Dimitrios Georgakopoulos
Liang-Jie Zhang (Eds.)

LNCS 10972

Internet of Things – ICIOT 2018

Third International Conference
Held as Part of the Services Conference Federation, SCF 2018
Seattle, WA, USA, June 25–30, 2018, Proceedings

ICIOT
Internet of Things

Springer

Dimitrios Georgakopoulos · Liang-Jie Zhang (Eds.)

Internet of Things – ICIOT 2018

Third International Conference
Held as Part of the Services Conference Federation, SCF 2018
Seattle, WA, USA, June 25–30, 2018
Proceedings

Springer

Editors

Dimitrios Georgakopoulos
Swinburne University of Technology
Hawthorn, VIC
Australia

Liang-Jie Zhang
Kingdee International Software Group Co., Ltd.
Shenzhen
China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-94369-5 ISBN 978-3-319-94370-1 (eBook)
https://doi.org/10.1007/978-3-319-94370-1

Library of Congress Control Number: 2018947338

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

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

Preface

This volume presents the accepted papers for the 2018 International Conference on Internet of Things (ICIOT 2018), held in Seattle, USA, during June 25–30, 2018. With the rapid advancements of mobile Internet, cloud computing, and big data, device-centric traditional Internet of Things (IoT) is now moving into a new era that is termed the "Internet of Things Services" (IOTS). In this era, sensors and other types of sensing devices, wired and wireless networks, platforms and tools, data processing/visualization/analysis and integration engines, and other components of traditional IoT are interconnected through innovative services to realize the value of connected things, people, and virtual Internet spaces. The way of building new IoT applications is changing. We indeed need creative thinking, long-term visions, and innovative methodologies to respond to such a change. The ICIOT conference is organized to continue promoting research and application innovations around the world.

For this conference, we accepted 13 papers, including 12 full papers and one short paper. In the category of full papers, we accepted nine research track papers and four application and industry track papers. Each was reviewed and selected by at least three independent members of the ICIOT 2018 international Program Committee.

We are pleased to thank the authors whose submissions and participation made this conference possible. We also want to express our thanks to the Organizing Committee and Program Committee members for their dedication in helping to organize the conference and reviewing the submissions. We owe special thanks to the keynote speakers for their impressive speeches.

Finally, we would like to thank Zhonghai Wu, Alexander Lenk, and Mohamed Sellami for their excellent work in organizing this conference.

May 2018

Dimitrios Georgakopoulos
Liang-Jie Zhang

Conference Committees

General Chair

Zhonghai Wu	Peking University, China

Program Chair

Dimitrios Georgakopoulos	Swinburne University, Australia

Application and Industry Track Chair

Alexander Lenk	BMW Group, Germany

Short Paper Track Chair

Mohamed Sellami	Institut Supérieur d'Électronique de Paris, France

Publicity Chair

Raju Vatsavai	North Carolina State University, USA

Services Conference Federation (SCF 2018)

General Chairs

Wu Chou	Essenlix Corporation, USA
Calton Pu	Georgia Tech, USA

Program Chair

Liang-Jie Zhang	Kingdee International Software Group Co., Ltd, China

Finance Chair

Min Luo	Huawei, USA

Panel Chair

Stephan Reiff-Marganiec	University of Leicester, UK

Tutorial Chair

Carlos A Fonseca	IBM T.J. Watson Research Center, USA

Industry Exhibit and International Affairs Chair

Zhixiong Chen	Mercy College, USA

Operations Committee

Huan Chen (Chair)	Kingdee, China
Jing Zeng	Tsinghua University, China
Yishuang Ning	Tsinghua University, China
Sheng He	Tsinghua University, China
Cheng Li	Tsinghua University, China

Steering Committee

Calton Pu	Georgia Tech, USA
Liang-Jie Zhang (Chair)	Kingdee International Software Group Co., Ltd, China

ICIOT 2018 Program Committee

Luca Cagliero	Politecnico di Torino, Italy
Tao Chen	University of Birmingham, Britain
Siobhan Clarke	University of Dublin, Ireland
Georgiana Copil	Austria
Rik Eshuis	Eindhoven University of Technology, The Netherlands
Pedro Furtado	University of Coimbra, Portugal
Chris Gniady	University of Arizona, USA
Alfredo Goldman	University of São Paulo, Brazil
Atishay Jain	Adobe, USA
Nagarajan Kandasamy	Drexel University, USA
Samantha Kumara	Sabaragamuwa University of Sri Lanka, Sri Lanka
Nuno Laranjeiro	University of Coimbra, Portugal
ShaoChun Li	IBM Research (China), China
Yutao Ma	Wuhan University, China
Massimo Mecella	University of Rome, Italy
Suoratik Mukhopadhyay	Louisiana State University, USA
Roberto Natella	Federico II University of Naples, Italy
Roy Oberhauser	Aalen University, Germany
Rui André Correia de Oliveira	CISUC, University of Coimbra, Portugal
Ashish Tanwer	Cisco, USA
Jian Wang	Wuhan University, China
Pengcheng Xiong	NEC Labs, USA

Contents

Research Track – Architecture

A Blockchain-Based Decentralized Security Architecture for IoT

Pelin Angin[1](✉), Melih Burak Mert[1], Okan Mete[1], Azer Ramazanli[1], Kaan Sarica[1], and Bora Gungoren[2]

[1] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
pangin@ceng.metu.edu.tr, {melih.mert,okan.mete, azer.ramazanli,kaan.sarica}@metu.edu.tr
[2] Portakal Teknoloji, Ankara, Turkey
bora.gungoren@gmail.com

Abstract. Advances in the Internet of Things (IoT) computing paradigm have made it a popular solution covering many areas such as smart cities, connected vehicles, smart farming etc. to provide major economic benefits, reduced resource consumption, smarter environments and increased sharing of resources among other advantages. However, the security issues arising from the scale of connectivity and heterogeneity of resources in IoT make it a hot target for attackers, where centralized security solutions fall short. The vulnerabilities in providing proper device authentication and data integrity in IoT networks have been shown to introduce devastating effects. This calls for designing a data security architecture for IoT, which can accurately authenticate devices by anyone in the network in a decentralized manner and prevent unauthorized modification of the stored data. In this paper, we propose a blockchain-based approach for IoT systems that introduces transparency and tamper-resistance into data storage and retrieval in IoT networks. Evaluation of a developed prototype demonstrates that the proposed solution is promising to present a unified framework for IoT data security.

Keywords: Data integrity · Blockchain · IoT security · Verification

1 Introduction

The Internet of Things (IoT), enabling the connectivity of physical and virtual objects to create smart environments, has witnessed exponential growth in the past decade with the advances in networking infrastructures and smart devices. According to a Gartner study, the number of IoT units worldwide is expected to reach around 20 billion by 2020 [7], and the IoT connections even only within the EU is estimated to reach 6 billion by 2020 [3], covering a market of over one trillion euros. Among major applications of IoT are smart homes integrating various sensors for security, elderly care and smart energy consumption, wearables for personal health monitoring, smart manufacturing expected to be prevalent in the European logistic chain and production line, smart energy grids, smart cities utilizing data from various sensors for long term development planning, connected vehicles, smart farming improving the agri-food supply chain, earth/ocean observation systems addressing environmental issues, and surveillance/safety warning systems for emergency response. Wide adoption of IoT is expected to provide immense economic benefits, as it enables crossing over borders between different industrial sectors, creating more efficient processes, reduced consumption and increased sharing of resources. IoT technology will also address societal challenges as in the cases of:

– At-home health monitoring in Netherlands resulting in significant efficiency gains for elderly care efforts [16]
– Auto-adjusting streetlights in Barcelona providing over 30% energy savings [1]
– UK's intelligent transport system reducing travel time by 25% and accidents by 50% [19]

Given the potential of IoT to create smarter, safer, and efficient systems as "the next major economic and societal innovation wave" after Internet's evolution, development of IoT infrastructures and services are a significant task.

Despite the many economic and societal benefits of IoT, the security issues it gives rise to are a concerning aspect of the technology, hampering wider adoption. The crosslinking of various types of objects in IoT enables each object to influence the behavior of others and the exchange of vast amounts of information taking place automatically without the users' awareness leads to security problems including violations of data privacy and integrity, vital for both proper functioning of critical systems and data protection rights of individuals. IoT-enabled devices have already created a large attack surface for hackers to exploit. Among major security incidents involving IoT devices are:

– CIA tools turning Samsung SmartTVs into secret listening devices [8]
– The massive distributed denial of service attack against a major DNS provider (Dyn) launched through security cameras [5]
– Hackers remotely taking control of a Jeep Cherokee [9]
– The Stuxnet virus destroying a fifth of Iran's nuclear centrifuges by spinning out of control [12]

When we consider Industrial Internet of Things (IIoT), involving organizations in the energy, utilities, government, healthcare and finance sectors, the seriousness of the situation becomes more obvious. Vulnerabilities in such systems not only create privacy violations, but also hurt safety and availability, as in the case of disrupting power supply for communities through hacking into energy grids, making authentication and data integrity paramount.

Achieving security in IoT faces new challenges peculiar to the technology in addition to previously known vulnerabilities:

- Many of the current IoT devices lack basic security requirements.
- There is a lack of standardization for IoT standards and protocols, which creates security loopholes.
- There is a lack of clarity regarding who is responsible for security violations [17].
- There are scalability problems associated with the sizes of machine-to-machine (M2M) networks.
- The low computing power/battery life of many devices make them unfit for computation-intensive cryptographic operations.

The challenges facing a secure IoT model partly stem from the centralized architecture of the current IoT ecosystem, which requires devices to be authenticated through trusted third parties (TTP) with high processing power in the cloud. Even if seemingly sufficient for today's networks, a centralized security architecture will be unable to meet the needs of the near future's huge IoT networks, with the central security servers being a single point of failure, therefore hot targets for attackers, and leading to performance bottlenecks and high maintenance costs. In order to seamlessly integrate into existing IoT systems, the security architecture to be developed needs to be energy-efficient, easy to operate across a variety of IoT devices, scalable for huge device networks, provide real-time authentication and data integrity verification, and ensure end-to-end security of disseminated data in line with the protection rights of individuals such that they remain in control.

In this work, we propose an adaptable data security architecture for IoT, addressing the specific requirements pertaining to the nature of IoT devices such as limited processing power, battery life and storage space, and provide a security architecture truly fit for the decentralized nature of IoT applications, addressing the shortcomings of existing state-of-the-art client-server based models. The architecture is based on the application of the blockchain technology to IoT in order to provide secure device authentication and protect/verify the integrity of data collected by sensors and other devices in an IoT network.

The rest of this paper is organized as follows: Sect. 2 provides an overview of data security approaches in IoT. Section 3 introduces the proposed data security architecture for IoT. Section 4 discusses the results of preliminary experiments for the feasibility of the approach. Section 5 concludes the paper.

2 Related Work

Current security architectures for IoT are generally extensions of legacy client-server based models, which provide authentication of devices by certification authorities through strong standards like the X.509 public key infrastructure (PKI) [11], and utilize extensions of TLS such as DTLS [14] for securing network communication, which are both heavy-weight for devices lacking sufficient energy, storage and processing power and rely on a centralized architecture for tasks including management of security keys and data integrity verification [2], resulting in large delays unsuitable for the distributed nature of IoT applications. Following the immense growth of the IoT technology in the past decade, there have been several proposals for IoT security and privacy, among which are:

- A distributed capability-based access control method based on PKI using the Elliptic Curve Digital Signature Algorithm (ECDSA), which falls short of satisfying the real-time requirements of many IoT applications due to large delays [18]
- An IP-sec and TLS based protocol to provide authentication and privacy, which is computationally-intensive [10]
- A privacy management method based on the measurement of the risks of disclosing sensitive data, which strictly relies on the accuracy of the risk estimation methods [20].

One recent technique proving successful in providing decentralized, verifiable security and privacy of data is blockchain. Blockchain is a promising technology for secure IoT, providing a decentralized architecture, anonymity of users, tamperproof record of data transactions and highly trusted data verification/device authentication [4]. There are already proposals for utilizing blockchain as the underlying security model for various IoT systems [15], and companies like IBM and Samsung have started exploring the potential of blockchain for IoT. Although blockchain is seen by many as the next disruptive technology with great promise for IoT, a direct application of blockchain to IoT has major shortcomings to be addressed:

- The ever-growing nature of the public ledger of transactions in blockchain is not suitable for the limited storage space available in many IoT devices.
- The distributed consensus protocol involving all nodes for each transaction takes a long time, not meeting the real-time needs of some IoT applications.
- The cryptographic processing on the devices consumes too much energy.

Our proposed architecture differs from purely blockchain-based architectures in that it takes into consideration the resource limitations of IoT devices in the blockchain network, and imposes a hierarchical blockchain structure for providing data security in IoT.

3 Proposed Approach

In order to address the data security challenges of IoT, in this work we propose a decentralized data security framework for IoT, adaptable to the very nature and context of IoT applications and devices. The framework has the goal of enabling the creation of a secure IoT ecosystem through standardization and interoperability, paving the way for further interdisciplinary research involving fields ranging from autonomous vehicles to smart energy distribution networks, removing the security barrier from wider adoption of IoT-enabled devices and services. The objectives of the framework are as follows:

1. To build a decentralized, anonymity-preserving, non-repudiation capable authentication and data verification framework, able to integrate IoT devices with various capabilities and generating data in various forms.
2. To ensure adaptable operation of the data security framework, such that it adjusts the processes involved in data verification and dissemination based on the capabilities of participating devices and operation context.
3. To ensure optimal performance in terms of energy consumption and end-to-end data communication delay.

We begin this section by describing a typical data generation scenario in IoT along with its data security requirements, provide an overview of blockchain and describe the hierarchical blockchain architecture we propose to provide decentralized data security in IoT networks.

3.1 The Smart Energy Trading Scenario

One specific use case of IoT in the energy sector is the building of distributed energy distribution networks automated with the help of smart energy meters that measure the energy consumption at individual units in the network. Thanks to the availability of renewable energy production tools like rooftop solar panels, houses can now even generate their own energy and perform energy trading with their neighbors without the need for an intermediary. In the presence of no trusted intermediate authority, the tracking of the amount of energy produced/-consumed by each unit will need to rely on a mechanism that does not require *trust* between the network participants. This kind of IoT scenario is prone to the following security issues:

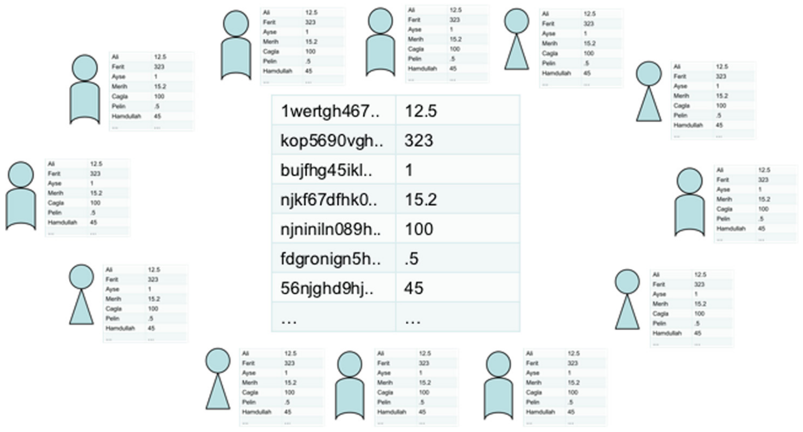
1. Especially in developing countries, fraudulent users could be involved in acts such as modifying the readings of the meters or even altering the behavior of the meters to incur negative usage costs, hence destroying the integrity of the data gathered by the IoT devices.
2. If the data gathered by all units are kept at a single site, that site becomes a single point of failure, and an attack on the site could result in the loss of all measurement data.
3. Malicious devices in the network could spoof other devices to hold them responsible for their own usage.
4. In the case of energy trading, devices buying/selling energy may not meet contractual obligations and it will be hard for the involved parties to track them down in case of using fake identities.

The last two of the above items require strong authentication mechanisms, however approaches based on trusted third parties (TTP) will not scale when the size of these networks is considered, and TTPs are still subject to the single-point-of-failure problem.

3.2 Blockchain

Blockchain, which is seen as the fifth disruptive technology in computing after mainframes, PC, the Internet and social media, is a distributed database (called *ledger*) shared and controlled by a group of networked independent parties as seen in Fig. 1. Within the context of the energy trading scenario, the ledger can be thought of as a database of all trading, production and consumption (as a result of energy consumption and production) transactions ever made by the network participants. For the sake of simplicity, the ledger in Fig. 1 shows the current energy surplus of the network participants in the database.

Blockchain provides an immutable record of data secured by a peer-to-peer (P2P) network of participants, who validate all transactions against the ever-growing database, using the cryptographic hash of each block of data in the chain that links it to its predecessor. The database updates to include new transaction records occur by broadcasting the transaction to the entire network as seen in Fig. 2 (which corresponds to sending a given amount of energy from A to B for our scenario) and running a distributed consensus algorithm typically involving the solution to a cryptographic puzzle (proof of work) by special participants of the network (miners) [13]. The whole network has a consistent copy of the ledger, which provides transaction transparency.



1wertgh467..	12.5
kop5690vgh..	323
bujfhg45ikl..	1
njkf67dfhk0..	15.2
njniniln089h..	100
fdgronign5h..	.5
56njghd9hj..	45
...	...

Ali	12.5
Ferit	323
Ayse	1
Merih	15.2
Cagla	100
Pelin	.5
Hamdullah	45
...	...

Fig. 1. Blockchain network and the ledger

Some important features of blockchain, which make it an important tool for distributed data security are as follows:

– It is a continuously growing list of records.
– It is protected from revision and tampering (i.e. it is immutable).
– It records all transactions that ever occurred.
– It provides non-repudiation (i.e. participants performing transactions cannot deny involvement in transactions they performed).

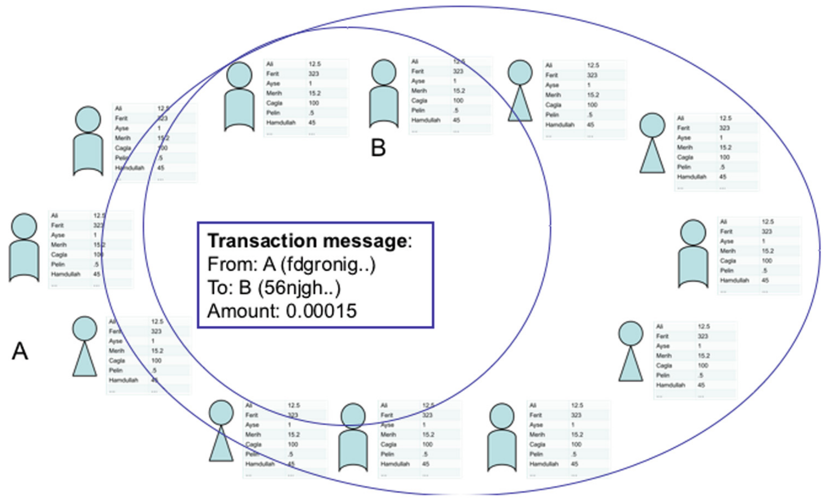


Fig. 2. Transaction in blockchain

– Upon a new transaction, the blockchain is validated across the distributed network before including the transaction as the next block in the chain (i.e. it relies on a distributed consensus mechanism).
– Blocks are made by miners (utilizing special equipment/software) coming to agreement on which transactions fit in each block, and which block will be the next in the chain.

One important feature of blockchain that makes it an effective technique for data security at scale is that it does not rely on *trust* between the participants in the network, and there is no central trusted authority involved. The security of blockchain relies purely on cryptography and distributed consensus in the network. In order to be able to write an alternative transaction history or replace any transaction in the chain, an attacker would have to invest in significant computational resources, which serves as a deterrent for adversaries. The transparency feature (i.e. everyone participating in the network has full visibility of all transactions) is a significant aspect for verifiability of all data transactions by all parties involved.

On the other hand, there are also some limitations of blockchain, which make it difficult to adapt to the problem of IoT security. Among the limitations are the following:

– The 50% + 1 rule: An attacker controlling over 50% of the network will be able to amend transactions, i.e. write an alternative history.
– Prohibitive power consumption: There are different consensus mechanisms (e.g. proof of stake, proof of work, proof of identity etc.) for blockchain, and proof of work, which requires the solving of cryptographic puzzles by the miners in the network, has been the most popular for public blockchains.

The security of this scheme relies on the complexity of the computations performed by the nodes, however this also makes it prohibitive in terms of energy consumption for resource-constrained devices to participate in the block formation process.

– Evergrowing ledger size: As the ledger keeps a record of all transactions that occurred since the formation of that blockchain, its size could become over-burdening for capacity-limited IoT devices.

3.3 Hierarchical Blockchain Architecture

In order to account for the storage, processing and energy limitations of IoT devices, we propose a hierarchical blockchain architecture for data security in IoT. In the proposed architecture, the resource-limited IoT devices are connected to an upper layer of "data collectors" that are powerful devices with larger storage and less energy constraints (e.g. cloud servers). In a classical approach, these collectors would normally be connected to a central server, which stores all the collected data. However, in such an architecture, there is a chance to lose the data on some nodes, and the server is vulnerable to attacks. In the proposed architecture, we solve these problems with the benefits of blockchain. Blockchain provides a secure distributed database and eliminates the necessity of a central server. A simplified view of the hierarchical blockchain architecture is provided in Fig. 3.

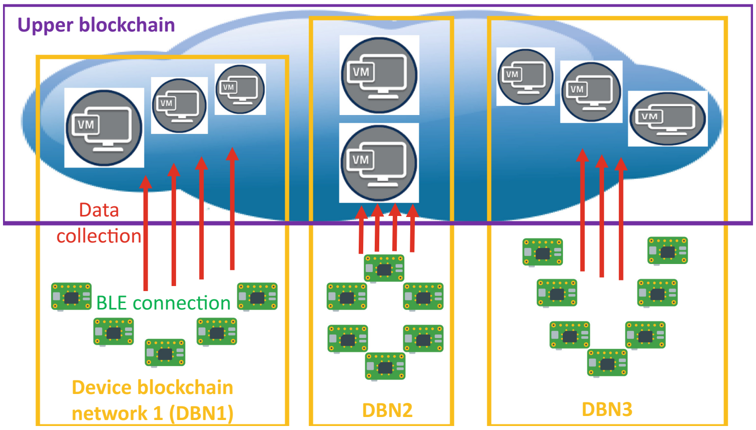


Fig. 3. High-level view of hierarchical blockchain architecture

The model ensures that the computation required for data verification is securely offloaded to nearby edge/cloud servers by resource-constrained devices in an adaptable manner considering device capacity, battery level and available storage space. This will lead to optimized resource utilization through a context-aware hybrid security architecture with computation-intensive work assigned to devices of greater capacity.

Upon issuance of a new transaction (e.g. a new energy purchase) request from a node in the IoT blockchain network, the following steps are taken:

1. The request issuer digitally signs the transaction (data update) with its own private key and broadcasts it in the device-level IoT blockchain network.
2. The transaction goes into a pool of pending transactions as seen in Fig. 4.
3. The miners in the local network verify the validity of the transaction origin by checking the digital signature on the transaction against the public key of the sender.
4. Miners pick out transactions from the pending pool, confirm the validity of the transaction logic (e.g. that the issuer has sufficient balance in his/her account to purchase the requested amount of energy) and start solving a cryptographic puzzle (using a secure hash algorithm such as SHA256 [6]). Solving the puzzle is achieved by finding a nonce value through repeatedly performing a cryptographic hash calculation involving the hash of the previous block and the current transaction until the target value is achieved.
5. The first miner able to solve the puzzle places the selected transaction in the chain, which needs to be approved by a majority of the network participants. The verification by the other participants is easy, as they are now given the correct nonce value and already have the hash of the previous block.
6. The linking of the transaction to the chain is provided by including the cryptographic hash (irreversible) of the previous block in the chain, so that any modifications to the previous data collected will not be possible (i.e. will require significant effort and computational resources).

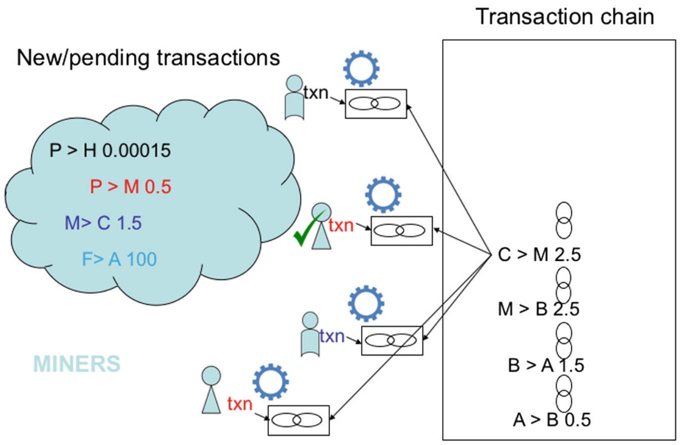


Fig. 4. Formation of blocks in blockchain

While the device-level blockchain operates as described above, the blockchain at the upper layer having powerful servers maintains the ledger of all transactions from the various blockchains at the layer below. This enables resetting of the device-level blockchains periodically to handle resource limitations, while still maintaining a transparent, data integrity-preserving complete history of transactions replicated at multiple sites.

Reconsidering the security issues with the scenario described in Sect. 3.1, we observe the proposed model mitigates them as follows:

1. Once a reading enters the blockchain, it will not be possible to change it, as the alteration would require building the whole chain from the beginning, relying on the utilization of expensive computational resources, which will outweigh the benefits of changing the meter reading/trade transaction for the adversary. Additionally, transactions that are not possible (such as negative meter readings) will not be validated by the blockchain network.
2. As the whole transaction history will be replicated at multiple sites in a transparent manner, there will be no single point of failure.
3. It will not be possible for malicious devices to spoof others, as transactions will require digital signatures, validated by all network participants.
4. Due to the transparency of transactions and the validation process, fraudulent transactions will not be possible.

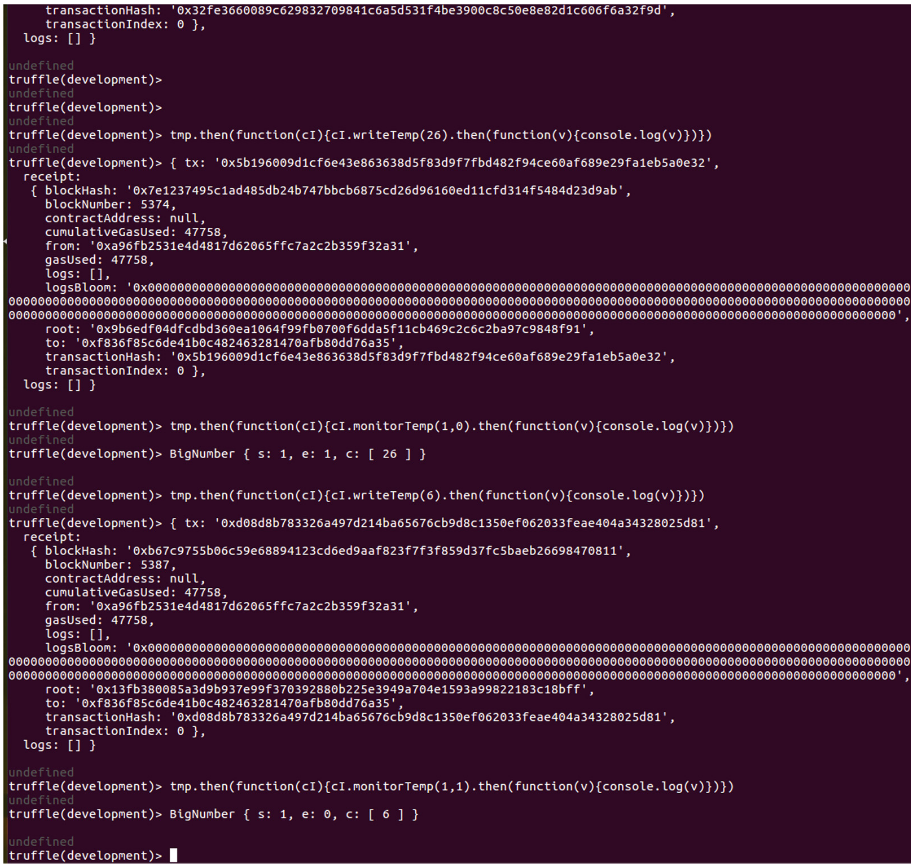
4 Implementation

In a prototype implementation of the proposed architecture, we used Raspberry Pi[1] devices that collect temperature data using their sensors. These devices are connected to cloud servers that serve as data collectors through WiFi. Raspberry Pi devices connected to the same data collector create a blockchain network between themselves, and all data collectors also create a blockchain network among themselves, without using a central server. Due to the storage limitations of the Raspberry Pi devices, ledgers are formed within the network formed at the device level, and the data are pushed to the layer above periodically. This results in a blockchain of blockchains to store the entire transaction history at the upper layer involving powerful cloud servers. The blockchain was implemented using the Ethereum blockchain platform[2].

In order to evaluate the feasibility of the proposed architecture for IoT security, we performed experiments with the developed IoT blockchain. Ethereum virtual machines were installed on Raspberry Pi devices, as well as on machine instances (t2.medium and c5.large instance types) in the AWS EC2[3] and connected to the same blockchain network through WiFi. The task of mining was offloaded to the cloud servers, as the devices were observed not to be capable of successfully completing the mining process when the difficulty level of the cryptographic puzzle to solve was high, which justifies the use of edge computing and a hierarchical approach in the proposed architecture[4].

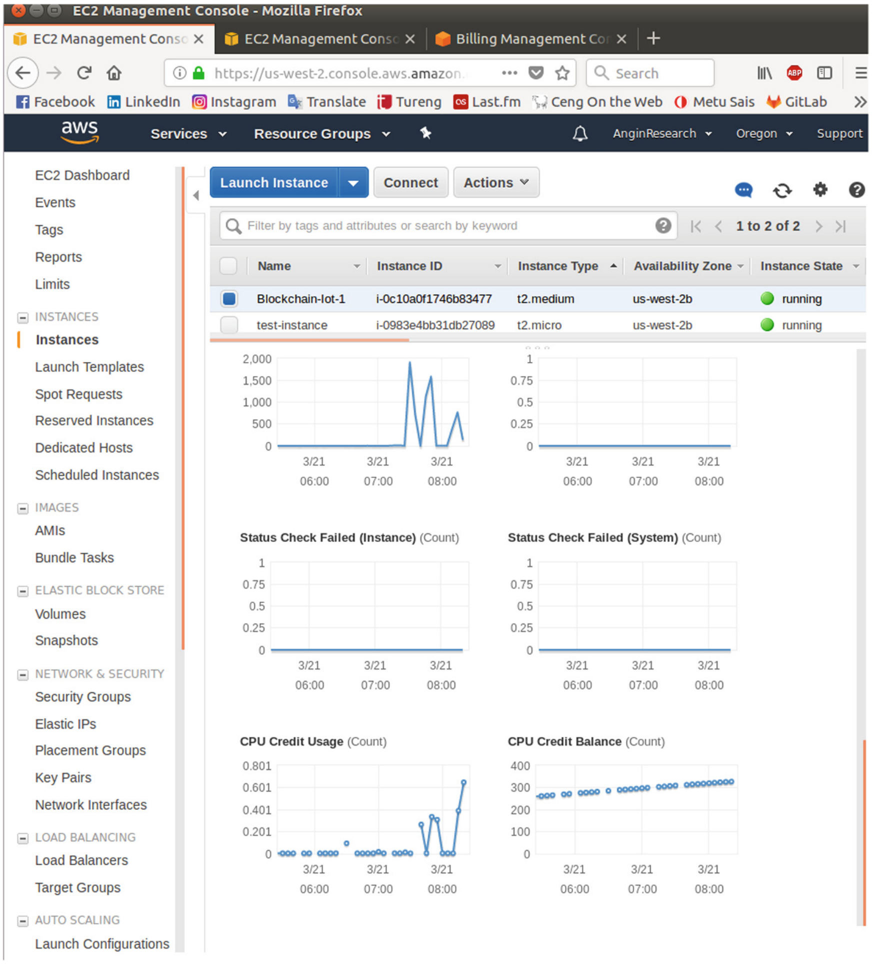
[1] https://www.raspberrypi.org/.
[2] https://www.ethereum.org/.
[3] https://aws.amazon.com/ec2/.
[4] Actually, even the t2.micro instance of AWS EC2 was not capable of successfully completing a mining task in the experiments.

