code is increasing.

This chapter presents our approach to evolving a systematic software engineering paradigm known as SEF-SCC which offers new service and cloud software engineering method, process, reference architecture, and techniques, and addresses some of the key research questions using BPMN (Business Process Model and Notation) modelling and simulation during requirements engineering for service computing, as follows:

- How do we determine that the proposed business requirements and processes will perform efficiently?
- How do we model and simulate business processes effectively?
- How long does it take to find a composable service?
- What is the optimum allocation of services and resources?
- Does the process work well against a large number of service requests?

This is where BPMN modelling and simulation tool helps us to model the business processes and study their performance effectiveness.

11.2 Service and Cloud Computing Paradigms

SOA is a formalised way of integrating applications (traditional applications and legacy systems) existing in an enterprise architecture and hence a suitable approach for interconnecting IoE (Internet of Everything) devices. In simple terms, it can be defined as an architecture based on reusable and well-defined services implemented as IT components where the components are loosely coupled. Some of the advantages include: reduced impact of change, platform independence, technology independence, language independence, and flexibility to design new solutions from existing IT solutions regardless of where they reside or how they were created.

Service providers build services and offer them via an intranet or Internet. They register services with service brokers and publish them in distributed registries. Each service has an interface, known as contract and functionality, which is kept separate from the interface. The service consumers search for services based on some criteria, and when found, a dynamic binding is performed. In this case, the service provides the consumer with the contract details and an endpoint address. The consumer then invokes the service.

Services, implemented as Web services (WS), are delivered using technologies such as eXtensible Markup Language (XML), Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description Discovery and Integration (UDDI). Enterprise service buses (ESBs) build on MOM (message-oriented middleware) to provide a flexible, scalable, standards-based integration technology for building a loosely coupled, highly distributed SOA. ESBs contain facilities for reliable messaging, Web services, data and message transformation, and content-based ‘straight through’ routing. Along with Web services, ESBs are proving to be the major technical enablers for actual SOA projects. Figure 11.1 shows the benefits of service and cloud computing. It supports multitude of devices, provides distributed services, seamless data, service intelligence, and platform integration. It also allows for managers to take business decision and make predictions based on the performances.

SOA is founded on the notion of services, and each service is characterised by three distinct layers:

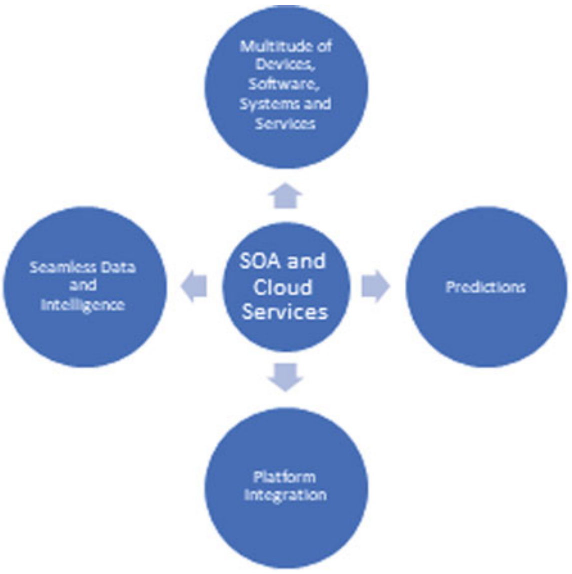

Multitude of Devices, Software, Systems and Services
Seamless Data and Intelligence
SOA and Cloud Services
Predictions
Platform Integration

Fig. 11.1 Service and cloud computing benefits

- Policies and service-level agreements (SLAs),
- Service interface, and
- Service implementation, with the first two (policies, SLAs and interfaces) forming what is generally referred to as service contract.

Every service supports the business/functional capability assigned to it through a set of well-defined operations that are offered on the interface of the service under the specified SLAs (often called QoS parameters). These operations consist of a set of messages that are exchanged between the consumers and providers during service invocation.

A service is an implementation of a clearly defined business function that operates independent of the state of any other service. It has a well-defined set of platform-independent interfaces and operates through a pre-defined contract with the consumer of the service. Services are loosely coupled (in the sense that a service need not know the technical details of another service in order to work with it), and all interaction between services takes place through service interfaces.

Data between the consumer and the service are passed in XML format over a variety of protocols. The main protocols that Web services use today are SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). While REST uses the existing Internet infrastructure (HTTP), SOAP is independent of the network layer and can use a variety of network protocols like HTTP, SMTP.

However, lack of building trust in the Internet-based technologies remains an outstanding issue for software engineering researchers which has been witnessed numerous cyber-attacks globally, in recent years. In addition, the core lesson learned from software failures points to the lack of application of established key software engineering principles and practices across the software development lifecycle. Therefore, this chapter presents a novel software engineering paradigm comprising methods, framework, and tools (e.g. SOA) for service and cloud computing. This chapter also distinguishes between software engineering for service and cloud computing and cloud software engineering as articulated in the following section. As mentioned before, the proposed SEF-SCC method is also applicable to cloud-based IoT environments and Fog and Edge computing paradigms [1–3].

11.2.1 SE for Cloud Computing Versus Cloud SE

SE for cloud should focus on engineering approaches to service development process, methods, developing reusable services, systematic approaches to cloud deployment, management, pricing, design for scalability and elasticity that needs to be built-in, tested, and deployed by cloud providers.

Cloud SE should focus on engineering approach to developing new services offered by a cloud with emphasis on built-in for scalability, service reusability, and elasticity. Some challenges for cloud software engineering as mentioned by Sommerville [4] are:

- Building a software development environment that radically simplifies hosting scientific applications on a range of clouds,
- Building applications that make use of the cloud providers' platform as a service (PaaS) APIs to access common services,
- Effectively using existing PaaS APIs for computationally intensive applications, and programming models for developing cloud services,
- Setting up systems that support Web-based applications,
- Investigating how to adapt applications that are computation/data intensive to run within the constraints set by the PaaS interfaces from cloud providers,
- Designing application architectures for the cloud: noting that underlying infrastructure management is done in the program itself, depending on performance and cost requirements, and abstractions needed for this,
- Developing innovative programming models for distributed environments,
- Building PaaS for high-performance/throughput computing, and
- Development of cloud-aware software engineering environments.

In addition, it is also the key foundation to innovate new abstraction of a service, similar to objects, components, and packages. There are other key challenges such as software engineering approaches to green cloud computing, in other words, to reduce the amount of power consumed by cloud data centres, emerging IoT, and Fog computing applications. It is important to make this distinction as, in this case, the cloud service providers will be able to follow strict engineering principles when they design and install new cloud services. Our earlier work on designing cloud services using service component models and the design of an independent security service component model can be used as a plug-in [5, 6]. This work has also demonstrated the issue of design for software security as a service (SSaaS) [5, 7]. Therefore, for these above reasons of engineering and achieving required quality of service (QoS) (accuracy, trustworthy, safe, and secure), SEF-SCC has been developed and applied to various applications and presented in the following sections.

11.3 Software Engineering Framework for Service and Cloud Computing (SEF-SCC)

SEF-SCC has evolved from various research projects on software components, reuse, and cloud security framework. We believe, by adopting a systematic framework for service and cloud computing, the trust in service and cloud computing will increase and be sustainable for all variety of applications from social media, entertainment, medical, financial, and migrating IT systems to the cloud. Figure 11.2 shows the SEF-SCC framework which consists of methods and design principles supporting service component models, the processes, reference architectures, tools, SE-SCC services, SEF-SCC adoption models, and evaluation techniques. Some of the components of the proposed model are briefly described below.

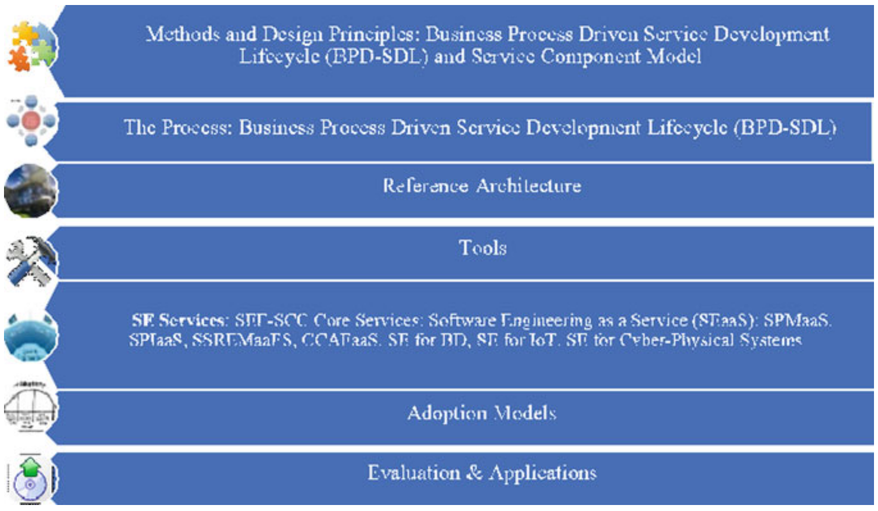

Methods and Design Principles: Business Process Driven Service Development Lifecycle (BPD-SDL) and Service Component Model
The Process: Business Process Driven Service Development Lifecycle (BPD-SDL)
Reference Architecture
Tools
SE Services: SEF-SCC Core Services: Software Engineering as a Service (SEaaS): SPMaaS, SPIaaS, SSREMaaFS, CCAFaaS, SE for BD, SE for IoT, SE for Cyber-Physical Systems
Adoption Models
Evaluation & Applications

Fig. 11.2 Software engineering framework for service and cloud computing (SEF-SCC)

Methods and Design Principles

Service component-based method and design strategies using UML components and SoaML have been developed and demonstrated as part of the Business Process-Driven Service Development Lifecycle (BPD-SDL). This has been demonstrated with key design principles of separation of concerns, service components, component-based architectures, interface design, reuse by service composition for Amazon EC2 case study which involved re-engineering EC2 architecture from the document review process with user guides and published research chapters [5, 6] and software project management as a service (SPMaaS) design with SoaML [8].

Processes

These provide a full lifecycle support starting with service requirements with BPMN modelling and simulation. SEF-SCC provides a Business Process-Driven Service Development Lifecycle (BPD-SDL).

Reference Architecture

This explicitly supports SOA-based design and therefore provides concrete design principles to be sustained for the product line of services.

Tools

These are set of tools such as Visual Paradigm, Bizaghi, and Bonitasoft BPMN modelling and simulation tool for capturing service requirements and to validate business process and performances.

SEF-SCC Services and Applications

These are services such as software engineering as a service (SEaaS), software project management as a service (SPMaaS) [8], software process improvement as a service (SPIaaS), software security requirements engineering management as a service (SSREMaaES) [6, 7], as well as Cloud Computing Adoption Framework for Financial Cloud (CCAFaaS) [9], SE for BD, SE for IoT, SE for cyber-physical systems. Adoption models supporting domain-specific application areas based on SEF-SCC have been developed and demonstrated for cloud computing, enterprise security, and improvement models. For example, we have developed a Cloud Computing Adoption Framework (CCAF) for cloud security and resiliency framework [10], enterprise security framework [11], and a framework for Internet security [12]. SEF-SCC applications refer to applications such as software project management as a service (SPMaaS) [8], SOA for big data analytics and business intelligence [13], SOA for E-Gov [14] and have been applied to validate the framework.

Integrated Framework for Service Computing

Service computing requires multidisciplinary skills from computing, social science, engineering, and other disciplines to understand what it involves developing a service rather than traditional software which is often not experienced by computer science and software engineering experts. In addition, there are three key challenges in

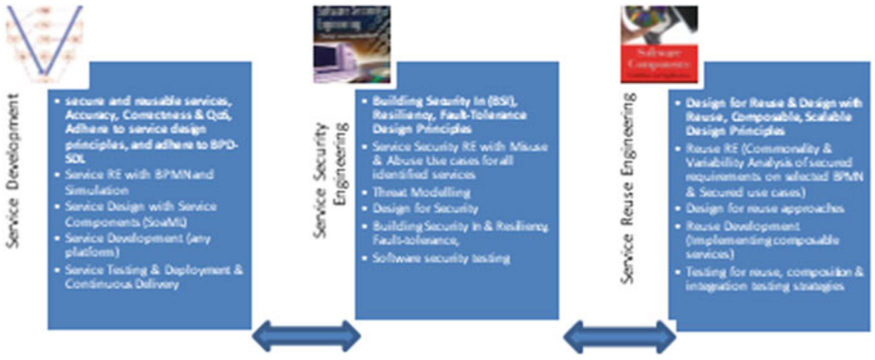

Service Development
secure and reusable services, Accuracy, Correctness & QoS, Adhere to service design principles, and adhere to BPD-SDL
Service RE with BPMN and Simulation
Service Design with Service Components (SoaML)
Service Development (any platform)
Service Testing & Deployment & Continuous Delivery
Service Security Engineering
Building Security In (BSI), Resiliency, Fault-Tolerance Design Principles
Service Security RE with Misuse & Abuse Use cases for all identified services
Threat Modelling
Design for Security
Building Security In & Resiliency Fault-tolerance,
Software security testing
Service Reuse Engineering
Design for Reuse & Design with Reuse, Composable, Scalable Design Principles
Reuse RE (Commonality & Variability Analysis of secured requirements on selected BPMN & Secured use cases)
Design for reuse approaches
Reuse Development (implementing composable services)
Testing for reuse, composition & integration testing strategies
Software Components

Fig. 11.3 SEF-SCC: service-security-reuse—an integrated service engineering framework

service computing: *service development, service security, and service reuse*. In this context, this chapter proposes three integrated frameworks for service computing as shown in Fig. 11.3. Service development aims to provide the development of a secure, safe, reusable, accurate, correct quality of service (QoS) based on engineering lifecycle development stages whereas the service security engineering focuses on Building Security In (BSI) based on the principles of secure service requirements techniques such as application of misuse and abuse service requirements process to the selected service requirement processes using BPMN modelling and simulation to validate consistent processes. The service reuse engineering provides design for reusable services, design with reusable services, application of commonality, and variability analysis techniques to selected BPMN services [15–17].

The three-tier integrated framework (Fig. 11.3) aims to emphasise the key service design principles of engineering service development, service reuse, and service security by supporting the following:

- Secure and reusable services,
- Accuracy, correctness, and QoS,
- Adherence to service design principles, and
- Adherence to BPD-SDL stages.

Each framework can be run in parallel for achieving engineering service development systematically, design for service reuse, and design for service security. However, this chapter aims to focus mainly on the service development framework.

11.3.1 SEF-SCC Process: Business Process-Driven Service Development Lifecycle (BPD-SDL)

As part of the SEF-SCC service development lifecycle, we have developed a Business Process-Driven Service Development Lifecycle (BPD-SDL) as shown in

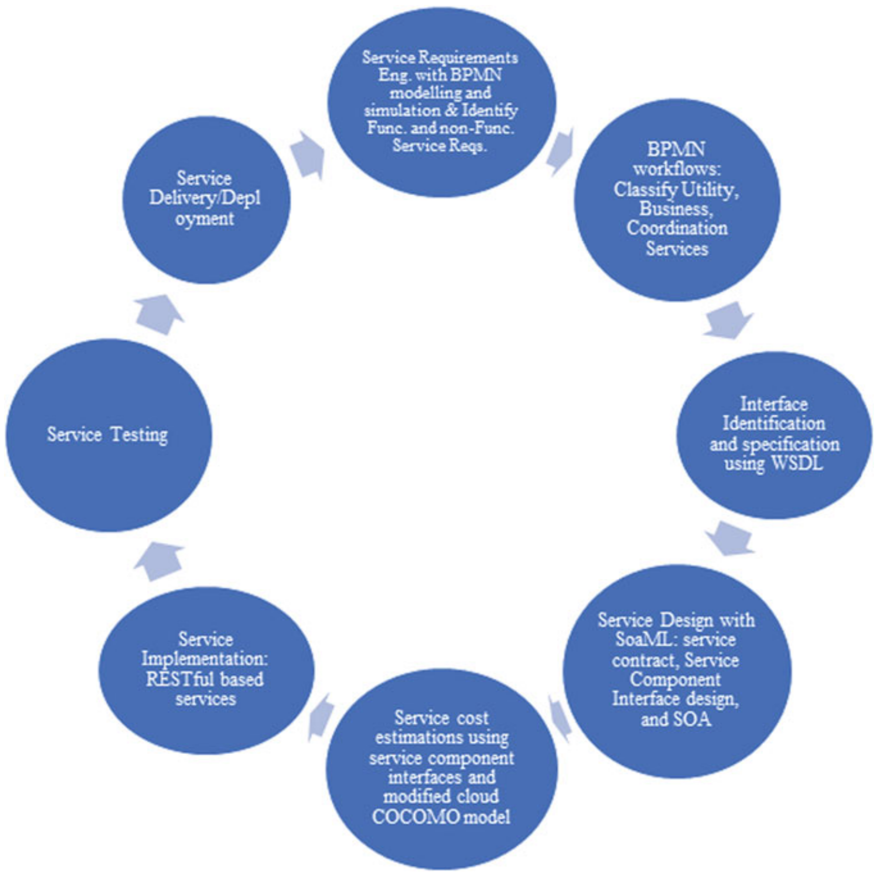

Service Requirements Eng. with BPMN modelling and simulation & Identify Func. and non-Func. Service Reqs.
BPMN workflows: Classify Utility, Business, Coordination Services
Interface Identification and specification using WSDL
Service Design with SoaML: service contract, Service Component Interface design, and SOA
Service cost estimations using service component interfaces and modified cloud COCOMO model
Service Implementation: RESTful based services
Service Testing
Service Delivery/Depl oyment

Fig. 11.4 Business process-driven service development lifecycle (BPD-SDL)

Fig. 11.4. The BPD-SDL is a key development in addressing the need for a modern software engineering for service and cloud computing (SE-cloud), and it consists of the following stages which can be iterative and agile; however, BDP-SDL reinforces the engineering approach:

1. Service requirements—This stage consists of eliciting service requirements from various stakeholders, modelling service requirements using BPMN, conduct simulation, and validate service requirements. Also, identifying service requirements into functional and non-functional services forms a set of criteria for performance evaluation of the acquired services. The performance evaluation of time, cost, and resources is the key aspects of service computing, and they will be deployed in a cloud which needs to meet those performance criteria. During this stage, it is also a good idea to classify business processes into various business to consumer matrix according to proposal by Chen [18]:

B2B (business-to-business), B2C (business-to-consumer), C2C (consumer-to-consumer), etc. This will be useful to maintain requirements traceability and consistency.