In order to keep the transaction processing time standard (around 12–15 s), the Ethereum blockchain builds a directed acyclic graph (DAG) and automatically adjusts the difficulty of the cryptographic puzzles, which makes it difficult to evaluate performance in terms of transaction processing time. It was observed that the collected sensor data was processed and stored securely on all nodes in the formed IoT network. Figures 5, 6, 7 and 8 show excerpts from the runtime environment of the implemented blockchain.



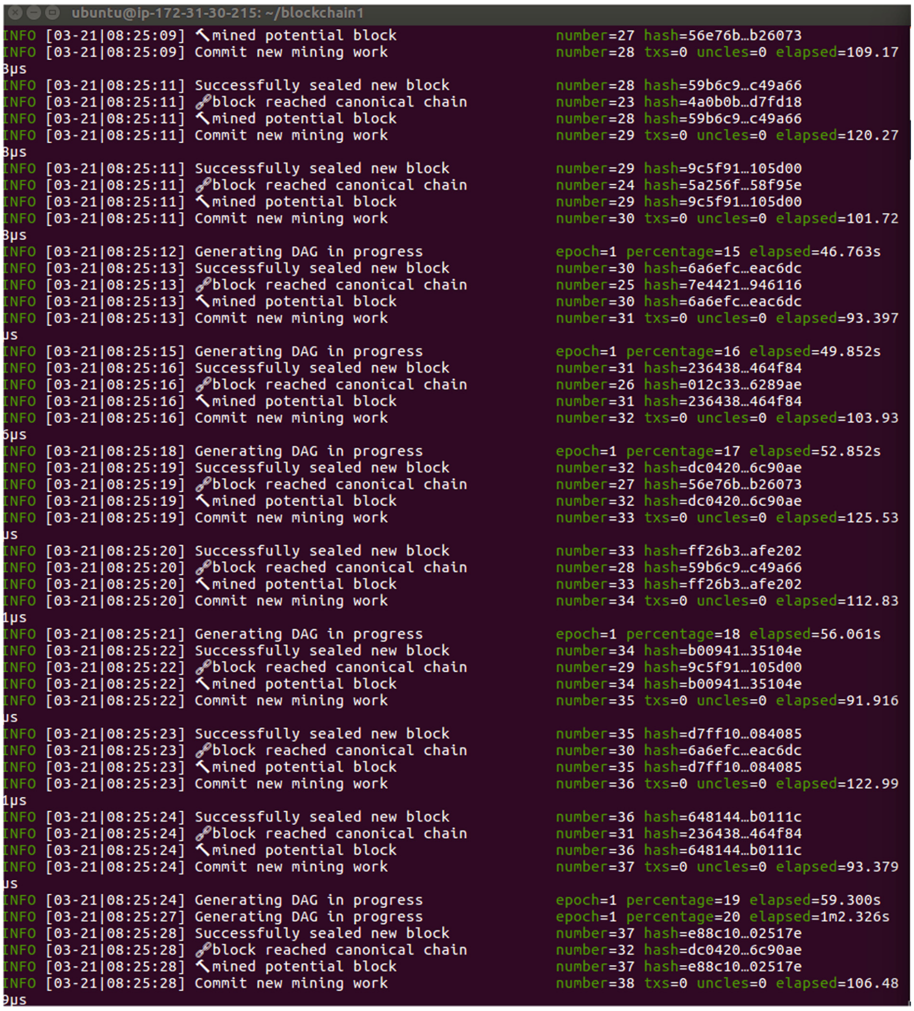
```
     transactionHash: '0x32fe3660089c629832709841c6a5d531f4be3900c8c50e8e82d1c606f6a32f9d',
     transactionIndex: 0 },
  logs: [] }

undefined
truffle(development)>
undefined
truffle(development)>
undefined
truffle(development)> tmp.then(function(cI){cI.writeTemp(26).then(function(v){console.log(v)})})
undefined
truffle(development)> { tx: '0x5b196009d1cf6e43e863638d5f83d9f7fbd482f94ce60af689e29fa1eb5a0e32',
  receipt:
   { blockHash: '0x7e1237495c1ad485db24b747bbcb6875cd26d96160ed11cfd314f5484d23d9ab',
     blockNumber: 5374,
     contractAddress: null,
     cumulativeGasUsed: 47758,
     from: '0xa96fb2531e4d4817d62065ffc7a2c2b359f32a31',
     gasUsed: 47758,
     logs: [],
     logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
     root: '0x9b6edf04dfcdbd360ea1064f99fb0700f6dda5f11cb469c2c6c2ba97c9848f91',
     to: '0xf836f85c6de41b0c482463281470afb80dd76a35',
     transactionHash: '0x5b196009d1cf6e43e863638d5f83d9f7fbd482f94ce60af689e29fa1eb5a0e32',
     transactionIndex: 0 },
  logs: [] }

undefined
truffle(development)> tmp.then(function(cI){cI.monitorTemp(1,0).then(function(v){console.log(v)})})
undefined
truffle(development)> BigNumber { s: 1, e: 1, c: [ 26 ] }

undefined
truffle(development)> tmp.then(function(cI){cI.writeTemp(6).then(function(v){console.log(v)})})
undefined
truffle(development)> { tx: '0xd08d8b783326a497d214ba65676cb9d8c1350ef062033feae404a34328025d81',
  receipt:
   { blockHash: '0xb67c9755b06c59e68894123cd6ed9aaf823f7f3f859d37fc5baeb26698470811',
     blockNumber: 5387,
     contractAddress: null,
     cumulativeGasUsed: 47758,
     from: '0xa96fb2531e4d4817d62065ffc7a2c2b359f32a31',
     gasUsed: 47758,
     logs: [],
     logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
     root: '0x13fb380085a3d9b937e99f370392880b225e3949a704e1593a99822183c18bff',
     to: '0xf836f85c6de41b0c482463281470afb80dd76a35',
     transactionHash: '0xd08d8b783326a497d214ba65676cb9d8c1350ef062033feae404a34328025d81',
     transactionIndex: 0 },
  logs: [] }

undefined
truffle(development)> tmp.then(function(cI){cI.monitorTemp(1,1).then(function(v){console.log(v)})})
undefined
truffle(development)> BigNumber { s: 1, e: 0, c: [ 6 ] }

undefined
truffle(development)> █
```

Fig. 5. Ethereum transaction sample



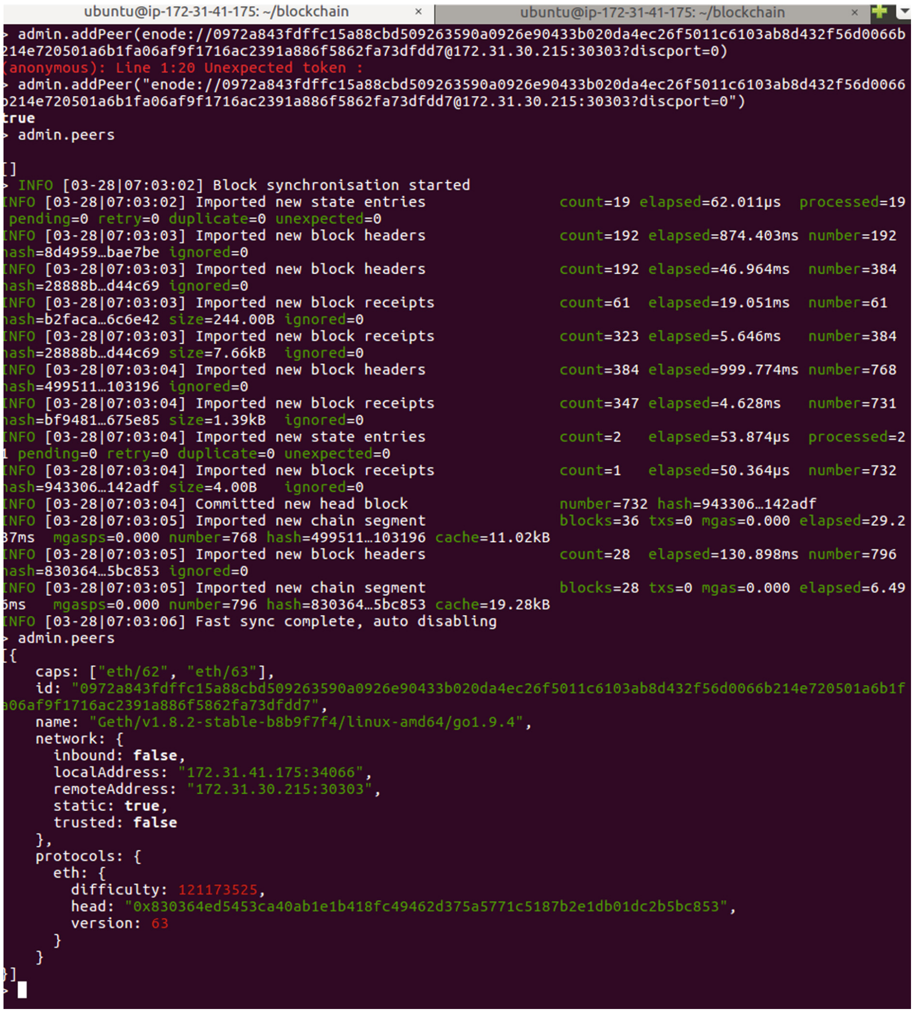
Name	Instance ID	Instance Type	Availability Zone	Instance State
Blockchain-Iot-1	i-0c10a0f1746b83477	t2.medium	us-west-2b	running
test-instance	i-0983e4bb31db27089	t2.micro	us-west-2b	running

Fig. 6. AWS instances in Ethereum blockchain



```
ubuntu@ip-172-31-30-215: ~/blockchain1
INFO [03-21|08:25:09] 🔨mined potential block                number=27 hash=56e76b…b26073
INFO [03-21|08:25:09] Commit new mining work                 number=28 txs=0 uncles=0 elapsed=109.17
3µs
INFO [03-21|08:25:11] Successfully sealed new block          number=28 hash=59b6c9…c49a66
INFO [03-21|08:25:11] 🔗block reached canonical chain         number=23 hash=4a0b0b…d7fd18
INFO [03-21|08:25:11] 🔨mined potential block                number=28 hash=59b6c9…c49a66
INFO [03-21|08:25:11] Commit new mining work                 number=29 txs=0 uncles=0 elapsed=120.27
3µs
INFO [03-21|08:25:11] Successfully sealed new block          number=29 hash=9c5f91…105d00
INFO [03-21|08:25:11] 🔗block reached canonical chain         number=24 hash=5a256f…58f95e
INFO [03-21|08:25:11] 🔨mined potential block                number=29 hash=9c5f91…105d00
INFO [03-21|08:25:11] Commit new mining work                 number=30 txs=0 uncles=0 elapsed=101.72
3µs
INFO [03-21|08:25:12] Generating DAG in progress             epoch=1 percentage=15 elapsed=46.763s
INFO [03-21|08:25:13] Successfully sealed new block          number=30 hash=6a6efc…eac6dc
INFO [03-21|08:25:13] 🔗block reached canonical chain         number=25 hash=7e4421…946116
INFO [03-21|08:25:13] 🔨mined potential block                number=30 hash=6a6efc…eac6dc
INFO [03-21|08:25:13] Commit new mining work                 number=31 txs=0 uncles=0 elapsed=93.397
µs
INFO [03-21|08:25:15] Generating DAG in progress             epoch=1 percentage=16 elapsed=49.852s
INFO [03-21|08:25:16] Successfully sealed new block          number=31 hash=236438…464f84
INFO [03-21|08:25:16] 🔗block reached canonical chain         number=26 hash=012c33…6289ae
INFO [03-21|08:25:16] 🔨mined potential block                number=31 hash=236438…464f84
INFO [03-21|08:25:16] Commit new mining work                 number=32 txs=0 uncles=0 elapsed=103.93
6µs
INFO [03-21|08:25:18] Generating DAG in progress             epoch=1 percentage=17 elapsed=52.852s
INFO [03-21|08:25:19] Successfully sealed new block          number=32 hash=dc0420…6c90ae
INFO [03-21|08:25:19] 🔗block reached canonical chain         number=27 hash=56e76b…b26073
INFO [03-21|08:25:19] 🔨mined potential block                number=32 hash=dc0420…6c90ae
INFO [03-21|08:25:19] Commit new mining work                 number=33 txs=0 uncles=0 elapsed=125.53
µs
INFO [03-21|08:25:20] Successfully sealed new block          number=33 hash=ff26b3…afe202
INFO [03-21|08:25:20] 🔗block reached canonical chain         number=28 hash=59b6c9…c49a66
INFO [03-21|08:25:20] 🔨mined potential block                number=33 hash=ff26b3…afe202
INFO [03-21|08:25:20] Commit new mining work                 number=34 txs=0 uncles=0 elapsed=112.83
1µs
INFO [03-21|08:25:21] Generating DAG in progress             epoch=1 percentage=18 elapsed=56.061s
INFO [03-21|08:25:22] Successfully sealed new block          number=34 hash=b00941…35104e
INFO [03-21|08:25:22] 🔗block reached canonical chain         number=29 hash=9c5f91…105d00
INFO [03-21|08:25:22] 🔨mined potential block                number=34 hash=b00941…35104e
INFO [03-21|08:25:22] Commit new mining work                 number=35 txs=0 uncles=0 elapsed=91.916
µs
INFO [03-21|08:25:23] Successfully sealed new block          number=35 hash=d7ff10…084085
INFO [03-21|08:25:23] 🔗block reached canonical chain         number=30 hash=6a6efc…eac6dc
INFO [03-21|08:25:23] 🔨mined potential block                number=35 hash=d7ff10…084085
INFO [03-21|08:25:23] Commit new mining work                 number=36 txs=0 uncles=0 elapsed=122.99
1µs
INFO [03-21|08:25:24] Successfully sealed new block          number=36 hash=648144…b0111c
INFO [03-21|08:25:24] 🔗block reached canonical chain         number=31 hash=236438…464f84
INFO [03-21|08:25:24] 🔨mined potential block                number=36 hash=648144…b0111c
INFO [03-21|08:25:24] Commit new mining work                 number=37 txs=0 uncles=0 elapsed=93.379
µs
INFO [03-21|08:25:24] Generating DAG in progress             epoch=1 percentage=19 elapsed=59.300s
INFO [03-21|08:25:27] Generating DAG in progress             epoch=1 percentage=20 elapsed=1m2.326s
INFO [03-21|08:25:28] Successfully sealed new block          number=37 hash=e88c10…02517e
INFO [03-21|08:25:28] 🔗block reached canonical chain         number=32 hash=dc0420…6c90ae
INFO [03-21|08:25:28] 🔨mined potential block                number=37 hash=e88c10…02517e
INFO [03-21|08:25:28] Commit new mining work                 number=38 txs=0 uncles=0 elapsed=106.48
9µs
```

Fig. 7. Mining in Ethereum blockchain



```
ubuntu@ip-172-31-41-175: ~/blockchain                ubuntu@ip-172-31-41-175: ~/blockchain
> admin.addPeer(enode://0972a843fdffc15a88cbd509263590a0926e90433b020da4ec26f5011c6103ab8d432f56d0066b
214e720501a6b1fa06af9f1716ac2391a886f5862fa73dfdd7@172.31.30.215:30303?discport=0)
(anonymous): Line 1:20 Unexpected token :
> admin.addPeer("enode://0972a843fdffc15a88cbd509263590a0926e90433b020da4ec26f5011c6103ab8d432f56d0066
b214e720501a6b1fa06af9f1716ac2391a886f5862fa73dfdd7@172.31.30.215:30303?discport=0")
true
> admin.peers

[]
> INFO [03-28|07:03:02] Block synchronisation started
INFO [03-28|07:03:02] Imported new state entries          count=19 elapsed=62.011µs  processed=19
 pending=0 retry=0 duplicate=0 unexpected=0
INFO [03-28|07:03:03] Imported new block headers          count=192 elapsed=874.403ms number=192
ash=8d4959…bae7be ignored=0
INFO [03-28|07:03:03] Imported new block headers          count=192 elapsed=46.964ms  number=384
ash=28888b…d44c69 ignored=0
INFO [03-28|07:03:03] Imported new block receipts         count=61  elapsed=19.051ms  number=61
ash=b2faca…6c6e42 size=244.00B ignored=0
INFO [03-28|07:03:03] Imported new block receipts         count=323 elapsed=5.646ms   number=384
ash=28888b…d44c69 size=7.66kB  ignored=0
INFO [03-28|07:03:04] Imported new block headers          count=384 elapsed=999.774ms number=768
ash=499511…103196 ignored=0
INFO [03-28|07:03:04] Imported new block receipts         count=347 elapsed=4.628ms   number=731
ash=bf9481…675e85 size=1.39kB  ignored=0
INFO [03-28|07:03:04] Imported new state entries          count=2   elapsed=53.874µs  processed=2
1 pending=0 retry=0 duplicate=0 unexpected=0
INFO [03-28|07:03:04] Imported new block receipts         count=1   elapsed=50.364µs  number=732
ash=943306…142adf size=4.00B   ignored=0
INFO [03-28|07:03:04] Committed new head block            number=732 hash=943306…142adf
INFO [03-28|07:03:05] Imported new chain segment          blocks=36 txs=0 mgas=0.000 elapsed=29.2
37ms  mgasps=0.000 number=768 hash=499511…103196 cache=11.02kB
INFO [03-28|07:03:05] Imported new block headers          count=28  elapsed=130.898ms number=796
ash=830364…5bc853 ignored=0
INFO [03-28|07:03:05] Imported new chain segment          blocks=28 txs=0 mgas=0.000 elapsed=6.49
6ms   mgasps=0.000 number=796 hash=830364…5bc853 cache=19.28kB
INFO [03-28|07:03:06] Fast sync complete, auto disabling
> admin.peers
[{
    caps: ["eth/62", "eth/63"],
    id: "0972a843fdffc15a88cbd509263590a0926e90433b020da4ec26f5011c6103ab8d432f56d0066b214e720501a6b1f
a06af9f1716ac2391a886f5862fa73dfdd7",
    name: "Geth/v1.8.2-stable-b8b9f7f4/linux-amd64/go1.9.4",
    network: {
      inbound: false,
      localAddress: "172.31.41.175:34066",
      remoteAddress: "172.31.30.215:30303",
      static: true,
      trusted: false
    },
    protocols: {
      eth: {
        difficulty: 121173525,
        head: "0x830364ed5453ca40ab1e1b418fc49462d375a5771c5187b2e1db01dc2b5bc853",
        version: 63
      }
    }
}]
> 
```

Fig. 8. Blockchain syncronization in Ethereum blockchain

5 Conclusion

In this paper we proposed a data security architecture for IoT networks. The solution is based on the application of the blockchain technology to IoT networks to provide decentralized device authentication and data security guarantees, taking into consideration the resource limitations of IoT devices and the heterogeneity of IoT networks, which enables utilization of powerful cloud servers for mining in the blockchain network. This work differs from previous work based on blockchain in that it proposes a hierarchical structure of blockchain (blockchain of blockchains) to overcome the resource problems in IoT. A prototype implementation of the proposed architecture demonstrates the feasibility and promise of the model to provide data security in IoT. Future work will include modifications to the structure of blockchain itself to utilize lightweight cryptography fit for the nature of IoT as well as amendments to the consensus protocol used, in order the achieve higher performance mining on IoT devices as well.

References

1. Barcelona: Barcelona ciutat digital. http://ajuntament.barcelona.cat/digital/ca. Accessed Mar 2018
2. Capossele, A., Cervo, V., Cicco, G.D., Petrioli, C.: Security as a CoAP resource: an optimized DTLS implementation for the iot. In: IEEE ICC, pp. 549–554 (2015)
3. European Commission: IDC and TXT solutions, smart 2013/0037 cloud and iot combination, study for the european commission. http://www.telit2market.com/wp-content/uploads/2015/02/TEL_14016_P_112-114.pdf. Accessed Mar 2018
4. Dorri, A., Kanhere, S.S., Jurdak, R.: Towards and optimized blockchain for IoT. In: 2nd ACM/IEEE IoTDI, pp. 173–178 (2017)
5. Dyn: Dyn analysis summary of friday october 21 attack. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/. Accessed Mar 2018
6. Eastlake, D., Hansen, T.: US Secure Hash Algorithms. RFC 6234, RFC Editor, May 2011
7. Gartner: Gartner says 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. https://www.gartner.com/newsroom/id/3598917. Accessed Apr 2018
8. Globe, B.: If the CIA can compromise our gadgets, can others do the same? https://www.bostonglobe.com/business/2017/03/08/wikileaks-hits-cia-secrecy-software-spying/EQdLVwseMu70HEYlZcowOO/story.html. Accessed Mar 2018
9. Greenberg, A.: Hackers remotely kill a jeep on the highway with me in it. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/. Accessed Mar 2018
10. Gross, H., Hölbl, M., Slamanig, D., Spreitzer, R.: Privacy-aware authentication in the internet of things. In: Reiter, M., Naccache, D. (eds.) CANS 2015. LNCS, vol. 9476, pp. 32–39. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26823-1_3
11. NIST Network Working Group: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor, May 2008
12. Kelley, M.B.: The stuxnet attack on iran's nuclear plant was 'far more dangerous' than previously thought. http://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11. Accessed Mar 2018
13. Laurence, T.: Blockchain for dummies. For Dummies (2017)
14. Modadugu, N., Rescorla, E.: The design and implementation of datagram TLS. In: Network and Distributed System Security Symposium (NDSS) (2004)
15. Moinet, A., Darties, B., Baril, J.L.: Blockchain based trust & authentication for decentralized sensor networks. CoRR abs/1706.01730 (2017)
16. Qorvo: Sensara senior lifestyle. http://www.qorvo.com/resources/d/qorvo-sensara-senior-lifestyle-white-paper. Accessed Mar 2018
17. Research, F.: A Mix of New and Existing Technologies Help Secure IoT Deployments. Technical report, Forrester Research (Q1 2017)

18. Skarmeta, A., et al.: A decentralized approach for security and privacy challenges in the internet of things. In: IEEE World Forum on Internet of Things (WF-IoT), pp. 67–72 (2014)
19. Road Traffic Technology: M42 active traffic management scheme, birmingham. http://www.roadtraffic-technology.com/projects/m42/. Accessed Mar 2018
20. Ukil, A., Bandyopadhyay, S., Pal, A.: Iot-privacy: to be private or not to be private. In: INFOCOM Computer Communications Workshops, pp. 123–124 (2014)

Pattern Based Integration of Internet of Things Systems

Bedir Tekinerdogan[1] and Ömer Köksal[1,2](✉)

[1] Wageningen University, Wageningen, The Netherlands
bedir.tekinerdogan@wur.nl
[2] ASELSAN Research Center, Ankara, Turkey
koksal@aselsan.com.tr

Abstract. The Internet of Things (IoT) is the network of physical devices embedded with sensors, actuators, and connectivity which enables these objects to connect and exchange data. Cleary the IoT has a pervasive impact on the society and an increasing number of systems are now based on IoT. One of the key challenges in IoT is coping with the heterogeneous set of systems and the integration of these systems in the same communication network. Several studies have focused on this integration aspect and addressed this at different levels of abstraction. Unfortunately, the different approaches are scattered and fragmented over the different studies and it is not clear how to cope with the integration concern within a single IoT system but also across multiple IoT systems that need to be integrated. To this end this chapter provides a comprehensive and systematic approach for identifying the key integration concerns in the IoT system architecture and describing the currently provided solutions. For this we adopt a pattern-based approach in which generic architecture solution structures are provided to these recurring integration concerns. We illustrate our approach for addressing the integration of IoT based systems within the context of smart city engineering.

Keywords: Internet of Things · Architecture design patterns
Smart city engineering

1 Introduction

The Internet of Things (IoT) is the result of technological progress in many parallel and often overlapping fields, including those of embedded systems, ubiquitous and pervasive computing, mobile telephony, telemetry and machine-to-machine communication, wireless sensor networks, mobile computing, and computer networking. In essence, IoT combines the concepts "Internet" and "Thing" and the provided definitions in the literature can be interpreted how these have addressed these two concepts. What is important is that IoT adds a new dimension "anything" to the current communication technologies (ICTs), which already provide "any time" and "any place" communication.

To support the design of IoT systems, various reference architectures have been provided in the literature. In general, IoT architecture is represented as a layered architecture with various set of layers representing a grouping of modules that offer a cohesive set of services. Based on the literature [1–4] we provide the reference architecture as shown in Fig. 1.

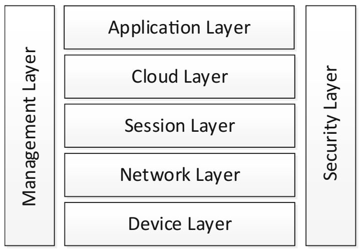


Fig. 1. IoT reference architecture

The reference architecture consists of seven layers including Device Layer, Network Layer, Session Layer, Cloud Layer, Application Layer, Management Layer, and Security Layer [5]. Usually these layers can be distributed in different ways over the different nodes in the IoT system. Using the IoT reference architecture various different IoT systems can be designed. Each such IoT system integrates the various devices within the same network. Yet, the scope of an IoT system is often within a particular scope and the integration with other IoT systems or non-IoT systems is not a trivial task.

Cleary the IoT has a pervasive impact on the society and an increasing number of systems are now based on IoT. One of the key challenges in IoT is coping with the heterogeneous set of systems and the integration of these systems in the same communication network. Several studies have focused on this integration aspect and addressed this at different levels of abstraction. Unfortunately, the different approaches are scattered and fragmented over the different studies and it is not clear how to cope with the integration concern within a single IoT system but also across multiple IoT systems that need to be integrated. To this end this chapter provides a comprehensive and systematic approach for identifying the key integration concerns in the IoT system architecture and describing the currently provided solutions. For this we adopt a pattern-based approach in which generic architecture solution structures are provided to these recurring integration concerns. We illustrate our approach for addressing the integration of IoT based systems within the context of smart city engineering.

The remainder of the paper is organized as follows. Section 2 presents the case study together with the problem statement. Section 3 presents the integration framework together with the adopted integration patterns. Section 4 presents the adopted process for preparing and designing an integrated IoT system. Section 5 illustrates the application of the approach for smart city engineering context. Section 6 concludes the paper.

2 Case Study: Smart City Engineering

In this section, we define a case study that will be used to illustrate the problem statement and the approach in further sections. The case study that we consider is within the context of smart city engineering [6]. One of the important applications in smart city engineering includes the development of smart traffic system (STS). STS provides different capabilities such as traffic light management, congestion detection, traffic regulation, shared parking platform, etc. The high-level reference architecture of STS is depicted in Fig. 2 [7].

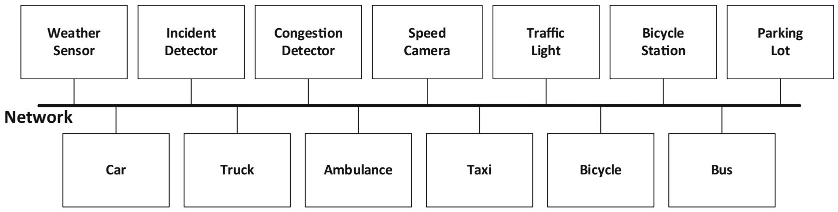


Fig. 2. Conceptual architecture for smart traffic system

Although the above case integrates many different entities it still deals with the design of a single system, in this case an STS. Very often it is required though to integrate the STS with other systems in the smart city engineering context, such as city energy consumption system, the weather information system, the security system, the air quality control system, the smart lighting system etc. (Fig. 3).

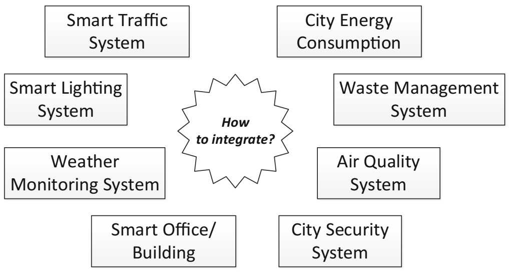


Fig. 3. Integration of different IoT-based systems in the context of smart city engineering

Integrating all these systems in a coherent manner is not trivial and requires careful consideration. We will elaborate on this in the next sections.

3 Integration Framework

The integration of IoT systems can be considered at different abstraction levels. We will discuss the integration based on the four layers of the architecture as defined in Fig. 1. To illustrate this need for integration at different levels Fig. 4 shows the integration of different IoT systems.

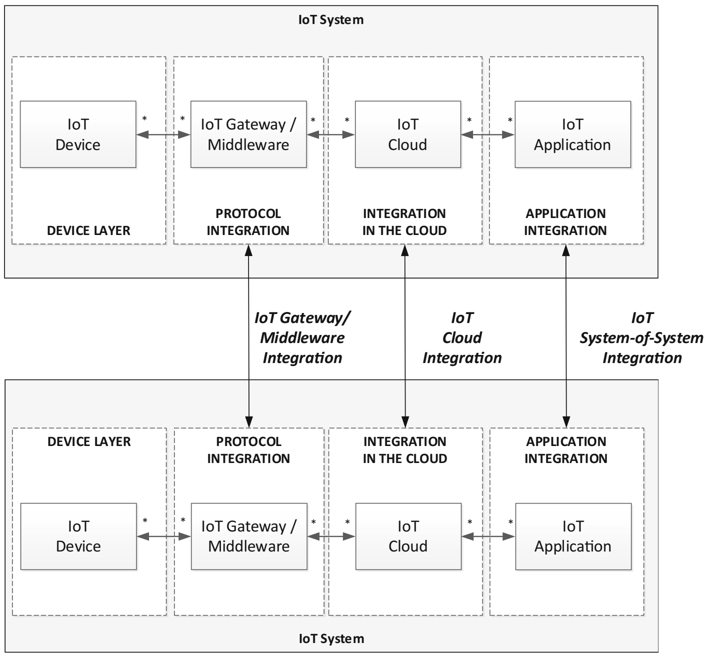


Fig. 4. System integration of IoT-based systems in different layers

As shown in Fig. 4 we distinguish the following types of integration in IoT systems (1) *Session Layer Integration: (1a) Protocol Integration via IoT Gateway (1b) Protocol Integration via Middleware, (2) Cloud Layer Integration,* and *(3) Application Layer Integration.* For describing the integration solutions, we will adopt a design pattern-based approach. A design pattern represents a generic solution to recurring problems. Design patterns play an important role in the engineering design process and can be applied at the different levels in the lifecycle including the architecture design, detailed design, and the code. In this paper, we will mainly focus on architectural patterns which focus on the gross-level structure of the system and its interactions [8]. In the following, for each level we will describe the possible architectural patterns that can be used in the integration of IoT systems. Hereby we will also shortly indicate the advantages and disadvantages of the adopted architecture design pattern.

3.1 Protocol Integration via IoT Gateway

Multiple session layer protocols exist in the IoT domain to integrate the different things in the IoT as shown in Table 1 [5]. The issue of heterogeneous devices adopting different communication protocols impedes the integration of these devices in the same IoT systems. An IoT gateway acts as a portal between two elements of one or multiple IoT systems, allowing them to share information by communicating between the adopted IoT protocols. An IoT gateway, as such, bridges the gap between devices, cloud, and the computer or mobile device providing a communication link between the devices and cloud.