2. Conduct BPMN workflows—This stage consists of categorising services into a number of classes such as task and entity, and enterprise-oriented services, as shown in Table 11.1. This table is useful in managing service requirements and to specify detailed WSDL description in the following stage on service interface identification and specification.
3. Service interface identification and specification using WSDL—WSDL provides a clear template for specifying message-oriented service interface abstractions (InMessage, Operations, OutMessage), port type (The bindings) which provides the details of the location and its binding.
4. Service design with SoaML: SoaML has been developed for service computing by extending the standard UML. SoaML provides different dimensions of the design aspects: Service contract, service component interface design, and SOA. This is illustrated in the next section on service design.
5. Service cost estimations using service component interfaces [19] and modified COCOMO model for cloud computing [20]—This stage aims to develop service effort estimation based on sum of the number of interfaces plus adding weighting factors. The second method is a modified form of COCOMO model of cost estimation.
6. Service implementation—RESTful-based services provide a lightweight implementation for service-based systems.
7. Service testing—Service testing should include traditional testing techniques such as unit testing, boundary value analysis testing, integration, and acceptance testing. In addition, for test cloud services as well as SOA, testing should include performance testing, model-based testing, symbolic execution, fault injection testing, random testing, and in particular privacy-aware testing as proposed by Priyanka, Chana, and Rana [21].
8. Service delivery/deployment—This stage aims to deploy and capture expected QoS and performances as predicted during service requirements simulation using BPMN models.

Table 11.1 Categorising and classifying service requirements workflows

Business types	Service types		
	Task-oriented services	Entity-oriented services	Enterprise services (B2B)
Utility services			
Business services			
Coordinations and choreographic services			
Human services			

The BPD-SDL provides a comprehensive set of guidelines based on the principles of service computing, and the detailed service elicitation, elaboration, and specification support consistent design, implementation, testing, and deployment of services.

11.3.2 SEF-SCC Design: Service Component-Based Design Method with SoaML and SOA-Based Reference Architecture

SEF-SCC design principles are based on the key concept of separation of concerns as part of the domain modelling process, design for service reusability, and design for service security. To achieve this, SEF-SCC design proposes to use high-level abstractions such as virtual machines, service components, and packaging. These design concepts are a natural extension of a software component concept by supporting the notion of provider and requires interfaces. Similarly, the concept of microservices, containers, and serverless programming abstractions, all are based on service components abstraction, as shown in Fig. 11.5.

SEF-SCC approach to service design emphasis is based on the foundation of design principles such as high-level abstraction to achieve loose coupling required for implementing cloud services and SOA-based reference architecture. In addition,

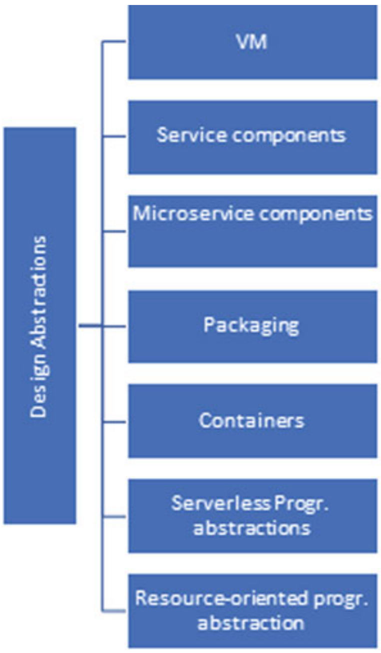


Fig. 11.5 Design abstraction principles

in order to maximise reuse, scalability, and elasticity, SEF-SCC emphasises on component-based design as it provides a natural extension of a software component to implement message-oriented architecture which is required for service and cloud computing. To support, service computing software engineers, SEF-SCC provides a design rationale for selecting appropriate design abstraction based on established set of design criteria, as shown in Table 11.2. We believe this is important for making design decisions based on the fundamentals of software engineering to achieve developing a trustworthy cloud service. The design rationale is understanding the definitions of abstractions, loose coupling versus simpler abstraction, lightweight versus heavyweight design, implementation, response time, interoperability of multi-clouds and cloud federation, infrastructure support, stateless versus stateful, end-to-end application versus flexible packaging, and more efficient cloud resources.

As discussed earlier, component-based design of cloud as well as Web services provides a natural dimension to design which incorporates most of the design foundations that are required such as interface-based design, high-level abstraction of supporting cohesion, and loose coupling through separation of concern and flexibility of interface coupling and decoupling to achieve and maximise reuse through service composition. A simple service component model for implementing SEF-SCC as a service (SEFaaS) is shown in Fig. 11.6, as a plug-in service component model; the SEF-SCC service security as a service (S3aaS) service component model is shown in Fig. 11.7, as a plug-in service security component model.

As shown in Fig. 11.6, a conceptual service design component model, SEFaaS service component model, can be packaged as a container in the cloud environment providing service interfaces (application services) for various SEF-SCC applications discussed earlier such as *IClassicalSEaas* which aims to provide traditional software engineering as a service on requirements engineering, cost estimation, software project management, design (structured, object-oriented, and software components, and packaging), and testing methods such as unit, integration, and acceptance. Service interface on ISEF-SCC as a service (I3SEaaS) aims to provide SEF-SCC as a service. Service interface on *ISEforBigData* as a service aims to provide software engineering for big data services of streaming, analysing, evaluating, pruning, visualising, analytics, and predictions.

Cost estimation for service computing is relatively a new area which remains unexplored in research. However, Gupta [19] has proposed a service point estimation model which aims to calculate the sum of service component interfaces with some weighting factor. Therefore, the service interface, *IServicePointCostEstimation* as a service (ISPCEaaS), aims to provide an autonomic service for computing the service cost and therefore achieving the estimation of service complexity analysis before actual implementation and the performance characteristics can also be evaluated and measured against initial BPMN simulations. In addition, Guha [20] has proposed a modified cloud COCOMO model with weighting for cloud computing projects which are: $a = 4$, $b = 1.2$, $c = 2.5$, $d = 0.3$. Therefore, the effort and cost estimation equations are:

Table 11.2 Design rationale for selecting suitable design abstraction

Design abstractions	Design criteria						
	Definitions	High-level loosely coupled abstraction	Lightweight	Response time	Heavyweight	Multi-cloud interoperability	Infrastructure
VM	VM is an abstraction and an instance of a specific OS	✓		5–10 min	✓		✓
Service components	A lightweight abstraction of software component and a natural extension of a service supporting interfaces explicitly to connect and compose new services	✓	✓	Seconds to minutes		✓	✓
Microservice components		✓	✓	Seconds to minutes		✓	✓
Packaging		✓	✓	Seconds to minutes		✓	✓
Containers		✓	✓	In seconds		✓	✓
Severless	Serverless provides an abstraction of a service functions without underlying infrastructure in a cloud environment. Specific programming language to be used such as Lambda, Cloud functions	✓	✓	In seconds		✓	✓

(continued)

Table 11.2 (continued)

Design abstractions	Design criteria							
	Stateless	Stateful	End-to-end application packaging	Flexible packaging	Simpler cloud abstraction	Faster and more efficient cloud resources	Example platform	References
VM							Azure, AWS, Google, etc	
Service components	✓	✓	✓	✓	✓	✓	Gravity	Ramachandran [5] [22] Gravity
Microservice components	✓	✓	✓	✓	✓	✓	Microservices as Docker containers running inside of Kubernetes, Dropwizard Wildfly Swarm	
Packaging	✓	✓	✓	✓	✓	✓		
Containers	✓	✓	✓	✓	✓	✓	Titus: Netflix (AWS) Opensource container management systems such as Kubernetes and Docker Swarm	Leung et al. [23]
Severless	✓	✓	✓	✓	✓	✓	Lambda (AWS), Google (Cloud Functions), OpenWhisk (IBM), and Azure	Savage [24]

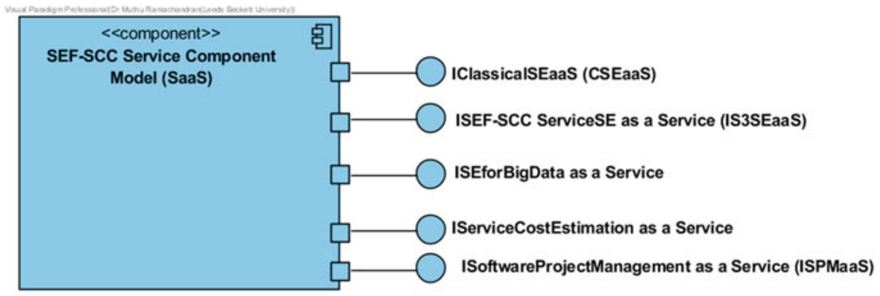

Visual Paradigm Professional(Dr Muthu Ramachandran(Leeds Beckett University))
<<component>>
SEF-SCC Service Component Model (SaaS)
IClassicalSEaaS (CSEaaS)
ISEF-SCC ServiceSE as a Service (IS3SEaaS)
ISEforBigData as a Service
IServiceCostEstimation as a Service
ISoftwareProjectManagement as a Service (ISPMaaS)

Fig. 11.6 SEF-SCC service component model [SEF-SCC as a service (SEFaaS)]: a plug-in service component model

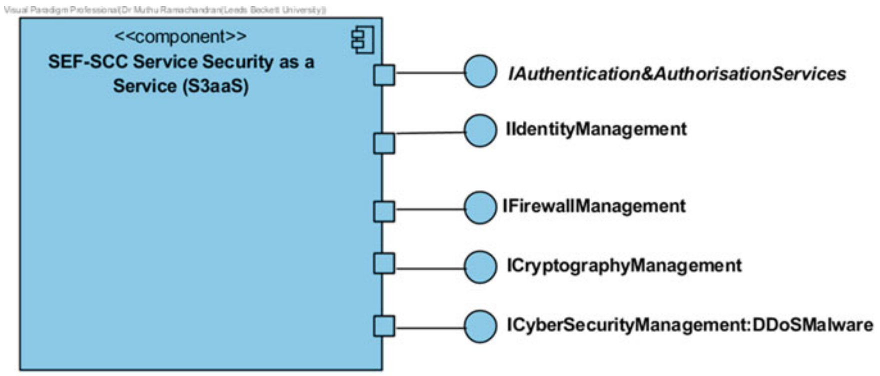

Visual Paradigm Professional(Dr Muthu Ramachandran(Leeds Beckett University))
<<component>>
SEF-SCC Service Security as a Service (S3aaS)
IAuthentication&AuthorisationServices
IIdentityManagement
IFirewallManagement
ICryptographyManagement
ICyberSecurityManagement:DDoSMalware

Fig. 11.7 SEF-SCC service security as a service (S3aaS): a plug-in service security component model

$$\text{Cloud computing project effort applied (EA)} = a \times (\text{Service Points})^{b} (\text{Human Months}) \tag{11.1}$$

$$\text{Cloud computing development time (dt)} = c \times (\text{Effort Applied})^{d}\ (\text{Months}) \tag{11.2}$$

$$\text{Number of Service Development Engineers Required} = \text{Effort Applied(EA)/Development Time(dt)} \tag{11.3}$$

Equations 11.1, 11.2, and 11.3 provide cloud project effort and cost estimations based on service points which are the sum of all interface function points.

Similarly, the service interface, *ISoftwareProjectManagement* as a service (ISPMaaS), aims to provide support for classical project management activities such as requirements management, scheduling, planning, managing resources, and efficient resource management. In addition, the best practice service component design principles are:

- Use port concept to define all interfaces, and
- Always follow best practice guidelines on interface design principles with service components which is a natural extension of a service.

As shown in Fig. 11.7, a conceptual service design component model, SEF-SCC service security as a service (S3aaS) component model, can be packaged as a container in the cloud environment that provides service interfaces (provider application services) for various SEF-SCC service security applications such as *IAutentication&Authrisation, IIdentityManagement, IFirewallManagement, ICryptograhyManagement,* and *ICyberSecurityManagement* for services against malware, adware, and DDoS attacks.

As part of our best practice design guidelines, BPMN modelling and simulation for capturing and validating requirements, SOA design method consists of designing service components, designing SOA-based reference architecture and designing SoaML SOA, and finally mapping service components to SOA reference architecture. The following section presents SOA-based SEF-SCC architecture design and best practices.

11.3.3 SEF-SCC SOA-Based Reference Architecture

SOA is a design method based on service provider, service consumer, and service directory model which has been established as a proven design method for service computing. One of the key importance of proposing SEF-SCC reference architecture is to standardise the evolution of SOA-based software products and cloud-based services for all software as a service paradigm design and implementation.

In addition, a reference software architecture provides a generic structure for a class of software systems and software product line engineering to achieve a standard practice [25]. Similarly, NIST reference architecture for cloud computing defines to provide an overall framework and to communicate accurately across cloud services, hence providing support for vendor neutral design [26]. In this context, SEF-SCC has developed a reference architecture based on SOA framework (RAG-SCC) as shown in Fig. 11.8.

RAG-SCC follows the design principles of SOA and provides three-layer architectural model. The top layer is orchestration and coordination services which focuses on new service composition for new business creation and choreography; the middle layer is known as the business layer which provides container for all business services components and their connections and focus on user-centric business services such as secure access, business utility and coordination services (task-oriented, entity-oriented, and enterprise-oriented services), the enterprise service bus which is the backbone of SOA architecture by providing common communication pathway (supporting the concept of unified modelling) for all services from various layers; and the bottom layer is known as the infrastructure layer which provides core infrastructure services, infrastructure utility services,

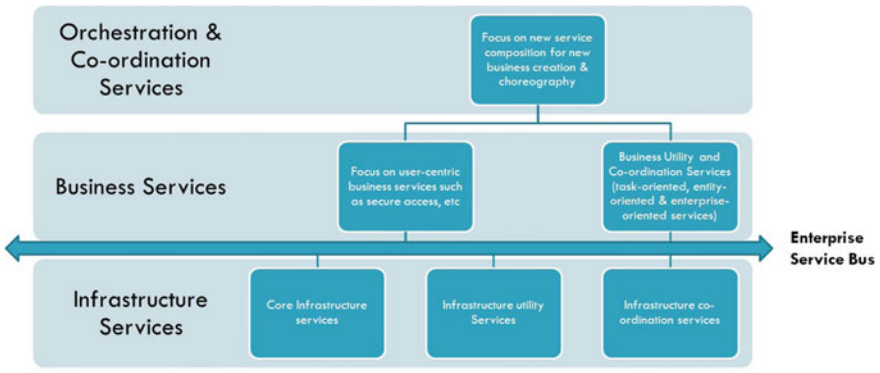

Orchestration & Co-ordination Services
Focus on new service composition for new business creation & choreography
Business Services
Focus on user-centric business services such as secure access, etc
Business Utility and Co-ordination Services (task-oriented, entity-oriented & enterprise-oriented services)
Enterprise Service Bus
Infrastructure Services
Core Infrastructure services
Infrastructure utility Services
Infrastructure co-ordination services

Fig. 11.8 Reference architecture for generic service and cloud computing (RAG-SCC)

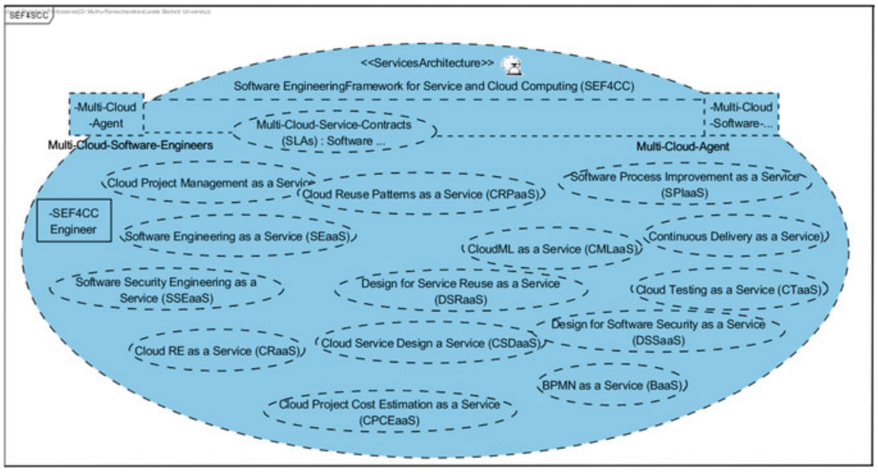

SEF4SCC
<<ServicesArchitecture>>
Software EngineeringFramework for Service and Cloud Computing (SEF4CC)
-Multi-Cloud -Agent
Multi-Cloud-Software-Engineers
Multi-Cloud-Service-Contracts (SLAs) : Software ...
-Multi-Cloud -Software-...
Multi-Cloud-Agent
Cloud Project Management as a Service
Cloud Reuse Patterns as a Service (CRPaaS)
Software Process Improvement as a Service (SPIaaS)
-SEF4CC Engineer
Software Engineering as a Service (SEaaS)
CloudML as a Service (CMLaaS)
Continuous Delivery as a Service
Software Security Engineering as a Service (SSEaaS)
Design for Service Reuse as a Service (DSRaaS)
Cloud Testing as a Service (CTaaS)
Design for Software Security as a Service (DSSaaS)
Cloud RE as a Service (CRaaS)
Cloud Service Design a Service (CSDaaS)
BPMN as a Service (BaaS)
Cloud Project Cost Estimation as a Service (CPCEaaS)

Fig. 11.9 SOA design for SEF-SCC services with SoaML

infrastructure coordination services. We also recommend service security is built-in security component as plug-in and in all layers in the reference architecture.

The next step in the SEF-SCC design process involves designing SOA design with SoaML which is one of the best practices for providing a completed design as shown in Fig. 11.9 for all SEF-SCC services such as the consumers are multi-cloud users, the provider is multi-cloud agent with cloud software engineers; there are SLAs for both parties to connect and communicate for using and composing new services, and the sample services are:

- Cloud project management as a service (CPLaaS),
- Cloud reuse patterns as a service,
- Software security engineering as a service,

- CloudML as a service,
- Software process improvement as a service,
- Cloud RE as a service,
- Cloud project cost estimation as a service,
- Design for service reuse as a service,
- Cloud testing as a service,
- BPMN as a service,
- Continuous delivery as a service, and
- Cloud service design as a service.

There are plenty of services that can be coordinated and composed for new services, and therefore, it is one of the best practices for service design offered by SEF-SCC design approach.

After SOA design with SoaML, the final step in the design process is to compose the classified service as shown in Table 11.1 and map them into the reference architecture as shown in Fig. 11.10.

The mapping of services into the SOA architecture requires architectural design skills to evolve the design rules that can be embedded into the services and the architectural layers. The following section shows a proposed automated CASE tool for developing secure cloud services based on the best practice guidelines presented in this chapter.

11.3.4 Autonomic CASE Tool for Service Computing

Autonomic service computing is emerging faster as it is SOA-based which supports autonomic computing using reuse of knowledge and services. For example, Bellur [27] proposes an autonomic service-oriented middleware for IoT-based systems

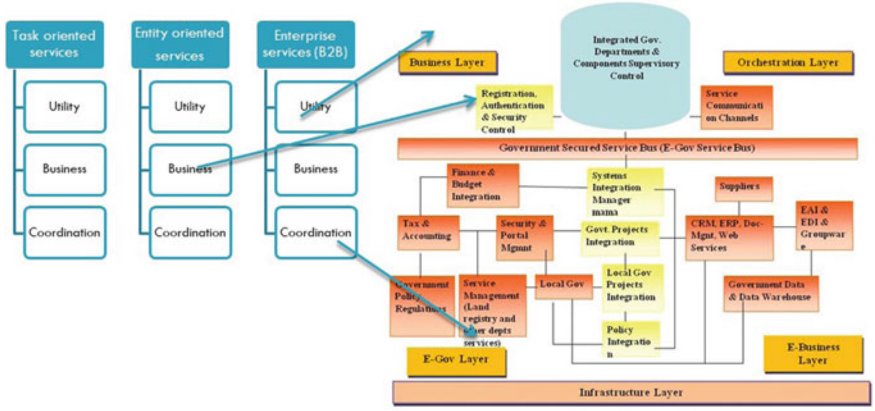

Task oriented services
Utility
Business
Coordination
Entity oriented services
Utility
Business
Coordination
Enterprise services (B2B)
Utility
Business
Coordination
Business Layer
Integrated Gov. Departments & Components Supervisory Control
Orchestration Layer
Registration, Authentication & Security Control
Service Communication Channels
Government Secured Service Bus (E-Gov Service Bus)
Finance & Budget Integration
Systems Integration Manager mama
Suppliers
Tax & Accounting
Security & Portal Mgmnt
Govt. Projects Integration
CRM, ERP, Doc-Mgnt, Web Services
EAI & EDI & Groupware
Government Policy Regulations
Service Management (Land registry and other depts services)
Local Gov
Local Gov Projects Integration
Government Data & Data Warehouse
Policy Integration
E-Gov Layer
E-Business Layer
Infrastructure Layer

Fig. 11.10 Mapping services to SEF-SCC reference architecture

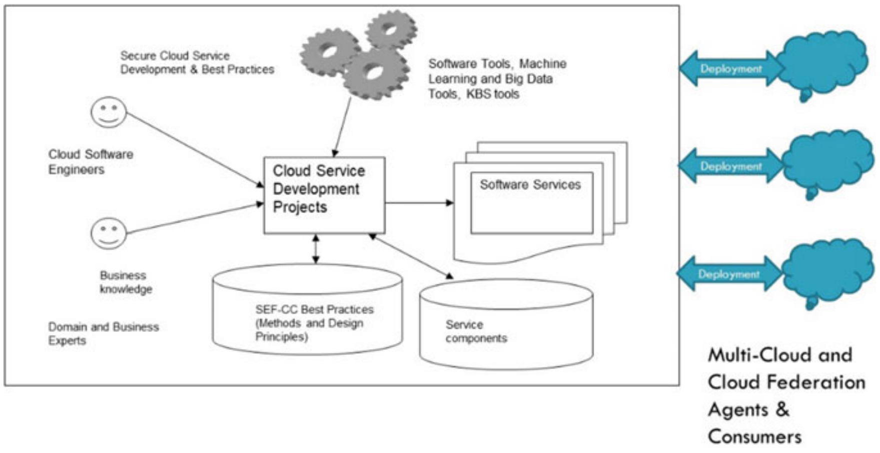

Secure Cloud Service Development & Best Practices
Software Tools, Machine Learning and Big Data Tools, KBS tools
Cloud Software Engineers
Cloud Service Development Projects
Software Services
Business knowledge
Domain and Business Experts
SEF-CC Best Practices (Methods and Design Principles)
Service components
Deployment
Deployment
Deployment
Multi-Cloud and Cloud Federation Agents & Consumers

Fig. 11.11 Service CASE tool for developing cloud services

(AUSOM) by applying MAPE-K loop (monitor, analyse, plan, act using stored knowledge). Similarly, Bocciarelli [28] proposed the design of business process modelling and simulation as a service (MSaaS). Therefore, with the emergence of AI, we believe we can develop an autonomic system as shown in Fig. 11.11 based on model services, design and develop services, test and deploy services using stored knowledge, best practices of SEF-SCC, and reuse of services by composition.

As shown in Fig. 11.11, SEF-SCC CASE aims to provide support for a complete business-driven service development lifecycle discussed in this chapter (BD-SDL) with service requirements modelling and simulation with BPMN engine; knowledge discovery engine provides best practices on service development and service component reuse.

11.4 SEF-SCC Framework Evaluation Case Study on BEPET and SEF-SCC Applications

This section aims to illustrate how BD-SDL and SEF-SCC best design practices have been evaluated using a case study on BEPET over 10-year period with more than 12 software engineers who are trained on the method and have been strict to use the BD-SDL principles. Chart 11.1 shows the empirical evaluation of the method proposed in this chapter. Chart 11.2 shows the number of service components developed for SPMaaS, SSEREaaS, SPIaaS, and SEaaS applications designed as part of the SEF-SCC applications.

There are further research challenges ahead in developing autonomic CASE presented in this chapter as well as developing numerous applications as a service.

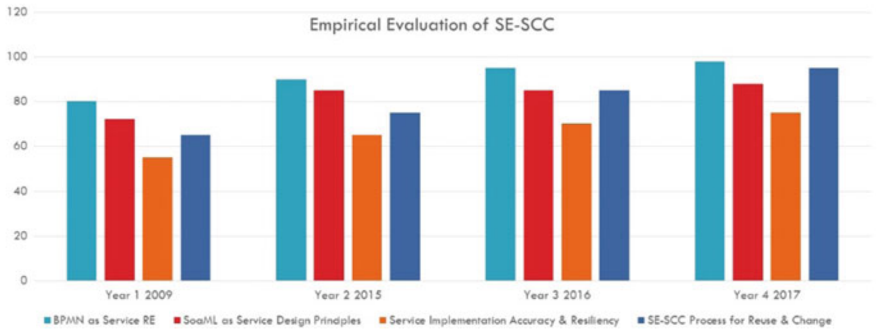


Chart 11.1 Evaluation results on the applicability of SEF-SCC service development lifecycle

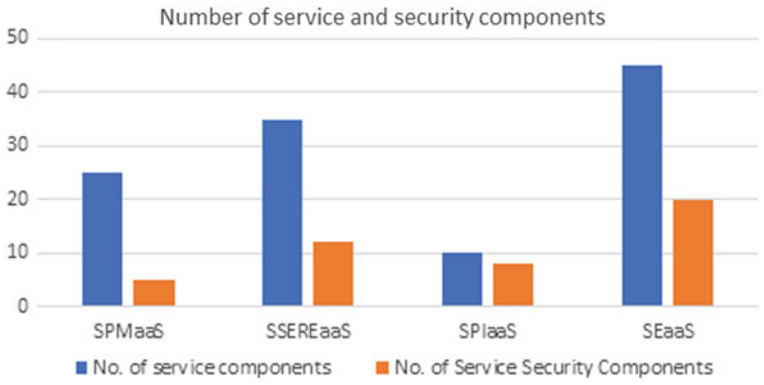


Chart 11.2 Number of service and service security components for SPMaaS, SSEREaaS, SPIaaS, and SEaaS

11.5 Conclusion

Service computing has emerged to offer greater business flexibility and globalising of economy as well as social mobility by connecting and communicating with people.

This chapter has presented our approach to Software Engineering Framework for Service and Cloud Computing (SEF-SCC) to address the need for a systematic approach to design and develop robust, resilient, and reusable services. This chapter presented SEF-SCC methods, techniques, and a systematic engineering process supporting the development of service-oriented software systems and software as a service paradigm. As a part of the SEF-SCC design process and its best practice, design guidelines have been valuable assets for creating an engineered approach to SOA and cloud services and also have been used to train software engineers for more than 10 years for developing their skills in service computing.

References

1. Hu P et al (2017) Survey on fog computing: architecture, key technologies, applications and open issues. J Netw Comput Appl 98(2017):27–42
2. Mahmud R, Ramamohanarao K, Buyya R (2010) Latency-aware application module management for fog computing environments. ACM Trans Embed Comput Syst 9(4), Article 39
3. Subramanya T et al (2017) A practical architecture for mobile edge computing. In: IEEE conference on network function virtualization and software defined networks (NFV-SDN)
4. Sommerville I (2012) Challenges for cloud software engineering. http://pire.opensciencedatacloud.org/talks/Cloud-Software-Challenges.pdf
5. Ramachandran M (2011) Software components for cloud computing architectures and applications. In: Mahmood Z, Hill R (eds) Cloud computing for enterprise architectures. www.springer.com/computer/communication+networks/book/978-1-4471-2235-7
6. Ramachandran M (2012) Software security engineering: design and applications. Nova Science Publishers, New York, USA, 2011. ISBN: 978-1-61470-128-6. https://www.novapublishers.com/catalog/product_info.php?products_id=26331
7. Ramachandran M (2016) Software security requirements engineering and management as an emerging cloud service. Int J Inf Manage 36(4):580–590. https://doi.org/10.1016/j.ijinfomgt.2016.03.008
8. Ramachandran M, Chuagle V (2016) Software Project Management as a Service (SPMaaS): perspective and benefits. In: Mahmood Z (ed) Software project management for distributed computing: life-cycle methods for developing scalable and reliable tools. Springer, Berlin
9. Ramachandran M, Chang V (2014) Modelling financial SaaS as service components. In: International workshop on emerging software as a service and analytics (ESaaSA 2014), the 4th international conference on cloud computing and services science, CLOSER 2014, 3–5th Apr, Barcelona, Spain
10. Chang V, Ramachandran M (2016) Towards achieving cloud data security with the cloud computing adoption framework. IEEE Trans Serv Comput 9(01):138–151
11. Ramachandran M (2014) Enterprise security framework for cloud data Security. In: Chang V (ed) Book chapter delivery and adoption of cloud computing services in contemporary organizations. IGI Global, Hershey
12. Ramachandran M, Mahmood Z (2011) A framework for internet security assessment and improvement process. In: Ramachandran M (ed) Chapter 13, Knowledge engineering for software development life cycles: support technologies and applications. IGI Global Publishers, USA. ISBN-13 978-1609605094
13. Ramachandran M (2016) Service-oriented architecture for big data and business intelligence analytics in the cloud. In: Sugumaran V, Sangagaiah A, Thangavelu A (eds) Computational intelligence applications in business intelligence and big data analytics. CRC Press (Taylor & Francis Group), Boca Raton
14. Ramachandran M, Mahmood Z, Raj P (2014) Service oriented architecture for e-government applications. In: Mahmood Z (ed) Emerging mobile and web 2.0 technologies for connected e-government. IGI Global, Hershey
15. Delgado A et al (2011) Business Process Service Oriented Methodology (BPSOM) with service generation in SoaML. In: Advanced information systems engineering—23rd international conference, CAiSE 2011, London, UK, 20–24 June 2011
16. Mahmood Z, Saeed S (eds) (2013) Software engineering framework for cloud computing paradigm. Springer, Berlin
17. Ramachandran M (2013) Business requirements engineering for developing cloud computing services. In: Mahmood Z, Saeed S (eds) Software engineering frameworks for cloud computing paradigm. http://www.springer.com/computer/communication+networks/book/978-1-4471-5030-5
18. Chen S (2005) Strategic management of e-business, 2nd edn. Wiley, Hoboken

19. Gupta D (2013) Service point estimation model for SOA based projects. http://servicetechmag.com/system/application/views/I78/1113-1.pdf
20. Guha R (2013) Cloud COCOMO/modified COCOMO for cloud service cost and effort estimation technique: impact of semantic web and cloud computing platform on software engineering. In Mahmood Z, Saeed D (eds) Software engineering framework for cloud computing paradigm. Springer, Berlin
21. Priyanka C, Chana I, Rana A (2012) Empirical evaluation of cloud-based testing techniques: a systematic review. ACM SIGSOFT Softw Eng Notes 37(3):1–9
22. Mirandola R et al (2014) A reliability model for Service Component Architectures. J Sys Soft 89(2014):109–127
23. Leung A, Spyker A, Bozarth T (2018) Titus: introducing containers to the Netflix cloud. Commun ACM 61(2)
24. Savage N (2018) Going serverless. Commun ACM 61(2):15–16
25. Angelov S, Grefen P, Greefhorst D (2012) A framework for analysis and design of software reference architectures. Inf Softw Technol 54(2012):417–431
26. Liu F (2011) NIST cloud computing reference architecture NIST special publication 500-292
27. Bellur U (2017) AUSOM: Autonomic Service-Oriented Middleware for IoT-based systems. In: IEEE 13th world congress on services
28. Bocciarelli P et al (2017) Business process modeling and simulation: state of the art and MSaaS opportunities. In: SummerSim '17 proceedings of the summer simulation multi-conference, Bellevue, Washington

Chapter 12
Data and Computation Movement in Fog Environments: The DITAS Approach

Pierluigi Plebani, David Garcia-Perez, Maya Anderson, David Bermbach, Cinzia Cappiello, Ronen I. Kat, Achilleas Marinakis, Vrettos Moulos, Frank Pallas, Stefan Tai and Monica Vitali

Abstract Data-intensive applications are becoming very important in several domains including e-health, government 2.0, smart cities, and industry 4.0. In fact, the significant increase of sensor deployment in the Internet of things (IoT) environments, in conjunction with the huge amount of data that are generated by the smart and intelligent devices such as smartphones, requires proper data management. The goal of this chapter is to focus on how to improve data management when data are produced and consumed in a Fog Computing environment, where both resources at the edge of the network (e.g., sensors and mobile devices) and resources in the cloud (e.g., virtual machines) are involved and need to operate seamlessly together. Based on the approach proposed in the European DITAS project, data and computation movement between the *edge* and the *cloud* are studied, to create a balance between such characteristics as latency and response time (when data are stored in edge-located resources) and scalability and reliability in case of data residing in the cloud. In this contribution, to enable data and computation movement, an approach based on the principles of Service-Oriented Computing applied to a Fog environment has been adopted.

P. Plebani (✉) · C. Cappiello · M. Vitali
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milan, Italy
e-mail: pierluigi.plebani@polimi.it

D. Garcia-Perez
Atos Spain SA, Pere IV 291, 08020 Barcelona, Spain

M. Anderson · R. I. Kat · S. Tai
IBM Research, Haifa University Campus, Mount Carmel, Haifa 3498825, Israel

D. Bermbach · F. Pallas
IS Eng Research, TU Berlin, Einsteinufer 17, 10587 Berlin, Germany

A. Marinakis · V. Moulos
NTUA—National Techinical University of Athens, Iroon Polytechniou Str, 15773 Zografou Campus, Athens, Greece

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_12

12.1 Introduction

In various fields, commercial and social, the amount of data produced and processed is ever increasing and so appropriate applications are required that are able to manage this vast amount of data, commonly known as Big Data. Scientific disciplines, such as bioinformatics or astrophysics, have already encountered this data deluge several years ago and have recognized the data-intensive science as the fourth research paradigm in addition to the experimental, theoretical, and simulation-based paradigms [1, 2]. For this reason, parallel computing, even based on specialized hardware—and where scaling-out approach was preferred to a scaling-up approach—has been adopted to produce tools for improving the efficiency of scientific computations.