Table 1. Characteristics of the IoT session layer protocols (adapted from: [5])

Property	AMQP	CoAP	DDS	MQTT	XMPP
Source-Target	S2S	D2D	D2D	D2S	D2S
Real-Time	No	No	Yes	No	Near RT
Broker/Bus-based	Broker	Broker	Bus	Broker	Bus
Communication pattern	Pub/Sub	Request-Reply	Pub/Sub	Pub/Sub	Pub/Sub
Transport	TCP	UDP	TCP/UDP	TCP	TCP
Message/Data centric	Message	Data	Data	Message	Data

Figure 5 shows the different gateway patterns used for integration of IoT systems. In the classical protocol integration hardware/software gateways are used to format and translate data coming from one protocol type to a different protocol type as given in Fig. 5a. This type of protocol integration is successful as long as the number of devices to be integrated is not excessive. However, for a large-scale set of devices it is not easy to handle all the heterogeneous protocols and technologies of the IoT and design a suitable gateway without causing anomalies such as timing and collusion problems. With the possible addition of even more protocols and technologies developed in IoT domain, this problem will become even less manageable [9]. In order to solve the scalability problems and to provide more efficient gateways the following solutions are proposed in the literature.

Distributed Multi-gateway Approach

In this approach multiple gateways are used to cope with the different set of protocols in the IoT system in Fig. 5b [9]. Hereby, the protocols are treated singularly or as a subset of the selected protocols in each gateway. Each gateway translates its protocol to a common shared protocol. The gateways themselves can communicate using the common protocol. By combining gateways dedicated to different technologies multi-protocol scenarios can be generated.

Web-Service Multi-protocol

Instead of having a gateway for each protocol it is also possible to provide a central gateway that is connected to a central conversion server. This so-called web-service multi-protocol pattern is shown in Fig. 5c [9]. In this approach, gateways receive raw data from sensors which are translated to a shared format by connecting with a web-service. In contrast to the distributed multi-gateway there is only one gateway which does the translation among the protocols.

Intelligent Gateways

For translating to different protocols, the gateway can be provided the required translation functionality as shown in Fig. 5d. In this case the gateway will not depend on a separate central server such as in the case of web-service multiprotocol gateway but include the functionality for translating the protocols. Hence, we can indicate this as an intelligent gateway solution. This solution is, for example adopted by Diaz-Cacho et al. [10] and Al-Fuqaha et al. [11]. In these solutions intelligent gateways convert the

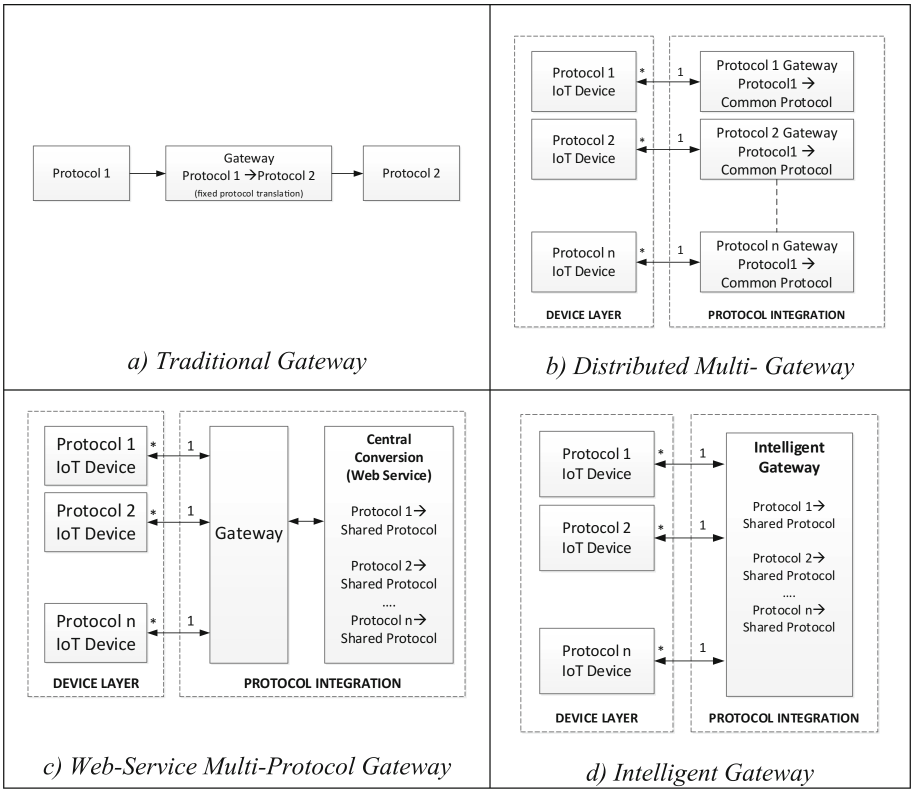

Protocol 1
Gateway
Protocol 1 → Protocol 2
(fixed protocol translation)
Protocol 2
a) Traditional Gateway
Protocol 1 IoT Device
Protocol 2 IoT Device
Protocol n IoT Device
Protocol 1 Gateway Protocol1 → Common Protocol
Protocol 2 Gateway Protocol1 → Common Protocol
Protocol n Gateway Protocol1 → Common Protocol
DEVICE LAYER
PROTOCOL INTEGRATION
b) Distributed Multi- Gateway
Gateway
Central Conversion (Web Service)
Protocol 1→ Shared Protocol
Protocol 2→ Shared Protocol
....
Protocol n→ Shared Protocol
c) Web-Service Multi-Protocol Gateway
Intelligent Gateway
d) Intelligent Gateway

Fig. 5. Different gateway patterns for integration in IoT systems

incoming protocol data to a common shared protocol data which is in this case is extended MQTT. However, the intelligent gateway can in principle also provide different kind of functionality and mapping of the protocols.

3.2 Protocol Integration via Middleware

The alternative way of overcoming the protocol heterogeneity other than using a gateway is the use of a middleware to be used as an abstraction layer. This pattern is shown in Fig. 6. This goes beyond the intelligent gateway solution that includes only functionality for translation among the protocols. In case of a middleware solution also additional functionality such as naming and directory services, security aspects, reliability and other functional and quality services can be also provided. The primary aim of using middleware is to provide seamless integration of systems by hiding the communication and various low-level acquisition aspects [12].

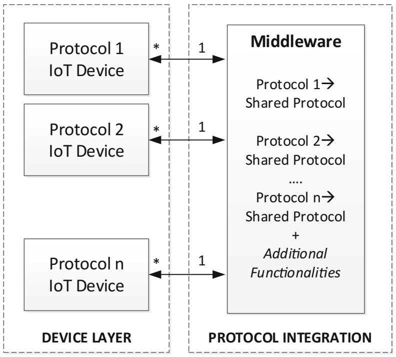


Fig. 6. Integration of the protocols using middleware

There are studies offering the use of an IoT middleware to integrate IoT-based systems in the literature. Ngu et al. [13] provides a survey about IoT middleware integration. Lomotely et al. [14] proposes a middleware to be used as an abstraction layer to address variation in device semantic and protocols that limit the interoperability of the systems. The proposed middleware uses enhanced environment features to match the appropriate communication protocol to aid pushing data from sensors to cloud infrastructure.

3.3 Integration in the Cloud

Another integral component of the IoT is cloud computing. In general, three types of cloud computing models are defined including Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [15–17]. The IaaS model shares hardware resources among the users. Cloud providers typically bill IaaS services according to the utilization of hardware resources by the users. The PaaS model is the basis for the computing platform based upon hardware resources. It is typically an application engine similar to an operating system or a database engine, which binds the hardware resources (IaaS layer) to the software (SaaS layer). The SaaS model is the software layer, which contains the business model. In the SaaS layer, clients are not allowed to modify the lower levels such as hardware resources and application platform. Clients of SaaS systems are typically the end-users that use the SaaS services on-demand basis. For adopting cloud-based integration the different clients are considered the individual systems in the overall IoT system. The integration pattern based on the IaaS, PaaS, or SaaS in the cloud layer is shown in Fig. 7a.

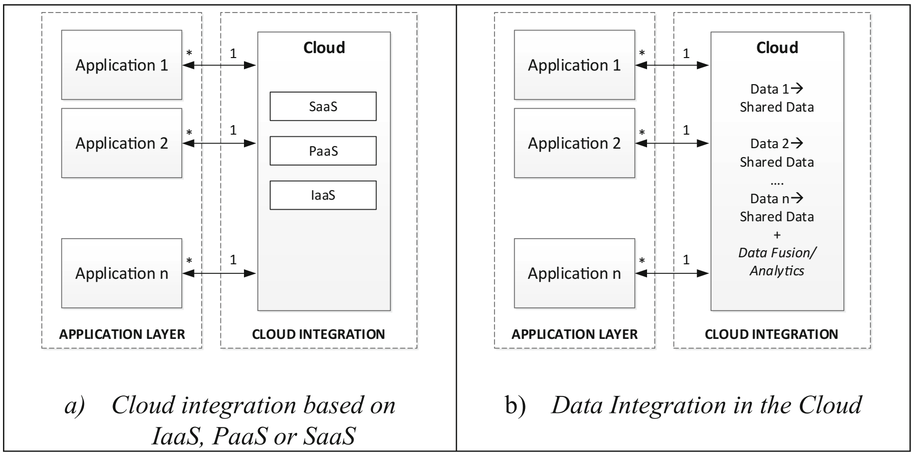


Fig. 7. Cloud-based integration of different IoT systems

An important benefit of IoT is the generation of data that can be further used to derive information to support the decision-making process. The data is typically stored in the cloud which can be used to support analytical and computational tasks on these data allowing centralized access to the generated IoT services [18].

Figure 7b shows the pattern for the data integration in the cloud. Hereby, the integration of the systems is primarily based on the integration of the data from the different IoT systems. Since each IoT system can use its own type of data platform and the corresponding data structures and formatting, the integration will need to support data interoperability. For this it is needed to adopt a common data format and platform that is adopted at a central cloud node. Incoming data from different nodes will be typically mapped to a shared data format. Subsequently a data fusion and/or data conversion process will be carried out to synthesize the data. The cloud node will typically include analytics modules for processing the data for descriptive, diagnostic, predictive or prescriptive analytics.

3.4 Integration in Application Layer

Besides of integration at the gateway or middleware level we can also achieve the integration of IoT systems at the application layer level. Much has been written about application integration and likewise we will borrow from the earlier concepts to define the integration of IoT systems. In the literature dozens of architecture patterns have been published regarding application integration [19–24]. In the following we will consider only those patterns that can be directly used for supporting the integration of systems, and in the context of this chapter, in particular the integration of IoT systems.

Peer-to-Peer

In the peer-to-peer architectural pattern, peers (IoT Systems) connect to each other directly, and there is no intermediate component between the IoT systems. The conceptual model is shown in Fig. 8a. The elements in the system are autonomous, equal peers that are both providers and consumers of data and processing power. Further, the primary content is provided by peers, are there are no central components providing content. In addition, peers can be added and removed from the system at any time.

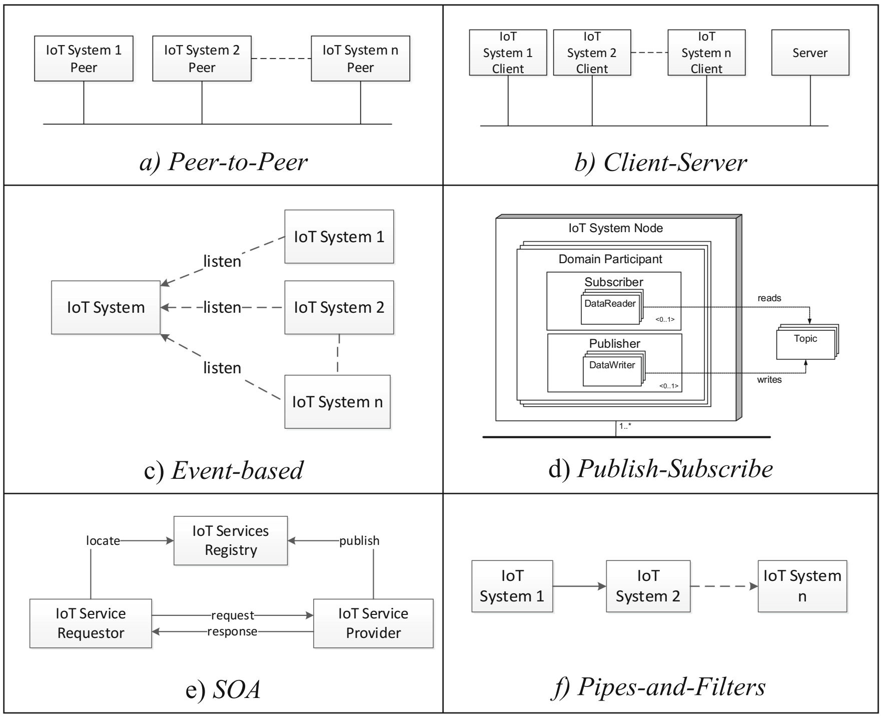


Fig. 8. Patterns for integration at the application layer

Client-Server

The Client-Server architectural pattern is a very common and well-known pattern for network-based applications. The conceptual model this pattern is shown in Fig. 8b. Hereby some systems play the role of Clients, while other adopts the role of a Server. One or multiple Client components initiate a request to a Server, which then performs some computations and responds to the Clients. Only clients can initiate communication, while servers only respond to requests from clients. If needed server components can be clients to other servers. Clients cannot communicate to each other. As we will see in the later sections this is different from the Event-Based and Streaming invocations since the Client decides itself when to initiate a request.

Event-Based

The conceptual diagram of the event-based software architectural pattern is given in Fig. 8c. This pattern is based on implicit invocations which are induced by events, i.e., when a certain event takes place it triggers the function calls. Event can be defined as a significant change in state. Typically, event-based systems are composed of event producers, event consumers and event channels. The events are sent to the listeners over the network even they are not on the same hardware. So, this pattern is well-suited for real-time applications, message-oriented middleware, and point-to-point communications. Further, the event-based pattern supports parallel execution of tasks and scalability.

Publish-Subscribe

This pattern is shown in Fig. 8d. It consists of mainly three elements including Publishers, Subscribers, and Topics. Publishers write to Topics and Subscribers read from the Topics on which they are registered. One Publisher can write to many Topics and one Subscriber can read from many Topics. Unlike the event-based pattern described above, the subscribers in this pattern are all interested in a type of event happening without knowing the publisher of the event. The adopted communication pattern provides space decoupling, time decoupling and synchronization decoupling [25]. The decoupling of producer and consumer participants increases scalability by removing explicit independencies between communicating parties. Removing these dependencies together with the asynchronous communication feature of this infrastructure makes this pattern well-suited for even large scale IoT systems.

Service Oriented Architecture (SOA)

The service-oriented architecture deals with composing applications by integrating distributed, separately maintained components aiming vendor and technology independence. This integration composed of three essential loosely coupled parts which are registry, service providers, and service requestors as shown in Fig. 8e. In this integration type, service publishes its description to the service registry that keeps the list of all services with their locations and functionalities. When a service requester requires a service, it gets the required information from the registry and communicates with the requested service over a standardized communication.

This type of decoupled integration is especially suitable for heterogeneous distributed systems supporting evolvability and interoperability. The disadvantage of this integration pattern is the complexity.

Pipes and Filters

This pattern is composed of two basic elements: pipes and filters as shown in Fig. 8f. Filters are connected to each other by pipes. Filters transform the data received from another filter into a new form and output this transformed data to the following filter. Pipes are the routes of data streams. Although filters are independent of each other and might execute parallel, they must use the data type agreed with pipe in order to communication takes place. This simple communication mechanism makes the pattern scalable and reusable supporting evolvability. On the other hand, this batch-type data processing cannot handle interactivity well and latency causes performance degradation.

4 Overall Approach

Table 2 shows the summary of the previously defined patterns that can be used to support integration of the concerns in the IoT system. Obviously, it is clear that for integrating multiple IoT systems many different issues need to be taken into account. To guide and support the integration of the IoT systems we propose the process as shown in Fig. 9.

Table 2. Identified list of patterns that can be used in the integration process

Layer	Pattern	Integration Approach
Session layer	Traditional gateway	Provides translation of a given protocol to a predefined protocol
	Multi-gateway	Provides multiple gateways each of which can translate to a dedicated protocol
	Web-service multi-protocol	Provides a single gateway that can provide the translation of the protocols to a common protocol through a central web server
	Intelligent gateway	Provides a gateway that includes the required functionality for translating to different protocols
	Middleware	Connects devices within or across IoT systems and provides additional services (e.g. naming and directory, quality of service, etc.)
Cloud layer	SaaS based	Multiple IoT systems are tenants of the cloud and use the SaaS
	PaaS based	Multiple IoT systems are tenants of the cloud and use the PaaS
	IaaS based	Multiple IoT systems are tenants of the cloud and use the IaaS
	Data integration	Multiple IoT systems use data fusion and analytics as cloud services
Application layer	Peer-to-Peer	Entities within an IoT system or across IoT systems communicate as peers, that is, autonomously as data providers and consumers
	Client Server	One IoT system or another system is defined as a server which is used by multiple other IoT systems
	Event-Based	IoT elements listen to other IoT elements. In case of changes events are triggered to the coupled IoT elements (system or devices)
	Publish-Subscribe	Multiple IoT systems communicate as participants that are interested in defined topics. If topics change, the loosely coupled IoT systems are notified which can take further actions
	SOA	IoT service providers define and register their services to the IoT Service Registry. The IoT service requestor can request and use the registered IoT services
	Pipes & Filters	Every IoT system is considered as a black box component that gets as input data, which is then processed by the IoT system, and further provided to the output. IoT systems can use data from other IoT systems and/or provide data to other IoT systems. IoT systems can be configured in multiple different ways but there is no shared state

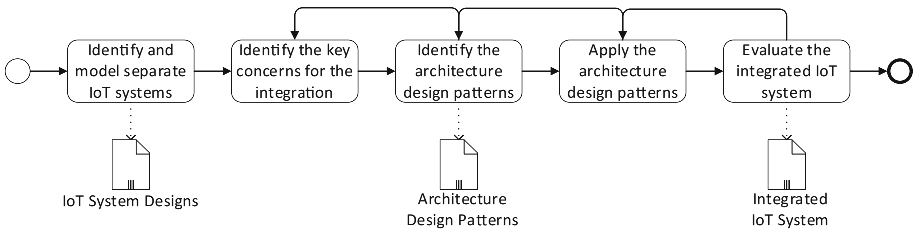

Identify and model separate IoT systems
Identify the key concerns for the integration
Identify the architecture design patterns
Apply the architecture design patterns
Evaluate the integrated IoT system
IoT System Designs
Architecture Design Patterns
Integrated IoT System

Fig. 9. BPMN model representing the steps for integrating multiple IoT systems

The first step in the process is the identification of the individual IoT systems that need to be integrated. The second step in the process includes the identification of the concerns for the integration. This will require checking the needs and the overall purpose for the SoS. Hereby it is important to describe the added value that is created using the integration of these systems. In the third step we identify the patterns that can be used for the integration. These will include the patterns that we have described in the previous section. For this we will adopt the criteria and consider the constraints, the advantages and disadvantages of the corresponding patterns. Once the patterns have been identified we apply and compose the patterns. In principle, more than one pattern can be applied which will require design decision for the composition. In the final step we evaluate the overall architecture of the SoS with respect to the initial objective and the stakeholder concerns in the SoS.

5 Integrating the Smart City Engineering Systems

In order to illustrate our approach, we will consider the smart city engineering case study as defined in Sect. 2. The provided solution is given in Fig. 10. Here it is assumed that Air Quality System and Weather Monitoring System reside at the same location, which are integrated using a smart gateway that realizes the translation of the adopted different protocols in these systems. The Smart Building and Smart Office are also considered in the same location. Hereby, a multi-protocol gateway solution has been used in which multiple gateways for different protocol translations are adopted. The Smart Traffic System, Smart Lighting System, and City Energy Consumption system are considered to be connected over a local area network and communicate through a middleware platform. The middleware provides the translation services and additional network and communication services. All the systems are integrated in the City cloud in which all the cloud integration patterns including SaaS, PaaS, IaaS and data integration is used. This is one solution in which different patterns have been applied to meet the requirements. For different other requirements other patterns can be used to integrate the IoT systems.

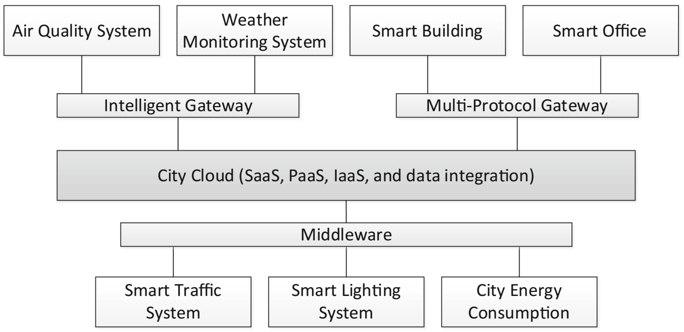


Fig. 10. Example pattern-based integration of smart city engineering systems

6 Conclusion

One of the key challenges in IoT is coping with the heterogeneous set of systems and the integration of these systems in the same communication network. Based on a layered reference architecture for IoT we have indicated that the integration can be at different layers including session layer, cloud layer and application layer. Further we have shown that the integration is typically carried out based on well-defined patterns, that is, generic solutions structures for recurring problems. We have not provided any new integration solution but rather systematically compiled and structured the integration patterns as defined in the literature. Our study has resulted in 15 different patterns which can be used in different combinations. To guide the application of the patterns we have provided a general process represented using the BPMN. The process and the patterns have been successfully applied to a smart city case study. Hence, we have shown that the systematic structuring of the integration patterns is useful for developing IoT systems that need to integrate heterogenous elements. Although we have identified and described the key patterns in the literature, this study could be further extended by considering other patterns. In our future work we will consider other type of IoT reference architectures and based on these enhance the set of patterns that we have described in this paper. Further, we will also consider IoT patterns beyond the integration concern such as security and safety patterns.

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutorials **17**(4), 2347–2376 (2015)
2. Gazis, V., et al.: A survey of technologies for the internet of things. In: IWCMC 2015 - 11th International Wireless Communications and Mobile Computing Conference, pp. 1090–1095 (2015)

3. Pandya, H.B., Champaneria, T.A.: Internet of things: survey and case studies. In: 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO), pp. 1–6 (2015)
4. Palattella, M.R., et al.: Standardized protocol stack for the internet of (important) things. IEEE Commun. Surv. Tutorials **15**(3), 1389–1406 (2013)
5. Köksal, Ö., Tekinerdogan, B.: Feature-driven domain analysis of session layer protocols of internet of things. In: Proceedings of the 2017 IEEE 2nd International Congress on Internet of Things, ICIOT 2017, pp. 105–112 (2017)
6. Yoshikawa, Y., Sato, A., Hirasawa, S., Takahashi, M., Yamamoto, M.: Hitachi's vision of the smart city. Hitachi Rev. **61**(3), 111–118 (2012)
7. Tekinerdogan, B., Celik, T., Köksal, Ö.: Generation of feasible deployment configuration alternatives for Data Distribution Service based systems. Comput. Stand. Interfaces **58**, 126–145 (2018)
8. Bushmann, F., Meunier, R., Rohnert, H.: Pattern-Oriented Software Architecture: A System of Patterns, vol. 1, p. 476. Wiley, New York (1996)
9. Olivieri, A.C., Rizzo, G.: Scalable approaches to integration in heterogeneous IoT and M2M scenarios. In: Proceedings of the 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2015, pp. 358–363 (2015)
10. Diaz-Cacho, M., Delgado, E., Falcon, P., Barreiro, A.: IoT integration on industrial environments. In: Proceedings of the IEEE International Workshop on Factory Communication Systems, WFCS 2015, vol. 2015, July 2015
11. Al-Fuqaha, A., Khreishah, A., Guizani, M., Rayes, A., Mohammadi, M.: Toward better horizontal integration among IoT services. IEEE Commun. Mag. **53**(9), 72–79 (2015)
12. Calbimonte, J.P., Sarni, S., Eberle, J., Aberer, K.: XGSN: an open-source semantic sensing middleware for the web of things. In: CEUR Workshop Proceedings, vol. 1401, pp. 51–66 (2014)
13. Ngu, A.H., Gutierrez, M., Metsis, V., Nepal, S., Sheng, Q.Z.: IoT middleware: a survey on issues and enabling technologies. IEEE Internet Things J. **4**(1), 1–20 (2017)
14. Lomotey, R.K., Pry, J., Sriramoju, S., Kaku, E., Deters, R.: Middleware framework for IoT services integration. In: Proceedings of the 2017 IEEE 6th International Conference on AI and Mobile Services, AIMS 2017, pp. 89–92 (2017)
15. Tekinerdogan, B., Öztürk, K., Doğru, A.: Modeling and reasoning about design alternatives of software as a service architectures. In: Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture, WICSA 2011, pp. 312–319 (2011)
16. Öztürk, K., Tekinerdogan, B.: Feature modeling of software as a service domain to support application architecture design. In: Proceedings of the Sixth International Conference on Software Engineering Advances (ICSEA 2011), pp. 142–148 (2011)
17. Tekinerdogan, B., Öztürk, K.: Feature-driven design of SaaS architectures. In: Mahmood, Z., Saeed, S. (eds.) Software Engineering Frameworks for the Cloud Computing Paradigm. Computer Communications and Networks, pp. 189–212. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5031-2_9
18. Botta, A., De Donato, W., Persico, V., Pescape, A.: On the integration of cloud computing and internet of things. In: Proceedings of the 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014, pp. 23–30 (2014)
19. Croes, E.: Software Architectural Styles in the Internet of Things. Radboud University Nijmegen (2015)
20. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley, New York (2003)

21. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine (2000)
22. David Garlan, M.S.: An Introduction to Software Architecture. World Scientific Publishing Company, New Jersey (1994)
23. Garlan, D., Clements, P., Little, R., Nord, R., Stafford, J.: Documenting software architectures: views and beyond (2010)
24. Clements, P., et al.: Documenting Software Architectures. Style DeKalb IL, p. 592 (2010)
25. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Comput. Surv. **35**(2), 114–131 (2003)

A Novel Wiki Mechanism of Engineering Empirical Knowledge Management

Zuhua Jiang(✉), Ying Huang, and Geng Li

School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
{zhjiang, hycqs0812, reaganli}@sjtu.edu.cn

Abstract. Wiki technology can support the collaborative accumulation of empirical knowledge and embodies the wisdom and experience information from a vast number of users. This paper proposes a novel Wiki mechanism for engineering empirical knowledge management, compares the new Wiki mechanism with the original one, introduces the differences between two mechanisms in the way of knowledge accumulation, and analyses two mechanisms from the maturity of knowledge structure.

Keywords: Engineering empirical knowledge · Media Wiki platform Knowledge management · Expert recommendation mechanism

1 Introduction

The engineering empirical knowledge (EEK) in engineering practice refers to the experience and skills acquired by engineers from both successful and failed cases in the design process, which is regarded as a precious asset of an enterprise. The characteristics of EEK are non-standard organized, highly professional, and difficult to be replicated. Therefore, the difficulty of managing and indexing the EEK items increases, and the threshold of replication of EEK raises. In particular, the requirement of identification and standardization of EEK make it important to improve the knowledge management.

The carrier of EEK mainly includes documents, pictures, videos, etc. Many enterprises' EEK is stored as document in the intranet, or even stored as hard copy in the reference room. According to the characteristics of document-based EEK, this paper defines the structure of EEK as a three-tier system from top to bottom: basic attributes, semantic features, and data contents. The basic attributes include contributors, ID numbers, names, and the time that a piece of EEK was created or last modified. Semantic features are keyword tags that reflect specific contexts and technologies in the engineering field. Data contents contain the characteristics of knowledge content.

Given the rapidly changing knowledge that owns a large number of files in broad areas, it is difficult for the original knowledge management information system to accumulate EEK accurately and efficiently. To describe a piece of EEK without ambiguity or misdirection, the accumulation process of EEK requires many engineers collaborating to refine and verify. Therefore, enterprises need domain experts to

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 34–41, 2018.
https://doi.org/10.1007/978-3-319-94370-1_3

provide professional modification and verification regularly to ensure the EKK of high quality. Additionally, enterprises want to take measures to provide the employees easier access to EEK, even to realize the personalized recommendation of EEK.

The current internet model supported by Web 2.0 technology and user-driven content dramatically improves the way of EEK accumulation and also realizes the process of self-organizational knowledge accumulation on an open platform. This paper studies the mechanism and technology of EEK management, and the accumulation mechanism of EEK mentioned in this paper is based on Wiki platform.

2 Related Works on Wiki

The users of Wiki platform are groups of individuals sharing the same hobbies and interests, or working in the same fields. Thus, they create and accumulate knowledge together in an open and collaborative manner. Wiki technology is one of the representative knowledge management tools, which provides a self-evolution method to obtain the wisdom of the cluster [1]. Therefore, based on Wiki technology, it is convenient for the knowledge-based enterprises or departments to create, edit, browse the knowledge.

The original working mechanism of Wiki can be defined as follows: any user is allowed to create a new structuralized knowledge item, which can be edited and modified by another user with permission of the platform. And Wiki system records the operations such as (create, modify) on the knowledge items and share the new version in the platform.

The major problems are found when Wiki is applied for enterprise knowledge management. (1) As for the content, the quality of knowledge is poor; (2) As for the accumulation, the employees' initiative to contribute knowledge item is not strong; (3) As for the knowledge diffusion, the enterprises don't have an effective and personalized mechanism to recommend knowledge.

Ding et al. pointed out that it is difficult for ordinary Wiki users to find the topics of interest or participate in the creation and contribution. And the expired contents on Wiki is still lack of updating mechanism. Therefore, enterprises still need to strengthen the knowledge accumulation and knowledge maintenance when using wiki [2].

Hwang et al. tried to solve the semantic conflicts when different users edited Wiki and proposed four dimensions of dynamic evolutions of knowledge [3].

Mariano et al. suggested a framework called fortunata to simplify the creation of context-based semantic web. Based on this framework, users can quickly create and edit knowledge items without the needs for advanced computer skills so they can promote knowledge reuse and interaction between creators through two application plugins based on fortunata [4].

To solve the maintenance problem of wiki's engine in the enterprise, Gorka et al. put forward to establish topic knowledge map through keyword nodes on the wiki, and reconstruct the map to complete the maintenance of enterprise's wiki based on context [5].

By combining semantic Wiki with predefined task ontology, defining and maintaining ontology, Baumeister et al. use the semantic wiki in medical emergency decision-making system, so that the decision-making system has an adaptive reasoning process, thus solving the problem [6].

Jung proposed a semantic wiki knowledge management system based on central wiki ontology, which can collect as many organized knowledge resources as possible, and continuously update and maintain the knowledge contexts. It can not only capture knowledge resources in language, but also capture conceptual structure effectively from multiple users, and integrate the captured new resources into knowledge management system [7].

WikiGenes [8, 9] and Knol [10] are two types of Wiki systems that record and identify the identity of knowledge revisers on the Wiki platform and increase the user's confidence in Wiki knowledge by evaluating the authority of the user's contents. In this way, the wiki is applied to the authoritative research field.

Many studies have proposed methods to improve the quality of Wiki contents. And an expert matching algorithm based on forwarding feedback neural network is used to find the most suitable modifier for the quality of Wiki content [1]. The characteristics of knowledge contents and users' behaviors are considered synthetically, and the effect of knowledge quality improvement is analyzed by simulation.

Above all, scholars around the world have combined the process and management concept of knowledge management based on the characteristics of an open source-Wiki platform, aiming at the application of knowledge management in enterprises. Most of the research aimed at the problem that the knowledge quality of Wiki platform is poor, and put forward the idea of applying Wiki to the practice of enterprises' empirical knowledge management under the premise of improving the knowledge quality.

3 The Novel Wiki Mechanism of Engineering Empirical Knowledge Management

3.1 The Structure of the Novel Wiki Mechanism

The original accumulation method of EEK can be described as follows: the engineers produce the documents for both successful and failed cases, which are stored either as electronic documents in the intranet or as hard copies in the reference room in order to be shared [4]. Thus, the conventional method increases the difficulties of inaccessibility and decreases the comprehensibility of EEK; also the EEK cannot be updated in time. In addition, if some incorrect concepts or even logical errors are contained in user's description of experience, the individuals will be less inclined to the use of EEK. Due to the original method, the utilization of enterprises' EEK accumulated in a long-term is rather low. Therefore, the requirement for improving the recommendation mechanism of enterprises' empirical knowledge emerges.