With the proliferation of smart devices, the same type of scenarios is turning up in other domains. Predictions on IoT (Internet of things) estimate 32 billion connected devices by the year 2020 that will produce in excess of 440 petabytes of data (accounting for approximately 10% of the world data) [3]. On top of such huge amounts of data, data processing and analysis are required, often in real time, to extract meaningful information. This becomes crucial when data comes from the operational infrastructure of large companies, and analysis is required to compute key performance indices (KPIs) and to produce reports for the management to support strategic business decisions. Combining this complexity with the need for being faster in management decisions, real-time processing of data streams becomes fundamental [4].

In cloud platforms, these data-intensive applications (DIA) have found the ideal environment for executing tasks and processing significant amounts of data, while scalability and reliability are guaranteed by the underlying infrastructure [5]. Furthermore, the adoption of cloud solutions has had a significant impact on costs, as storing and managing data in the cloud are typically inexpensive compared to on-premises solutions. The development of this type of applications took advantage of several approaches that had been proposed in the recent years. Especially within the Apache community [6], several projects have been started to deal with batch processing, e.g., Hadoop and real-time processing, e.g., Storm, Spark, or Kafka, just to name a few. With these tools, data-intensive application developers can rely on a specific programming model that simplifies access to heterogeneous data sources storing data in different formats and significantly reduces the effort required to make data processing scalable and reliable. To reduce the inherent latency of the cloud, specific architectures, e.g., the Lambda Architecture [7], have been proposed for real-time analytics.

Although these approaches are now widely adopted, there are situations in which relying on a cloud infrastructure for implementing data-intensive applications is not beneficial, especially when data are produced outside of the cloud by devices (such as smart objects, laptops, servers, dedicated systems) residing on the premises of who want to analyze the data thus produced. Firstly, when data are produced at the edge of the network but processed in the cloud using the solutions mentioned above, bandwidth could become a bottleneck thus increasing the latency. Secondly, security and privacy issues are still one of the main reasons why cloud adoption remains limited especially in application domains where data processing mainly involves sensitive information (e.g., e-health) that usually cannot be moved to the cloud as they are. On the other hand, applying the data processing solutions developed for cloud-based infrastructure directly to the edge could be very challenging: Each infrastructure has its own characteristics, could be managed differently, and involve a variety of devices with different characteristics that make a one-size-fits-all approach really difficult.

In such a scenario, Fog Computing [8], often also referred to as Edge Computing [9] becomes attractive models. Fog Computing is an emerging paradigm that aims to extend the Cloud Computing capabilities to fully exploit the potential of the edge of the network where traditional and commonly used devices (e.g., handheld devices, laptops, sensors) as well as new generations of smart devices (e.g. smart wearables and mobile devices) are interconnected within the IoT environment. Especially for data-intensive applications, since the IoT is a source for enormous amounts of data that can be exploited to provide support in a multitude of different domains (e.g., predictive machinery maintenance, patient monitoring), this new paradigm has opened new frontiers [10]. In fact, with Fog Computing, we can take advantage of resources living on the edge of the network and the cloud environment to exploit the respective advantages. Data could be stored closer to where they are produced using edge-located resources if the data cannot leave the boundary of the organization that owns them; conversely, cloud solutions could be adopted only after data are transformed to preserve privacy. Orthogonally, data processing could occur on the edge when the available information, computational power, and the response time do not require a scalable solution. On the other hand, data processing will be moved to the cloud when limitation of network bandwidth is not an issue.

The goal of this chapter is to discuss how data management in data-intensive applications can be improved through data and computation movement in Fog environments. In particular, this chapter focuses on the experience of the European DITAS project [11] which aims to improve, through a combined cloud and fog platform, the development of data-intensive applications by enabling information logistics in Fog environments for delivering information at the right time, the right place, and with the right quality [12] using both resources belonging to the cloud and the edge. The resulting data and computation movement are enabled by virtual data containers (VDCs) that provide an abstraction layer, adopting the Service-Oriented Computing [13] principles and hiding the underlying complexity of an infrastructure made of heterogeneous devices. Applications developed using the DITAS toolkit are able to exploit the advantages of both cloud-based solutions in terms of reliability and scalability, as well as edge-based solutions with respect to latency and privacy.

The rest of this chapter is structured as follows. Section 12.2 introduces the characteristics of data-intensive applications when immersed in a Fog Computing-based environment. Section 12.3 discusses the approach adopted in the DITAS project to support the deployment and execution of a data-intensive application in

Fog environments, while Sect. 12.4 specifically focuses on the data and computation movement actions. Finally, Sect. 12.5 discusses related work, while Sect. 12.6 concludes the work outlining the future work that the DITAS project focuses on.

12.2 Data-Intensive Applications in Fog Computing

Data-intensive applications are becoming more and more crucial elements of IT systems due to the ever-increasing amount of data that needs to be managed. Several types of data-intensive applications can be developed to cover one or more phases of the data management, i.e., data capture and storage, data transmission, data curation, data analysis, and data visualization [5].

In recent years, usually under the umbrella of Big Data, researchers and practitioners have focused on providing tools, methods, and techniques for efficiently managing extensive amounts of data in various formats and schemas. As a result, distributed file systems (e.g., HDFS), new generations of DBMS where the relational model is no longer adopted (e.g., MongoDB [14], Cassandra [15]), and new programming models (e.g., MapReduce), as well as new architectures (e.g., Lambda Architecture) have been proposed. Regardless of the specific solution, most of them usually rely on resources available on the cloud, thus offering the possibility to easily scale applications in or out with the amount of data to be processed. Actually, in some cases, relying only on cloud infrastructures cannot be feasible for two main reasons: (i) data cannot be moved from where they are collected due to privacy/security issues, and (ii) the time required to move data to the cloud could be prohibitive. In such a scenario, Fog Computing [7] aims to support the required synergy between the cloud—where the application usually runs—and the devices at the edge of the network—where the data are generated—especially in IoT environments. In fact, cloud and edge are usually seen as two distinct and independent environments that, based on the specific needs, are connected to each other to move data usually from the edge to the cloud. To create a synergy between these two paradigms, Fog Computing has been coined—initially in the telco sector—to identify a platform able “to provide compute, storage, and networking services between cloud data centers and devices at the edge of the network” [16]. Based on this, and also in the light of the definition proposed by the OpenFog Consortium [8], we consider Fog Computing as the sum of Cloud and Edge Computing where these two paradigms seamlessly interoperate to provide a platform where both computation and data can be exchanged in both downstream and upstream direction [17] (refer to Fig. 12.1).

Based on these definitions, Cloud Computing is mainly related to the core of the network, whereas Edge Computing is focused on supporting the owners of resources through the local “in-situ” means for collecting and preprocessing data before sending it to cloud resources (for further utilization), thus addressing typical constraints of sensor-to-cloud scenarios like limited bandwidth and strict latency

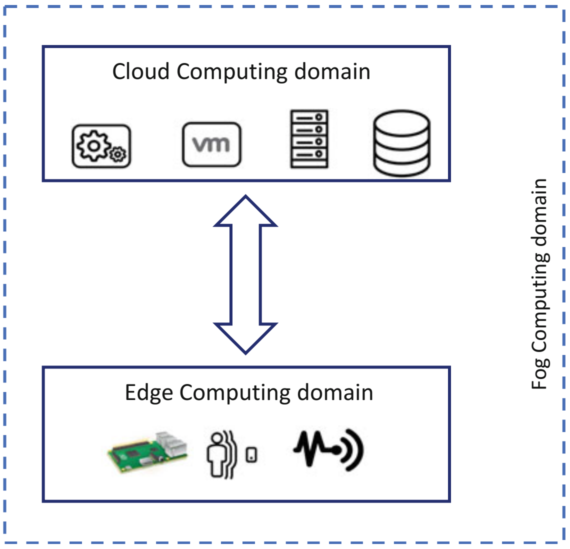


Fig. 12.1 Fog Computing environment

requirements. Furthermore, cloud resources include physical and virtual machines which are capable of processing and storing data. On the other hand, smart devices, wearables, or smartphones belong to the set of edge-located sources. While Cloud Computing is devoted to efficiently managing capabilities and data in the cloud, Edge Computing is focused on providing the means for collecting data from the environment (which will then be processed by cloud resources) to the owner of the available resources.

Exploiting the Fog Computing paradigm, these two environments seamlessly interoperate to provide a platform where computation and data can be exchanged in both downstream and upstream direction. For instance, when data cannot be moved from the edge to the cloud, e.g., due to privacy issues, then the computation is moved to the edge. Similarly, when there are insufficient resources at the edge, data are moved to the cloud for further processing. The DITAS project aims to provide tools, specifically designed for developers of data-intensive applications, which are able to autonomously decide where to move data and computation resources based on information about the type of the data, the characteristics of applications as well as the available resources at both cloud and edge locations along with application constraints such as the EU GDPR privacy legislation [18].

To achieve this goal, DITAS adopts Service-Oriented Computing principles [13] where data used by data-intensive application developers are provided through the Data as a Service (DaaS) paradigm. As shown in Fig. 12.2, we assume the existence of several data providers which take care of optimizing the data collection and provisioning. Data sources could be deployed on the edge (e.g., data coming from sensors) or on the cloud (e.g., data about business transactions). The goal of the data provider is to develop and deploy a DaaS which hides the complexity of managing his/her data sources and to provide them through APIs. Such APIs are used by data

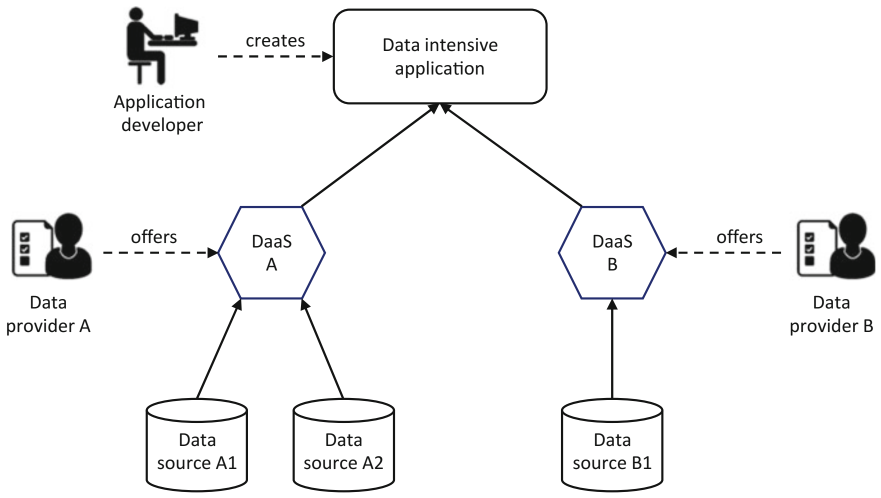


Fig. 12.2 DITAS approach

consumers that, in our scenario, are represented by data-intensive application developers which process the obtained data in order to generate added-value applications.

Focusing on these two main standpoints, the next paragraphs focus on how Service-Oriented Computing can be adopted in a Fog environment.

12.2.1 Data Provisioning

With respect to the typical data management life cycle, data providers are usually in charge of collecting, storing, and supporting access to data over which they have complete control. For instance, in IoT scenarios, data are generated at the edge of the network (e.g., sensors, mobile devices), and the data provider has to set up an environment allowing the data consumers to properly access them. This requires moving the data to the cloud, where a seemingly unlimited amount of resources is available for efficient storage and processing of data as well as exposing it through APIs. Although cloud resources ensure high reliability and scalability, network capacity might negatively influence latency when data movements among resources on the cloud and the edge occur; thus, the advantage of the fast processing at the cloud might be wasted, and the offered service quality might be negatively affected.

As an example, a data provider could be a highway manager that offers data about the status of the traffic, or time series about the number of vehicles, their type, accidents, etc. Assuming that this information is obtained by reading values of sensors in the field, or based on the information coming from applications, these data are usually moved to the cloud so that they are easily and widely accessible.

While cloud platforms provide solutions where interoperability among different infrastructures is now easy to achieve, we cannot say the same for the edge part. Indeed, agreement about protocols, data formats, and interfaces of smart devices, sensors, and smartphones has not been achieved yet. This results in difficulties when data providers have to deal with heterogeneous devices that need to communicate or, at least, need to send the data to the cloud for further processing.

Focusing on the processing, a significant issue to be taken into account concerns the ever-increasing computational and storage capabilities provided by the resources on the edge. Regarding data storage, once the data are created, it is not required to immediately move them to a capable storage in the cloud. Conversely, it is possible to leave the data where they are produced and, in addition, to exploit the computational power to perform some preprocessing directly on the edge.

As a last step in the data management life-cycle, data providers have to make data available to data consumers also considering that they could have different needs and different capabilities. In this light, principles of Service-Oriented Computing can be adopted to define the DaaS interfaces that data providers have to propose to allow the final users to properly consume the data. Such interfaces have to consider both functional (i.e., how the data can be accessed) and non-functional (i.e., which is the quality of the data and the service) aspects.

Regarding functional aspects, the offered APIs can adopt the REST [19] architectural style, typical SQL-based interfaces or others. Concerning non-functional aspects, data quality dimensions (e.g., timeliness, accuracy) and service quality dimensions (e.g., response time, latency, data consistency) need to be computed and balanced according to consumer expectations.

12.2.2 Data Consumption

From the consumer standpoint, data are accessible by invoking the available DaaS. Assuming that there could be several data providers, each of them in charge of managing different data sources, data consumers have to deal with a plethora of DaaS, each of them providing specific data with different quality of data and service.

Data-intensive applications are built on top of these DaaS with the goal of analysing and processing of the provided data to create added value for the customer. For instance, data coming from the highway manager in the example above can be combined with weather information to analyze the correlation between accidents and severe weather conditions.

A particular aspect considered in DITAS relates to the combination between the Fog Computing paradigm and the Service-Oriented Computing approach. Consequently, we assume that while resources required for service provisioning are known in advance and under the control of the service provider, there are additional resources that live on the premises of the customers which will be known by the provider right before starting the data consumption along with the quality of data and service requirements. With respect to the typical approach, these additional resources are not part of the client infrastructure, but they can be included in the set of resources belonging to the service provisioning infrastructure. In this way, the data provider can exploit them to improve the user experience of that specific customer. Among the different opportunities that this scenario could open, in this chapter, we focus on the possibility of hosting part of the application logic which composes the data provisioning. Similarly, the resource on the edge of the network can be used to host the data that are considered relevant for the consumer and thus to reduce the latency when users request them.

12.3 DITAS Approach

From the perspective of a data provider, exposing data following a DaaS paradigm requires to decide where to store data, in which format, how to satisfy the security constraints, and many other aspects. This situation becomes even more complex when dealing with heterogeneous systems where different devices are involved in the data management. For instance, over time, different versions of smart devices might be used to collect data from the sensors installed in manufacturing plants. This implies that the developers have to manage this heterogeneity as well as to properly distribute the data among edge and cloud, to make applications as efficient as possible.

Moving to the data consumer perspective, the development of data-intensive applications requires to select the proper set of DaaS, considering both functional and non-functional aspects, to connect and start interacting with them, ensuring that the agreed quality of data and service is respected, etc. All of these aspects could distract the attention of the data-intensive application developer from the business logic, i.e., to organize the actual data processing.

For this reason, in order to improve the productivity of application developers with the DITAS platform, we aim to offer tools for smart data management trying to hide the complexity related to data retrieval, processing, and storage. To this end, data-intensive applications in DITAS are not directly connected to the data sources containing the necessary data, but the access to these data sources is mediated by a specific component called Virtual data container (VDC) (see Fig. 12.3), which represents the concrete element able to provide a DaaS. In more detail, a VDC:

- Provides uniform access to data sources regardless of where they run, i.e., on the edge or on the cloud.
- Embeds a set of data processing techniques able to transform data (e.g., encryption, compression).
- Allows composing these processing techniques in pipelines (inspired by the node-RED programming model) and executing the resulting application.
- Can be easily deployed on resources which can live either on the edge or in the cloud.

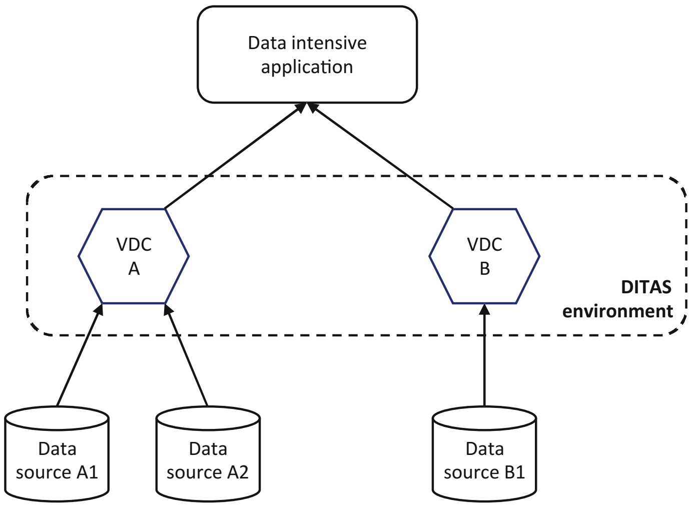


Fig. 12.3 Role of VDC in the DITAS data management

To manage this indirect access between data sources and the data-intensive application, DITAS distinguishes between the data sources' life cycle and the data-intensive application's life cycle. The former defines the relationship between a data source and a VDC, whereas the latter defines the relationship between the data-intensive application and the VDCs which give access to the required data.

For this reason, in the terms adopted in DITAS, a data administrator has a complete knowledge of one or more data sets and is responsible for making them available to applications that might be managed by other developers through a DaaS. On the other hand, a DIA developer defines the functional and non-functional aspects of the data-intensive application and selects the best fitting DaaS needed to compute the analysis. Moreover, the DIA developer is in charge of developing the business logic of the data-intensive application. Exploiting the DITAS-SDK, application developers only focus on data management having the virtual data container (VDC) handle the burden of selecting the best location where to store the data and the most suitable format to satisfy both the functional and non-functional requirements specified by the application designer.

Based on the work done by the data administrator and the DIA developer, DITAS offers two environments able to support the data-intensive application life cycle. More specifically:

- An SDK assisting both the data administrators and DIA application developers.
- An execution environment (EE) where the deployed VDCs and DIA operate.

The next paragraphs detail the steps that compose the data-intensive application life cycle, as shown in Fig. 12.4. In particular, we have identified two main phases:

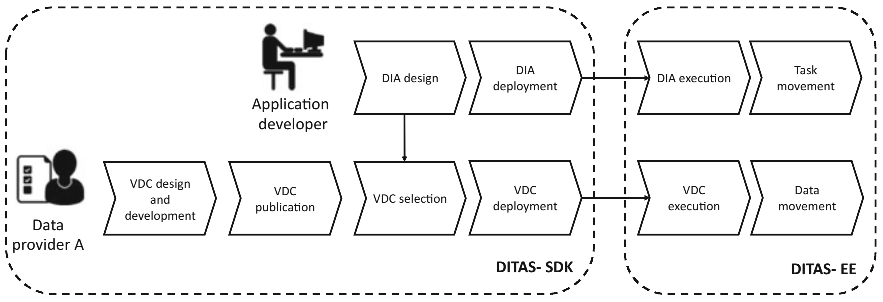


Fig. 12.4 Data-intensive application life cycle in DITAS

the design and development phases, which take advantage of the DITAS-SDK and the execution phase which relies on the DITAS-EE.

12.3.1 Design and Development Phase

The first step of the application life-cycle concerns the work performed by the data administrator (a.k.a. data provider), who—based on the managed data sources—creates a VDC Blueprint which specifies the characteristics of a VDC in terms of following:

- The exposed data sources.
- The exposed APIs.
- How the data from the data sources needs to be processed in order to make them available through the API.
- The non-functional properties defining the quality of data and service.
- The components cookbook: a script defining the modules composing the container as well as their deployment.

As DITAS follows the Service-Oriented Computing principles, the visibility principle requires to publish a description of a service to make it visible to all the potential users. As a consequence, the data administrator publishes the VDC Blueprint. At this stage, no specific approach for the VDC discovery has been adopted (i.e., centralized registry, distributed publication), as it is an issue to be taken into account for future work.

Once published, the DIA developers come into play. In fact, their role is to search for the data that are relevant for the applications they are developing. As the information included in a VDC Blueprint also concerns functional and non-functional aspects, a DIA developer relies on this information to select the most suitable VDC according to its purposes. It is worth noticing that, based on the nature of the DIA, the developer could select different VDCs referring to different data. A peculiar aspect of the DITAS approach concerns the data utility, that is defined as the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics [20]. In this way, data utility considers both functional (i.e., which data are needed) and non-functional (e.g., the data accuracy, performance) aspects.

Finally, the developer designs and develops the DIA and deploys it on the available resources which can be located on the edge or in the cloud. The initial deployment is the key element in the approach; as in this phase, it is required to know which are all the possible resources on which the VDC can be executed. As introduced in Sect. 12.2, the considered Fog environment implies that DaaS can be provided using resources belonging to both the provider and the consumer. Without loss of generality, we can assume that the provider resources are always in the cloud, while the consumer resources are always on the edge. In this way, a VDC living in the cloud has more capacity and it probably lives close to the data source to which it is connected. Conversely, a VDC living on the edge has the advantage of living closer to the user, thus reducing latency when providing the requested data. Deciding where to deploy the VDC depends on the resources required by the VDC (e.g., it might happen that the amount of resources to process the data before making them available to the user cannot be provided at the edge), the network characteristics (e.g., the connection at the consumer side can support a high-rate transmission), and security (e.g., not all the data can be moved to the consumer side, thus even the processing cannot be placed at the edge).

Once the DIA has been deployed, DITAS supports a flexible execution that initiates data and computation movement when necessary to ensure the fulfillment of the non-functional requirements.

12.3.2 Execution Phase

Before introducing the steps of the DIA life cycle related to the execution, it is worth introducing some of the elements composing the DITAS Execution Engine (DITAS-EE). As reported in Fig. 12.5, the DITAS-EE solution is built on top of a Kubernetes [21] cluster. In fact, given a VDC Blueprint, based on the cookbook section, a docker [22] container is generated and deployed. Furthermore, given a VDC Blueprint, many application developers can select it for their own application. As a consequence, the DITAS-EE has to manage several DIAs which operate with different VDCs. Moreover, as the same VDC Blueprint can be adopted in different applications, each of these applications includes instances generated from the VDC Blueprint; thus, they are connected to the same data sources.

To properly manage this concurrent access, given a VDC Blueprint, the DITAS-EE includes a virtual data manager (VDM) that controls the behavior of the different instances of the same VDC Blueprint to correctly operate on the data sources and no conflict arises when enacting data and/or computation movements.

Thanks to the abstraction layer provided by the VDC, applications deployed through the platform can access the required data regardless of their nature and

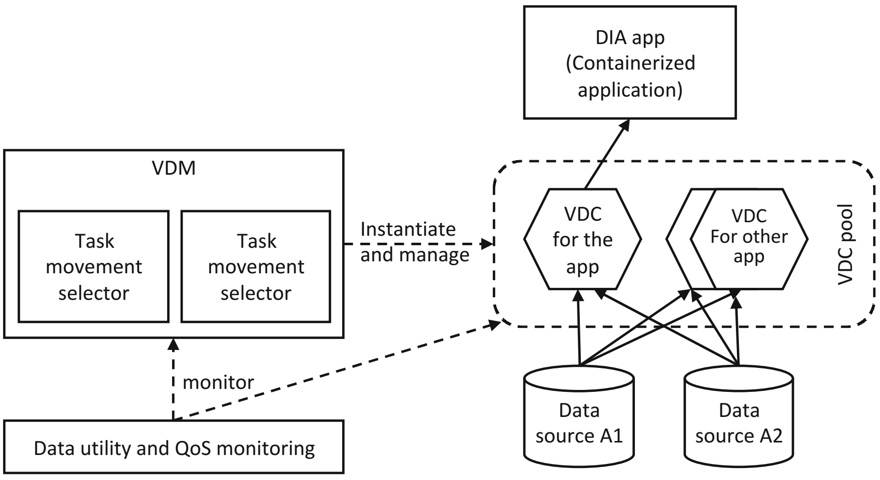


Fig. 12.5 DITAS execution environment

location (cloud or edge). Due to the distributed nature of the applications to be managed, to the execution environment being distributed by definition and because of the different computational power offered by the devices, it might happen that only a subset of the modules can be installed on a specific edge device. For this reason, at deployment time, not only the data-intensive application is distributed over the cloud and edge federations, but also the execution environment is properly deployed and configured to support the data and computation movement. The decision on where to locate both the application and the data required by the application itself is taken at design time, but can be updated during the application execution, according to the detected state of the execution environment. Some details about the approach followed to support the data and computation movement are introduced in the next section.

12.4 Data and Computation Movement in Fog Environments

Movement strategies provide solutions for moving data and computation in a Fog environment, taking into consideration all the factors that affect the application execution, data usability, and trying to keep the QoS and the data quality at the levels required by the application designer.

Data and computation movement strategies are used to decide where, when, and how to save data—on the cloud or on the edge of the network—and where, when, and how to move tasks composing the application to create a synergy between traditional and cloud approaches able to find a good balance between reliability, security, sustainability, and cost.

The driver for data and computation movement is the evaluation of the data utility [20]. When an application is deployed through the DITAS platform, the application designer expresses application requirements about the QoS and quality of data used both to lead the data source selection and to select a proper computation and data location. In the application requirements, both hard and soft constraints are expressed. When the evaluation of the data utility does not respect the application designer requests, the VDM will enact the most suitable data and computation movement strategies to balance the posed requirements such as reducing the latency or the data size, ensuring a given accuracy, while maintaining—if requested—privacy and security. Data and computation movements are executed to satisfy all the hard constraints and, as much as possible, soft constraints and requirements expressed by the user with the final objective of executing the requested functionality having in mind also the maximization of the user experience. As the computation movement requires a dynamic deployment of the data processing tasks, data movement could require only a transformation of the data format (e.g., compression or encryption) or could also affect the quality of the data (e.g., data aggregation).

Data and computation movement are managed over the entire life cycle of the application, from its deployment to its dismissal. During this time, the application and its data sources are monitored and evaluated in order to satisfy the hard and soft requirements expressed by the application developer. As the decision of when, how, and where data and computation movement must occur depends on the current situation in which the data-intensive application operates, the execution environment includes a distributed monitoring system.

The management of data and computation movement is a life cycle composed of the steps of the monitor-analyze-plan-execute (MAPE) control loop, as briefly discussed below:

- Monitor: a DIA is monitored (using the data utility and QoS monitor module) through a set of indicators providing information about both the application behavior and the data source state. This set of indicators can be enriched by a dependency map where such indicators are related to each other, giving a more refined knowledge about the execution environment.
- Analyze: The result of the previous phase is used to compare the current situation with the required data utility values. If the data utility provided to the application does not satisfy the application requirements, an exception is raised. Such an analysis is one of the main tasks to be executed by the VDM.
- Plan: According to the detected violation, some movement actions should be enacted. These actions are in fact data movement and computation movement strategies. To support the planning phase, we will study dependencies among the different data and computation movement techniques in order to identify positive or negative effects to the indicators related to aspects such as reliability, response time, security, and quality of data that need to be measured during the

execution of applications. Knowing these relations, it is possible to select the proper action for a violation. The impact on data utility derived from the enactment of the data and computation movement strategies will be analyzed and predicted applying data mining techniques to logs obtained from previous executions. Referring to the DITAS-EE architecture, the task movement selector and the data movement selector are the two modules in charge of the planning.
- Execute: Once the strategies have been selected, they can be enacted in order to fix violations. For data movement, specific modules are executed to move the data from the source to the destination, whereas for computation movement, the set of possible actions corresponds to the Kubernetes capabilities.

We define a movement strategy as a modification in the placement of a computing task or a set of data. The abstract movement strategy is characterized by one movement action and an object category (e.g., data, computation). The abstract movement strategy is also characterized by the effects on the environment that the enactment of such a strategy will cause. This information can be retrieved by a knowledge base built from the observation of previous enactments using machine learning techniques like reinforcement learning techniques.

In order to enact a movement strategy, it is necessary to specify also the actual object of the movement and its initial and final location. We define this as movement strategy instance. A movement strategy instance may be subject to some constraints given by the object of movement. If we consider data movement, a constraint can be related to privacy and security policies on the moved data. These policies are independent from the context and need to be applied anytime a movement action is applied to an object affected by them.

In order to enact a movement action, several alternative techniques may be used. A movement technique is a building block of a movement strategy. Strategies will combine these building blocks according to the specific needs of an application. Similarly, computation movement techniques will be proposed to define how to distribute the tasks to be executed among the available nodes taking into account the requirements of the task, the resource made available by a node, and general requirements at application level. For instance, in case of a movement strategy requiring data aggregation, one of a set of different movement techniques may be selected, each technique implementing a different aggregation algorithm.

The selection of the most suitable movement action for a given context should be driven by the expected utility improvement, which is computed on the basis of the detected violation and the known effects of the strategy on the environment.

Based on this definition of the movement strategy, the primary goal of DITAS is to find a coherent mechanism for deciding which is the best data/computation movement action to be enacted based on the application characteristics, the nature of the data, privacy and security issues, and the application's non-functional requirements. Creating rules and selecting parameters for an automatic selection of data movement actions in different contexts represents now a major challenge. In Fig. 12.6, we show a preliminary model specifying the influences that need to be mapped between several elements of the environment. More specifically, through

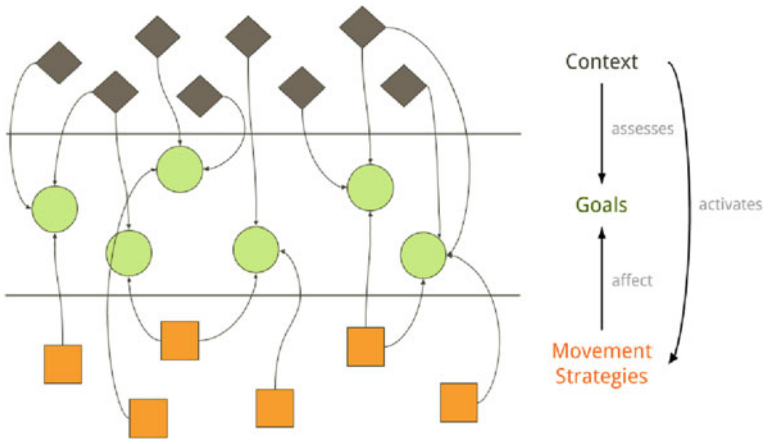


Fig. 12.6 Influences among context goals and movement strategies

the analysis of the data collected by the monitoring system, some events are raised by violations in the data utility, which compose the context in the top layer of the figure. These events are linked to the goals in the middle layer, which are a representation of the user requirements composing the data utility evaluation. At the bottom level, we represent the available movement strategies together with their effects on the goals.

12.5 Related Work

Since the 1990s, when interconnection of heterogeneous information systems managed by different owners became easier and when the Web started managing significant amounts of information, the problem of delivering such information has become more and more relevant: The more the data are distributed, the more difficult it is to find the information needed. Thus, tools are required to guide the users in this task. In this scenario, information logistics has emerged as a research field for optimizing the data movement, especially in networked organizations [12]. As discussed in [23], information logistics can be studied from different perspectives, e.g., how to exploit the data collected and managed inside an organization for changing the strategy of such organizations, how to deliver the right information to decision makers in a process, or how to support supply chain management. In our case, according to the classification proposed in [23], we are interested in user-oriented information logistics, i.e., the delivery of information at the right time, the right place, and with the right quality and format to the user [24]; thus, data movement becomes crucial. As a general framework, there are three sets of data movement techniques:

- The first includes techniques that affect neither the content nor the structure of the moved data. In this case, most of the approaches are application-driven; i.e., applications accessing this data heavily influence the adoption of the techniques [25–28]
- The second set concerns techniques that do not modify the content of the data but only their format. Lossless, both spatial and temporal compressions as well as data encryption mechanisms fall into this category with the objective of either reducing the amount of data or making the communication secure [29, 30].
- Finally, the third set includes techniques that operate on the data transmitted aiming to improve the performance of data movement while maintaining a sufficient level of data quality [24, 31, 32].

To support data movement in a Fog environment which has to deal with heterogeneous devices, data virtualization becomes fundamental. Data virtualization [33] is a data integration technique that provides access to information through a virtualized service layer regardless of data location [34]. Data virtualization [35] allows applications to access data, from a variety of heterogeneous information sources, via a request to a single access point so that it provides a unified, abstracted, real-time, and encapsulated view of information for query purposes and can also transform the information to prepare it for consumption by various applications. Data virtualization solutions add levels of agility (business decision agility, time-to-solution agility, or resource agility) that are difficult to achieve with traditional ETL solutions.

Container-based virtualization is one of the two approaches of lightweight virtualization [36], which minimize the use of processor and memory resources by sharing system calls with the host operating system. Managing data in containers can be done either by keeping the data with the container or by implementing a dedicated data layer [37]. Keeping the data with the container requires the use of techniques that move the data within the container. An example is from ClusterHQ's Flocker [38], which ensures that when an application container moves, the data container moves with it. By implementing a dedicated data layer for the storage container, data services (databases, file systems) can be implemented on more persistent entities such as virtual machines and physical servers.

12.6 Concluding Remarks

Fog Computing is an emerging paradigm able to support the development, deployment, and execution of distributed applications. This chapter has focused on a specific type of applications: data-intensive applications which have to deal with the gathering, processing, provisioning, and consumption of data. Following the Service-Oriented Computing principles, this chapter introduces the approach provided by the DITAS project that allows a flexible execution of data-intensive applications. Such flexibility is provided through data and computation movement actions allowing the data-intensive application to change the way in which the processing is distributed at run time, as well as to optimize how the data are distributed among the different nodes involved in the execution which may belong to edge and cloud environments. Computation movement is ensured by the adoption of a containerized solution which creates self-contained modules, i.e., VDC, that can be easily moved around the Fog environments. Data movement is supported by the execution of specific actions that are driven by the data utility that defines the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics.

Since the DITAS platform is still under development, future work will focus on the validation in real applications. In particular, the approach will be tested in a real case study concerning an industry 4.0 scenario. Here, data coming from sensors installed on some machinery need to be quickly processed exploiting both the computational power provided at the edge, which ensures reduced latency, and the computational power available in the cloud, which ensures significant scalability. In particular, the impact in terms of overhead introduced the DITAS platform will be analyzed.