In order to obtain the EEK of high quality and provide the employees with an easier access to the EEK, enterprises are supposed to build a knowledge accumulation mechanism supported by the collaboration of multi-disciplinary knowledge workers, and also be responsible for informing the domain experts to modify and verify this mechanism accurately and timely.

Based on Wiki, this paper studies the mechanism and technology of EEK management. Wiki stress great importance of user participation; thus the community mainly built by its users allows the individuals sharing the same interests and the groups majored in similar specialties to create and accumulate knowledge in a collaborative way (Fig. 1).

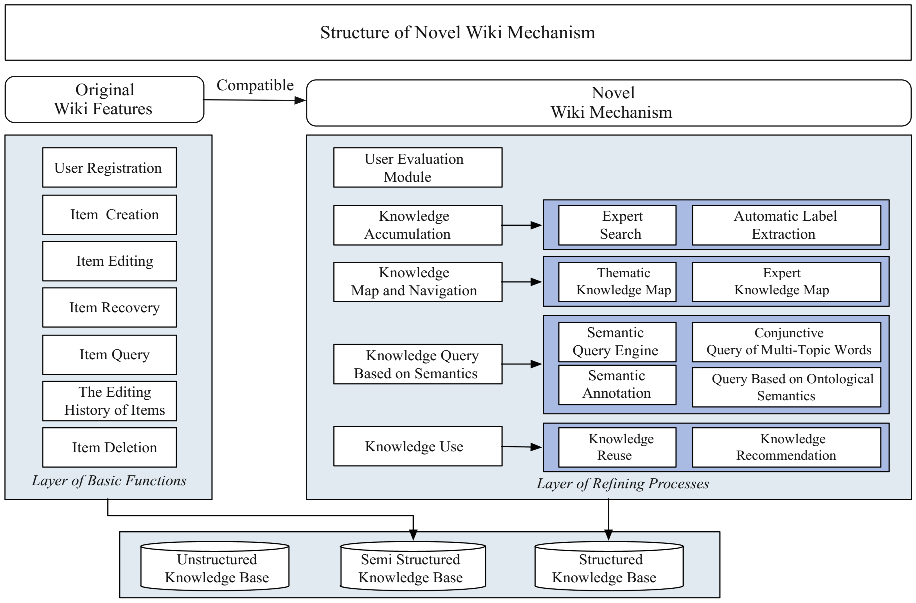


Fig. 1. Structure of the novel Wiki mechanism

Besides the basic functions of the original mechanism, the novel Wiki mechanism can collect the revision suggestion and feedback from users, and reflect their interests in a particular item. Based on that, novel Wiki mechanism can expand other new functions.

3.2 The Characteristics of the Novel Wiki Mechanism

The basic functions of original Wiki mainly focuses on the accumulation processes of primary knowledge, transforming the unstructured EEK into the semi-structured one, which cannot form a closed loop of knowledge accumulation.

The novel Wiki mechanism retains the basic functions of the original Wiki mechanism, improves the knowledge accumulation, and expands a series of new functions, thereby refining the processes of experiential knowledge management and facilitating knowledge sharing. The function modules of novel Wiki mechanism are shown in Fig. 2.

Compared with the original Wiki mechanism, the novel mechanism implements the function mainly on account of the user participation. Specifically, the novel mechanism

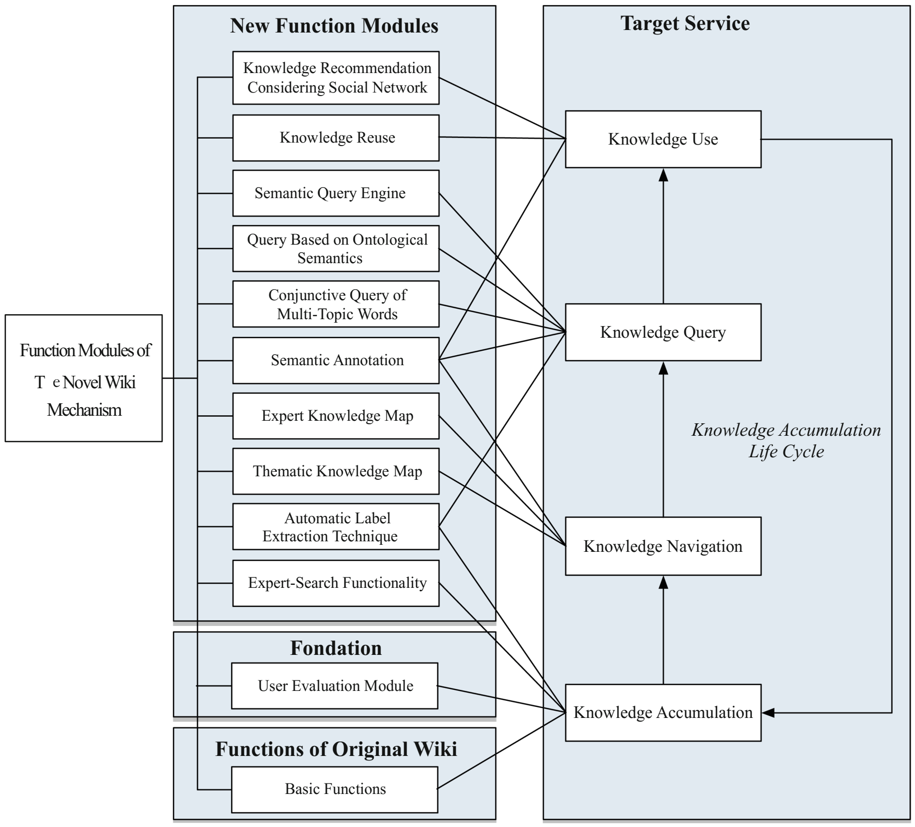

New Function Modules
Knowledge Recommendation Considering Social Network
Knowledge Reuse
Semantic Query Engine
Query Based on Ontological Semantics
Conjunctive Query of Multi-Topic Words
Semantic Annotation
Expert Knowledge Map
Thematic Knowledge Map
Automatic Label Extraction Technique
Expert-Search Functionality
Fondation
User Evaluation Module
Functions of Original Wiki
Basic Functions
Function Modules of T e Novel Wiki Mechanism
Target Service
Knowledge Use
Knowledge Query
Knowledge Accumulation Life Cycle
Knowledge Navigation
Knowledge Accumulation

Fig. 2. Function modules of novel Wiki mechanism

collects the feedback and advice from users by the evaluation module, which can reflect users' interests in a particular item. Therefore, the user evaluation module is regarded as a prerequisite for the novel Wiki mechanism. On the DreamWeaver8.0, the PHP language is used to display the evaluation module as well as realize the statistical functions on the Media Wiki platform.

The goal of novel Wiki mechanism for knowledge management is to achieve better utilization of knowledge. As for the functionality, the novel Wiki mechanism not only includes but also extends the basic operations and functions of the original one. For example, the new functions of knowledge navigation, knowledge query, and knowledge use have been added. With regard to the structure of EEK, different from the original mechanism, the results of knowledge accumulation in the novel Wiki mechanism are structured EEK (Table 1).

On Wiki platform, the enterprise engineers can create and accumulate EEKs, and the lifecycle of a piece of EEK including the knowledge acquisition, organization, improvement and sharing. The scattered EEKs will firstly be organized and classified

Table 1. Comparison between the novel Wiki mechanism and the original one

	Original Wiki mechanism	New Wiki mechanism
Function	Basic function	Basic and advanced function
Essential difference	• Users are unable to send and receive feedback • Level of user participation is low	• Users are able to interact with the system • Level of participation is improved
Knowledge structure	Semi-structured	Structured
Target	• Knowledge accumulation • Knowledge query	• Knowledge accumulation • Knowledge navigation • Knowledge query • Knowledge use
Lifecycle of the knowledge accumulation	No knowledge accumulation life cycle	Completed knowledge accumulation life cycle
Knowledge quality	Low	High
Improvement method	• Users revise the item proactively, or even anonymously • The modified version is not guaranteed to be superior to the previous one	• System invite domain experts to revise the item • The modified version can be guaranteed to be superior to the previous one

into a structured form; each item will undergo several revisions to achieve higher quality; once up to the standard, the piece of EEK will be published and shared.

The new Wiki mechanism is based on the original basis of considering the score of the user group, which is regarded as a statistical measure to judge the quality of the item. However, under this new mechanism, assessing the quality of a piece of EEK is merely based on the rating of the item judged by all the users, which can be easily influenced by artificial factors. Also, there is no guarantee that the quality of the changed item is better than the previous one.

Therefore, the expert recommendation should be promoted by recommending the domain experts to modify the items on Media Wiki platform, thereby enhancing the knowledge quality and improving the accumulation efficiency. The knowledge accumulation mechanism in the New Wiki platform is shown in Fig. 3.

In the new Wiki platform, the history of item creation and modification can be seen by all users on the platform, when a piece of EEK is created and submitted, or an existing item is modified. Additionally, the users who have the permission can edit these existing EEKs for times to ensure the high quality of the item. If the quality of an item has left much to be desired, the expert recommendation system can find the most suitable user to modify and check this item.

Once a user is selected as the domain expert according to the algorithm of this paper, he/she will be recommended to improve the quality of EEK. The process of

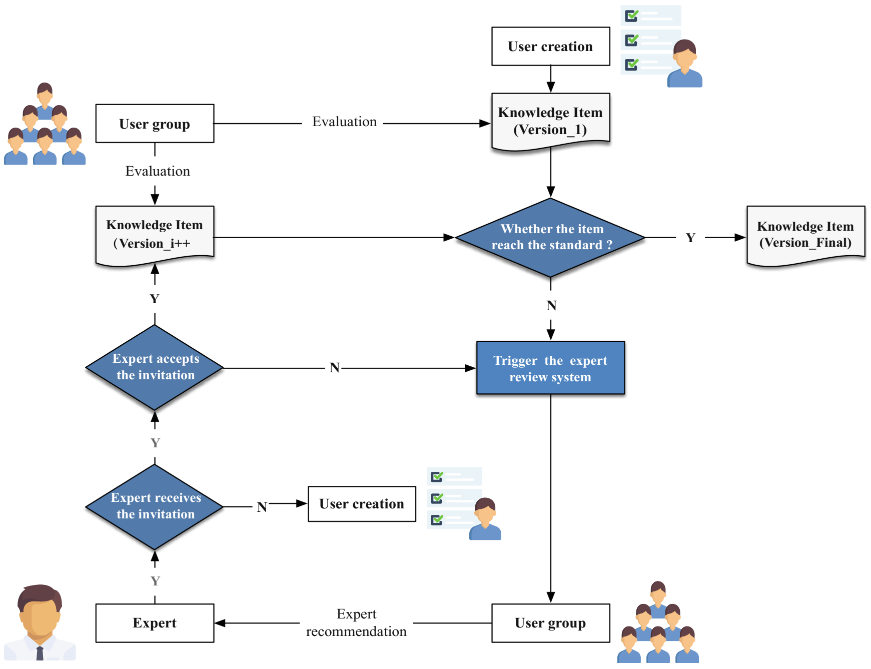

User creation
Knowledge Item (Version_1)
User group
Evaluation
Evaluation
Knowledge Item (Version_i++
Whether the item reach the standard ?
Y
Knowledge Item (Version_Final)
Y
N
Expert accepts the invitation
N
Trigger the expert review system
Y
Expert receives the invitation
N
User creation
Y
Expert
Expert recommendation
User group

Fig. 3. Knowledge accumulation mechanism in the New Wiki platform

"recommending experts to modify items" may be repeated many times until the piece of incomplete EEK reaches its expected quality. Only at this point can the piece of EEK be shared and utilized by enterprise engineers.

Generally, the original Wiki mechanism can realize the processes of knowledge organization, improvement, and sharing, yet there are some differences between the original and novel mechanism with regard to the accumulation of EEK: (1) the knowledge improvement process under the original one can only rely on the user's independent modification, but it is unknown whether the revised version quality is better; (2) the knowledge sharing process of the original Wiki is carried out at the same time as knowledge creation is completed, and it does not guarantee that the quality of knowledge has met the standard; (3) original wiki cannot receive users' feedback, although all users have the permission to modify the knowledge items created by other users, in fact, the users' participation in the platform and their enthusiasm to modify the items are rather low.

4 Conclusion

The mechanism of knowledge accumulation is the prerequisite of the expert recommendation system. On the one hand, in order to take full advantages of the expert recommendation system, it requires a large number of extensive knowledge items as well as high user participation. Open mode and operability are the main characteristics of the Media Wiki, and this management mode only not accelerates the accumulation of EEK, but also standardized the organization of EEK. On the other hand, in order to ensure that the quality of the revised version is better than the previous one, two aspects of user information are worthy of consideration: the correlation between user's knowledgeability and item knowledge, and the user's authority in the user group. The knowledgeability includes user's professional expertise and user's knowledge proficiency. As for the user authority, it can be evaluated by the user's interaction and evaluation on the platform. The accumulation mechanism of EEK on the Media Wiki can also support the establishment of the information system of user's knowledgeability.

Acknowledgements. This research was supported by National Natural Science Foundation of China (No. 70971085, 71271133, 71671113).

References

1. Lykourentzou, I., et al.: CorpWiki: a self-regulating Wiki to promote corporate collective intelligence through expert peer matching. Inf. Sci. **180**(1), 18–38 (2010)
2. Ding, X., et al.: Visualizing an enterprise Wiki. In: Extended Abstracts on Human Factors in Computing Systems, CHI 2007. ACM (2007)
3. Hwang, D., et al.: A semantic Wiki framework for reconciling conflict collaborations based on selecting consensus choice. UCS **16**(7), 1024–1035 (2010)
4. Mariano, R., David, C., Oscar, C.: A contribution-based framework for the creation of semantically-enabled web applications. Inf. Sci. **180**, 1850–1864 (2010)
5. Gorka, P., Oscar, D., Maider, A.: Refactoring affordances in corporate wikis: a case for the use of mind maps. Enterp. Inf. Syst. **9**(8), 7785–7834 (2015)
6. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE: a semantic Wiki for knowledge engineering. Appl. Intell. **35**(3), 323–344 (2011)
7. Jung, J.: Semantic wiki-based knowledge management system by interleaving ontology mapping tool. Int. J. Softw. Eng. Knowl. Eng. **23**(1), 51–63 (2013)
8. Hoffmann, R.: A Wiki for the life sciences where authorship matters. Nat. Genet. **40**(9), 1047–1051 (2008)
9. Benjamin, M.G., Douglas, G.H., Simon, M.L., Warren, A.K., Su, A.I.: Mining the Gene Wiki for functional genomic knowledge. BMC Genom. **12**, 603–616 (2011)
10. O'Leary, D.E.: Wikis: from each according to his knowledge. Computer **41**(2), 34–41 (2008)

Research Track – Smart IoT

Activity Recognition Using Graphical Features from Smart Phone Sensor

Syeda S. Akter(✉), Lawrence B. Holder(✉), and Diane J. Cook(✉)

Washington State University, Pullman, WA 99164, USA
{selina.akter,holder}@wsu.edu, cook@eecs.wsu.edu

Abstract. We develop a graphical feature-based framework that collects data from different kinds of sensor networks, represents the sensor network data as a graph, extracts graphical features from the graph representation, and adds those features to a set of non-graphical features that are typical for the application. Our hypothesis is that the addition of a structural representation and transitional features will improve performance for the corresponding prediction tasks of different networks. We apply our graphical feature-based approach on smart phone GPS sensor data to predict activities performed by phone users. We represent the location category corresponding to each GPS value as a node and movement of users from one GPS location to another as an edge in graph. Then we extract graphical features such as existence of nodes and edges from the graph representation and add them to basic sensor data features coming from the smart phone. We find that using this augmented feature set improves activity recognition accuracy by 7.27% compared to using only basic non-graphical features with feature set augmented with existence of nodes performing the best.

Keywords: Graph mining · Sensor networks · Smart phone sensors · GPS Graphical features · Activity recognition

1 Introduction

We envision a graphical feature-based framework to represent and analyze smart phone data. This framework collects data from sensor networks, uses graph structure to represent movement-related data, and employs selected graphical features to improve corresponding prediction tasks of those sensor networks. In our previous work [1], we designed an algorithm to perform activity recognition from smart home motion sensor data in which we represented the motion sensor data as a graph, extracted graphical features from it, and classified activities performed by residents. The approach achieved significant improvement in classification accuracy compared to the benchmarks in the area. Next [2], we applied the graphical feature based framework on Nokia Smart Phone sensor data, represented GPS information as a graph, extracted and selected useful graphical features, and trained a support vector machine to perform classification for the target variables of gender, age-group and job type. Our approach outperformed most of the benchmarks in the field of demographic prediction from sensor data while using only one type of generic sensor data through leveraging graph structure and movement

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 45–55, 2018.
https://doi.org/10.1007/978-3-319-94370-1_4

patterns. As part of our goal of evaluating the use of graph representations and graph mining to improve performance on recognition and prediction tasks for sensor networks, in this paper we represent smart phone sensor data as a graph to enhance the task of activity recognition.

Activity recognition from sensor data is an active area of research. Automatically detecting human activities from a range of sensors can have broad applications such as remote patient monitoring and medical diagnosis to shorten hospital stay, child and elderly care, emergency assistance both at home and at assisted living, reminder system for people with cognitive disorder and chronic conditions, and recognition of sports and leisure activities in order to increase the lifestyle quality of people. Smart phones are becoming a ubiquitous part of our daily and social life. Incorporation of diverse and powerful sensors such as GPS, gyroscope, accelerometer, light sensors, temperature sensors, and Bluetooth make it a useful tool for activity recognition from smart phone sensor data. Added benefits include unobtrusiveness, low installation cost, ease of use, and ability to monitor inside and outside the home [4].

We hypothesize that representing smart phone sensor information as a graph and adding transitional features to basic non-graphical sensor data features will improve activity recognition performance. We used the dataset collected through an activity learning mobile app called Activity Learner (AL) designed by Aminikhanghahi et al. [3] We chose GPS information to represent as a graph, extract graphical features, and use these graphical features along with typical features based on non-graphical sensor data such as accelerometer and gyroscope to predict activities. The experiment shows that inclusion of graphical features significantly improves performance over typical non-graphical features with nodes features providing the best results. Analyzing the confusion matrix shows that the addition of edges may improve the performance for some activities. Using only selected features has the potential to improve the performance with the addition of edges.

2 Previous Works

The first and second age of the internet connected people to the internet. The third age of the internet connects not only people but also things to the internet [13]. According to CISCO, 50 billion things will be connected to the internet in 2020 [14]. These things range from very small and static devices (e.g., RFIDs) to large and mobile devices (e.g., vehicles). According to Khalil et al. [14], the role of Wireless Sensor Networks (WSN) in the IoT is that of a virtual skin that connects the things with the network and with each other, makes them aware of their surroundings, and shares this information with other things in order to make informed decisions. In [15], Zeng and Min approached how to build such a complicated system and presented a systematic design framework for IoT Enabled Systems.

A smartphone is light, inexpensive, user-friendly, multipurpose, and portable device that can be easily used by people in their daily lives and includes all technologies needed for IoT [16]. Smartphones are equipped with a range of built-in sensors such as accelerometers, motion sensors, position sensors, environmental sensors, microphone, and camera providing images and videos. Special sensors measuring health vital signs, such as body temperature, ECG value, blood glucose level, stress level, body fat percentage, heart rate, etc., can be integrated into the smartphone. In [16], Khaddar et al. termed smart phone as the brain of the IoT world as smart phones come with a variety of connectivity technologies such as NFC, Bluetooth, Wi-Fi, and cellular allowing it to connect and interact with other devices and sensors. Next we discuss related works that used accelerometer, GPS and other sensor information from smart phones for the task of activity recognition.

Bouchard et al. [5] discussed different types of spatial features such as distance, position, shape and gesture. Their experiments with a user's raw latitude and longitude as features showed improved accuracy for activity recognition. Aminikhanghahi et al. [3] developed an algorithmic approach called Thyme for adapting prompt timing based on the context of the user's activity. In the first phase of Thyme, an Activity Learner (AL) smart phone app collects smart phone sensor data and learns a mapping from raw sensor data to activity labels through the use of classification algorithms Linear Support Vector Classification (SVC), Naïve Bayes (NB), K-Nearest Neighbour (KNN), Decision Tree (DT) and Random Forest (RF) with Random Forest resulting in the highest 82% accuracy with leave-one-out validation. This result is for training and testing done for each user separately; for combined users the average accuracy is 78%. Along with extracting standard signal processing features from raw sensor data, AL also computes higher-level information about the five-second data window, including heading change rate (percentage of points in the sequence that change direction), stop rate (percentage of points in the sequence that exhibit a significant drop in velocity), overall trajectory from start to finish of the data sequence, and normalized distance to the user's mean location. They also have used a cost-sensitive classifier by adding weight to each data point during training to help the learning algorithm handle the imbalanced class problem. Our method shows that use of generic graphical features can improve prediction accuracy without use of well thought-out, application-specific features or adding special methodologies to handle an imbalanced class distribution.

Liao et al. [8] approached location-based activity recognition from GPS traces. They train a conditional random field to iteratively re-estimate significant places and activity labels. Initial activity estimation consists of a sequence of locations and the most likely activity performed at that location, and these estimates are inferred by applying belief propagation. A set of significant places are extracted from this activity estimate and then used to classify individual activities again based on whether they belong to a significant place. This process is repeated until the activity sequence does not change. Their proposed method is evaluated on a fairly small dataset for four participants. The data does contain complete GPS traces for seven days of data per person, but the method is prediction task dependent. The method provided good results on this particular dataset, but it still needs to be explored whether this method can be applied in general to different sensor networks and different prediction tasks.

Chetty et al. [6] and Garcia-Ceja et al. [7] both performed activity recognition from three-dimensional accelerometer data and showed improved accuracy in predicting simple [6] and complex activities [7]. We would like to show that using a graph representation of spatial features and extracting transitional features from the representation improve the activity recognition accuracy even further over these non-graphical accelerometer-based features.

3 Graphical Feature Based Framework

We collect location-related sensor information, namely latitude and longitude from smart phone sensor data in order to predict activities performed by smart phone users. We converted this geolocation information to location categories. We used OpenStreetMap (OSM), which is a map of the world built by a community of mappers that contribute and maintain data about roads, trails, cafes, railway stations and much more. [11] It is time consuming to access existing world maps that are available only online to probe about categories of each geo-location. To address this issue, we used a tool called Nominatim [12] through which we can download OSM data, import to a local database, and perform reverse geo-coding using another tool, geopy [9] for large amounts of geo-location data locally in significantly less time.

Whenever any user visits a place, we represent that location category as a node in the graph. When a user moves from one location to another, we add an undirected edge between the corresponding two nodes in the graph. From this graph representation, we extract graphical features that we present in Table 1. In Table 1, we also show some basic features that we can directly obtain from smart phone sensors as a typical set of features. We add selected graphical features to this basic feature set and feed this augmented feature set to a classifier for predicting activities. The workflow for activity classification from GPS data based on the Graphical Feature Based Framework is shown in Fig. 1.

Table 1. List of features

Types of features	List of features	Raw sensor data that are used
Basic features (statistics applied to raw sensor data)	Max, min, sum, mean, median, standard deviation, median absolute deviation, zero crossings, mean crossings, interquartile range, coefficient of variation, skewness, kurtosis, simple moving average (SMA), log SMA, power, autocorrelation	Accelerometer data across x, y, z-axis, rotation across x, y, z-axis, yaw, pitch, roll, month, day of week, hour, minute and seconds
Graphical features (generated from graph representations)	Existence of nodes, existence of edges, and existence of both nodes and edges	Latitude and longitude

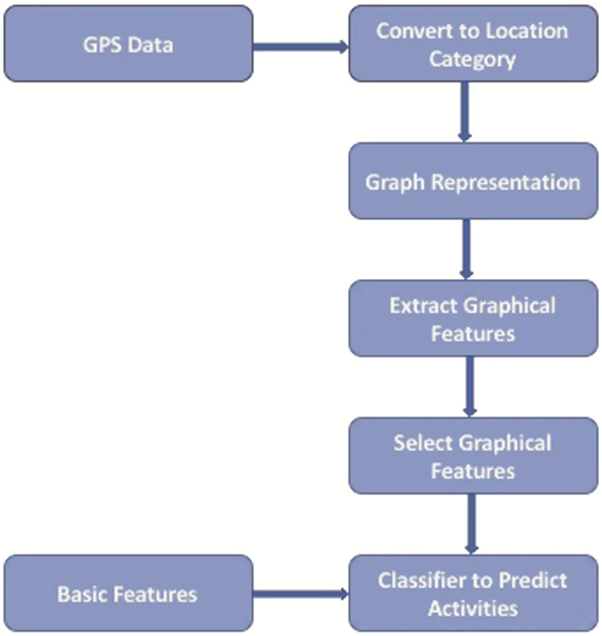

GPS Data
Convert to Location Category
Graph Representation
Extract Graphical Features
Select Graphical Features
Basic Features
Classifier to Predict Activities

Fig. 1. Graphical feature-based framework for activity prediction

4 Computational Details and Results

4.1 Dataset

Aminikhanghahi et al. [3] designed a mobile app that collects 5 s of data at intervals specified by the user on iOS and Android platforms. The app, called Activity Learner (AL), collects the following 16 types of raw sensor data: accelerometer data across x, y and z-axis, rotation across x, y and z-axis, yaw, pitch, roll, course, speed, horizontal accuracy, vertical accuracy, latitude, longitude and altitude along with time and date information (month, day of week, hour, minute and seconds). The total number of instances in this dataset is 17933. We convert the GPS sensor data to location category and represent each location category as a node in the graph representation. Some example location categories are library, parking, motel, cycle way, hotel, park, supermarket, place of worship, school, bar, restaurant, bus stop, and road.

We construct one undirected graph for each activity performed by users in this dataset. The app AL guesses and periodically queries the participant about their current activity to obtain the labels of these activities. The user can agree to the predicted activity through the AL interface. Alternatively, they can proactively provide input about the activity they are currently performing. [3] In the dataset we analyze, there are a total of 214 unique activities performed by 47 participants. In many cases they labeled the same activities with different names or spellings. We map similar activities with slightly different names or with different spellings into a general name. For example, mapping

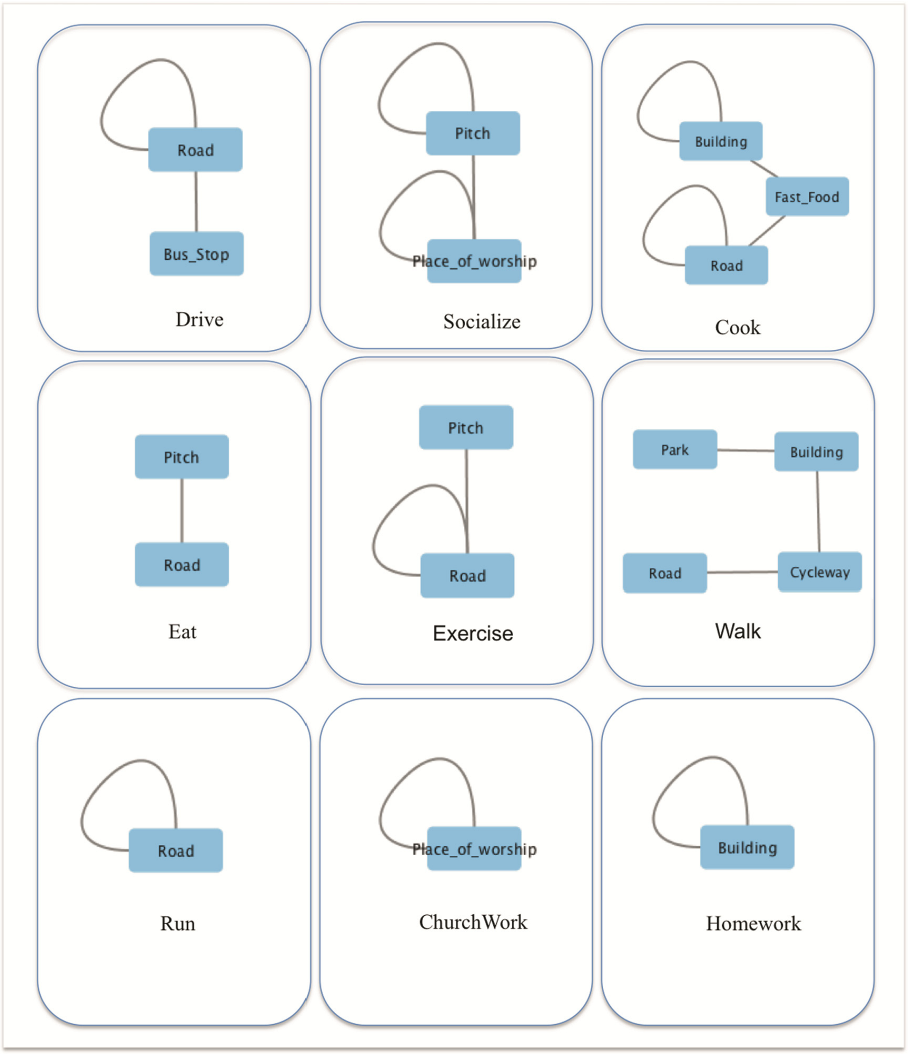


Fig. 2. Graph representations of different activities

both ‘Driving’ and ‘Drive’ to one activity ‘Drive’, mapping ‘Errands’, ‘RunErrands’, ‘Store’, ‘Walmart’ to ‘Errands’, etc. AI comes with an option to provide this activity mapping file. After activity mapping, there are 116 unique activities in the dataset. In Fig. 2, we show example graph representations for instances of different activities out of these 116 activities such as ‘Drive’, ‘Socialize’, ‘Cook’, ‘Eat’, ‘Exercise’, ‘Walk’, ‘Run’, ‘ChurchWork’ and ‘HomeWork’. After representing each activity as a graph, the total number of unique nodes is 42 and the total number of unique edges including self-loops is 94. We use this set of unique nodes and edges as our graphical feature set and add it to the basic feature set for each instance. For each activity, we construct a graph and when a node exists in this user-activity graph, we mark that corresponding feature as 1, otherwise we mark it as 0. Similarly, for each edge in this user-activity graph, we mark the corresponding feature in our feature set as 1, otherwise it is marked as 0. In this way we construct and add the graphical feature set for each instance to the basic feature set. We tried Decision Tree, Random Forest, Gradient Boost, Extra Tree, Bagging and Ada Boost for classifying with Basic Features. Extra Tree Classifier performed the best among these six classifiers. Initially, we select the 100 best features for inclusion in the model. To classify activities, we apply the ExtraTreeClassifier using the SelectKBestFeatures feature selection method with Mutual Information as the scoring function for the features.