References

1. Szalay A, Gray J (2020) Computing: science in an exponential world. Nature 440:413–414
2. Bell G, Hey T, Szalay A (2009) Beyond the data deluge. Science 323(5919):1297–1298
3. Turner V, Reinsel D, Gatz JF, Minton S (2014) The digital universe of opportunities: rich data and the increasing value of the internet of things. IDC White Paper
4. Plattner H, Zeier A (2012) In-memory management. Springer, Verlag
5. Chen CLP, Zhang C (2014) Data-intensive applications, challenges, techniques and technologies: a survey on. Inf Sci 275:314–347
6. Apache Software Foundation (2008) http://www.apache.org
7. Marz N, Warren W (2013) Principles and best practices of scalable realtime data systems. Manning Publications, New York
8. OpenFog Consortium Architecture Working Group (2016) OpenFog Architecture Overview. http://www.openfogconsortium.org/ra. Accessed 31 Jan 2018
9. Shi W, Dustdar S (2016) The promise of edge computing. Computer 49(5):78–81
10. Bermbach D, Pallas F, Garcıa Pérez D, Plebani P, Anderson M, Kat R, Tai S (2017) A research perspective on Fog Computing. ICSOC-ISYCC Workshop, Malaga
11. ATOS Spain (2018) http://www.ditas-project.eu
12. Sandkuhl K (2008) Information logistics in networked organizations: selected concepts and applications. Springer, Heidelberg
13. MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R (2006) Reference model for service oriented architecture 1.0. Tech rep, OASIS
14. MongoDB Inc (2009) http://www.mongodb.com
15. Apache Software Foundation (2008) https://cassandra.apache.org
16. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the 1st edn of the MCC workshop on mobile cloud computing, pp 13–16

17. Plebani P, Garcia-Perez D, Anderson M, Bermbach D, Cappiello C, Kat RI, Pallas F, Pernici B, Tai S, Vitali M, (2017) Information logistics and fog computing: the ditas approach. In: Proceedings of the forum and doctoral consortium papers presented at CAISE 2017, Essen, Germany, pp 129–136. CEUR vol 1848
18. EU (2018) http://www.eugdpr.org/
19. Fielding R, Taylor R (2002) Principled design of the modern web architecture. ACM Trans Internet Technology, 2(2), pp 115–150
20. Cappiello C, Pernici B, Plebani P, Vitali M (2017) Utility-driven data management for data-intensive applications in fog environments. In: Advances in Conceptual Modeling, ER Workshops, pp 216–226
21. Kjubernetes (2018) http://www.kubernetes.io
22. Docker (2018) http://www.docker.com
23. Michelberger B, Andris RJ, Girit H, Mutschler B (2013) A literature survey on information logistics. In: Proceedings of 16th international conference of business information systems, BIS 2013, Poznan, Poland, June 19–21, Springer International Publishing, Heidelberg
24. D'Andria F, Field D, Kopaneli A, Kousiouris G, Garcia-Perez D, Pernici B, Plebani P (2015) Data movement in the internet of things domain. In: Service oriented and cloud computing: 4th european conference, ESOCC 2015, Taormina, Italy, 15–17 Sep 2015, Springer, Switzerland
25. Cappiello C, Hinostroza A, Pernici B, Sami M, Henis E, Kat RI, Meth KZ, Mura M (2011) ADSC: Application-driven storage control for energy efficiency. ICT-GLOW 2011, pp 165–179
26. Logothetis D, Olston C, Reed B, Webb KC, Yocum K (2010) Stateful bulk processing for incremental analytics. In: Proceedings of the 1st ACM symposium on Cloud computing (SoCC'10), ACM, New York, USA, pp 51–62
27. Leyshock P, Maier D, Tufte K, (2014) Minimizing data movement through query transformation. In: Proceedings of IEEE international conference on (Big Data), 27–30 Oct 2014, pp 311–316
28. Lim H, Herodotou H, Babu S (2012) Stubby: a transformation-based optimizer for MapReduce workflows. Proc VLDB Endowment 5(11):1196–1207
29. Srisooksai T, Keamarungsi K, Lamsrichan P, Araki K (2012) Practical data compression in wireless sensor networks: a survey. J Netw Comput Appl 35(1):37–59
30. Lu P, Zhang L, Liu X, Yao J, Zhu Z (2015) Highly efficient data migration and backup for applications in elastic optical inter-data-center networks. IEEE Netw 29(5):36–42
31. Ho TTN, Pernici B (2015) A data-value-driven adaptation framework for energy efficiency for data intensive applications in clouds. In: Proceedings of IEEE conference on technologies for sustainability (SusTech), pp 47–52
32. Peng X, Pernici B (2016) Correlation-model-based reduction of monitoring data in data centers. In: Proceedings of smart cities and green ICT systems (SMARTGREENS), pp 135–153
33. Davis JR, Eve R (2011) Data virtualization: going beyond traditional data integration to achieve business agility. Nine Five One Press
34. Mousa AH, Shiratuddin N, Abu Bakar MS (2015) RGMDV: An approach to requirements gathering and the management of data virtualization projects. In: Proceedings of the 2nd innovation and analytics conference & exhibition (IACE 2015), vol 1691, AIP Publishing, New York
35. Hopkins B, Cullen A, Gilpin M, Evelson B, Leganza G, Cahill M (2011) Data virtualization reaches the critical mass. Forrester Report
36. Vaughan-Nichols SJ (2006) New approach to virtualization is a lightweight. Computer 39 (11):12–14
37. Sahoo SK, Agrawal G (2004) Supporting XML based high-level abstractions on HDF5 datasets: a case study in automatic data virtualization. Languages and Compilers for High Performance Computing. Springer, Heidelberg, pp 299–318
38. ClusterHQ (2018) https://github.com/ClusterHQ/flocker

Chapter 13
Toward Edge-based Caching in Software-defined Heterogeneous Vehicular Networks

Adnan Mahmood and Hushairi Zen

Abstract Considerable technological advancements and pervasive use of smart connected vehicles have highlighted the true potential of vehicular networks, which, in the realm of smart cities, assists in improvement of road safety and traffic management and efficiency. Coupled with this is the massive innovation realized in wireless networking technologies, and today, heterogeneous networks encompassing 4G LTE, Wi-Fi, WiMAX, DSRC, and Terahertz communication are taking shape, synergy of which not only promises higher bandwidth but also ensures low latent infrastructure critical for supporting diverse range of safety applications. Heterogeneity, on the other hand, has introduced new challenges too, e.g., in terms of network fragmentation, an unbalanced traffic flow in a multi-path topology, and inefficient utilization of network resources. Accordingly, the emerging yet promising paradigm of software-defined networking (SDN) has, of lately, been introduced to vehicular networks for addressing such bottlenecks, which through its logically centralized control ensures programmability, scalability, elasticity, and agility. Nevertheless, a centralized control in a distributed environment like vehicular networks could become a *single point of network failure* and may also result in significant network management overhead in case of extremely dense traffic scenarios. Therefore, leveraging edge-based caching in heterogeneous vehicular networks is indispensable. This chapter brings forward the notion of SDN-based heterogeneous vehicular networking and argues that edge-based caching can help overcome the bottlenecks posed by traditional networking architectures especially in terms of ensuring low latency for safety-critical applications. Finally, open challenges and probable solutions are discussed.

A. Mahmood (✉) · H. Zen
Communications Research Group, Faculty of Engineering,
Universiti Malaysia Sarawak, Kota Samarahan, 94300 Sarawak, Malaysia
e-mail: adnanqureshi@ieee.org; amahmood@tssg.org

A. Mahmood
Emerging Networks Laboratory, Telecom. Software and Systems Group,
Waterford Institute of Technology, Waterford, Ireland

Z. Mahmood (ed.), *Fog Computing*,
https://doi.org/10.1007/978-3-319-94890-4_13

13.1 Introduction

Over the past few decades, vehicular ad hoc networks (VANETs) have gained a significant level of attention within both academia and the industry. Subsequently, a considerable amount of research has been carried out particularly in the domains of intra- and inter-vehicular communication, multi-hop routing protocols, clustering and/or platooning, vehicular cloud computing, and network security [1]. All of these research domains primarily assist in mitigating massive number of accidents on the roads, thereby enhancing the safety of passengers. For instance, if a vehicle *A* is moving in front of vehicle *B* on a high-speed expressway and suddenly makes a lane change due to a potential road hazard ahead, it should immediately send a warning (i.e., a cooperative message) to vehicle *B* for alerting it of any imminent threat. Also, if vehicle *A* at high velocity is approaching a slow-moving queue ahead, it needs to slow down and avoid erratic behavior in order to prevent any potential crash, a timely intimation for which becomes highly indispensable [2, 3]. An illustration of the said scenarios is depicted in Fig. 13.1.

This kind of opportunistic communication occurs in various diverse forms, and vehicles generally communicate with other vehicles within their immediate vicinity via vehicle-to-vehicle (V2V) communication, with their supporting infrastructure and/or network over vehicle-to-infrastructure/-network (V2I/V2N) communication, with its roadside units via vehicle-to-roadside (V2R) communication, and with vulnerable pedestrians through vehicle-to-pedestrian (V2P) communication—hence strengthening the foundations of a futuristic notion of vehicle-to-everything (V2X) communication [4], as highlighted in Fig. 13.2. Such sort of V2X communication is indispensable for forming a highly robust and scalable networking architecture for

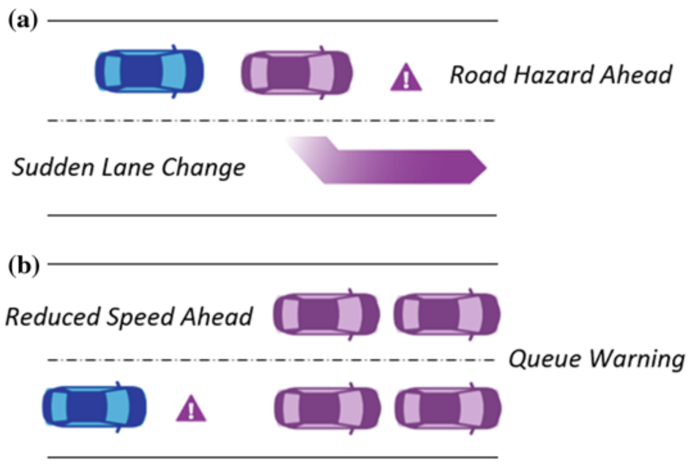


Fig. 13.1 Illustration of vehicular cooperative messages. (**a**) Sudden lane change and (**b**) Queue warning

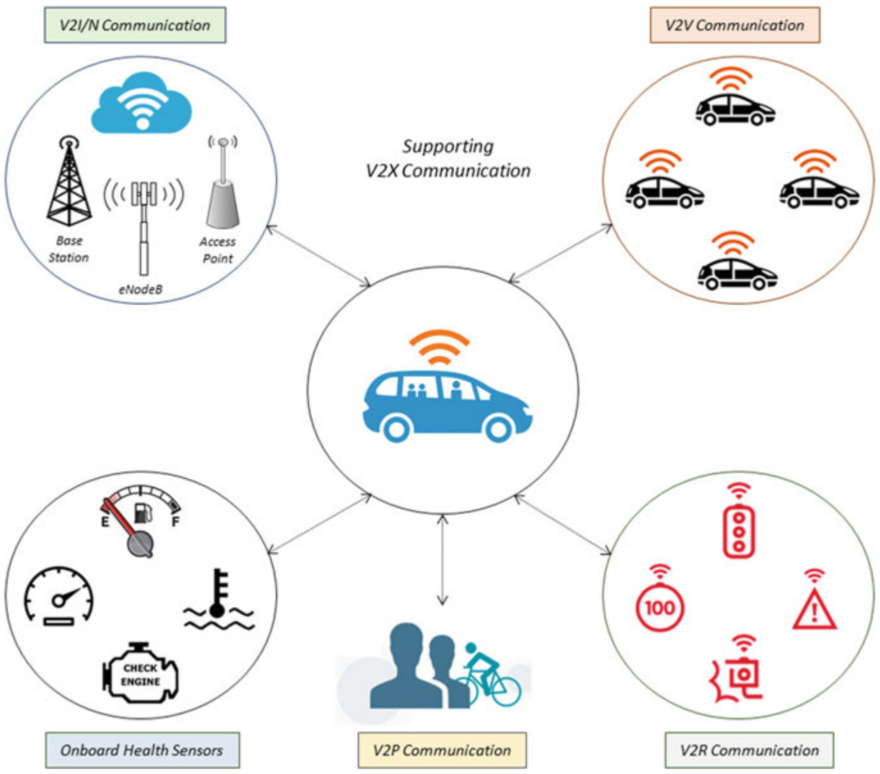


Fig. 13.2 Vehicle-to-everything (V2X) communication

next-generation ITS platforms so as to ensure an extended communication range, extremely higher data rates, and low latency critical for successful implementation of numerous safety-critical applications and services in connected vehicles.

Also, in case of non-safety (i.e., infotainment applications), vehicular users intend to acquire requisite contents with ultra-low latency and higher bandwidths. In case, when a user's requested content is not served owing to one reason or another, an *intelligent recommendation system* needs to be in place in order to recommend the alternate content depending on the local content popularity at a given time and location. However, it should be noted that vehicles are highly dynamic in nature, and accordingly, the content popularity in their neighborhood also changes at the same scale, and which if not properly addressed could lead to serious deterioration in a user's quality-of-experience (QoE). In order to overcome this bottleneck, the emerging and promising notion of edge and/or fog computing has, of lately, been coined in the context of vehicular networks [5, 6], whose primary purpose is to bring intelligence nearer to the edge so as to not only make intelligent decisions on run-time basis but to also serve vehicular requests within a tolerable delay threshold. The Third Generation Partnership Project (3GPPP) recommends that the maximum threshold for safety-critical vehicular applications should be within a range of 20–100 ms [7] as illustrated in Table 13.1.

Coupled with this are the massive amounts of data generated and consumed by (both) the vehicles and their respective users. Vehicles on average possess approximately 100 sensors onboard and are anticipated to host around 200 by the year 2020 [8]. According to an estimate by Intel Corporation, autonomous vehicles are expected to generate and consume 40 TB of data for every eight hours of driving and the primary reason for this appetite is onboard sensors with automotive cameras and radars alone generating 20–40 Mbps and 100 kbps of data, respectively [9]. For instance, road maps in the case of autonomous vehicles would not be downloaded as *one-time download*; rather, they have to be continuously updated in real time due to a dynamic and highly distributed nature of vehicular networks and their associated details should be extremely detailed, i.e., down to the nearest inch, as they would be used for efficient traffic management and mitigating of road hazards. This is further coupled with a second type of data stream, i.e., data generated by humans (vehicular users and roadside pedestrians). As per an estimate, vehicular users on average consume around 650 Mbps per day of videos, chat, and Internet usage and this is expected to escalate to up to 1.5 Gbps per day by 2020. The third stream comes from the crowdsourced traffic data arising from certain specialized platforms, i.e., Google's Waze [10].

This flood of data needs to be managed and appropriately analyzed. Passing it via one '*single*' network would not only be impractical, but also quite impossible due to numerous limitations inherent in existing wireless networking technologies (i.e., in terms of their geographical coverages, data rates, and bandwidth) and keeping in view the ever-increasing number of vehicles (i.e., network nodes) on the roads. Therefore, a heterogeneous networking architecture is highly indispensable, wherein an amalgamation of numerous wireless technologies is considered as an optimal solution, as this helps to offset the disadvantages of one another. However, heterogeneity itself is quite a complex situation to tackle as it leads to network fragmentation and inefficiency in network resources utilization. Moreover, it is trivial to pre-process and accordingly filter appropriate data at the physical plane (i.e., data or network infrastructure plane), as a significant chunk of onboard data is only for reporting the health status at localized levels and thus need not to be transmitted to the backhaul. Furthermore, it is important to cache optimal data as decisions are generally taken on instantaneous set of data and not the historical one, the latter being used for prediction purposes. Therefore, this chapter also serves as a viewpoint for addressing the challenging issue of massive big data in next-generation heterogeneously connected vehicles. Our contribution, in this chapter, primarily lies in proposing a heterogeneous vehicular networking architecture by leveraging the paradigms of software-defined networking (SDN) and edge-based caching to guarantee an intelligent and efficacious network resource management of a distributed and densely populated vehicular networking environment.

Table 13.1 Example link layer parameters for V2X services

Scenarios	Effective range (m)	Absolute speed of a UE (km/h)	Relative speed between two UEs (km/h)	Maximum tolerable latency (ms)	Minimum radio layer message reception reliability (%)	Cumulative transmission reliability (%)
Major roads/ suburban	200	50	100	100	90	99
Motorway/freeway	320	160	280	100	80	96
Autobahn	320	280	280	100	80	96
Non-Line-of-Sight/ urban	150	50	100	100	90	99
Urban intersections	50	50	100	100	95	–
Shopping area/ campus	50	30	30	100	90	99
Imminent crashes	20	80	160	20	95	–

Referring to Table 13.1, in relation to V2X services, it should be noted that there are various parameters involved in these scenarios, viz., (1) physical, e.g., relative speed and effective range; (2) incidental, e.g., line-of-sight occlusion; and (3) service-related, e.g., minimum application layer message reception reliability and maximum tolerable latency. These parameters could be derived from the principle that there should be more than 95% probability that a vehicle has received at least one V2X message from two consecutive V2X messages notifying it of any impending incident.

The remainder of this chapter is organized as follows. Section 2 illustrates an overview of the vehicular safety applications and its developments over the years. An outline of envisaged SDN-based heterogeneous vehicular networking architecture duly assisted with edge-based caching is depicted in Sect. 3, followed by a discussion on open challenges and their probable solutions in Sect. 4. Finally, conclusions are drawn in Sect. 5.

13.2 Overview of Vehicular Safety Applications and Developments

Since the early days of automobiles, the safety of drivers and passengers has been regarded as one of the most crucial aspects of road transport. Therefore, the prominent invention of seatbelts as a safety measure happened as early as 1950s [11] and is still being enforced in the interest of road safety across the globe. Seatbelts represent a passive safety measure that can minimize the impact of traffic accidents on the passengers. Though passive safety measures[1] have contributed to a significant extent in minimizing the road injuries and fatalities, significant effort is being made to prevent the road accidents in the first place. Accordingly, numerous efforts have been made by a number of governments to enhance road traffic safety by either enforcing the speed limits or by building (more) safer roads coupled with diverse education campaigns. There remains much room for improvement [12]. Meanwhile, active safety measures, i.e., anti-locking braking system, adaptive cruise control, and traction control systems, have been installed in vehicles to reduce the probability of accidents per travelled kilometers.