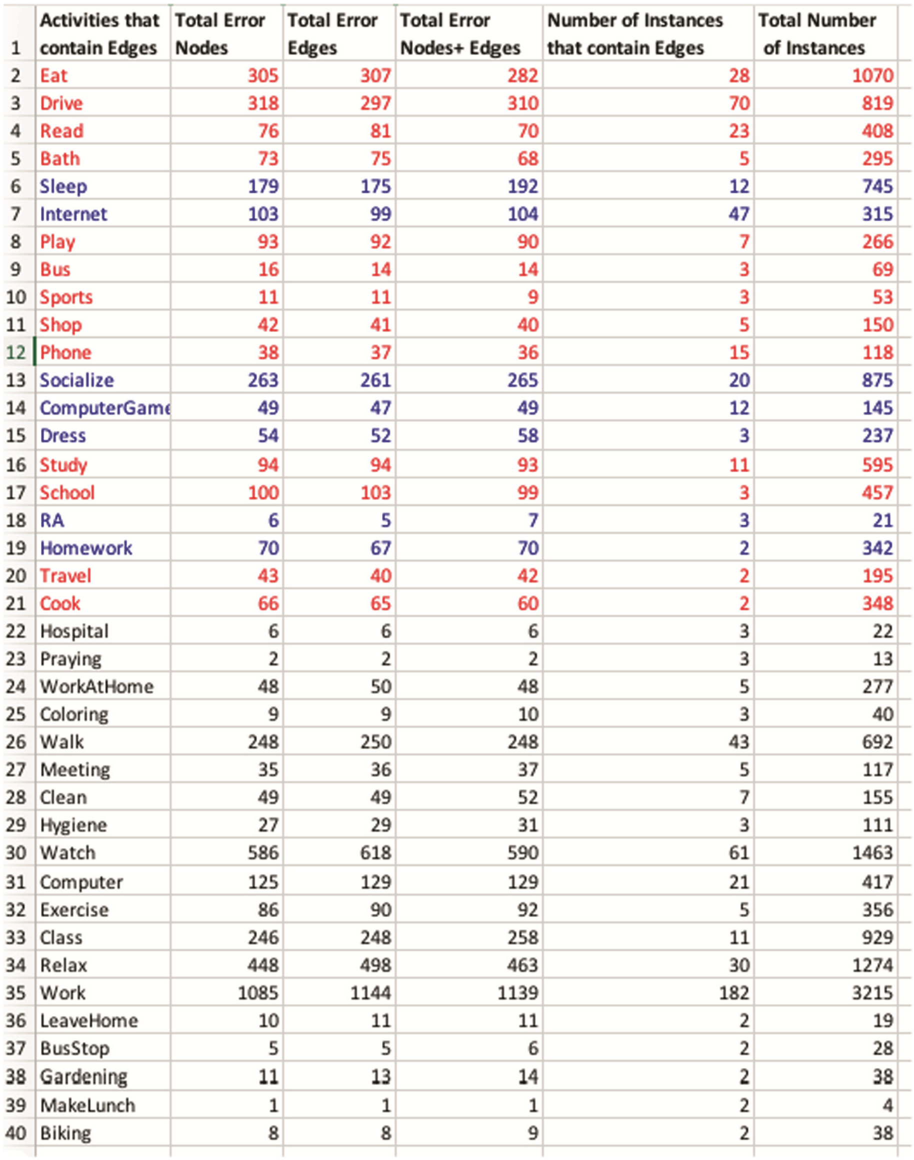
4.2 Result

In Table 2, we compare performance of graphical features with basic features for classifying activities. As demonstrated in Table 1, basic features include basic statistical computation (max, min, sum, mean, median, standard deviation, median absolute deviation, zero crossings, mean crossings, interquartile range, coefficient of variation, skewness, kurtosis, simple moving average (SMA), log SMA, power, autocorrelation) of accelerometer data across x, y, z-axis, rotation across x, y, z-axis, yaw, pitch, roll, and date information (month, day of week, hour, minute and seconds). Graphical Features include existence of nodes, existence of edges, and existence of both nodes and edges.

Table 2. Accuracy in percentage for basic features vs graphical features for six classifiers

Features	Basic	Basic + Nodes	Basic + Edges	Basic + Nodes + Edges
Decision tree	45.27	46.25	**46.64**	46.18
Random forest	50.39	**50.96**	49.56	49.92
Extra tree	51.45	<u>**55.19**</u>	53.86	53.73
Gradient boost	33.91	**37.17**	37.05	36.69
Bagging	50.93	51.90	52.19	**52.31**
Ada boost	**16.53**	**16.53**	16.13	16.13

We apply six different classifiers, namely, Decision Tree, Random Forest, Extra Tree, Gradient Boost, Bagging and Ada Boost with 3-fold cross-validation to classify 116 activities performed by 47 participants. The best performing feature set along a row (for each classifier) is boldfaced. We observe that adding only nodes or only edges or both nodes and edges improve the result compared to only basic features except Ada Boost. Decision Tree provides the best result with combination of basic features augmented with edges. For Bagging, combination of basic, nodes and edges as feature sets is the best performer. Random Forest and Extra Tree produce the best result with basic and nodes feature sets. However, the best result among all classifiers and feature sets is produced by Extra tree with basic feature set augmented with nodes which is 7.27% improvement over using only basic features.



1	Activities that contain Edges	Total Error Nodes	Total Error Edges	Total Error Nodes+ Edges	Number of Instances that contain Edges	Total Number of Instances
2	Eat	305	307	282	28	1070
3	Drive	318	297	310	70	819
4	Read	76	81	70	23	408
5	Bath	73	75	68	5	295
6	Sleep	179	175	192	12	745
7	Internet	103	99	104	47	315
8	Play	93	92	90	7	266
9	Bus	16	14	14	3	69
10	Sports	11	11	9	3	53
11	Shop	42	41	40	5	150
12	Phone	38	37	36	15	118
13	Socialize	263	261	265	20	875
14	ComputerGame	49	47	49	12	145
15	Dress	54	52	58	3	237
16	Study	94	94	93	11	595
17	School	100	103	99	3	457
18	RA	6	5	7	3	21
19	Homework	70	67	70	2	342
20	Travel	43	40	42	2	195
21	Cook	66	65	60	2	348
22	Hospital	6	6	6	3	22
23	Praying	2	2	2	3	13
24	WorkAtHome	48	50	48	5	277
25	Coloring	9	9	10	3	40
26	Walk	248	250	248	43	692
27	Meeting	35	36	37	5	117
28	Clean	49	49	52	7	155
29	Hygiene	27	29	31	3	111
30	Watch	586	618	590	61	1463
31	Computer	125	129	129	21	417
32	Exercise	86	90	92	5	356
33	Class	246	248	258	11	929
34	Relax	448	498	463	30	1274
35	Work	1085	1144	1139	182	3215
36	LeaveHome	10	11	11	2	19
37	BusStop	5	5	6	2	28
38	Gardening	11	13	14	2	38
39	MakeLunch	1	1	1	2	4
40	Biking	8	8	9	2	38

Fig. 3. Total errors in confusion matrix for activities that contain edges

To investigate the reason for performance decrease when edges are added to feature set, we look at the confusion matrix for activities that contain edges. There are 51 activities in the dataset where transitions between locations occurred and hence these activities contain edges. We remove 12 activities that have only one instance that contain edges. Figure 3 shows the list of these activities that have at least two instances in the dataset that contain edges. It also shows the total number of instances for each activity and the number of instances where edges occurred. We find false positive, false negative values for each of these activities from the confusion matrix and compute the total error (false positive + false negative).

We compute the total error for three cases: nodes added to basic features, edges added to basic features, nodes and edges added to basic features and present this information in Fig. 3 in columns ‘Total Error for Nodes’, ‘Total Error for Edges’ and ‘Total Error for Nodes and Edges’. Among them 20 activities that are colored in red and blue in Fig. 3 demonstrated reduced error using either edges or both nodes and edges. Blue colored activities showed reduced error with addition of edges and red colored activities showed reduced error with use of both nodes and edges. 19 activities that are colored black in Fig. 3 showed increased error with the use of nodes and edges.

As a next step, we tried some basic feature selection techniques to test whether it may improve the result. Through feature selection, we may be able to keep and use only useful features and eliminate features that had a negative effect on accuracy. This may help with the problem of overfitting as well. We present the result of our initial experiment of feature selection with Extra Tree classifier in Table 3. We used the k-best features selector from scikit-learn based on the mutual information criteria to select 100-best features from the basic feature set, nodes added to the basic feature set, edges added to the basic feature set and nodes and edges added to the basic feature set. As demonstrated in Table 3, better overall accuracy is achieved with the addition of edges only and both nodes and edges compared to basic features. The basic features with edges showed the best performance. In future work, we plan to experiment with varying the value of k and with other feature selection methods to see whether the overall accuracy can be improved further.

Table 3. Accuracy in percentage for basic features vs graphical features with feature selection

Features	Accuracy in percentage
Basic	57.65
Basic + Nodes	57.13
Basic + Edges	**57.67**
Basic + Nodes + Edges	57.50

5 Discussion

Compared to a non-graphical typical feature set, graphical feature sets provide improvements over non-graphical features with nodes performing the best among all graphical feature sets. However, adding edges decreased the performance in some cases.

In the current dataset all sensor events were collected at one second intervals. Both basic and graphical features of each instance are computed from sensor events collected in a five second window. Each five second window has an activity label provided by the user. Most activities continue past one window. We already obtained better performance for some activities using transitional information available in only a five-second window. Some activities may benefit from a larger window in order to allow for more transitions in the activity graph, but that would require an additional data collection effort in the future.

Also, there may be noise both in labeling activities and in determining location categories. Activity labels are dependent on getting correct information from users about the activity they are doing. There are many instances of activities in the dataset during which no geolocation data is collected. While extracting the location category from raw latitude and longitude values using the Nominatim Database, for some location categories "None" is returned, indicating unknown location category. The ability to get more accurate location category information may improve predictions.

We are predicting classes across behavior of all users in this experiment. User-wise activity prediction may give better result because movement patterns can vary from user to user.

6 Conclusion

We present a Graphical Feature based Framework with the goal to improve prediction tasks for different sensor networks by representing the sensor networks in graph form, extracting graphical features from these graphs, and adding those features to the typical set of features for the task, to be fed to classifiers. In this work, we apply this framework to smart phone sensor data for the task of activity recognition. The results demonstrate that adding spatial and transitional features improves the activity recognition accuracy compared to typical non-graphical and non-spatial feature sets. Without feature selections, nodes perform the best. However, analyzing the confusion matrix shows that adding edges can decrease total error in many activities. Initial investigation with feature selection shows that use of feature selection may help through eliminating extra and non-helpful edges and also may help with overfitting due to the large number of graphical features. We plan to try other feature selection methods and find the set of helpful selected graphical features. In the future, using larger window sizes to help extract more transitions for ongoing activities can be used to further improve the performance of adding graphical features. User-wise activity prediction along with graphical features will reflect an individual's movement pattern and hence may improve activity prediction. Along with the use of edges and the combination of nodes and edges, larger sub-graphs can be considered as part of the graphical features to discriminate among activities.

References

1. Akter, S.S., Holder, L.B.: Activity recognition using graphical features. In: 13th International Conference on Machine Learning and Applications (ICMLA), Michigan (2014)
2. Akter, S.S., Holder, L.B.: Using graphical features to improve demographic prediction from smart phone data. In: Proceedings of the 2nd International Workshop on Network Data Analytics, Chicago (2017)
3. Aminikhanghahi, S., Fallahzadeh, R., Sawyer, M., Cook, D.J., Holder, L.B.: Thyme: improving smartphone prompt timing through activity awareness. In: 16 th International Conference on Machine Learning Applications, Cancun (2017)

4. Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., Havinga, P.: Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: a survey. In: 23rd International Conference on Architecture of Computing Systems (ARCS), Hannover (2010)
5. Bouchard, K., Holder, L.B., Cook, D.J.: Extracting generalizable spatial features from smart phones datasets. In: AAAI Workshop: Artificial Intelligence Applied to Assistive Technologies and Smart Environments, Phoenix (2016)
6. Chetty, G., White, M., Akther, F.: Smart phone based data mining for human activity recognition. Proc. Comput. Sci. **46**, 1181–1187 (2015)
7. Garcia-Ceja, E., Brena, R.: Long-term activity recognition from accelerometer data. Proc. Technol. **7**, 248–256 (2013)
8. Liao, L., Fox, D., Kautz, H.: Location-based activity recognition. In: Advances in Neural Information Processing Systems (2006)
9. Python Software Foundation, "geopy1.11.0". https://pypi.python.org/pypi/geopy
10. Hall, M.A.: Correaltion-based feature selection for machine learning (1999)
11. Haklay, M., Weber, P.: Openstreetmap: user-generated street maps. IEEE Pervasive Comput. **7**(4), 12–18 (2008)
12. https://nominatim.openstreetmap.org/
13. Manrique, J.A., Rueda-Rueda, J.S., Portocarrero, J.M.: Contrasting internet of things and wireless sensor network from a conceptual overview. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 252–257. IEEE, December 2016
14. Khalil, N., Abid, M.R., Benhaddou, D., Gerndt, M.: Wireless sensors networks for internet of things. In: 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6. IEEE, April 2014
15. Zeng, J., Min, J.: A systematic framework for designing IoT-enabled systems. Serv. Trans. Internet Things **1**, 23–31 (2017). https://doi.org/10.29268/stsc.2017.0002
16. El Khaddar, M.A., Boulmalf, M.: Smartphone: the ultimate IoT and IoE device. In: Smartphones from an Applied Research Perspective. InTech (2017)

Algebraic Service Composition for User-Centric IoT Applications

Damian Arellanes(✉) and Kung-Kiu Lau

School of Computer Science, The University of Manchester,
Manchester M13 9PL, UK
{damian.arellanesmolina,kung-kiu.lau}@manchester.ac.uk

Abstract. The Internet of Things (IoT) requires a shift in our way of building applications, as it is aimed at providing many services to society in general. Non-developer people require increasingly complex IoT applications and support for their ever changing run-time requirements. Although service composition allows the combination of functionality into more complex behaviours, current approaches provide support for dealing with one IoT scenario at a time, as they allow the definition of only one workflow. In this paper, we present DX-MAN, an algebraic model for static service composition that allows the definition of composite services that encompass multiple workflows for run-time scenarios. We evaluate our proposal on an example in the domain of smart homes.

Keywords: IoT applications · Algebraic service composition
Scalability · Exogenous connectors · End-user development · DX-MAN

1 Introduction

The Internet of Things (IoT) promises a new era in which every physical world object and all living entities will be interconnected through innovative distributed services. Thus, the scale of IoT applications will go beyond human mind expectations.

IoT applications are mainly aimed at providing value to society in general. People with no development expertise are able to control, manage and customize their own applications [6,11]. For this reason, IoT requires a shift in our way of building applications: a developer must be able to create a generic application that encompasses multiple scenarios, in order to accommodate as much as possible the run-time user requirements. Thus, users will be able to autonomously choose a behaviour among the alternative ones.

Although some scenarios are simple, many others require the combination of a huge number of services. Hence, service composition is crucial for building complex IoT applications. However, designing a generic composite that accommodates multiple IoT scenarios is not trivial, since user requirements may vary from one scenario to another. Moreover, the dynamism of IoT applications causes an increase in the number of possible scenarios as the number of services grows.

Current composition approaches do not fulfill the demands of IoT applications that require user-centric compositions of a huge number of services. This is because their semantics allows the definition of only one workflow at a time. Thus, tackling a new scenario would require an entirely new workflow or the modification of an existing one. This paper proposes DX-MAN, an algebraic model for static IoT service composition, which enables the development of composite services that encompass many workflows so as to accommodate multiple run-time scenarios.

The rest of the paper is structured as follows. Section 2 presents the related work. Section 3 presents a motivating example. Section 4 describes our model for algebraic IoT service composition. Section 5 presents examples to show the feasibility of our model. Section 6 presents a discussion of our results as well as challenges related to this research. Finally, Sect. 7 presents the conclusions and the future work.

2 Related Work

Current composition approaches include orchestration, nested orchestrations, choreography, data flows and nested data flows.

Orchestration [13,22] and *choreography* [8,26,27] have been used for many years in Service Oriented Architecture (SOA) and are now gaining attention for IoT applications. Orchestration defines a central coordinator for the invocation of operations in services. In order to eliminate the performance bottleneck caused by the central coordinator or to support multiple administrative domains, a number of sub-workflows can be defined in *nested orchestrations* [7,16]. On the other hand, a choreography realizes a workflow through the collaborative and decentralized exchange of messages between the services involved. Regardless of the underlying mechanics, (nested) orchestration and choreography allow the definition of only one workflow at a time.

A data-driven workflow, or *data flow*, allows the combination of data streams from different IoT sources. It is basically a graph where nodes represent computation and edges represent data paths: a node receives data, then performs some computation and finally passes data on. Although data flows are increasingly popular for IoT applications thanks to the emergence of *mashups* [5,14,24], they allow the creation of one workflow at a time; and this is also true in *nested data flows* [12].

Like orchestration, data flows are considered as exogenous composition mechanisms because a workflow is defined with no knowledge of the services involved [18]. Reo [19,23] is a declarative language for data flows, which also has the notion of exogenous connectors. Unlike DX-MAN, the composition of two Reo connectors yields a more complex connector, but not a service. Of course, a Reo connector can be transformed into a service, but this would require an extra step as it is not part of Reo semantics. More importantly, Reo allows the creation of one workflow at a time, like other data flow approaches.

Automatic service composition [1,9,10,22] consists of discovering, selecting and combining services at run-time, in order to construct a workflow that fulfills a given specification [17,25,28]. It does not provide new composition semantics, but it is built on top of existing ones: data flows [9], orchestration [22], choreography [1], or any combination thereof [10]. Therefore, automatic service composition also allows the definition of only one workflow at a time.

Other approaches [11,15,29] do not provide any composition constructs, because they are only *frameworks* or *software tools* for end-user development. Some of them provide support to define only one straightforward workflow at a time, typically a sequential one.

3 Motivating Example

To motivate our approach, this section introduces a running example. The example is in the domain of smart homes and is based on the case study presented in [22]. It consists of two independent services shown in Fig. 1: (i) a *Windows* service for opening and closing windows and (ii) a *Climate* service to turn dehumidifiers on and off. For simplicity, we only show two operations per service. The distribution of services over IoT nodes is out of the scope of this paper.

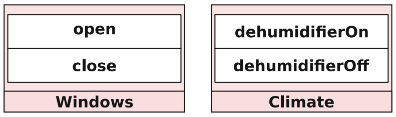


Fig. 1. Services involved in our motivating example.

Imagine a user requires different workflows at run-time depending on climatic conditions. Automatic service composition is the best approach currently available to generate workflows on the fly. However, it allows the definition of only one workflow at a time, since it is built on top of existing composition semantics.

For example, on a sunny day the user may want to open the windows and turn the dehumidifier off. The workflow depicted in Fig. 2 is generated by an automatic composition mechanism so as to accommodate this user requirement. Suppose it suddenly starts raining so the user decides to close the windows and turn the dehumidifier on. Thus, the automatic composition mechanism would need the generation of the entirely new workflow shown in Fig. 3.

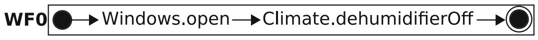


Fig. 2. Workflow for a sunny day.

Of course, the user can express all his needs in a single step. The workflow generated by the automatic composition mechanism for this scenario is shown in

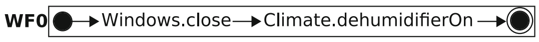


Fig. 3. Workflow for a rainy day.

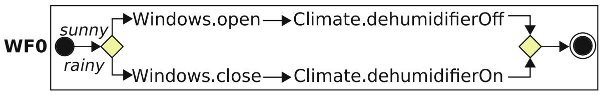


Fig. 4. Workflow for a sunny and rainy day.

Fig. 4, and it includes the scenarios depicted in Figs. 2 and 3. If the user changes his mind again, a new workflow would be needed.

It might seem that nested workflows are an alternative solution to this problem, as shown in Fig. 5. However, their composition semantics also allow the definition of only one workflow at a time. Thus, individual nested workflows are required whenever user requirements change.

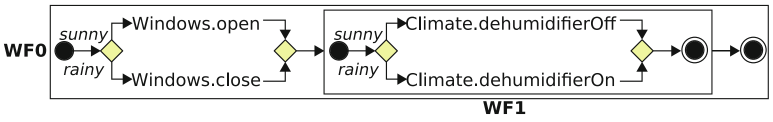


Fig. 5. Nested workflows for a sunny and rainy day.

4 DX-MAN

We propose DX-MAN (Distributed X-MAN) [3] to mitigate the impact of change of run-time user requirements. It is a multi-level service composition model [4] inspired by algebra and the X-MAN component model [20,21], where services and exogenous connectors are first-class entities. Figure 6 illustrates the DX-MAN constructs which we further describe in this section.

A DX-MAN service is a distributed software unit that exposes a set of operations through a well-defined interface. It can be deployed in any IoT node such as a Cloud, an edge device or a sensor. Distribution semantics are out of the scope of this paper, but we refer the reader to another paper on that matter [3].

An atomic service is the most primitive kind of DX-MAN service. It is formed by connecting an invocation connector with a computation unit (see Fig. 6). The invocation connector provides access to the operations implemented in the computation unit, and the computation unit is not allowed to call other computation units. The atomic service interface has all the operations implemented in the computation unit. Formally, an atomic service $AS \in \mathbb{S}$, where $\mathbb{S}$ is the type of services, is a set of operations defined as follows:

$$AS = \{op_i \mid i \in \mathbb{N}\} \tag{1}$$

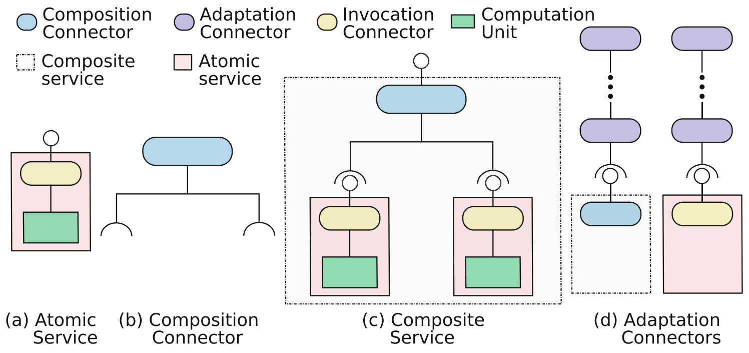


Fig. 6. DX-MAN constructs.

Exogenous connectors are architectural elements that define explicit control flow and encapsulate a network communication mechanism, in order to coordinate the execution of an IoT application from outside services. So, services are unaware they are part of a larger piece of behaviour.

Our notion of algebraic composition is inspired by algebra where functions are composed hierarchically into a new function of the same type, using the operator $\circ$. The resulting function can be further composed with other functions, yielding a more complex one.

Algebraic service composition means that a composition connector is used as an operator ($\circ$) *to hierarchically compose* $\geq$1 *services, atomic or composite, into a (composite) service. As it is constructed from sub-service interfaces, the composite interface has all the sub-service operations. Like an algebraic function, a composite service is a generalization of a particular problem because it implicitly contains multiple workflows whose formation is constrained by the composition connector being used.* Formally, a composite service $CS \in \mathbb{S}$, where $\mathbb{S}$ is the type of services, is a set of services defined as follows:

$$CS = \{S_i \mid i \in \mathbb{N} \wedge S \in \mathbb{S}\} \tag{2}$$

DX-MAN provides composition connectors for sequencing, branching and parallelism. A sequencer connector (SEQ) allows the invocation of sub-service operations in a user-defined order. A sub-service operation can be associated with $\geq$0 orders. Sub-service operations with no given order are never invoked, and when no sub-service operation has an order assigned, an empty workflow is thrown at run-time. Any sub-service operation can be invoked any number of times within a workflow. Thus, a sequencer connector defines a composite service that contains an infinite number of sequential workflows. Figure 7 shows an example of a composite service constrained by a sequencer connector.

A selector (SEL) connector chooses the sub-service operations to be invoked, according to user-defined conditions which are evaluated concurrently. A sub-service operation can be associated with exactly zero or one condition. Sub-service operations with no condition associated are never invoked. When no

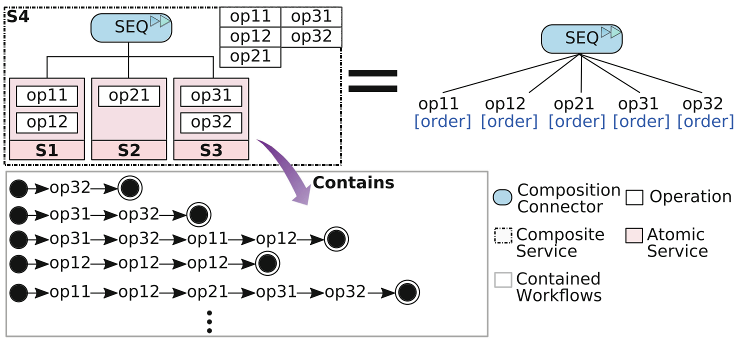


Fig. 7. Sequencer connector.

sub-service operation has a condition associated or all conditions hold false, an empty workflow is thrown at run-time. A selector connector defines a composite service that contains $2^{|\bigcup_{i=1}^{|CS|} S_i|}$ workflows. For example, Fig. 8 shows a composite service that contains 32 possible branching workflows as there are five sub-service operations. We do not show all possible workflows because of space constraints.

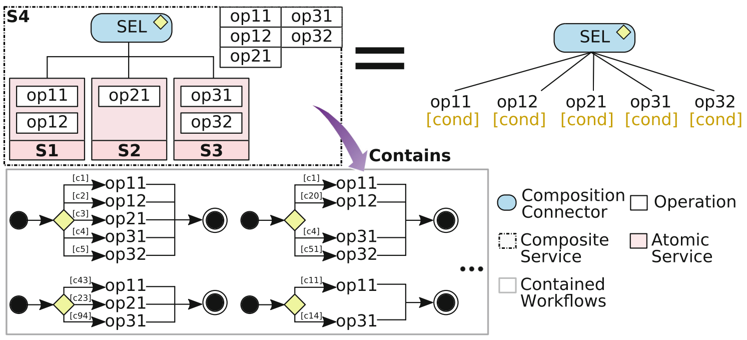


Fig. 8. Selector connector.

A parallel connector (PAR) allows the parallel invocation of sub-service operations. A sub-service operation can be invoked multiple times in parallel within a workflow; to do so, the user needs to specify the number of jobs for each sub-service operation. When no sub-service operation has jobs assigned, an empty workflow is thrown at run-time. A parallel connector defines a composite service that contains infinite parallel workflows. Figure 9 shows a composite service constrained by a parallel connector.

Although they do not compose services, adapters can also constrain workflows by applying additional control structures over an individual service. A looping adapter can be used to iterate a number of times over a sub-workflow, while a

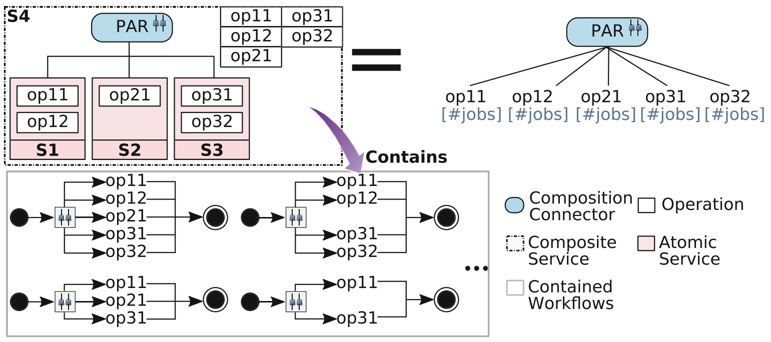


Fig. 9. Parallel connector.

user-defined condition holds true. A guard adapter invokes a sub-workflow only if a user-defined condition is true.

Selection trees are abstract templates that allow the selection of workflows at run-time. They are implicitly created from a composite service during design-time. Figures 7, 8 and 9 show examples of selection trees for a sequencer connector, selector connector and parallel connector, respectively. In the next section, we present examples that show how to choose workflows using selection trees.

5 Examples

This section presents two examples of using DX-MAN for user-centric IoT applications. The first example describes how a one-level composite service accommodates the run-time user requirements described in our motivating example (see Sect. 3). The second example describes how a two-level composite service enables more complex workflows by hierarchically composing services. For both examples, we distinguish between developers and users. Developers design, deploy and execute DX-MAN services, while users choose the workflow they need at run-time. To do so, we developed a platform prototype [2].[1] Composite services and selection tree instances are defined using JavaScript Object Notation (JSON) documents. Due to space constraints and clarity, we omit the JSON documents used for the examples. Instead, we show a graphical representation of composite services and selection trees.

5.1 One-Level Composition

At design-time, the developer uses a sequencer connector *SEQ0* to compose the services *Windows* and *Climate* into a composite service *C0* which contains infinite sequential workflows (see Fig. 10). At run-time, the user only chooses the workflow he needs from the composite *C0*.

[1] https://gitlab.cs.man.ac.uk/mbaxrda2/DX-MAN.

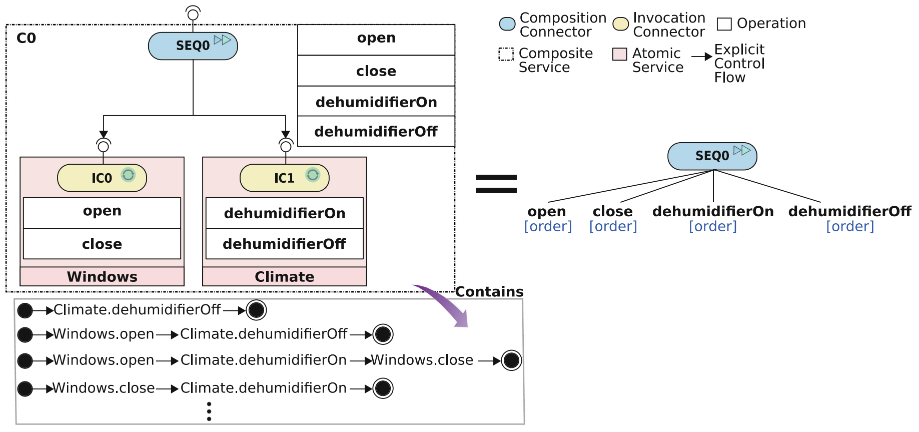


Fig. 10. DX-MAN architecture for the scenarios of our motivating example.

For example, on a sunny day the user chooses the workflow depicted in Fig. 2 in Sect. 3, by assigning execution order as shown in Fig. 11. Suddenly, it starts raining so the user chooses the workflow illustrated in Fig. 3 in Sect. 3, by assigning execution order as shown in Fig. 12. Thus, there is clearly no need of creating an individual workflow or a new composite service whenever user requirements change, but only defining an instance of the respective selection tree.

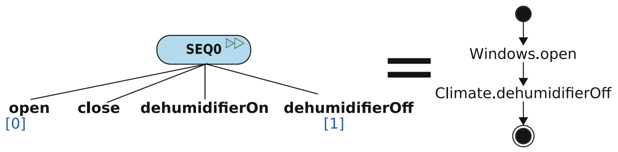


Fig. 11. Choosing a workflow for a sunny day.

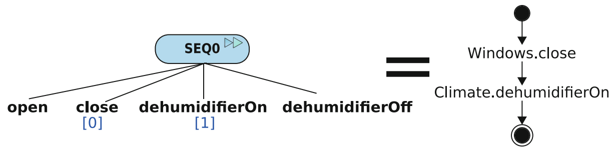


Fig. 12. Choosing a workflow for a rainy day.

As another example, on a cold day the user may want to only close the windows. To do so, the user assigns the execution order shown in Fig. 13. Again, without the need of creating an individual workflow or a new composite service.

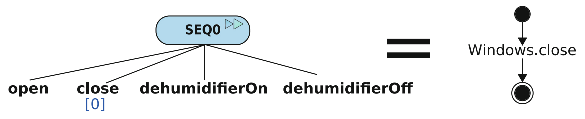


Fig. 13. Choosing a workflow for a cold day.

5.2 Two-Level Composition

In the previous subsection, we presented a one-level composition as a solution for our motivating example. Nevertheless, DX-MAN allows more complex workflows by hierarchically composing services into multi-level structures.

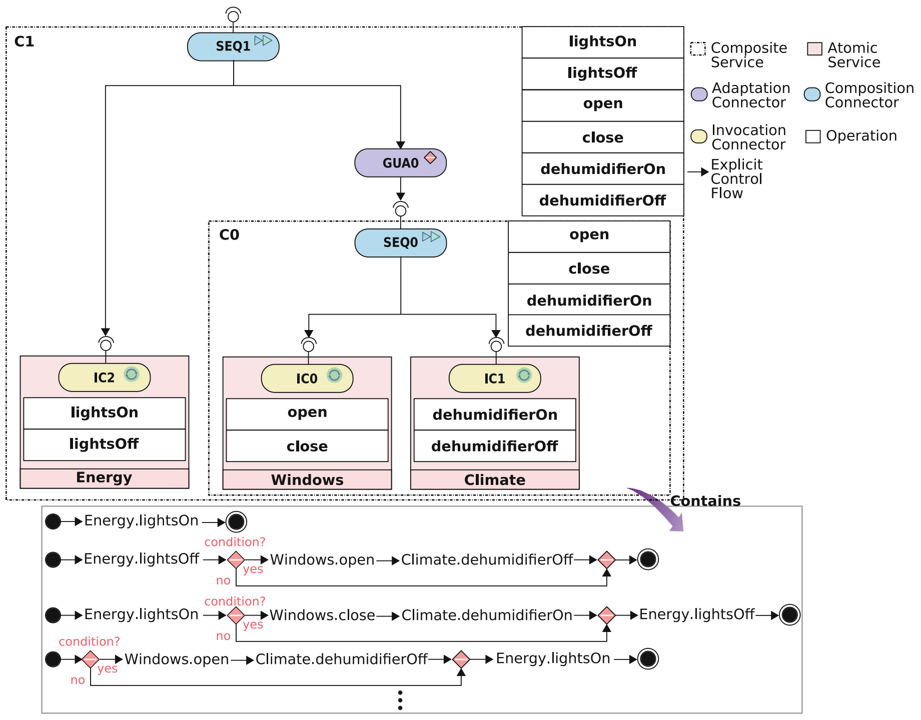


Fig. 14. Two-level DX-MAN architecture.

Suppose there is an atomic service *energy* for turning lights on and off. The developer uses a sequencer *SEQ1* to compose the existing composite *C0* and the atomic service *energy* into a new composite *C1*. He also adds a guard adapter to invoke *C0* if a user-defined condition holds true. Figure 14 shows the resulting two-level DX-MAN composition, and Fig. 15 shows the respective selection tree.

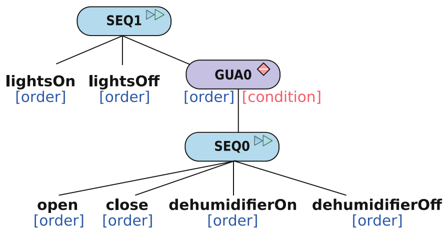


Fig. 15. Resulting tree from the two-level DX-MAN architecture.

Unlike nested workflows, a DX-MAN composite service enables an entirely new world of alternative workflows as shown in Fig. 14. For example, the user may want the following workflow before sleeping: turn the lights off and, if it all the lights were successfully turned off, close the windows and turn the dehumidifier off. To choose that workflow from *C1*, the user assigns the execution order shown in Fig. 16. A condition is represented as a JSON document and specifies the name of the parameter, the operator (only "==" and "!=" are supported at this stage) and the value to compare with. For example, the condition for *GUA0* would be {*"parameterName":"lightsStatus", "operator":"==", "value":"off"*}.

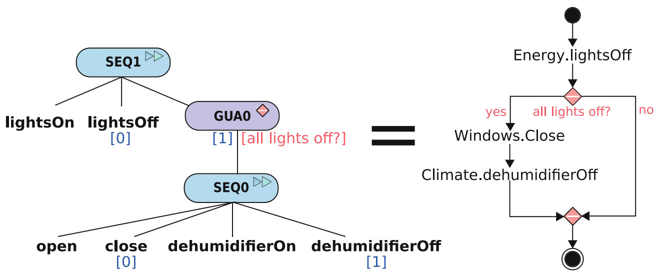


Fig. 16. Choosing a workflow before sleeping.

6 Discussion

We presented a preliminary version of DX-MAN in another paper [3]. In this paper, we present additional semantics that allows the selection of workflows at run-time. We also present a comparison between DX-MAN and current composition approaches in the context of user-centric IoT applications.

Developers can use current composition semantics (e.g., orchestration or choreography) to define a workflow that accommodates as many run-time scenarios as possible. However, it is impossible for them to predict all possibilities during the design-phase and, even if they try, the resulting workflow would potentially require a lot of computing resources, because it becomes larger, more complex and cumbersome as the number of possible scenarios increases. This is in fact highly likely in IoT applications where the number of available services is always growing.

Although automatic service composition mechanisms could mitigate the ever changing run-time user requirements, their overhead increases exponentially as the number of available services grows [9]. Thus, they are only suitable for a small number of services and straightforward workflows. A large number of services would require a user to wait hours (or even days) before getting a responsive application. For that reason, current automatic composition mechanisms are not yet ready to tackle the imminent scale of user-centric IoT applications.

Even though it is focused on static composition, DX-MAN provides semantics to enable multiple workflows at run-time. In some cases, it may be necessary to change a DX-MAN composition at run-time so as to support even more scenarios. This can be done using automatic composition or dynamic reconfiguration techniques on top of DX-MAN semantics.

In contrast to other composition approaches, DX-MAN does not entail much composition overhead, since there is no need to deploy individual workflows, but only a composite service from which a workflow is chosen (not created) at run-time. In fact, IFTTT or any similar tool can be used on top of DX-MAN to choose a workflow, according to a set of user-defined rules.

At this point, the reader may notice that there are clearly many challenges for future work. We discuss some of them below.

Automatic Service Composition. We believe that our work opens new opportunities for automatic service composition, as this technique can be applied on top of DX-MAN semantics. Services (with all their implicit workflows) can be composed to find more possible workflows at run-time, rather than attempting to construct only one workflow at a time. We are particularly interested in decentralized approaches for automatic service composition, since decentralization is crucial to unleash the full potential of IoT.

Self-adaptive Behaviour. Self-adaptive mechanisms can built on top of DX-MAN to autonomically choose a workflow out of the alternative ones, e.g., based on QoS requirements. A DX-MAN composite service can mutate so as to accommodate changes in the context. However, changing a composition at run-time is not trivial, specially when the response time is critical for the user.

Workflow Validation at Run-Time. As a sequencer connector currently allows the invocation of any operation in any order, there is a need for avoiding invalid sequences (e.g., opening a window three consecutive times). At this stage, it is up to the user to decide which workflows are valid.

Concurrency. DX-MAN only provides support for basic concurrency in parallel invocations. However, many IoT scenarios require active services that can be operating on their own (e.g., using a scheduler). Extending DX-MAN with concurrent capabilities requires further investigation.

Data flows at Run-Time. In DX-MAN, data flow is orthogonal to control flow. Current DX-MAN semantics only allow one data flow for every possible workflow within a composite service. For that reason, at this stage DX-MAN can only be used in scenarios where data flow is unimportant, e.g., actuator triggering. In more complex IoT scenarios, different data flows per workflow will be required. Nevertheless, determining data flows at run-time according to user requirements is a challenging task.

7 Conclusions and Future Work

Users may want to customize their own IoT applications. However, current composition approaches allow the definition of only one workflow at a time. This is not desirable for IoT applications where run-time user requirements are always changing. Although automatic composition is a promising technique to tackle this problem, it is still based on existing composition semantics, thus allowing the definition of only one workflow at a time. For that reason, we need to accommodate run-time user requirements as much as possible during the design phase. In this paper, we presented DX-MAN as a solution for this issue.

The algebraic nature of DX-MAN is suitable to mitigate the impact of change in run-time user requirements. We showed with a small example how DX-MAN allows the definition of (general) composite services that contain multiple workflows. Users only choose the workflow they need out of the alternative ones, rather than resort to the cumbersome and inefficient task of creating individual workflows at run-time.

In the short term, we plan to extend the DX-MAN semantics, in order to enhance the flexibility of composite services. Additionally, as workflows are chosen using JSON documents at this stage, we would like to allow the selection of workflows in a more interactive way (e.g., using a visual tool or voice commands). We are in fact currently working on a visual Web editor to fill this gap.

We believe that DX-MAN opens new research directions to tackle the challenges that user-centric IoT applications pose. Given the novelty of DX-MAN, in what creative ways can you define composite services during the design-phase, in order to accommodate as much as possible run-time user requirements?

References

1. Ahmed, T., Tripathi, A., Srivastava, A.: Rain4Service: an approach towards decentralized web service composition. In: IEEE International Conference on Services Computing (SCC 2014), pp. 267–274 (2014)
2. Arellanes, D., Lau, K.K.: D-XMAN: a platform for total compositionality in service-oriented architectures. In: 7th IEEE International Symposium on Cloud and Service Computing (SC2 2017), pp. 283–286 (2017)
3. Arellanes, D., Lau, K.K.: Exogenous connectors for hierarchical service composition. In: 10th IEEE International Conference on Service Oriented Computing and Applications (SOCA 2017), pp. 125–132 (2017)

4. Arellanes, D., Lau, K.K.: Analysis and classification of service interactions for the scalability of the internet of things. In: IEEE International Congress on Internet of Things (IEEE ICIOT 2018) (2018)
5. Blackstock, M., Lea, R.: WoTKit: a lightweight toolkit for the web of things. In: 3rd International Workshop on the Web of Things (WoT 2012), pp. 1–6 (2012)
6. Brambilla, M., Umuhoza, E., Acerbis, R.: Model-driven development of user interfaces for IoT systems via domain-specific components and patterns. J. Internet Serv. Appl. **8**(1), 14 (2017)
7. Chafle, G., Chandra, S., Mann, V.: Decentralized orchestration of composite web services. In: 13th International World Wide Web Conference (WWW 2004), pp. 134–143 (2004)
8. Cherrier, S., Ghamri-Doudane, Y., Lohier, S., Roussel, G.: D-LITe: distributed logic for internet of things services. In: International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing (ITHINGSCPSCOM 2011), pp. 16–24 (2011)
9. Ciortea, A., Boissier, O., Zimmermann, A., Florea, A.M.: Responsive decentralized composition of service mashups for the internet of things. In: 6th International Conference on the Internet of Things (IoT 2016), pp. 53–61 (2016)
10. Dar, K., Taherkordi, A., Vitenberg, R., Rouvoy, R., Eliassen, F.: Adaptable service composition for very-large-scale Internet of Things systems. In: 11th Middleware Doctoral Symposium (MDS 2011), pp. 1–2 (2011)
11. Ghiani, G., Manca, M., Paternò, F., Santoro, C.: Personalization of context-dependent applications through trigger-action rules. Trans. Comput. Hum. Inter. (TOCHI) **24**(2), 14:1–14:33 (2017)
12. Giang, N.K., Blackstock, M., Lea, R., Leung, V.C.M.: Developing IoT applications in the fog: a distributed dataflow approach. In: 5th International Conference on the Internet of Things (IOT 2015), pp. 155–162 (2015)
13. Glombitza, N., Ebers, S., Pfisterer, D., Fischer, S.: Using BPEL to realize business processes for an internet of things. In: 3rd International Conference on Ad-Hoc Networks and Wireless (ADHOCNETS 2011), pp. 294–307 (2011)
14. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the Web of Things. In: Internet of Things (IOT 2010), pp. 1–8 (2010)
15. IFTTT: IFTTT (2018). https://ifttt.com/
16. Jaradat, W., Dearle, A., Barker, A.: Towards an autonomous decentralized orchestration system. Concurr. Comput. Pract. Exp. **28**(11), 3164–3179 (2016)
17. Jatoth, C., Gangadharan, G.R., Buyya, R.: Computational intelligence based QoS-aware web service composition: a systematic literature review. IEEE Trans. Serv. Comput. **10**(3), 475–492 (2017)
18. Johnston, W.M., Hanna, J.R.P., Millar, R.J.: Advances in dataflow programming languages. ACM Comput. Surv. **36**(1), 1–34 (2004)
19. Jongmans, S.S., Santini, F., Sargolzaei, M., Arbab, F., Afsarmanesh, H.: Orchestrating web services using Reo: from circuits and behaviors to automatically generated code. Serv. Oriented Comput. Appl. **8**(4), 277–297 (2014)
20. Lau, K.K., Tran, C.M.: X-MAN: an MDE Tool for Component-Based System Development. In: 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, pp. 158–165 (2012)
21. Lau, K.K., Velasco Elizondo, P., Wang, Z.: Exogenous connectors for software components. In: 8th International Conference on Component-Based Software Engineering, pp. 90–106 (2005)
22. Lee, C., Wang, C., Kim, E., Helal, S.: Blueprint flow: a declarative service composition framework for cloud applications. IEEE Access **5**, 17634–17643 (2017)

23. Palomar, E., Chen, X., Liu, Z., Maharjan, S., Bowen, J.: Component-based modelling for scalable smart city systems interoperability: a case study on integrating energy demand response systems. Sensors **16**(11), 1810 (2016)
24. Persson, P., Angelsmark, O.: Calvin – merging cloud and IoT. Procedia Comput. Sci. **52**, 210–217 (2015)
25. Sheng, Q., Qiao, X., Vasilakos, A., Szabo, C., Bourne, S., Xu, X.: Web services composition: a decade's overview. Inf. Sci. **280**, 218–238 (2014)
26. Taušan, N., Markkula, J., Kuvaja, P., Oivo, M.: Choreography in the embedded systems domain: a systematic literature review. Inf. Softw. Technol. **91**, 82–101 (2017)
27. Thuluva, A., Bröring, A., Medagoda Hettige Don, G.P., Anicic, D., Seeger, J.: Recipes for IoT applications. In: 7th International Conference on the Internet of Things (IOT 2017) (2017)
28. Wang, S., Zhou, A., Yang, M., Sun, L., Hsu, C.H., Yang, F.: Service composition in cyber-physical-social systems. IEEE Trans. Emerg. Top. Comput. (2017)
29. Zapier: Zapier — The easiest way to automate your work (2018). https://zapier.com/

QoS-Aware Resource Allocation for Mobile IoT Pub/Sub Systems

Raphael Gomes[1](✉), Georgios Bouloukakis[2,3], Fábio Costa[4], Nikolaos Georgantas[2], and Ricardo da Rocha[5]

[1] Instituto Federal de Goiás - Câmpus Goiânia, Goiânia, Brazil
raphael.gomes@ifg.edu.br

[2] Inria, Paris, France
{georgios.bouloukakis,nikolaos.georgantas}@inria.fr

[3] Donald Bren School of Information and Computer Sciences, University of California, Irvine, USA
gboulouk@ics.uci.edu

[4] Instituto de Informática, Universidade Federal de Goiás, Goiânia, Brazil
fmc@inf.ufg.br

[5] Instituto de Biotecnologia, Universidade Federal de Goiás, Catalão, Brazil
rcarocha@acm.org

Abstract. IoT applications are usually characterized by large-scale demand and the widespread use of mobile devices. Similarly, performing interaction among application and system components in a decoupled and elastic way, and enforcing Quality of Service (QoS) usually also become issues. Hence, paradigms such as pub/sub on top of cloud resources represent a suitable strategy for application development. However, management of QoS-aware resource allocation for pub/sub systems remains challenging, especially when system peers connect in an intermittent way. In this paper, we propose a new approach for resource allocation focusing on end-to-end performance in face of peers' disconnections. We evaluate and demonstrate the benefits of our approach using simulations. QoS enforcement was achieved in almost all scenarios, and we have shown that our approach can help reasoning about efficient resource allocation.

Keywords: Publish/Subscribe middleware · Resource allocation
Mobile connectivity · Quality of Service

1 Introduction

The widespread use of smart mobile devices paved the way for the development of smartphone applications that can be accessed anywhere anytime. Additionally, such applications access embedded sensors/actuators for providing environmental-related information, opening new business and market opportunities in the so-called *Internet of Things* (IoT) [4]. However, mobile devices may connect and disconnect intermittently for energy saving purposes, and may be forcefully disconnected due to connectivity issues in underlying wireless network. Accordingly, asynchronous messaging patterns, such as the *Publish/Subscribe* (pub/sub) paradigm, enable event-buffering during disconnected periods and event-delivery during connected periods. In particular, pub/sub provides loosely coupled interaction in both time and space among publishing data sources and subscribing data sinks [15]. As a result, several industry standards have adopted pub/sub as part of their interfaces.

Existing middleware protocols such as MQTT [6] and AMQP [30], as well as tools and technologies such as RabbitMQ [27], Kafka [3] and JMS [26] follow the pub/sub paradigm. To support the development of effective applications under the constraints found in the IoT (i.e., intermittent connectivity, obsolete data feeds, etc.), pub/sub protocols provide several *Quality of Service* (QoS) features [10]. These features aim to enable application developers to tune an application by configuring its QoS metrics at different levels such as end-to-end response times and delivery success rates.

Despite the fact that the QoS features of pub/sub may enable timely data delivery, this can be affected (as well as other QoS metrics) by the hardware resources used to deploy software components such as message brokers [19]. Besides, although mobile devices have increasing processing, storage, and communication capabilities, they are not able to handle all sorts of tasks in a proper way [8], making imperative the use of external resources. The use of *Cloud computing* is the most promising solution to enable the deployment of scalable and reliable software components. This further enhances the loosely coupled interaction between publishers and subscribers, also improving reliability as deployed message brokers in the cloud are always connected to receive/forward events.

Different solutions have been proposed for the problem of resource allocation under specific QoS constraints in order to support the development of message brokers in the Cloud [16,25,28]. However, these solutions mainly focus on the satisfaction of local QoS requirements (e.g., employing load balancing strategies for individual jobs) lacking support for end-to-end non-functional properties. Proposed strategies for end-to-end QoS enforcement in pub/sub systems [7,12,19] have limited scope (e.g., changing on transport protocols) or present limitations such as incompatibility with existing middleware protocols. Additionally, in pub/sub systems, the subscribers' intermittent connectivity affects the existing resource allocation algorithms. For instance, message brokers with local subscribers that disconnect for long periods, demand additional processing and buffer resources. Thus, the design of QoS-aware resource allocation algorithm for mobile IoT applications remains an important challenge as it is necessary to satisfy elastic demands while keeping costs under control.

To deal with the above limitations, in this paper we present a QoS-aware approach for the allocation of resources for IoT applications. We consider IoT applications that run over a pub/sub system with mobile publishers/subscribers, along with message brokers that are deployed in the cloud. We formalize the resource allocation problem and estimate response times by relying on *Queueing Network Models* [20]. Our approach provides system designers with resource allocation estimates at design time enabling the achievement of accurate end-to-end runtime behavior. We evaluate and demonstrate the benefits of our approach using simulations based on both probability distributions and parameters derived from real-world workload traces. The core contributions of the paper are:

1. Formalization of the resource allocation problem in mobile pub/sub systems.
2. A model for the response time from publishers to subscribers, which is obtained by connecting queuing models that represent cloud resources and peers' disconnections.
3. A cost-effective resource allocation strategy based on the splitting of end-to-end QoS thresholds and on resource selection in a multi-cloud environment.
4. A simulation environment for the evaluation of QoS properties in pub/sub systems with awareness of the connectivity status of communicating peers.

The rest of this paper is organized as follows: Sect. 2 introduces the pub/sub system model and the problem of allocating resources for message brokers. In Sect. 3, a formalization of this problem is presented. Our resource allocation approach is provided in Sect. 4, which takes into account the peers' connectivity. Finally, the validation and evaluation of our approach is considered in Sect. 5, followed by conclusions and future work in Sect. 6.

2 System Model

Mobile IoT applications are typically deployed on resource-constrained devices with intermittent network connectivity. To support the deployment of such applications, the pub/sub interaction paradigm is often employed, as it decouples mobile peers in both time and space. In a pub/sub system, multiple peers interact via an intermediate *broker* entity – publishers produce *events* characterized by a specific *filter* to the broker. Subscribers subscribe their interest for specific filters to the broker, who maintains an up-to-date list of subscriptions.

To support distributed applications spanning a wide-area, the pub/sub system has to be implemented as a set of independent, communicating brokers, forming the *broker overlay*. As depicted in Fig. 1, in such architectures [5,9], peers can access the system through any broker that becomes their home broker. Then, based on the peers' input the pub/sub system performs several processes: *(i)* subscriptions are spread to a subset of existing brokers through a *subscription partitioning* process; *(ii)* published events correspond to subsets of subscribers determined through a *matching* process; *(iii)* the produced events are delivered to all the matched subscribers through an *event routing* process. Regardless of the event routing algorithm in use, produced events pass through a specific path of brokers towards the subscribers. For example, in Fig. 1 the produced events pass through brokers b_2, b_5 and, finally, b_7, which is s_1's home broker.

Building an IoT application over a pub/sub infrastructure, requires the selection of an appropriate protocol (e.g., MQTT). Such a protocol enables peers to access the broker overlay and push/receive events. Additionally, to create the

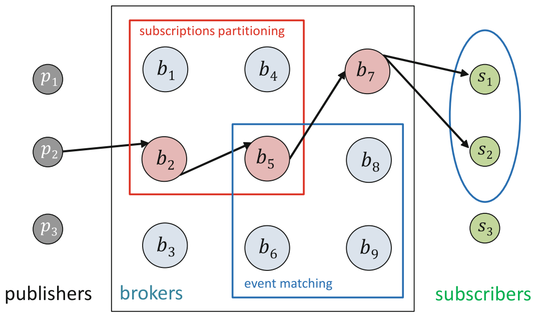


Fig. 1. Pub/Sub system.

pub/sub broker overlay, it is essential to select the corresponding *message broker* implementation (e.g., ActiveMQ, VerneMQ, HiveMQ, etc.). Every message broker has different capabilities, such as: compatibility with different protocols, support for clustering (i.e., forming a broker network), provision of performance features, etc. Finally, an important design-time decision is the deployment of each broker to an appropriate machine. A common tactic of pub/sub developers is to deploy or assume the deployment of brokers in the cloud.

To perform cloud resource allocation, different technologies (e.g., Virtual Machines - VMs [28] and containers [25]) can be used, each with its strengths and weaknesses. In this paper, we use VMs as the unit of allocation, although we keep the solution as generic as possible to admit other alternatives. Terms such as *resource* and *VM* are used interchangeably in the text. For simplicity, we assume that each resource is used by a single broker via on-demand resource instantiation. Accordingly, the pricing of resource allocation is estimated taking the one-hour utilization of an instance of the given resource type.

In IoT scenarios, system deployment usually aims to achieve co-location of *things* and allocated resources. This strategy is used to decrease the communication demand on devices with hardware and power limitations. Another motivation is that whilst data-processing speeds have increased rapidly, the bandwidth to carry data to and from datacenters has not increased at the same speed [29]. For these reasons, in our approach resource allocation is limited to resources found inside a single region. For cloud vendors such as Amazon [2], which splits a region into multiple availability zones, we adopt a single availability zone. Hence, we neglect data propagation delays inside pub/sub system when evaluating QoS.

Based on the above, several questions arise: *(i)* what is the amount of resources that a developer should allocate for an IoT application that runs over a pub/sub system? *(ii)* Can we ensure specific end-to-end response times between publishers and subscribers by relying on the resulting resource allocation? *(iii)* What is the cost to achieve such end-to-end QoS? *(iv)* Does the intermittent connectivity of peers affect the cost of resource allocation?

In the next section, we delineate our approach using a formal notation for the problem with respect to the questions raised above.

3 QoS-Aware Resource Allocation for Pub/Sub Systems

The lack of a direct producer/consumer relationship in pub/sub interactions makes the definition and enforcement of any end-to-end QoS policy very hard [14]. Additionally, the peers' intermittent connectivity makes the design of a fully decoupled QoS-driven pub/sub system that can scale to different dimensions without under- or over-provisioning of resources even more challenging.

Resource allocation is carried out by translating QoS constraints into infrastructure level parameters, which are then mapped to a number of predefined resource types, described in terms of hardware capacity, cost, and location. We call the joint execution of these tasks ***resource synthesis***. It is not a straightforward task due to the large number of resource type options and intersections among them. Yet, even considering a single region of any given cloud provider, a possibly large number of resource types may need to be inspected and the best cloud vendor must be selected. To elucidate the main elements of the resource synthesis problem and precisely outline its scope, we first formalize it.

3.1 Problem Formalization

Given a pub/sub system with intermittently disconnected peers, the ***Resource Allocation Problem*** **(RAP)** consists in selecting the proper resource type and the number of resource instances, in order to deploy a network of brokers aiming at QoS enforcement and cost savings. To mathematically represent event streams, we use a topic-based subscription model, since it is efficient and simple in terms of event classification. Nevertheless, our approach can be used for any model where several classification techniques are applied. We model the parameters concerning the RAP as follows:

$R = \{r_j : j \in [1..|R|]\}$ is the set of event topics that correspond to IoT sensor data, device commands, etc.

$S = \{s_i : i \in [1..|S|]\}$ is the set of subscribers: devices or persons interested in event notifications. They subscribe to any number of event topics in R. We denote the set of topics that s_i subscribes as $R_{s_i} \subseteq R$. Let $\lambda^{sub}_{s_i,r_j}$ be the delivery rates of events to each subscriber s_i for publications to topic $r_j \in R_{s_i}$.

$P = \{p_i : i \in [1..|P|]\}$ is the set of publishers: devices/persons that publish events on some set of topics. We denote the set of topics that p_i publishes as $R_{p_i} \subseteq R$. Let $\lambda^{pub}_{p_i,r_j}$ be the publication rate of events published to topic r_j by publisher $p_i \in P$. We assume that each p_i publishes events according to a Poisson process at each topic r_j.

We model the connectivity of pub/sub peers as follows: let ON and OFF be the states where the peer is connected and disconnected, respectively. A given peer, is connected (ON state) for an exponentially distributed time period with parameter θ^{ON} ($\theta^{\mathrm{ON}} = 1/T^{\mathrm{ON}}$). Upon the expiration of this time, the peer disconnects (OFF state) and stops sending or receiving relevant events for an exponentially distributed time period with parameter θ^{OFF} ($\theta^{\mathrm{OFF}} = 1/T^{\mathrm{OFF}}$). Accordingly, let $T^{\mathrm{ON}}_{p_i}$, $T^{\mathrm{OFF}}_{p_i}$ be the connection/disconnection average periods for p_i and $T^{\mathrm{ON}}_{s_i}$, $T^{\mathrm{OFF}}_{s_i}$ for s_i.

$B = \{b_k : k \in [1..|B|]\}$ is the set of pub/sub brokers. A broker forwards events from publishers to interested subscribers or to other brokers in the *broker network* for eventual consumption by a subscriber. We assume that each publisher/subscriber connects with a single broker that we refer to as its *home broker*: b_{p_i} is the broker that publisher p_i publishes to and b_{s_i} is the broker that subscriber s_i receives events from. Furthermore, we define the set of publishers and subscribers connected with b_k as $P_{b_k} = \{p_i \in P : b_k = b_{p_i}\}$ and $S_{b_k} = \{s_i \in S : b_k = b_{s_i}\}$ respectively.

$V = \{v_j : j \in [1..|V|]\}$ is the set of resource types (e.g., VMs).

$\zeta[]_{v_j}$ is a vector used to describe a resource type. It contains information about hardware capacity (e.g., CPU cores and clock speed, bandwidth, etc.).

c_{v_j} is the allocation cost of an instance created using the resource type v_j.

Δ_{s_i} is the end-to-end *response time* of events matching subscriber's topics R_{s_i} from the moment they are published until s_i receives them.

Δ_{p_i,s_i} is the end-to-end response time of events matching the common subscriber's and publisher's topics (i.e., $R_{p_i} \cap R_{s_i}$) from the moment they are published by p_i until s_i receives them.

The above delays include event processing times and network delays inside the broker network. Note that in this paper we do not model processing delays of publishers/subscribers. Our main purpose is to estimate delays inside the broker network in order to allocate their resources accordingly.

$\Delta_{b_k,v_j}^{p_i,s_i}$ is the response time contribution from broker b_k on the end-to-end path between p_i and s_i. This contribution depends on the resource v_j selected for broker deployment.

Δ^{thr} is the response time threshold inside the pub/sub system for any end-to-end path between p_i and s_i. When setting this parameter, resource allocation should be performed in such a way that the end-to-end response time (Δ_{p_i,s_i}) of all events between any p_i and s_i does not exceed this limit.

x_i is an integer variable that indicates if Δ^{thr} has been satisfied for a given end-to-end path, as follows:

$$x_i = \begin{cases} 1, & \text{if } \Delta_{s_i} \le \Delta^{thr}; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

$F = \{f_k : k \in [1..|F|]\}$ is the set of mappings (result of RAP), where each f_k is a pair (b_k, v_j) representing the mapping of a broker to an allocated resource type (which includes the number of deployed resource instances).

φ_{v_j} is an objective function depicting the profit in allocating resource v_j. It is used to distinguish the best option among feasible solutions.

$\varphi(F)$ is the composed profit of all resources in F. In this way, the optimal solution satisfies the QoS threshold for all subscribers and gives the best $\varphi(F)$.

RAP can be more formally expressed as an optimization problem as follows:

$$\begin{aligned}
\text{Maximize} \quad & \varphi(F) = \sum_{b_k \in B} \sum_{v_j \in V} \varphi_{v_j} y_{kj}; & \\
\text{subject to} \quad & \sum_{s_i \in S} x_i = |S|; & \\
& \sum_{v_j \in V} y_{kj} = 1, & k \in [1..|B|]; \\
& y_{kj} \in \{0, 1\}, & k \in [1..|B|], \\
& & j \in [1..|V|].
\end{aligned}$$

The variable y_{kj} is either 1, implying b_k is mapped to v_j, or 0 otherwise.

As RAP is clearly an optimization problem, we argue that it is not possible find an optimal solution for an instance of this problem in polynomial time. Nevertheless, in the next section, we provide a feasible approach based on a reduction to a version of the knapsack problem and on an analytical model for QoS estimation for solving this problem.

4 Resource Synthesis with Intermittent Connectivity

Our approach focuses on the QoS-aware allocation of resources for deploying message brokers at design time. Initially, a system designer provides the topology and characteristics of the pub/sub system (brokers, peers' intermittent connectivity, etc.), its routing algorithm [5] (e.g., rendezvous nodes) and the end-to-end response time threshold (Δ^{thr}). Then, our approach provides a specification of the corresponding resources, in terms of types and quantity. After deploying brokers based on the specification, end-to-end response times must be below the specified Δ^{thr} at runtime. In this paper, we are agnostic with respect to the algorithm used to route events. Hence, we assume that the end-to-end paths between publishers and subscribers (i.e., the broker paths) are estimated somehow using the provided routing algorithm. Additionally, we acknowledge that the resulting resource allocation may be invalidated if different routing paths are used at runtime. However, we argue that complementary solutions may benefit from cloud elasticity to perform adaptation of the resource allocation.

To provide an efficient solution for the above-described problem without the costs associated with an exhaustive search strategy, we rely on well-known solutions to the Multiple-choice Multi-dimension Knapsack Problem (MMKP) [24]. MMKP is a NP-Complete problem defined as follows: given a set H of items divided in h categories Q_q, where each category $Q_q : q \in [1..h]$, has $\varkappa_q = |Q_q|$ items such as $\forall\ 1 \leq i, j \leq h$ and $i \neq j, Q_i \cap Q_j = \emptyset$ and $\cup_{i=1}^{h} Q_i = H$. Each item $o \in [1..\varkappa_q]$, from category Q_q has a non-negative profit ϱ_{qo}, and has dimensions given by a weight vector $W_{qo} = \{w_{qo}^a : a \in [1..\ell]\}$, where each element w_{qo}^a is also a non-negative value. The knapsack dimensions are given by the vector $A = \{A^a : a \in [1..\ell]\}$. The goal in MMKP is to pick exactly one item from each category in order to maximize the total profit, subject to knapsack dimensions.