Active road safety measures are intended to curtail the risk of potential vehicular crashes, with low latency and reliability as critical requirements. Safety messages are classified into two main classes, i.e., periodic and event-driven, and the European Telecommunications Standards Institute (ETSI) in its documentation refers to them as Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs), respectively [13]. Literature also refers to both of them, collectively, as Basic Safety Messages [14].

CAMs are also known as *heartbeat messages* or *beacons*. These are typically short messages which are periodically broadcasted by a vehicle for interchanging its state information (basic status, presence, position, and kinematics) with other vehicles or pedestrians in its vicinity. This information is critical to the neighboring vehicles, in order to let them make more informed decisions before initiating any particular action. Furthermore, they are broadcasted to the roadside infrastructure for data analysis and gathering traffic flow statistics [15]. Contrarily, DENMs are event-driven warning notifications broadcasted to alert drivers on the roads concerning any hazardous event. Once an event is detected, the relevant ITS station would broadcast the respective DENMs to other ITS stations situated in a particular geographical region and which are concerned by the events. DENM broadcasting persists as long as the particular event remains prevalent [16]. Both CAMs and DENMs are broadcasted over a certain geographical area, i.e., in the awareness range (i.e., the immediate vicinity) for CAMs and relevance region (potentially affected by the hazardous event) in case of DENMs. The author's interpretation of the characteristics of both CAMs and DENMs along with their pertinent examples is depicted in Table 13.2 for illustration purposes.

The automotive manufacturers have also been collaborating for many years to enhance vehicular safety by developing and subsequently implementing state-of-the-art countermeasures for collision avoidance [17, 18]. One such consortium between Ford Motors and General Motors Corporation was Crash Avoidance Metrics Partnership (CAMP) which was initiated in 1995 and addressed

[1]Other passive safety measures include provision of airbags and enhancements in physical structure of cars.

considerable technical challenges pertinent to both V2V and V2I communications. Several projects have been undertaken under the CAMP with one of the most prominent being Vehicle Safety Communications—Applications (VSC-A).[2] The primary intent of VSC-A was to determine whether DSRC along with vehicle positioning could improve upon the autonomous vehicle-based safety systems or could enable new communications-based safety applications [19]. In 2002, Car-to-Car Communication Consortium (C2C-CC) was conceived with the primary aim of developing a European standard for enhancing traffic efficiency and road safety by utilizing Cooperative Intelligence Transportation Systems [20]. Safe Road Trains for the Environment (SARTE) was a European Commission co-funded project initiated in 2009 and addressed safety, environmental, and traffic congestion issues by investigating the feasibility of vehicle platoons on traditional public highways with full interaction with the other vehicles [21]. Likewise, numerous road transport regulatory bodies have been involved in enhancing the ITS. In the USA, the Department of Transportation National Highway Traffic Safety Administration (DoT-NHTSA) in 2014 suggested a V2V technological mandate for developing consensus standards for a coordinated national deployment of V2V technology in all new lightweight vehicles. This mandate was a result of a decade-long partnership with several automotive industries and academic institutions [22].

These projects have undoubtedly played a pivotal role in motivating the V2X-related activities; however, most of them primarily focused on actuators, sensors, or application layer V2X message designs [23]. Therefore, in order to make this technology readily feasible for the market, it is necessary to ensure its ubiquitous deployment over a wide geographical area on a cost-effective basis. This is where the 3GPP comes into play because of its proposals to guarantee low latency and highly reliable LTE-based V2X communication. Since its initial standardization in 2008, LTE has comprehensively evolved over the years not only in terms of uplink or downlink enhancements for one-to-one communication, but also for one-to-many communications (single-cell point-to-multipoint transmission—SCPTM and multimedia broadcast multicast services—MBMS) and for the sidelink-based device-to-device (D2D) communication [24–26].

[2]VSC-A consortium member includes Ford Motors, General Motors Corporation, Mercedes-Benz Research & Development North America, Inc., Honda R&D Americas, Inc., and Toyota Motor Engineering & Manufacturing North America, Inc.

Table 13.2 Characteristics of vehicular safety messages

Message classification	Salient features	V2X Use cases/ scenarios	Supporting examples
Cooperative awareness messages (CAMs)	Periodic state messages *(frequency: 1–10* Hz*, packet length: 800 bytes; maximum latency: 100* ms*)* Message contents *(position, dynamics, attributes)*	Forward collision warning, Blind intersection warning, Emergency vehicle warning, Notification of speed limits, Lane change assistance, Pre-crash sensing, Vulnerable road user safety, Slow vehicle intimation, and Overtaking vehicle warning	Vehicles traveling on the road(s) broadcast periodic state messages at certain appropriate intervals to the nearby vehicles or to vulnerable pedestrians for mitigating any risks of collision. Similarly, in case of emergency vehicle warning services, the location and dynamic status of the emergency vehicle (ambulance) is passed onto the nearby vehicles to assist in the safety operation by allowing the ambulance path to set free
Decentralized environmental notification messages (DENMs)	Event-triggered Messages *(packet length: typically shorter than CAMs; maximum latency: 100* ms*)* Message contents *(road hazard warnings)*	Abnormal status warning, Stationary vehicle warning, Wrong way drive warning, Roadwork warning, Signal violation warning, Traffic condition warning, Hazardous location alert, and Precipitation, road adhesion	DENMs are used in order to alert the drivers or pedestrians of any hazardous events on the roads. For example, once a vehicle is immobilized either due to a breakdown or an accident, DENMs are signaled in order to alert approaching vehicles of risk associated to this hazardous situation. Similarly, once a vehicle's safety function (i.e., steering or braking) gets out of order, an 'abnormal vehicle state warning' is broadcasted

Referring to Table 13.2, it should be noted that classification of use cases/ scenarios have been adapted from the European Telecommunications Standards Institute (ETSI) Technical Report No: ETSI TR 102 638 V1.1.1.

The 3GPP has already started the standardization process for LTE-based V2X services and conducted a feasibility study for identifying the D2D proximity-based services that could be provided by the user equipments (UEs) when in proximity to one another [27]. A technical report illustrating the probable V2X use cases along with their potential requirements, and example link layer parameters for V2X services under diverse scenarios (Table 13.1), has also been released [7]. Subsequent to outcome of this technical report, a feasibility study to evaluate new functionalities needed to operate LTE-based V2X services identified in [7] has been conducted. The study further included the identification and evaluation of the enhancements to the LTE physical layer, protocols, and interfaces [28]. Furthermore, a technical study for identifying and evaluating the potential architecture enhancements needed to operate the LTE-based V2X services has been recently launched [29]. The 3GPP has also initiated studies regarding the key early verticals and market segments to be addressed by the next-generation wireless communication technology (i.e., fifth generation). The use cases under initial consideration apply to critical communications, enhanced mobile broadband, network operation, massive machine type communications, and autonomous driving [30].

However, due to high mobility and frequent topological changes in vehicular networks, it is almost impossible for a single wireless technology to cater for the QoS requirements of the diverse ITS services. IEEE 802.11p (DSRC) has long been regarded as a *de facto* standard for supporting ITS applications in vehicular networks; offering mature technology, low cost, ease of deployment, and the ability to natively support V2V communication among its key benefits [31]. Nonetheless, DSRC technology can offer only intermittent and short-lived V2I connectivity owing to its limited radio coverage and unavailability of a pervasive roadside communication infrastructure. These concerns have lately shifted the interest toward LTE as being one of the potential wireless broadband access technology to support vehicular communication as it guarantees broad radio coverage, enhanced data and penetration rates, and high-speed terminal support in comparison with the DSRC. Yet, LTE is not capable of providing gigabyte-per-second data rates for exchanging of raw sensor data information among the vehicles and the use of IEEE 802.11ad (mmWave) is currently the most viable option for accessing extremely high bandwidth communication channels [32]. The mmWave V2X communication systems would help to achieve the bandwidth objectives of next-generation connected vehicles; however, issues such as the mmWave beam alignment overhead, mmWave vehicular channel modeling, and insufficient penetration rates remain unsolved and limit the technology's ability to achieve its full potential. The ideal approach though is to ensure a synergy of several radio access technologies in order to provide an appropriate heterogeneous vehicular networking platform to meet the challenging communication requirements of diverse ITS applications and services. In a heterogeneous vehicular networking environment, numerous radio access technologies serve as multi-access points to the Internet for the mobile users. The main desire is that vehicles should always remain connected with the optimal network and thus efficacious vertical handovers are needed to achieve seamless ubiquitous transmission.

13.3 Toward Software-defined Heterogeneous Vehicular Networking Architecture Assisted with Edge Computing

The emerging yet promising paradigm of SDN was originally conceived for the orchestration of data-centric networks [33]. SDN de-couples the data plane from the control plane, and network intelligence is passed to a logically centralized controller for intelligently automating and orchestrating the network resources. Unlike conventional SDN networks that are governed via a single point of logical control, vehicular networks are highly distributive in nature owing to a highly dynamic nature of vehicles and logically controlling it from a single point is quite inefficient especially in case of dense environments, wherein the network management overhead could exceed a tolerable threshold. This would subsequently transform the '*single point of network control*' into a '*single point of network failure*'. Thus, a localized logical control in addition to the centralized control is quite crucial for ensuring the stringent performance requirements of (both) safety-critical and non-safety vehicular networking applications.

It is therefore indispensable to design a layered-based networking architecture, which should facilitate a localized view in addition to a globalized view. This has thus strengthened the notion of edge and fog-based computing in ITS. For the sake of reader's clarification, it is also pertinent to mention that several research studies have used the latest buzzwords of edge and fog computing as being quite controversial in the context of vehicular networks. With the advent of IoT, both fog and edge computing are taking center stages as strategic concepts for tackling a plethora of vehicular data to be analyzed and acted upon in real time. They are particularly used for migrating data center applications nearer to the edge and also for data offloading purposes in critical networking scenarios, i.e., in case of a network broadcast storm in dense vehicular environments [34]. To the author's best understanding, both fog and edge computing in traditional networks are the name of same paradigm but coined by two different proponents, i.e., CISCO being the one that coined the term of fog computing and Nokia being the proponent of mobile edge computing and now accelerating the same with IBM as its strategic partner [35].

There are a number of perspectives to fog and edge computing. However, the key difference in their architecture depends on where the intelligence is precisely placed. In terms of fog computing, intelligence is pushed down to the local area network level of a networking architecture or IoT gateway, while edge computing pushes the intelligence, processing power, and communication capabilities directly into the devices or commonly referred to as *'things'*. In this context, autonomous and/or connected vehicles could be referred to as *edge entities* with each of them anticipated to be equipped with hundreds of onboard sensors generating critically important information for diverse vehicular networking applications. This subsequently constitutes big data and sending such sort of flood of data (in case of large number of vehicles on an expressway or dense road intersections) to a centralized cloud is not only insecure but rather unnecessary and quite impractical. Thus, it is important to sense, think, and act in a real time in order to ensure ultra-low latency for guaranteeing the QoS of diverse vehicular safety and non-safety applications, and QoE of vehicular users. For example, in the event of a vulnerable pedestrian unexpectedly coming in front of an autonomous vehicle, data analytics and decision making should transpire at the local level as such sort of scenario requires ultra-low latency and subsequent actuation. However, this would prove quite catastrophic in case the data analysis and decision making transpire in the centralized cloud. Similarly, in case of a company's truck fleet traversing on an expressway in the form of a platoon[3], it should aggregate and send only their critical health data to the localized or centralized cloud instead of each truck communicating its state information directly, the latter being not only cost inefficient but also yield significant management overhead both at the localized cloud and backhaul. Therefore, intelligent networking schemes, wherein an entire vehicular platoon only forwards its key aggregate details to the cloud as a single edge entity would enable to maintain optimal network performance at an economical cost and with higher efficiency.

SDN typically comprises three planes—*data plane, control plane, and applications plane.* The data plane (network infrastructure plane) comprises of vehicles, vehicular users, and base stations (BSs) and access points (APs) of numerous wireless networking technologies, i.e., 4G LTE, Wi-Fi, WiMAX, millimeter wave (mmWave), and may accommodate recently proposed terahertz communication once it comes into play in the near future [36]. This also compliments the vision set out by the 5G PPP group [37] which characterizes futuristic 5G systems to support a heterogeneous set of existing and new wireless networking technologies in order to increase network availability and reliability. V2X communication also transpires at this plane. The control plane constitutes an SDN controller, which is responsible for maintaining the status(es) of all the underlying networking entities (i.e., abstracted in the form of SDN switches) via a *status manager*. Once an event transpires, the SDN controller would issue a status update request. *Trajectory prediction* module plays a critical role as it tackles two critical challenges—reachability and mobility. In order to collect the status of the vehicles, it is critical to ensure that they are reachable. Also, even if the vehicles are reachable, tracking their positions in real time is also an uphill task and could lead to significant management overhead. Also, the *frequency manager* assists to determine and maintain the frequency of a requested vehicular

[3]Vehicle platooning aims to significantly minimize the inter-vehicular distances in contrast to inter-vehicular distances recommended for manual driving, by partially or fully automating the driving tasks, and with the main intent of effectively utilizing the road infrastructure by facilitating more number of vehicles for utilizing any stretch of the road. Hence, the smaller the inter-vehicular distance is, the more the number of vehicles could be packed, which subsequently leads to reduction in the aerodynamic drag thereby enhancing energy efficiency.

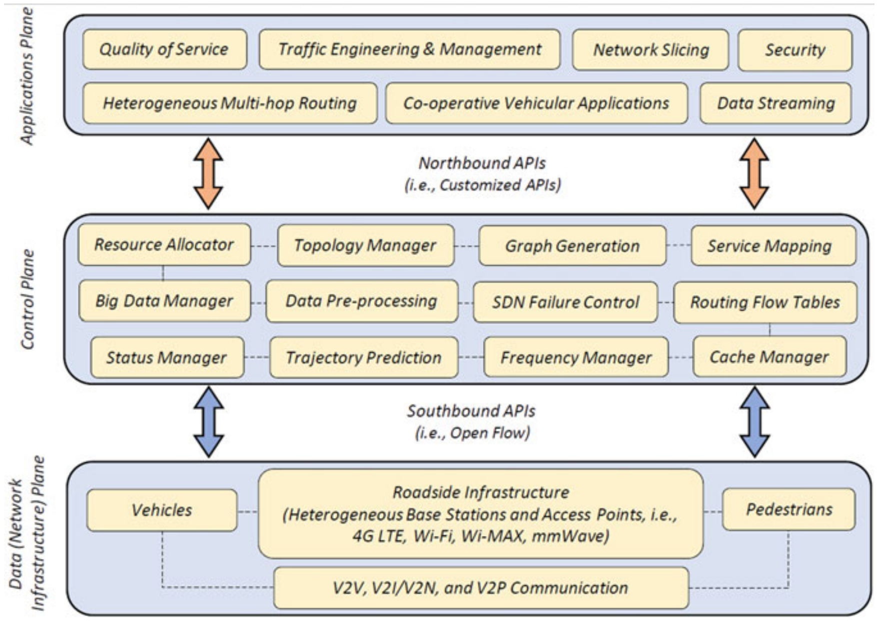


Fig. 13.3 A logical architecture of software-defined heterogeneous vehicular networks

application and service along with the particular time it has been requested for. This facilitates in caching frequently accessed applications and services for a certain geographical coverage area via a *cache manager*, and for predictive caching along the anticipated trajectory of the vehicles.

A logical architecture of software-defined heterogeneous vehicular networks is depicted in Fig. 13.3.

Routing flow tables is another key entity of the control plane as it ensures packet forwarding as per the anticipated trajectory of the vehicles. *SDN failure control* facilitates the vehicular networks to shift down to a decentralized approach in case the centralized controller fails and accordingly helps to reinstate it by carrying out necessary re-medial steps. Furthermore, since vehicles are highly dynamic in their nature and traverse from one wireless networking technology to another, *vertical handovers* become indispensable. Nevertheless, too many handovers could lead to wastage of precious network resources and significant network management overhead in terms of repetitively determining the necessity of handover, opting for the optimal network to handover to, and determining the exact instance for triggering the handover process. Therefore, an equilibrium needs to be maintained in terms of opting for the optimal network and mitigating the wastage of excessive network resources and excessive network management overhead.

Since vehicles act as nodes of a network and collaboratively forms an IoV network, the massive volume of big data generated by them (primarily from onboard sensors) and their respective vehicular users (via handheld devices) becomes part of the network. Passing this all data to a centralized cloud would bring excessive overhead on the backhaul and is quite impractical. Existing radio access technologies are also unable to cater for such a massive big data transfer to the clouds. The ideal approach is thus to pre-process this IoV big data with the help of a *data pre-processing* module which should filter out the right content to be further passed to the network. Also, a large amount of data generated by the vehicles has to be tackled at the local level (i.e., by the vehicular clouds[4] or the edge network) as vehicular data changes at a rapid pace and decisions should be made on the instantaneous data rather than the historical data. Thus, an efficient and intelligent *big data manager* in close coordination with the *edge cache manager* could opt for the critically important content to be cached at the edge and the one to be passed to the global cache via the network backbone. Accordingly, the *resource allocator* allocates and maintains the necessary compute and storage resources for such operations.