In the provided approach, we construct an MMKP instance from a RAP instance. To do so, we rely on results related to solving MMKP. We use the WS-HEU heuristic [31] to efficiently select the resource types in order to reach the thresholds for each constraint. We chose this heuristic because it solves the MMKP problem with an accuracy of 96% compared to the optimal value, and can provide worst-case complexity of $O(|S| \times |B|^2 \times (|V| - 1)^2)$. The construction of an MMKP instance from a RAP instance is carried out as follows. Let n be the number of categories (h). We map each $b_k \in B$ as a category Q_q with same number of items (generated from resource types, i.e., each $v_j \in V$ is mapped as a different item o for each category). In this way, we remove the property of empty set in the items intersection of categories, but this change does not affect the problem since constraints are evaluated separately for each category. We generate the weight vector W_{qo} for each item by using the average delays of each routing path in which the corresponding broker participates, so that $[\Delta^{p_1,s_1}_{b_k,v_j}..\Delta^{p_{|S|},s_{|S|}}_{b_k,v_j}]$ is mapped to $[w^1_{qo}..w^{\ell}_{qo}]$, and φ_{v_j} is mapped to ϱ_{qo}. Finally, the response time threshold is multiply mapped as knapsack dimensions $A = \{A^a : a \in [1..\ell]\}$.

End-to-end QoS evaluation is carried out in our approach by separately addressing each possible routing path in the pub/sub system. Although this may sound inefficient, we argue that by using the chosen heuristic we can reach the resource synthesis result in admissible time. Additionally, this task is performed at design time, when low overhead is not a major concern. We discuss this further in Sect. 5 when presenting the experimental evaluation. By using the WS-HEU heuristic, an initial solution is pursued using a greedy strategy based on the ratio between resource cost and QoS offering. In cases where even considering all resource types no feasible solution is found, an event of unaccomplished resource synthesis is triggered. We deal with this problem through the inclusion of new broker instances. To this end, we assume a round-robin load balancer with negligible overhead. The selection of the broker to be replicated is carried out by taking the broker with the worst QoS, which is most likely one of the edge brokers due to the effects of peers' disconnections. Additional broker instances are added until a feasible solution is found. When this happens, there is an attempt to improve the solution by performing simulated annealing [1]. The entire procedure is performed for the resource types from each cloud provider.

The resource synthesis output is a mapping of each broker (and its additional instances) to a selected resource type, taking the less costly result. By using this strategy, resource allocation is performed through the balancing of the contribution to the end-to-end constraints among all participating brokers. As can be seen, a key aspect of this approach is QoS estimation. For this purpose, we propose the performance model described next.

4.1 Pub/Sub QoS Estimation

In this subsection, we present our queueing network model which is used as input to the resource allocation algorithm. By relying on our previous work [9], we model the performance inside a pub/sub broker network by relying on queueing theory. In particular, each queue processes events through a dedicated *server*.

Each server supports a specific service rate (λ) which is exponentially distributed. All queueing centers apply a first-come-first-served (FCFS) queueing policy. We model the network transmission delay between brokers ($b_k \rightarrow b_i$) using an $M/M/1$ queue and between b_{s_i} (s_i's home broker) $\rightarrow s_i$ (subscriber) using an ON/OFF queue. With regard to [9], in this paper we slightly modify our formal model and queueing models. More specifically, in order to process incoming events at a broker node b_k we use the $M/M/c$ queueing model: c is the number of servers of the queue and each server has an independently and identically distributed exponential service-time distribution with mean $1/\mu$. We correspond c to the number of VM cores in which the broker is deployed.

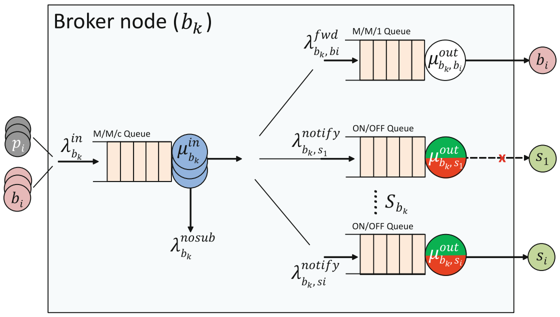


Fig. 2. Queueing network for pub/sub broker node.

As depicted in Fig. 2, we model each broker b_k using a single inbound queue $q^{in}_{b_k}$ (M/M/c) and multiple outbound queues $q^{out}_{b_k}$ (M/M/1 and ON/OFF). Let $\lambda^{in}_{b_k}$ be the arrival rate of events at $q^{in}_{b_k}$, represented as the sum of all event publication/forwarding rates over all publishers/brokers. Forwarding, replication, or dropping of events based on current subscriptions occurs at the exit of $q^{in}_{b_k}$. Let $\mu^{in}_{b_k}$ be $q^{in}_{b_k}$'s service rate for analyzing an incoming event and determining where to forward it (e.g. based on a topic routing tree). We assume that the service rate is exponentially distributed across all topics with parameter $\mu^{in}_{b_k}$. Events that do not match b_k's subscriptions are dropped with rate $\lambda^{nosub}_{b_k}$.

Broker b_k forwards events matching subscriptions served by other brokers with rate $\lambda^{fwd}_{b_k,b_i}$ through an outbound M/M/1 queue $q^{out}_{b_k,b_i}$. These events are sent from b_k to any $b_i \in B, b_i \neq b_k$. Let $\mu^{out}_{b_k,b_i}$ be $q^{out}_{b_k,b_i}$'s service rate for transmitting an event to b_i. For each of b_k's local subscribers, $s_i \in S_{b_k}$ & $b_{s_i} = b_k$, b_k forwards events matching subscriptions to $q^{out}_{b_k,s_i}$ with rate $\lambda^{notify}_{b_k,s_i}$ for transmission to s_i. We model each $q^{out}_{b_k,s_i}$ using an ON/OFF queue for transmitting events based on s_i's intermittent connectivity (i.e., $T^{\mathrm{ON}}_{s_i}$, $T^{\mathrm{OFF}}_{s_i}$). Let $\mu^{out}_{b_k,s_i}$ be $q^{out}_{b_k,s_i}$'s service rate for transmitting an event to s_i. Note that $\mu^{out}_{b_k,s_i}$ and $\mu^{out}_{b_k,b_i}$ rates model the transmission delay of events inside the network.

To build the broker model of Fig. 2, we use three different types of queueing models. Based on standard solutions, each queueing type evaluates several performance metrics (e.g., response time) through analytical models. Particularly, with regard to response time in M/M/1 queues [20], the time that an event remains in the system (corresponding to queueing time + service time, also called average delay) is given by:

$$\Delta_{q_{mm1}}(\mu, \lambda) \;=\; \frac{1}{(\mu \;-\; \lambda)} \tag{2}$$

Regarding the M/M/c queue, the time that an event is remain in the system is given by [18]:

$$\Delta_{q_{mmc}}(\mu, \lambda, c) \;=\; \frac{1}{\mu} + \left(\frac{r^c}{c!(c\mu)(1-\rho)^2}\right)p_0 \tag{3}$$

where c is the number of servers, $r = \lambda/\mu$, $\rho = r/c$ and the steady-state probability p_0 can be found in [18]. Finally, regarding the ON/OFF queue, the time that an event is remain in the system is given by [9]:

$$\Delta_{q_{onoff}}(\mu, \lambda, T^{\mathrm{ON}}, T^{\mathrm{OFF}}) \;=\; \frac{\frac{T^{\mathrm{OFF}\,2}}{T^{\mathrm{ON}}+T^{\mathrm{OFF}}} + \frac{T^{\mathrm{ON}}+T^{\mathrm{OFF}}}{\mu T^{\mathrm{ON}}}}{1 - \lambda\frac{T^{\mathrm{ON}}+T^{\mathrm{OFF}}}{\mu T^{\mathrm{ON}}}} \tag{4}$$

In the next subsection, we show how these analytical models can be used to estimate end-to-end response times.

End-to-End Response Time. We propose a strategy to allocate resources for deploying pub/sub brokers based on the response time threshold (Δ^{thr}) applied inside the pub/sub system. Hence, in order to allocate resources it is essential to calculate the expected delay between p_i and s_i (Δ_{p_i,s_i}) inside the broker network by considering the intermittent connectivity of peers. Based on the broker's queueing model (see Fig. 2), s_i's connectivity affects the resulting delay – i.e., during $T_{s_i}^{\mathrm{OFF}}$, b_k buffers events. Furthermore, events may pass through a subset of brokers, until they arrive to s_i's home broker (b_{s_i}). Therefore, for a given s_i, the average delay inside a broker node b_k can be estimated as follows:

$$\Delta_{b_k} = \begin{cases} \Delta_{q_{b_k}^{in}} + \Delta_{q_{b_k,b_i}^{out}}, & \text{if } b_k \neq b_{s_i} \\ \Delta_{q_{b_k}^{in}} + \Delta_{q_{b_k,s_i}^{out}}, & \text{if } b_k = b_{s_i} \end{cases} \tag{5}$$

Subsequently, for a given interaction between p_i and s_i, events pass through a subset of intermediate brokers (b_i) and s_i's home broker (b_{s_i}). Let λ_{p_i,s_i} be the rate of events from p_i to s_i. To estimate Δ_{p_i,s_i}, it is essential to apply the so called *effective* service rate ($\mu_{b_k-eff}^{in/out}$) at each inbound/outbound queue of b_k. Such a service rate is calculated only for the λ_{p_i,s_i} rate but by taking into account other rates of events $\lambda_{b_k}^{in/out-oth}$ (lack of subscriptions, forwarding rates to other brokers, etc.) [9].

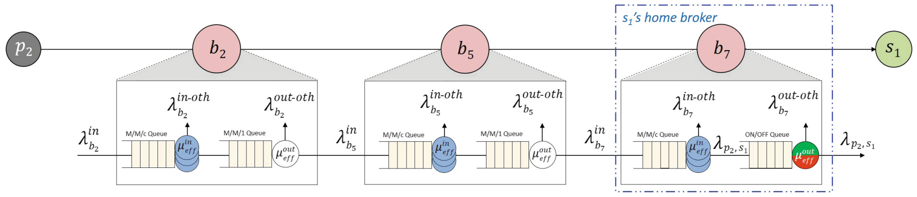


Fig. 3. Queueing network for the end-to-end interaction: p_2 to s_1.

To demonstrate our approach, we provide the following example: to Δ_{p_2,s_1}, the resulting queueing network is depicted in Fig. 3. Hence, Δ_{p_2,s_1} is given by:

$$\Delta_{p_2,s_1} = \Delta_{b_2} + \Delta_{b_5} + \Delta_{b_7}$$

Based on Fig. 3, b_2, b_5 are intermediate brokers and b_7 is s_1's home broker ($b_7 = b_{s_1}$). Hence, by relying on (5), we estimate the end-to-end delay using the corresponding queues. As already pointed out, $q^{in}_{b_k}$ is an M/M/c queue, $q^{out}_{b_k,b_i}$ is an M/M/1 queue and $q^{out}_{b_k,s_i}$ is an ON/OFF queue. Hence, by relying on (2), (3) and (4) the average delay of an intermediate broker, e.g., b_2 is given by:

$$\Delta_{b_2} = \Delta_{q_{mmc}}(\mu^{in}_{b_2-eff}, \lambda_{p_2,s_1}, c_{b_2}) + \Delta_{q_{mm1}}(\mu^{out}_{b_2-eff}, \lambda_{p_2,s_1})$$

And the average delay of b_{s_1} is given by:

$$\Delta_{b_{s_1}} = \Delta_{q_{mmc}}(\mu^{in}_{b_7-eff}, \lambda_{p_2,s_1}, c_{b_7}) + \Delta_{q_{onoff}}(\mu^{out}_{b_7-eff}, \lambda_{p_2,s_1}, T^{\mathrm{ON}}_{s_1}, T^{\mathrm{OFF}}_{s_1})$$

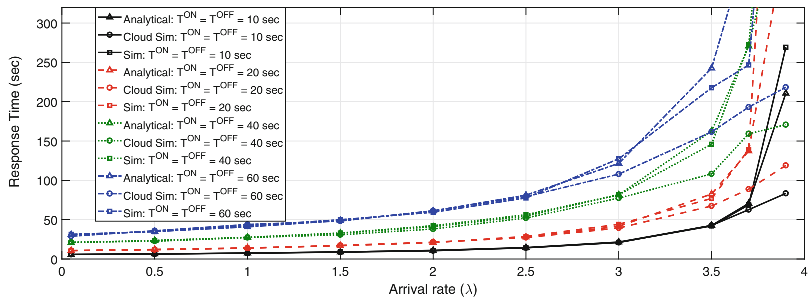


Fig. 4. Comparison of results from M-CloudSim, analytical model and MobileJINQS.

5 Experimental Evaluation

The goal of our experiments is to study the effects of our approach on QoS enforcement, on the reduction of the resource allocation cost, and on performance. In this section, we first introduce a simulation environment developed to enable this evaluation and then discuss the experiments and achieved results.

5.1 Simulating Intermittent Connectivity over Pub/Sub Systems

Experimental evaluation requires tools that facilitate the design of experiments while making them repeatable and precise. Simulators are useful tools to build such evaluation environments in the cloud context [32]. However, to the best of our knowledge, there is no simulation environment for cloud-based pub/sub systems enabling the modeling of the peers' connectivity. For this reason, we developed our own simulator, called *Mobile CloudSim* (M-CloudSim)[1], as an extension of the CloudSim simulation toolkit [11] with the following features.

(1) Application model: Simulations in CloudSim are implemented by modeling a set of cloudlets (application tasks) and VMs. On top of it, a cloud mediator is defined to schedule cloudlets according to the available VMs in a discrete-event simulation dynamics. Although this model is feasible to represent the main aspects of cloud applications, it does not allow indirect communication modeling as well as routing in message brokers. To solve this limitation, in our extension we have distinguished two cloudlet categories: event processing and event transmission. We explicitly represent the events routing paths, creating and submitting cloudlets from both categories to simulate event forwarding.

(2) Modeling intermittent peer connectivity: one of the main goals in our evaluation is to evaluate the effects of peer disconnection. We handle this by simulating changes on peer connection state. On the simulation, peer connection is checked before the processing, transmission, and delivery of events on home brokers. We only simulate changes of connectivity states for subscribers since during the publishers' disconnections there is no event-transmission to brokers and, therefore, no relation with resource allocation.

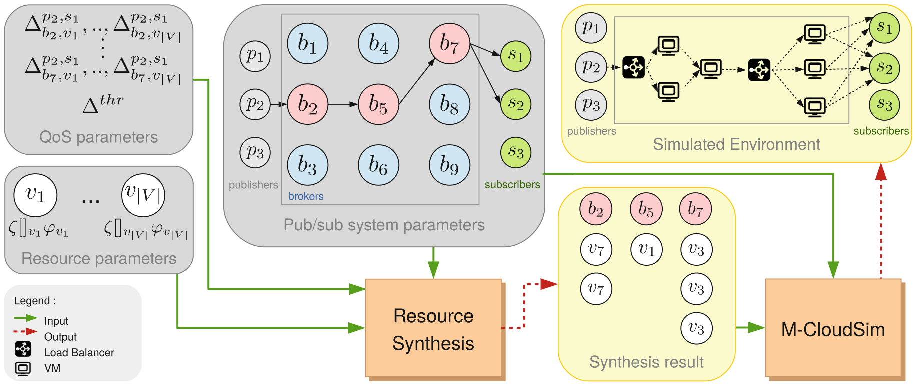


Fig. 5. Resource synthesis validation methodology.

We kept the M-CloudSim functionalities limited to the scope of this paper. As a way of demonstrating the validity of the results provided, we have performed

[1] https://github.com/raphaeldeaquino/mcloudsim.

an experimental comparison with two other tools: using the analytical model presented in this paper and using MobileJINQS [9], an open-source library for building simulations encompassing the constraints of mobile systems (but lacking resource simulation). In this evaluation, we have computed the response time in a specific path of a pub/sub system. In the simulated environment the subscriber remains in the ON and OFF states for exponentially distributed time periods $T^{\mathrm{ON}} = T^{\mathrm{OFF}} = 10/20/40/60\,\mathrm{s}$. For the analytical model and the MobileJINQS simulation, processing and transmission of events are served with a mean service demand of 0.0625 s and 0.125 s, respectively. Accordingly, in the M-CloudSim simulation we set CPU and bandwidth characteristics in the simulated VMs to provide the same timing. We assume sufficient buffer capacities so that no events are dropped. Also events are generated by the publisher with a mean rate varying from 0.05 to 4 events per sec. By applying λ rates greater than 4 events/sec, the system saturates both in the analytical model and in the MobileJINQS simulation. Differently, M-CloudSim can simulate any number of input rates since it works for a predefined time slice by using waiting queues with "infinite" size. The mean response time of 100 independent executions (for each scenario) is depicted in Fig. 4. Confidence intervals of the simulation results are found to be very small and are not presented in the figure.

By comparing the curves for the three strategies response times, we notice that results match with high accuracy for arrival rates below to 3.5 events/sec. Differences are noticed for rates equal to or higher than 3.5 events/sec. This is acceptable since the system is close to saturation at these rates.

5.2 Resource Synthesis Validation

In this experiment, we check whether QoS enforcement is achieved as a result of the resource synthesis approach. We also evaluate the resulting profit by using the resource allocation cost as a parameter. On top of it, we check whether the peer's intermittent connectivity affects the resource allocation cost. For this purpose, we carry out the resource synthesis in different scenarios and use M-CloudSim to examine the achieved response times. We compare the following scenarios: *(i)* with always-connected pub/sub peers; and *(ii)* with intermittently connected peers due to network issues. In both cases, we apply the methodology illustrated in Fig. 5: we first perform resource synthesis by using as input an end-to-end response time threshold (Δ^{thr}), the pub/sub system topology (i.e., publishers, subscribers), the flow of events, the end-to-end routing path of events, the peers intermittent connectivity and the available resource types.

For all scenarios, we applied as input the VM types provided by the market-leading vendors according to Gartner's latest report on cloud Infrastructure as a Service [21] – Amazon [2], Microsoft Azure [23], and Google Cloud Platform [17]. In this experiment, we use the available VM configurations in the São Paulo, Brazil, region on Jan-2018, with a total of 94 resource types. For the pub/sub system we use the same topology of Fig. 1 by taking only events on path $p_2 \rightarrow s_1$. We isolate other flows of this path by estimating the corresponding effective service rate (μ_{eff}) for $p_2 \rightarrow s_1$. Larger scale scenarios (with more paths) are used to

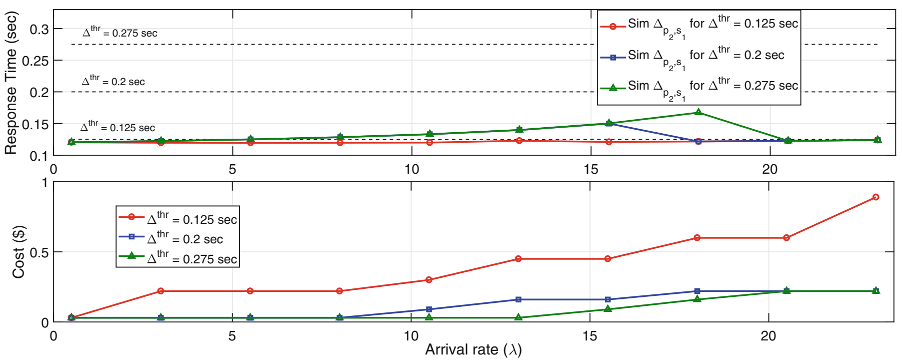


Fig. 6. Results of synthesis validation for always connected subscriber.

evaluate the scalability of the approach in the next section. We assume that the broker network is reliable, with no broker overload. In addition, all events have the same size, which is equal to 200 bytes (approximately 200 characters of application payload). Also event processing requires 33.2 CPU instructions (average number of instructions in the method body of general Java programs [13]). We vary the arrival rate of events from 0.5 to 23 events per second. We set Δ^{thr} with an initial value of 0.025 s. Then, we increase Δ^{thr} with increments of 0.025 s. We perform evaluations until the synthesis result is the same for at least 5 consecutive arrival rate variations (half of the analyzed scenarios). We present the results for the two scenarios next.

Always Connected

As already pointed out, in this scenario we assume that a subscriber is always connected ($T_{s_i}^{\mathrm{OFF}} = 0$). We use this case to analyze what is the minimum achieved response time in the given scenario and to generate a comparison baseline for the next case. Figure 6 presents the results of simulated response times and resource allocation cost for selected thresholds. Although we have analyzed several thresholds, for better readability we present only three cases: *(i)* the minimum achieved response time ($\Delta^{thr} = 0.125$); *(ii)* the upper threshold analyzed ($\Delta^{thr} = 0.275$), i.e., when synthesis result is stable for at least half of the arrival rates; and *(iii)* an intermediary value between them ($\Delta^{thr} = 0.2$). We discuss only these thresholds because it is not possible to achieve smaller values with the available resource types (even with multiple instances). On the other hand, higher thresholds do not affect the synthesis result when using the considered arrival rates. As the main result, QoS enforcement was achieved in all scenarios. It is worth noting that the simulated response times are, on average, 30% better than the set of thresholds, while the resource allocation cost was, on average, \$0.40, \$0.12, and \$0.09, respectively for each threshold. As expected, the allocation cost increases for higher arrival rates. As a remark, the resource types on Amazon provided the best results in all scenarios for the São Paulo region.

Network Issues

The actual connectivity of mobile peers depends on the network coverage and capacity, as well as the type of peer mobility. In this scenario, we evaluate the case of subway passengers in order to analyze the effects of the intermittent connectivity of peers on resource allocation. For this purpose, we use our dataset[2] of mobile connectivity for the subway system in Paris. The used data show that subway travelers loose and recover network connection for periods that range from several seconds to 5 min, maximum. On average, connected periods are 1.5 times larger than the disconnected periods. We randomly select the path *Dugommier → Cité Universitaire* for this study of subscriber mobility. By analyzing the complementary cumulative distribution functions of connections and disconnections in the above subway path, we conclude that our traces fit best with an exponential distribution, such as $T^{\mathrm{ON}}_{s_i} = 155.8\,\mathrm{s}$ and $T^{\mathrm{OFF}}_{s_i} = 96\,\mathrm{s}$. Figure 7 presents the results for selected thresholds.

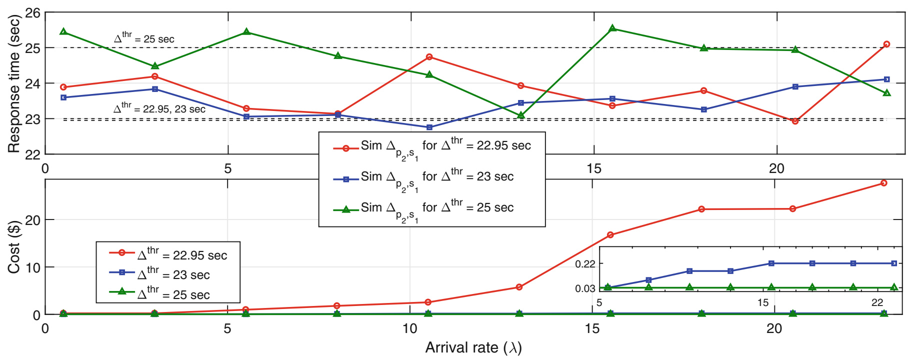


Fig. 7. Results of synthesis validation for subscriber's intermittent connectivity.

QoS enforcement is more challenging with intermittent connectivity. For the given parameters it is not possible to achieve response times lower than 22.95 s due to the overhead in the processing of buffered events. For this threshold, there is QoS violation in some cases but the values were on average only 3.11% different from the thresholds. The resource allocation cost was, on average, \$10.03, \$0.14, and \$0.03, respectively. This result enforces the need for such a solution for resource allocation tunning since a small change in response time threshold (an increasing of 8.95% on tolerated response time on this case) can result in a substantial decrease in resource allocation cost (a saving of 99.7% on cost).

5.3 Performance Evaluation

In the previous experiments, we demonstrated the effectiveness of the resource synthesis approach on QoS enforcement. However, we performed this evaluation

[2] https://github.com/boulouk/mobile-jinqs/tree/master/experiments/metro.

using a single event routing path. In order to extend this analysis, we measured the approach performance for higher scale scenarios. For this purpose, we analyzed the processing time to get results when taking many routing paths as input. We do this evaluation using the scenario from the previous experiment with always-connected subscribers, and a publishing rate of 23 events/sec. We set $\Delta^{thr} = 10\,\text{s}$ to make QoS fulfillment possible as the scale increases.

In this experiment, we create synthetic pub/sub topologies by varying $|B| = [1..6]$. By taking paths of different scales (i.e., number of participating brokers) we generate all possible options in the pub/sub system. In the context of our future work, we intend to use specific event routing algorithms (e.g., selective routing or event gossiping [5]) in order to scale our approach (the set of possible end-to-end paths will be reduced). For each scenario, we generate all routing paths with same number of communicating message brokers as follows: we generate all combinations without repetition taking $l = [1..|B|]$ message brokers per path. For each resulting set we then create all possible permutations. Using this strategy, each scenario has a topology with $\binom{|B|}{l} l! = \frac{|B|!}{(|B|-l)!}$ routing paths what allow us to evaluate the scalability of the approach.

Table 1. Resource synthesis processing time (in sec)

Total number of brokers	Number of brokers in each routing path					
	1	2	3	4	5	6
1	0.03					
2	0.04	0.09				
3	0.06	0.12	0.16			
4	0.08	0.21	0.68	0.99		
5	0.07	0.37	2.14	10.04	14.69	
6	0.10	0.55	5.91	62.41	346.03	489.36

We run the synthesis approach for the same input 100 times and collect the average processing time and confidence interval. All executions were performed on a machine with the following configuration: Intel CPU® Core™ i5 2.67 GHz, 4 GB RAM, Ubuntu 14.04. The average processing time for each scenario is presented in Table 1. Resource synthesis processing time varies from 0.03 s to 8.16 min. The high processing time in the worst case scenario is due to a high number of routing paths. In order to evaluate the feasibility of the solution, the processing time must be compared with the duration of the other tasks in the pub/sub system deployment. Regarding this, since we do not use previously instantiated VMs, VM startup is the main bottleneck. The reason for this is that VMs have to be instantiated and configured, which includes communication with the cloud provider, VM loading, and installation of software components. Even though we have not conducted any experiments in this regard, there are a number of results in the literature that indicate an overhead from 44.2 s to 13.5 min on

VM startup in public clouds [22]. We argue that the overhead imposed by the resource synthesis is acceptable, especially if we consider the achieved benefits.

6 Conclusions and Future Work

In this paper, we propose a new approach for resource allocation for IoT applications in the Cloud. Such applications run over pub/sub systems and their *things* can be intermittently disconnected. Our approach can be used as a tool by pub/sub system designers to allocate resources for QoS enforcement. We have evaluated the proposed solution using scenarios based on probability distributions of the events, as well as scenarios based on connectivity parameters derived from real traces. QoS enforcement was achieved in almost all scenarios, which showed only 3.11% of QoS violation in some cases. We conclude that resource demand increases with higher arrival rates but also due to peer disconnections. Consequently, the cost of deploying the system is also affected by these parameters. However, we have shown that our approach can help reasoning about efficient resource allocation in a variety of scenarios, saving up to 99.7% on resource allocation cost. Finally, our approach has a feasibly processing time, as corroborated by the experiments.

As future work, we intend to identify the limits for system saturation, given the available resources. We also aim to carry out a more comprehensive performance evaluation using real pub/sub protocols. As further unfolding of the work, we aim to extend this proposal with support for QoS enforcement when using multiple regions.

References

1. Aarts, E., Korst, J.: Simulated Annealing and Boltzmann Machines. Wiley, New York (1988)
2. Amazon: Amazon Web Services - Broad & Deep Core Cloud Infrastructure Services (2018). http://aws.amazon.com. Accessed 28 Mar 2018
3. Apache Kafka (2018). http://kafka.apache.org/. Accessed 28 Mar 2018
4. Atzori, L., Iera, A., Morabito, G.: The Internet of Things: a survey. Comput. Netw. **54**, 2787–2805 (2010)
5. Baldoni, R., Querzoni, L., Virgillito, A.: Distributed event routing in publish/subscribe communication systems: a survey. Technical report, Universita di Roma La Sapienza (2005)
6. Banks, A., Gupta, R.: MQTT Version 3.1. 1. OASIS standard (2014)
7. Barazzutti, R., Heinze, T., Martin, A., Onica, E., Felber, P., Fetzer, C., Jerzak, Z., Pasin, M., Rivière, E.: Elastic scaling of a high-throughput content-based publish/subscribe engine. In: IEEE 34th ICDCS, pp. 567–576. IEEE (2014)
8. Botta, A., De Donato, W., Persico, V., Pescapé, A.: Integration of cloud computing and IoT: a survey. Future Gener. Comput. Syst. **56**, 684–700 (2016)
9. Bouloukakis, G., Georgantas, N., Kattepur, A., Issarny, V.: Timeliness evaluation of intermittent mobile connectivity over pub/sub systems. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, pp. 275–286. ACM (2017)

10. Bouloukakis, G., Moscholios, I., Georgantas, N., Issarny, V.: Performance modeling of the middleware overlay infrastructure of mobile things. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2017)
11. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011)
12. Carvalho, N., Araujo, F., Rodrigues, L.: Scalable QoS-based event routing in publish-subscribe systems. In: Fourth IEEE International Symposium on Network Computing and Applications, pp. 101–108. IEEE (2005)
13. Collberg, C., Myles, G., Stepp, M.: An empirical study of java bytecode programs. Softw. Pract. Exp. **37**, 581–641 (2007)
14. Corsaro, A., Querzoni, L., Scipioni, S., Piergiovanni, S.T., Virgillito, A.: Quality of service in publish/subscribe middleware. In: Global Data Management (2006)
15. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Comput. Surv. **35**, 114–131 (2003)
16. Gascon-Samson, J., Garcia, F., Kemme, B., Kienzle, J.: Dynamoth: a scalable pub/sub middleware for latency-constrained applications in the cloud. In: IEEE ICDCS, pp. 486–496 (2015)
17. Google: Compute Engine (2018). https://cloud.google.com/compute. Accessed 28 Mar 2018
18. Gross, D., Shortle, J., Thompson, J., Harris, C.: Fundamentals of Queueing Theory. Wiley, New York (2008)
19. Hoffert, J., Schmidt, D.C., Gokhale, A.: Adapting distributed real-time and embedded pub/sub middleware for cloud computing environments. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 21–41. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16955-7_2
20. Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C.: Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Prentice-Hall, Inc., Upper Saddle River (1984)
21. Lydia, L., Raj, B., Craig, L., Dennis, S.: Magic quadrant for cloud infrastructure as a service, Worldwide (2017). https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519. Accessed 28 Mar 2018
22. Mao, M., Humphrey, M.: A performance study on the VM startup time in the cloud. In: IEEE CLOUD, pp. 423–430 (2012)
23. Microsoft: Microsoft Azure (2018). https://azure.microsoft.com. Accessed 28 Mar 2018
24. Moser, M., Jokanovic, D.P., Shiratori, N.: An algorithm for the multidimensional multiple-choice knapsack problem. IEICE **80**(3), 582–589 (1997)
25. Nguyen, P., Nahrstedt, K.: Resource management for elastic publish subscribe systems: a performance modeling-based approach. In: IEEE CLOUD (2016)
26. Oracle: JMS Specifications (2018). http://www.oracle.com/technetwork/java/jms/index.html. Accessed 28 Mar 2018
27. Pivotal: RabbitMQ (2018). https://www.rabbitmq.com/. Accessed 28 Mar 2018
28. Setty, V., Vitenberg, R., Kreitz, G., Urdaneta, G., van Steen, M.: Cost-effective resource allocation for deploying pub/sub on cloud. In: IEEE ICDCS (2014)
29. Shi, W., Dustdar, S.: The promise of edge computing. Computer **49**, 78–81 (2016)
30. Standard, O.: Oasis advanced message queuing protocol (AMQP) version 1.0
31. Yu, T., Zhang, Y., Lin, K.J.: Efficient algorithms for Web services selection with end-to-end QoS constraints. In: ACM TWEB, vol. 1 (2007)
32. Zhao, W., Peng, Y., Xie, F., Dai, Z.: Modeling and simulation of cloud computing: a review. In: IEEE APCloudCC, pp. 20–24 (2012)

ProCal: A Low-Cost and Programmable Calibration Tool for IoT Devices

Chia-Chi Li[1,2](✉) and Behnam Dezfouli[1](✉)

[1] Internet of Things Research Lab, Department of Computer Engineering, Santa Clara University, Santa Clara, CA, USA
{cli1,bdezfouli}@scu.edu

[2] Intel Corporation, Santa Clara, CA, USA

Abstract. Calibration is an important step towards building reliable IoT systems. For example, accurate sensor reading requires ADC calibration, and power monitoring chips must be calibrated before being used for measuring the energy consumption of IoT devices. In this paper, we present ProCal, a low-cost, accurate, and scalable power calibration tool. ProCal is a programmable platform which provides dynamic voltage and current output for calibration. The basic idea is to use a digital potentiometer connected to a parallel resistor network controlled through digital switches. The resistance and output frequency of ProCal is controlled by a software communicating with the board through the SPI interface. Our design provides a simple synchronization mechanism which prevents the need for accurate time synchronization. We present mathematical modeling and validation of the tool by incorporating the concept of Fibonacci sequence. Our extensive experimental studies show that this tool can significantly improve measurement accuracy. For example, for ATMega2560, the ADC error reduces from 0.2% to 0.01%. ProCal not only costs less than 2% of the current commercial solutions, it is also highly accurate by being able to provide extensive range of current and voltage values.