Also, the upper layer network protocols and services are generally interested in the network topology and are unable to directly use the status information provided by the status manager. Thus, a *topology manager* could be used for generating an up-to-date network topology by using the SDN-switch status and which may be described in the form of a graph. Once a vehicle is connected to the SDN controller, the controller predicts its future trajectory and accordingly estimates the topology change, based on which flow tables undertake the routing process. The control plane communicates with the data (network infrastructure) plane via a southbound application programming interface (API) and with the applications plane through a northbound API. At present, *Open Flow* [38] is regarded as a standard communication protocol for the southbound APIs, whereas no particular protocol has been to-date mutually agreed for the northbound APIs and hence customized APIs are being used.

Applications plane constitutes diverse vehicular applications and services in order to cater to the requests of vehicular users and normally includes QoS, traffic engineering and management, network slicing, security, heterogeneous multi-hop routing, cooperative vehicular applications, data streaming, etc.

13.4 Open Challenges and Probable Solutions

While SDN brings numerous benefits to traditional VANETs, there still exist a number of barriers to the successful deployment of SDN paradigm in a heterogeneous vehicular networking environment which significantly hinders its intelligent and efficacious decision making capabilities. Existing literature has primarily pointed toward the extension of SDN in vehicular networks by providing a single globalized view of the underlying network. However, unlike traditional networks,

[4]The notion of vehicular clouds is similar to that of vehicular platoons as vehicles with similar objectives (or interests) form a collaborative group and communication is being done via the cloud head, schemes for whose selection is similar to that of cluster head selection in wireless sensor networks.

VANETs are highly distributive in nature and thus demand for extremely low latency and higher bandwidth for a number of safety-critical applications. Some of these challenges and their probable solutions are now described below.

13.4.1 Avoiding a 'Single' Point of Network Failure

Traditional SDN networks are governed via a 'single' point of centralized control in order to ensure programmability, agility, scalability, flexibility, and elasticity. However, unlike traditional networks (i.e., in data centers), VANETs are highly distributive in nature and possesses high mobility. Hence, controlling vehicular nodes from a single point of logical control could lead to a potential network failure especially in case if the centralized controller fails unexpectedly. Also, placing all the network load on one centralized controller is not feasible since it leads to an excessive amount of network management overhead which could choke the entire network too. Thus, an upper threshold of a SDN controller needs to be chalked out as to what it can oversee, a concept that has been still neglected in the literature.

Hence, a locally distributed network intelligence in addition to a logically centralized control is imperative for supporting safety-critical applications demanding low latency. Edge caching is perhaps one of the possible solutions for caching the frequently requested applications and services, and content popularity plays a critical role in caching desirable contents. Nevertheless, owing to the high mobility of VANETs, the content popularity also changes at a drastic pace. A typical alternate to overcome this bottleneck is to deploy an intelligent and efficient *recommender system* which recommends alternate contents to the vehicular users, and especially for the time, the originally desired content has been requested from the remote end servers. Also, while it might take some time to fetch the desired contents, an intelligent recommender system could help to increase the overall QoE.

Even though the edge caches supported by edge/localized controllers can help to temporarily hold the network in place (i.e., once the centralized controller fails), nevertheless, if the edge controller itself fails, then an automatic fall back to traditional decentralized VANET approaches supported by intimation to vehicular users with sufficient reaction time might be an optimal solution by the time the complete system (i.e., both edge and centralized controllers) gets restored.

13.4.2 Network Security

Security is one of the critical concerns for SDN-based heterogeneous vehicular networks. Since SDN is vertically split up into three major functional layers, malicious attacks could transpire on all of these three layers and could be classified as: (1) data (network infrastructure) plane attack, (2) control plane attack, and (3) applications plane attack, as illustrated in Fig. 13.4. On the data (network

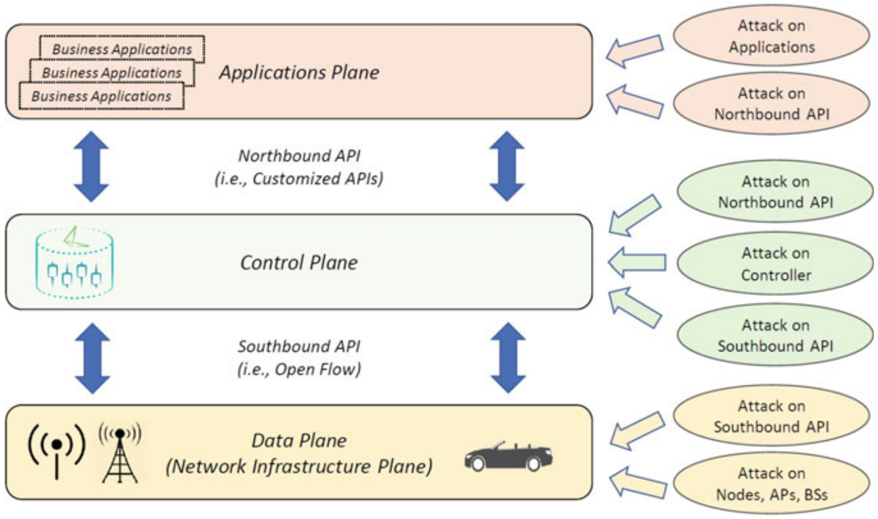


Fig. 13.4 Potential attacks on an SDN-based heterogeneous vehicular networks

infrastructure) plane, either any network node (vehicle or AP/BS) or the southbound API could be maliciously attacked hence disrupting the entire communication infrastructure. In conventional networks, this would lead to privacy breaks, identify thefts, alterations, and disruption of network traffic. However, its impact in VANETs is much more severe as it could result in human fatalities. Therefore, a secure network with features of data authentication, data integrity, application/ services access control, anti-jamming, data confidentiality, and data non-repudiation is indispensable for realizing the potential of SDN-based vehicular networking in its real essence. As the controller (both local and central/global) could be seen as the single point of network failure, it could either be directly attacked or via south-bound and northbound APIs. Lastly, the applications plane could be attacked either via the northbound customized API or by targeting some specific applications. Some specialized attacks, e.g., Distributed Denial of Service (DDoS), generally takes the advantages of botnets and high-speed wireless access technologies, and the size of such attacks has grown dramatically over the past years and traditional data analysis techniques have not been quite effective in defeating these attacks [39].

13.4.3 Big Data

Big data has recently become one of the hottest topics in both academia and industry primarily due to its capacity and complexity. It has been traditionally characterized in terms of 5Vs, i.e., size of data (volume), diverse range of data types and sources (variety), speed of data (velocity), usefulness of data (valueness), and quality of data (veracity). Trustworthiness of data is also a fundamental characteristic and thus needs to be ensured [40]. With the advent of IoT, modern-day connected vehicles are getting seamlessly connected with the Internet and the notion of *Internet of Vehicles* has already set in. However, with hundreds of thousands of vehicles and vehicular users generating, consuming, and transmitting terabytes of data per second to both localized and centralized clouds not only leads to excessive network management overhead but also considerably decreases the effectiveness of network by wasting its precious computing and storage resources.

In order to address the same, intelligence needs to be employed which could address the challenges posed by massive big data traversing in a vehicular networking environment.

13.4.4 Heterogeneous Radio Access Technologies

It is likely that Internet access would soon become one of the common features of connected and autonomous vehicles. At present, cellular networks are considered as a reliable and ubiquitous form of providing a high-speed Internet access. However, merely relying on cellular networks would not address the challenging performance requirements of numerous vehicular applications. This has called for heterogeneity of diverse wireless networking technologies and futuristic smart cities are expected to be deployed with densely populated overlapping heterogeneous radio access technologies. Hence, in order to ensure seamless ubiquitous communication and to meet the challenging communication requirements, it is necessary to ensure that vertical handovers (handovers between disparate wireless networking technologies) transpire at the right time and to a best *optimal* network. However, too many unnecessary handovers and the commonly transpired *ping-pong effect* could lead to performance degradation and should be avoided.

For any vertical handover to transpire successfully, three major processes need to be taken into consideration, viz., handover necessity estimation, handover target selection, and handover triggering condition estimation. Handover necessity estimation determines if a handover from the currently associated radio access technology is indispensable to another radio access technology; handover target selection opts for the optimal network out of the available radio access technologies at a given time and location, and handover triggering condition estimation depicts an exact instance to trigger handover to a selected network.

SDN can also facilitate in ensuring a seamless handover as the global controller has access to the underlying network topology along with vehicles anticipated trajectories, and the optimal network along with the exact time of handover could be accordingly instructed by the controller. Flow tables can play a critical role in realizing this objective.

13.5 Conclusion

Convergence of VANETs and heterogeneous wireless networking technologies together with SDN has led to the emergence of a promising computing paradigm, referred to as SDN-based heterogeneous vehicular networks. This has recently received a considerable amount of attention from academic and industrial communities. The key features of SDN can help overcome the limitations posed by the traditional VANET architectures. This chapter accordingly investigated, highlighted, and reported the origins, fundamentals, and developments of VANETs and scientifically brought forward the need for SDN integration into a vehicular networking environment. A SDN-based heterogeneous vehicular networking architecture assisted with edge-based caching is proposed along with a detailed discussion on its salient characteristics. Furthermore, several challenges that should be addressed to promote SDN-based heterogeneous vehicular networks have also been discussed.

Acknowledgements The corresponding author would like to acknowledge the generous support of Ministry of Higher Education, Government of Malaysia, for partly supporting the said research work via its prestigious Malaysian International Scholarship Grant, KPT. 600-4/1/12JLID 2(8).

References

1. Zheng K, Zheng Q, Chatzimisios P, Xiang W, Zhou Y (2015) Heterogeneous vehicular networking: a survey on architecture challenges, and solutions. IEEE Commun Surv Tutorials 17(4):2377–2396
2. Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W (2017) A survey on mobile edge networks: convergence of computing caching and communications. IEEE Access 5:6757–6779. https://doi.org/10.1109/access.2017.2685434
3. Dey KC, Rayamajhi A, Chowdhury M, Bhavsar P, Martin J (2016) Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network—performance evaluation. Transp Res Part C: Emerg Technol 68:168–184. https://doi.org/10.1016/j.trc.2016.03.008
4. Camacho F, Cárdenas C, Muñoz D (2018) Emerging technologies and research challenges for intelligent transportation systems: 5G, HetNets, and, Int J Interact Des Manuf (IJIDeM) 12(1):327–335. https://doi.org/10.1007/s12008-017-0391-2
5. Azizian M, Cherkaoui S, Hafid AS (2017) Vehicle software updates distribution with and cloud computing. IEEE Commun Mag 55(8):74–79. https://doi.org/10.1109/mcom.2017.1601161
6. Fontes RDR, Campolo C, Rothenberg CE, Molinaro A (2017) From theory to experimental evaluation: resource management in software-defined vehicular networks. IEEE Access 5:3069–3076. https://doi.org/10.1109/access.2017.2671030
7. GPP (2015) Third generation partnership project; technical specification group services and system aspects; study on LTE support for vehicle to everything (V2X) services, 3GPP TR 22.885 V14.0.0 (Technical Report)
8. Choi J, Va V, Gonzalez-Prelcic N, Daniels R, Bhat CR, Heath RW (2016) Millimeter-wave vehicular communication to support massive automotive sensing. IEEE Commun Mag 54(12):160–167

9. Nelson P (2016) just one autonomous car will use 4,000 GB of Data/Day. network world. Available Online at: https://www.networkworld.com/article/3147892/internet/one-autonomous-car-will-use-4000gb-of-dataday.html. Accessed 5 Mar 2018
10. Zheng X et al (2016) Big data for social transportation. IEEE Trans Intell Transp Syst 17(3): 620–630. https://doi.org/10.1109/tits.2015.2480157
11. Joerer S (2016) Improving intersection safety with inter-vehicular communication, Ph.D. Thesis, The University of Innsbruck, Innsbruck, Austria
12. Zheng K, Hou L, Zheng Q, Lu N, Lei L (2016) Soft-defined heterogeneous vehicular network: architecture and challenges. IEEE Netw 30(4):72–80. https://doi.org/10.1109/mnet.2016.7513867
13. ETSI (2010) Intelligent transport system (ITS); communication architecture, ETSI EN 302 665 V1.1.1 (European Standard Telecommunication Series)
14. USDoT, 2013, Concept of Operations for Road Weather Connected Vehicle Applications, U.S. Department of Transportation's Technical Report (FHWA-JPO-13-047)
15. Zheng K, Zheng Q, Yang H, Zhao L, Chatzimisios P (2015) Reliable and efficient autonomous driving: the need for heterogeneous vehicular networks. IEEE Commun Mag 53(12):72–79
16. Araniti G, Campolo C, Condoluci M, Iera A, Molinaro A (2013) LTE for vehicular networking: a survey. IEEE Commun Mag 51(5):148–157. https://doi.org/10.1109/mcom.2013.6515060
17. Seo H, Lee KD, Yasukawa S, Peng Y, Sartori P (2016) LTE evolution for vehicle-to-everything services. IEEE Commun Mag 54(6):22–28. https://doi.org/10.1109/mcom.2016.7497762
18. Bergenhem C, Huang Q, Benmimoun A, Robinson T (2010) Challenges of platooning on public motorways, 17th ITS world congress. Busan, Korea, pp 1–12
19. NHTSA (2011) Vehicles safety communications—applications (VSC-A), Final Report, U.S. Department of transportation national highway traffic safety administration's report (DOT HS 811 492A)
20. C2C-CC (2015) Co-operative road traffic—foresight, safety, and comfort, The CAR 2 CAR communication consortium status report
21. Davila A, Aramburu E, Freixas A (2013) Making the best out of aerodynamics: platoons, Society of automotive engineering (SAE International) Technical Paper 2013- 01-0767
22. NHTSA (2014) Advance notice of proposed rulemaking (ANPRM) notice of availability of technical report. Fed Reg 79(161):49270–49278
23. Xu W et al (2018) In big data era. IEEE/CAA J Automatica Sinica 5(1):19–35. https://doi.org/10.1109/jas.2017.7510736
24. Wu Y, Guo W, Yuan H, Li L, Wang S, Chu X, Zhang J (2016) Device-to-device meets lte-unlicensed. IEEE Commun Mag 54(5):154–159. https://doi.org/10.1109/mcom.2016.7470950
25. Sun S-H, Hu J-L, Peng Y, Pan X-M, Zhao L, Fang J-A (2016) Support for vehicle-to-everything services based on LTE. IEEE Wirel Commun 23(3):4–8. https://doi.org/10.1109/mwc.2016.7498068
26. Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surv Tutorials 19(3):1628–1656. https://doi.org/10.1109/comst.2017.2682318
27. 3GPP (2014) Third generation partnership project; technical specification group radio access network; study on LTE device to device proximity services; radio aspects, 3GPP TR 36.843 V12.0.1 (Technical Report)
28. GPP (2016) Third generation partnership project technical specification group radio access network; Study on the LTE-based V2X Services. 3GPP TR 36.885 V14.0.0 (Technical Report)
29. GPP (2016) Third generation partnership project; technical specification group services and system aspects; study on architecture enhancements for LTE support of V2X services. 3GPP TR 23.785 V14.0.0 (Technical Report)

30. GPP (2016) Third generation partnership project; technical specification group services and system aspects; feasibility study on new services and markets technology enablers. 3GPP TR 22.891 V14.2.0 (Technical Report)
31. Mir ZH, Filali F (2014) LTE and IEEE 802.11p for vehicular networking: a performance evaluation. EURASIP J Wireless Commun Networking 89:1–15. https://doi.org/10.1186/1687-1499-2014-89
32. Cui X, Gulliver TA, Li J, Zhang H (2016) Vehicle positioning using 5G millimeter-wave systems. IEEE Access 4:6964–6973. https://doi.org/10.1109/access.2016.2615425
33. Robinson M, Milosavljevic M, Kourtessis P, Stafford GP, Burrell MJ, Senior JM (2016) Software defined networking for heterogeneous access network*s*. In: 18th international conference on transparent optical networks (ICTON), Trento, Italy, pp. 1–4, https://doi.org/10.1109/icton.2016.7550412
34. Aujla GS, Chaudhary R, Kumar N, Rodrigues JJPC, Vinel A (2017) Data offloading in 5g-enabled software-defined vehicular networks: a stackelberg-game-based approach. IEEE Commun Mag 55(8):100–108. https://doi.org/10.1109/mcom.2017.1601224
35. Secinti G, Canberk B, Duong TQ, Shu L (2017) Software defined architecture for: a testbed implementation with wireless access management. IEEE Commun Mag 55(7):135–141. https://doi.org/10.1109/mcom.2017.1601186
36. Mumtaz S, Jornet JM, Aulin J, Gerstacker WH, Dong X, Ai B (2017) Terahertz communication for vehicular networks. IEEE Trans Veh Technol 66(7):5617–5625. https://doi.org/10.1109/tvt.2017.2712878
37. G PPP (2015) 5G vision (the next generation of communication networks and services), The 5G infrastructure public private partnership [Available Online at: https://5g-ppp.eu/wpcontent/uploads/2015/02/5G-Vision-Brochure-v1.pdf]
38. He Z, Cao J, Liu X (2016) SDVN: enabling rapid network innovation for heterogeneous vehicular communication. IEEE Netw 30(4):10–15. https://doi.org/10.1109/mnet.2016.7513858
39. Cui L, Yu FR, Yan Q (2016) When big data meets software-defined networking: for big data and big data for SDN. IEEE Netw 30(1):58–65. https://doi.org/10.1109/mnet.2016.7389832
40. Gai F, Zhang J, Zhu P (2017) Ratee-Based Trust Management System for. In: Ma L, Khreishah A, Zhang Y, Yan M (eds) Wireless algorithms, systems, and applications (WASA). Lect Notes Comput Sci 10251:344–355. Springer, Cham

Index