Keywords: Sensors · Accuracy · Measurement · ADC
Power emulation · Interfacing

1 Introduction

One of the main capabilities of IoT devices is sensing. To this end, sensors such as temperature, pressure, humidity, and acoustic, are connected to analog-to-digital converters (ADC) of IoT devices. Confidence about accuracy is particularly important for mission-critical applications such as monitoring a tank's temperature, a pipe's pressure in a chemical factory, or human vital signs [6]. Without calibration, for example, a 10-bit ADC might result in a 5 °C measurement error. In addition, as IoT devices are usually battery powered, the energy of these devices should be carefully measured and optimized based on application requirements. Studies show that it is important for the embedded power

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 88–105, 2018.
https://doi.org/10.1007/978-3-319-94370-1_7

monitor to continuously keep track of power consumption, so that node failures can be diagnosed and eliminated [28]. The existing power monitoring devices make this possible by including ADCs that measure both voltage and current [5,7–9,12,15,31]. Consequently, it is essential to ensure that both sensor reading and power measurement are performed with high accuracy. Multiple factors such as the inaccuracy of the ADC used, temperature, the supply voltage of the sensor, and the length of the sensor-to-ADC path can contribute to measurement inaccuracy [2]. For example, the measurement error of INA219 power monitoring chip increases as its supply voltage reduces [5].

From the ADC point of view, measurement accuracy is influenced by its non-linearity. Two of the key static specifications that define the accuracy of an ADC are *differential non-linearity* (DNL) and *integral non-linearity* (INL). DNL describes deviations from the ideal transition voltage in the converter's transfer functions. The deviation from ideal transition is the differential linearity error for that unique code, which translates to *least significant bit* (LSB). INL is the overall shape of the transfer function of ADC. This error is also referred to as *static linearity* or *absolute linearity.* The maximum deviation from the ideal transfer function to INL line is the worst case integral non-linearity [25]. Non-linearity causes data distortion while the signals are being digitized. By thorough characterization of an ADC's non-linearity and transfer function, we can minimize its data distortion. There are several different approximations of ADC non-linearity commonly used for calibration; such as common polynomials, Chebyshev polynomials, and Fourier series [27]. ADC calibration requires characterization of both increasing and decreasing input levels, which requires a wide range of samples [20]. Since all components, including ADCs, in an analog chain demonstrate deviations from their normal value, a full calibration process is essential.

The two common approaches for calibrating an analog input are: (i) using different resistors [9,13], and (ii) a potentiometer [7,31] to generate variable loads. Although the former approach provides an accurate and stable load, it requires a large number of fixed-value resistors and a lot of effort to collect enough samples for a quality calibration. Due to these challenges, often times, researchers use only a few resistors to calibrate the full range of an ADC [9,13]. This insufficient use of resistors provides a limited calibration range and does not eliminate data distortion across a wide range. The latter approach relies on a mechanical potentiometer to generate loads across a range. A mechanical potentiometer varies its resistance as a result of angular movement. Although some COTS potentiometers offer the ability to withstand large voltage and current values, they cost more than $100 [26]. More importantly, as mechanical potentiometers are analog devices, the actual changes in resistance cannot be controlled in a discrete manner. Therefore, it is difficult to match the readings of a potentiometer versus a baseline. Furthermore, mechanical potentiometers are prone to performance changes and reliability concerns over time, as they are sensitive to shocks and vibrations.

In addition to the aforementioned approaches, an another solution is to use a commercial DC power analyzer [12,15], such as Agilent N6705X [21]. This device has the capability of acting as a target that generates variable current draws and voltage outputs. It has a built-in meter and a logger to capture measurement results. N6705X can produce accurate and stable current and voltage outputs. However, the cost is more than $14,000 for a complete solution.

In this paper, we present a power calibration tool, named ProCal, which is a scalable tool for calibrating analog-to-digital interfaces. ProCal is a low-cost and programmable board that provides a wide range of discrete voltage and current outputs for calibration. The basic idea of ProCal is to combine a digital potentiometer with a controllable resistor network. In particular, the digital potentiometer is used for generating small changes in resistance, and the resistor network is controlled by digital switches to enable large resistance jumps. This is a low-cost approach to overcome the limited range of digital potentiometers currently available. Therefore, one of the key advantages of ProCal is its large dynamic output range for both current and voltage. In addition, the entire ProCal tool only costs about $100, including manufacturing. This is less than 2% of the cost compared to current commercial solutions.

The operations of the digital potentiometer and the resistor network are controlled by software. This software programs ProCal's output transitions and records the output settling instances. If a new current or voltage value is set at a time t, the software records $t + \frac{T}{2}$ as the time stamp of output settling time, where T is the reconfiguration period. Recording the settling time provides the ability to ensure the readings of the target IoT device and measurement equipment, such as the digital multimeter (DMM) are used as the ground truth and correlate with a stable value, without requiring a precise time synchronization. This feature notably simplifies calibration because various IoT devices demonstrate different setup times and sampling rates, depending on factors such as ADC speed, processor speed, and available memory. For example, the sampling rate of an ADC depends on its serial interface, device driver, and programming techniques used. Therefore, when ProCal simultaneously triggers a DMM and an IoT device to sample their analog inputs, we do not need to worry about the exact start times and sampling rates when matching the pairwise values.

One of the salient features of ProCal is providing output scalability to support a wide range of calibration requirements. Precisely, the range of the digital output must match the scale of analog input. For example, assume that an ADC can measure between 0 to 5 V, however, the light sensor connected to ADC has sensing range between 0 to 3 V, thus creating a void from 4 to 5 V. Therefore, the digital output range needs to map between analog input from 0 to 3 V to maximize the resolution of calibration. To show the scalability of ProCal, we incorporate the Fibonacci sequence to formulate a mathematical model of scalability. Therefore, ProCal enables the user to adjust resolution depending on the application at hand.

We have implemented a prototype of ProCal and conducted an extensive set of experimental evaluations. Our results show that ProCal provides satisfactory measurement fidelity under a wide range of operations. In particular, ProCal produces a dynamic current range 0.476 mA to 1 A, and dynamic voltage range 0.06 mV to 5 V. The minimum resolutions of current and voltage are 1.82 µA and 0.01 µV, respectively. We also present case studies where ProCal is used for calibrating high-resolution ADCs. Our calibration shows improvement across different devices. For example, for ATMega2560, the ADC error reduces from 0.2% to 0.01%, and for INA219, the ADC error reduces from 0.42% to 0.02%.

The rest of the paper is organized as follow. Section 2 presents the design challenges of scalable calibration, as well as the design, mathematical modeling, and implementation of ProCal. Evaluation of ProCal through various case studies is given in Sect. 3. Section 4 discusses the related work. We conclude the paper in Sect. 5.

2 Design Challenges, Considerations and Implementation

The primary objective of designing a calibration tool is to calibrate the analog inputs of IoT devices as well as those devices that perform data conversion between digital and analog, such as analog-to-digital converter or digital-to-analog converter. However, designing a scalable calibration tool poses the following challenges:

- **Time synchronization.** The power calibration tool should provide a time synchronization mechanism between the target device and a high precision measurement equipment, such as DMM or oscilloscope. Otherwise, target and DMM may be calibrating values belonging to different voltage or current settings. With measurement equipment sampling rates of 1 MHz, these event samples must be time stamped within 1 µs accuracy or better.
- **High accuracy and dynamic range.** The power calibration tool must provide highly accurate current and voltage samples over a dynamic range that spans five orders of magnitude in current draw or voltage drop. The changes in the current or voltage events must be stable, otherwise pairwise correlation of values would be meaningless.
- **Volume calibration steps.** One of the challenges for data converter calibration is its non-linear distortion. A well-characterized voltage profile can accomplish the correction of data distortion. The power calibration tool should provide high resolution current and voltage change steps to give full coverage for data converter characterization.
- **Portability and low cost.** Compared to high-cost commercial power analyzers, the tool should be easy to integrate, both electronically and mechanically. It should also be less expensive and easy to use, compared to commercial products.

In the rest of this section, we present our design and formulate mathematical models to prove the operating range and resolution of ProCal. We then present the hardware and software implementation of ProCal.

2.1 Design Considerations and Solutions

Potentiometer Selection. The initial objective for calibration is to generate variable loads to provide current and voltage variations. Both digital and mechanical potentiometers can satisfy this objective. Table 1 compares these two types based on resolution, programmability, accuracy, and operating range.

Table 1. Comparison of mechanical and digital potentiometers. Digital potentiometers provide resistance resolution that is programmable and repeatable.

Potentiometer type	Resolution	Programmable	Accuracy	Operating range
Mechanical	Infinite	None	Low	Wide Range
Digital	Limited	SPI/I2C	High	Limited Range

Theoretically, the resolution of mechanical potentiometers is infinite. In practice, however, the effectiveness of the resolution is determined by the skill level of the potentiometer adjuster. Besides, mechanical potentiometers use a wiper to contact a resistive element that moves along its length to vary resistance. Because of the mechanical contacts involved, the resistive could vary by vibration, shock, and pressure.

On the other hand, digital potentiometers consist of CMOS transmission gates. Although digital potentiometers cannot offer an infinite resolution, they can provide a resolution that appears to be continuous across the supported range. More importantly, resolution of digital potentiometers is fully specified, repeatable, and predictable. Due to these benefits, as well as their tolerance for vibration, shock, and pressure [4], our design employs a digital potentiometer.

Dynamic Output Range. The standard analog-to-digital conversion has analog input $V_{in}(t)$ and digital output $D_{out}(t)$, where t represents time. The purpose of voltage calibration is to find a function $f(d)$ to be implemented in the correction module, where d is digital output. Function $f(d)$ is to minimize the error between the corrected output $v_c = f(D_{out}(t))$ and ground truth measurement $V_{in}(t)$ at time t. Similarly, for current calibration, a function $I_{in}(t)$ is derived.

Equation 1 defines the minimum resolution of ADC per digital bit (LSB), as follows,

$$LSB = \Delta = \frac{V_{FS}}{2^n} \tag{1}$$

where V_{FS} is the full-scale range of ADC determined by reference voltage and n is the maximum number of bits the input value can be translated to. Equation 2 defines the final digital output value of the ADC,

$$D_{out}(t) = \left\lfloor \frac{V_{in}(t)}{\Delta} \right\rfloor \bigg|_{t=n\times T_s} = \left\lfloor 2^n \times \frac{V_{in}(n \times T_s)}{V_{FS}} \right\rfloor \tag{2}$$

where T_s is sampling period. Based on this equation, normalization and truncation of analog inputs are involved in the ADC quantization process [1]. The maximum instantaneous value of distortion is $\frac{LSB}{2}$, and the total range of variation is from $-\frac{LSB}{2}$ to $+\frac{LSB}{2}$. In addition to quantization errors, there are two static specifications that define the accuracy of analog-to-digital conversion: *differential non-linearity* (DNL), and *integral non-linearity* (INL). DNL error is defined as the difference between an actual step width and the ideal value of 1 LSB. INL error is described as the deviation, in LSB or percentage of full-scale range, of an actual transfer function from ideal straight conversion line. More importantly, the INL error magnitude directly depends on the correlation position chosen for ideal straight line. Since $f(d)$ is non-linear, it is crucial to characterize ADC's full scale instead of only a few positions.

Typical energy estimation for an IoT device requires the measurement of both current and voltage. Current range is 0 mA to 1 A and voltage range is 0 mV to 5 V (i.e., CMOS operation range). A calibration tool should generate both variable currents and voltages. However, the critical challenge is to create various currents in a wide dynamic range. Specifically, since the digital potentiometer, wiper, and potentiometer connection are limited to the bounds of the power-supply rail, the potentiometer can only carry limited current and voltage. Most of the existing digital potentiometers support current in the range 0.6 to 3 mA, and only a few products can accommodate up to 20 mA. In any case, the potentiometer operating condition must be within the range of CMOS operation. This limitation is against the desired current range. To overcome this challenge, we connect a resistor network in parallel with the digital potentiometer to increase the total current flow. We first express the current output of digital potentiometer as an equation. Then, this equation is incorporated into a mathematical model, which models the scalability of ProCal. The current output range for a n-bit digital potentiometer can be expressed as follows

$$\frac{V_{in}(t)}{R(2^n) + R_b} \leq I_{pot}(t) \leq \frac{V_{in}(t)}{R(0) + R_b} \tag{3}$$

where t is time, $I_{pot}(t)$ is current output of digital potentiometer, $R(0)$ is resistance when digital potentiometer is programmed to 0, $V_{in}(t)$ is the input voltage, R_b is the resistor connected to digital potentiometer in serial, and $R(2^n)$ is digital potentiometer's maximum resistance value when programmed to 2^n. Through connecting R_b with the digital potentiometer's input rail in serial, we protect the digital potentiometer from overcurrent. The maximum current the digital potentiometer can support is expressed as follows

$$I_{potmax}(t) = \frac{V_{in}(t)}{R(0) + R_b} = \frac{V_{in}(t)}{R_w + R_b} \tag{4}$$

where I_{potmax} is maximum current digital potentiometer can accommodate, R_w is the constant wiper resistance when digital potentiometer is programmed to 0.

Base on Kirchoff's current law, the current flow into a device equals the current flow out of the device. Since digital potentiometer can only accommodate up to $I_{potmax}(t)$ continuous current, we add a resistor network in parallel with the digital potentiometer to increase the total current flow in order to support larger current range. The total current digital potentiometer and parallel resistor network can support is expressed as follows,

$$I_{ProCal}(t) = I_{potmax}(t) + \sum_{j=1}^{n} I_j(t) \quad \forall n \in Z^+ \tag{5}$$

where $I_{ProCal}(t)$ is the total current ProCal can support, $\sum_{j=1}^{n} I_j(t)$ is the sum of the current that the resistor network can support, and $I_j(t)$ is the current that a resistor R_j belonging to the resistor network can support. For any positive integer n, we can expand the current output range $I_{ProCal}(t)$ to satisfy any calibration range requirement as long as the given input voltage $V_{in}(t)$ is within the operation range of digital potentiometer and resistor network. With the capability to expand the output range, the current scalability of ProCal is proven.

Similarly, for voltage scalability, we connect a resistor R_c with the resistor network in serial. Based on Ohm's law, ProCal's voltage output range can be expressed as follows,

$$V_{ProCal}(t) = I_{ProCal}(t) \times R_c \tag{6}$$

where $V_{ProCal}(t)$ is the total voltage ProCal can support. With the capability to expand the $I_{ProCal}(t)$ range, we proved the voltage scalability of ProCal.

In addition to scalability, we want the ProCal's current and voltage output to be as close as possible to a linear function. Because ADC nonlinearity requires curve-fitting to achieve accuracy, we can improve calibration accuracy by providing a linear output over a wide range. In order to generate a linear output, $I_j(t)$ should have a linear relationship with $I_{potmax}(t)$. The linear relationship between $I_j(t)$ and $I_{potmax}(t)$ is expressed as follows

$$I_j(t) = k \times I_{potmax}(t) \quad \forall k \in N \tag{7}$$

where $I_{potmax}(t)$ is the maximum current of digital potentiometer, and each $I_j(t)$ current that resistor network can support is k times of $I_{potmax}(t)$. To support gradual increase of output current, $I_j(t)$ can be expressed as follows

$$I_j(t) = I_{j-1}(t) + I_{j-2}(t) \tag{8}$$

where every $I_j(t)$ is the sum of previous two terms, also known as the Fibonacci sequence. Therefore, we use induction to prove the maximum current as shown below

$$\sum_{j=1}^{n} I_j(t) = I_{n+2}(t) - 1 \quad \forall n \geq 2 \tag{9}$$

With the mathematical model proven, we implement our prototype with $n = 3$. However, based on Eq. 5, scalability of ProCal is not necessarily limited to $n = 3$. We select digital potentiometer AD5200 [16] to generate $I_{pot}(t)$ current. The reason behind selecting AD5200 is that it supports up to 20 mA, which is larger than most of the digital potentiometers currently available in the market. According to Eq. 3, we connect a 220 Ω resistor with AD5200 in serial as overcurrent protection resistor R_b.

Since $I_{potmax}(t) = I_1(t) = 20$, we choose $I_2(t) = 80$ and $I_3(t) = 100$ to implement our prototype, where $I_3(t) = I_1(t) + I_2(t)$ satisfies Eq. 8, $I_2(t) = 4 \times I_{potmax}(t)$ and $I_2(t) = 5 \times I_{potmax}(t)$ satisfy the linearity relationship of Eq. 7. We use ADG1612 [17] digital switches to control the number of resistors in the resistor network that are connected to the circuit to provide more significant current jumps, compared to what is provided by AD5200. In particular, initially, the digital potentiometer is used for introducing small changes in the current by programming the resistance value from high to low. After reaching the maximum current of digital potentiometer, a resistor on the resistor network, controlled by the digital switch, is enabled to provide additional current. Repeating this process results in gradually increasing the current through enabling additional resistors until reaching ProCal's maximum current output. Based on Eq. 6, a similar analysis can be employed for voltage.

Validation of Minimum Resolution. The minimum resolution of our design depends on the digital potentiometer's maximum end-to-end resistance R_{max} and the number of bits supported by digital potentiometer. The equation that determines the digitally-programmed output resistance is expressed as follows

$$R(x) = \frac{x}{2^n} \times R_{max} + R_w \quad 0 \le x \le 2^n \tag{10}$$

where n is the number of bits supported by digital potentiometer, x is the value programmed into the digital potentiometer, and R_w is the wiper resistance. Output resistance $R(x)$ is proportional to the digital potentiometer's end-to-end resistance R_{max}.

The minimum resolution of current output can be expressed as follows

$$I_{res}(t) = \frac{(R(x) - R(x-1))}{R(x) \times R(x-1)} \times V_{in}(t) \quad 0 < x \le 2^n \tag{11}$$

For example, ProCal uses an 8-bit digital potentiometer, AD5200 which has maximum end-to-end resistance 10 kΩ. Based on Eq. 10, resistance $R(256)$ is 10.282 KΩ and $R(255)$ is 10.242 kΩ. Considering ProCal has input voltage 5 V, and a 220 Ω resistor connected to the digital potentiometer in serial, we can calculate ProCal's minimum current output as 0.476 mA and the minimum current resolution as 1.82 µA, based on Eq. 11. Minimum current resolution and output current range can be further reduced or increased by choosing different specs of the potentiometer or supply voltage. A similar approach is used to calculate ProCal's minimum voltage resolution and voltage output range. We validated that ProCal's minimum voltage output is 0.06 mV and minimum voltage resolution is 0.01 µV.

Time Synchronization. The differences in the sampling offset and sampling rates of DMM and target device make correlating the readings a challenging task. For example, when an output value is configured, ADC might miss that value due to its lower sampling rate compared to DMM. If the design does not provide enough duration during two consecutive output changes, ADC and DMM can collect samples belonging to different output values. Therefore, we need to ensure that both DMM and IoT device sample output at least once during a configuration period. The next challenge is to ensure that when we compare two sampled values, the values belong to the same interval when the output is stable. Therefore, ProCal should also provide mechanisms to synchronize the readings of ADC and DMM. To overcome these challenges, ProCal stores voltage or current settling time information after applying each configuration that results in current or voltage change. For example, assume that the minimum sampling rate of ADC and target device is r_{min}. We first ensure that $T > \frac{1}{r_{min}}$, where T is the period of changing the output. ProCal provides this feature to support the sampling rate of various ADCs. Second, assuming that a new output value is configured at time t_i, to correlate the values collected by DMM and target IoT device, we find in their readings the values that are closest to time $t_i + \frac{T}{2}$. This synchronization approach ensures that the two values belong to the same output value.

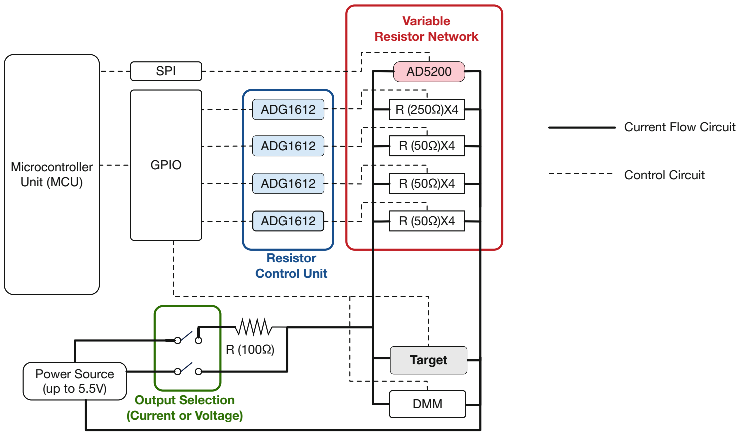


Fig. 1. Block diagram of ProCal. MCU provides synchronized trigger signals to the resistor control unit, target ADC, and DMM. AD5200 is a digital potentiometer, and ADG1612 is the digital switch connected to the resistor network.

2.2 Hardware Implementation

ProCal's hardware consists of four main units: (i) microcontroller unit (MCU), (ii) resistance control unit (RCU), (iii) variable resistor network, and (iv) output selection jumpers. MCU is used to control RCU, synchronize the operation of the target device and DMM, and record settling time of each output value generated by ProCal. RCU is responsible for controlling the variable resistor network. The output selection jumpers on the board enable the user to select current or voltage output for calibration.

Figure 1 presents the block diagram of ProCal. Figure 2 shows a ProCal board. We implement MCU function by developing a Python configuration code running on a Raspberry Pi 3 board. MCU connects to RCU through a SPI [19] bus and GPIO pins. MCU controls the RCU to perform the desired changes. MCU also provides a trigger signal through GPIO to trigger target device and DMM to start and stop sampling simultaneously.

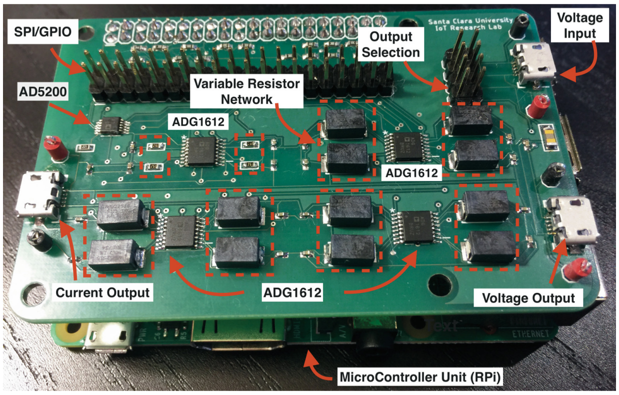


Fig. 2. ProCal prototype. ProCal is installed on top of a RPi. The RPi communicates with ProCal through SPI and GPIO interfaces. The dashed lines show the resistor network.

RCU contains one AD5200 [16] digital potentiometer and four ADG1612 [17] digital switches. AD5200 offers 256 different resistance values that can be digitally programmed by the configuration code through SPI. These different resistances provide full calibration coverage, from a low sleeping current to a high active current of IoT device, as we proved in Sect. 2.1. We connect a 220 Ω resistor with AD5200 in serial as the overcurrent protection. The salient features of AD5200 are its short setup time and 50 MHz SPI speed. These features enable ProCal to provide stable output current and voltage within 1 µs period.

The variable resistor network is a circuit connecting 16 resistors and a digital potentiometer, in parallel. ADG1612 contains four terminals, where each terminal connects to a resistor. MCU sends logic signals to ADG1612 terminals through GPIO. When the terminal receives a logic 1 signal, the resistor controlled by the terminal is connected to the variable resistor network, which increases the total current flow. If the terminal receives a logic 0 signal, the resistor is disconnected from the variable resistor network, which reduces the total current flow. There are four ADG1612s used in the RCU: one is connected to four 250 Ω high-precision (1%) resistors and the other three are connected to

Algorithm 1. Configuration Software of ProCal

```
function main($T$, $I_{max}(t)$, $V_{min}(t)$, $V_{in}(t)$, $n$)
    initialize GPIO and disconnect all resistors from the resistor network;
    setup SPI driver and bus speed to 50MHz;
    send start trigger to target through GPIO;
    $breakflag = 0$;
    for $i \leftarrow 0$ to number of $50\Omega$ resistors do
        if $breakflag == 1$ then
            break;
        if $i$ != $0$ then
            connect $50\Omega$ terminal[$i$] to the resistor network;
        for $j \leftarrow 0$ to number of $250\Omega$ resistors do
            if $breakflag == 1$ then
                break;
            if $i$ != $0$ and $j == 0$ then
                disconnect all $250\Omega$ from the resistor network;
            else if $j$ != $0$ then
                connect $250\Omega$ terminal[$j$] to the resistor network;
            for $x \leftarrow 2^n$ to $0$ do
                $R(x) = \frac{x}{2^n} \times R_{max} + R_w + R_b$;
                $I(t) = \frac{V_{in}(t)}{R(x)} + i \times 100 + j \times 20$;
                $V(t) = V_in(t) - R_c \times I(t)$;
                if $I(t) \geq I_{max}(t)$ then
                    $breakflag = 1$;
                    break;
                if $V(t) \leq V_{min}(t)$ then
                    $breakflag = 1$;
                    break;
                program AD5200 with value $x$;
                delay for $\frac{T}{2}$;
                write time stamp $\frac{T}{2}$, $I(t)$ and $V(t)$ into file;
                delay for $\frac{T}{2}$;
    send stop trigger to target and DMM through GPIO;
    return;
```

four 50 Ω high-precision (1%) resistors. These high-precision resistors connected to ADG1612 are part of the variable resistor network, which can provide 20 mA and 100 mA changes in current. It is worth noting that the effect of ADG1612 on current draw is minimal as it only shows a 1 Ω resistance when operating.

The output selection jumpers are used to switch ProCal's calibration output between voltage and current. When voltage output is selected, output selection jumper enables a high power 100 Ω resistor to connect in serial with the resistor network. It provides the same dynamic range for the voltage change between 0 to 5 V as we proved in Sect. 2.1. ProCal generates trigger signals through GPIO to start and stop data collection of target and DMM simultaneously. To improve stability and remove the alternating current caused by ripple voltage, we add low pass filters to the resistor network.

2.3 Software Implementation

ProCal's software runs on a flavor of Linux operating system called Raspbian and uses BCM and SPI library. The operations of the digital potentiometer and the resistor network are controlled by the software. This is the basis for setting up SPI and GPIO drivers, accurate time stamping of resistance change events, and sending synchronization signals. The software uses a collection of Python scripts that are used by MCU to trigger scheduled actions to start and stop a calibration process. The software programs the AD5200 and ADG1612 to provide output transitions, and records the settling time between two consecutive output changes.

As shown in Algorithm 1, the controller software first initializes all GPIO ports, sets up the SPI driver, and adjusts the SPI clock speed to 50 MHz. Users need to provide: (i) a desired time interval between configurations T, (ii) the maximum current $I_{max}(t)$ or minimum voltage $V_{min}(t)$ to calibrate, (iii) supply voltage to ProCal $V_{in}(t)$, and (iv) number of bits the digital potentiometer can support. ProCal can support output frequency (i.e., changes of current/voltage values) up to 50 MHz. After MCU sends the trigger signals to target device and DMM through GPIO, it starts to change resistance until the desired maximum current or minimum voltage is reached. The software time stamps every current/voltage change events and saves them into a log file. This log is used for time synchronization between the target device and DMM measurements.

3 Performance Evaluation and Case Studies

In this section, we evaluate ProCal's dynamic range, resolution, and stability. We show the effectiveness of ProCal to address the most challenging requirements of calibrating IoT devices. In particular, we present case studies of using ProCal to calibrate voltage and current with commercial ADCs.

3.1 Evaluation of Dynamic Range

Dynamic range and resolution are important performance metrics for a calibration platform. ProCal's output range and resolution can be customized to within the target ADC's range of interest. To measure dynamic range, resolution, and static accuracy, a high accuracy DMM, Keithley DMM7510 [23], is used to validate the results. DMM7510 provides picoampere-level sensitivity and sampling rate 1 Msps, which can precisely validate the performance of ProCal. When validating dynamic range, we connected DMM to ProCal's output to measure ground-truth current and voltage. For this experiment, output current changes every 5ms from 0.4 mA to 900 mA and voltage changes every 5 ms from 5 V to 0.06 mV.

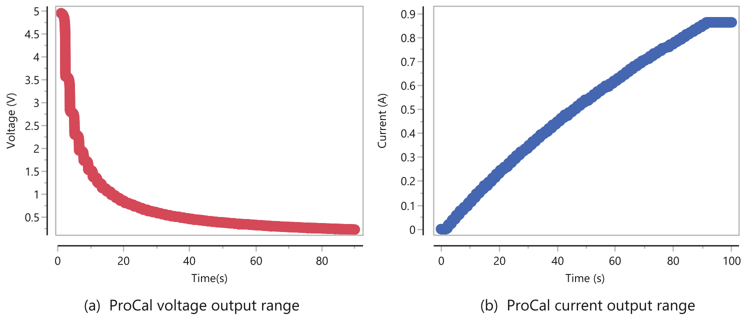


Fig. 3. Experimental results of ProCal prototype. (a) ProCal voltage output. Our prototype supports voltage output from 0.06 mV to 5 V. (b) ProCal current output. Our prototype supports current output from 0.476 mA to 900 mA. These current and voltage values address the requirements of calibrating various types of IoT devices.

Figure 3 shows the experimental results pertaining to ProCal's dynamic range validation. It can be seen that ProCal output current spans from 0.5 mA to 900 mA and voltage spans from 5 V to 0.06 mV. The results in this section validate the wide dynamic range and high resolutions.

3.2 Case Studies

In this section, we present case studies to demonstrate the benefits of calibration using ProCal. In these case studies, we use ProCal to perform real-world calibration of the most popular COTS ADCs, namely ATMega2560 [18], MCP3208 [22], and INA219 [24]. Our ground truth measurement is obtained by a high-accuracy DMM, Keithley DMM7510. In particular, we evaluate the performance of current and voltage calibration. To this end, we use ProCal to generate a dynamic range of voltage and current values. We use the target ADC and DMM to measure the output of ProCal simultaneously. After completing the measurements, we use the saved time stamp log to correlate the measurements between ADC and DMM. For each experiment, we generate a calibration function $f(d)$ or a calibration table to correlate ADC and DMM measurements. Note that as the log file reflects the stability time stamp of each configuration, we can safely compare two closest entries of the traces collected by ADC and DMM without requiring precise time synchronization of the two devices.

Figure 4 shows the current measurement error of INA219 and MCP3208 conducted in a normal indoor temperature 25 °C. As it can be observed, the error of both ADCs increases linearly versus current. Therefore, we find the best polynomial fitting curve to calibrate the errors.

Figure 5 shows the voltage measurement error of ATMega2560 and MCP3208. The error of both ADCs increases non-linearly versus voltage. For these cases, we used a calibration table to find the best fit of the ADC measurement.

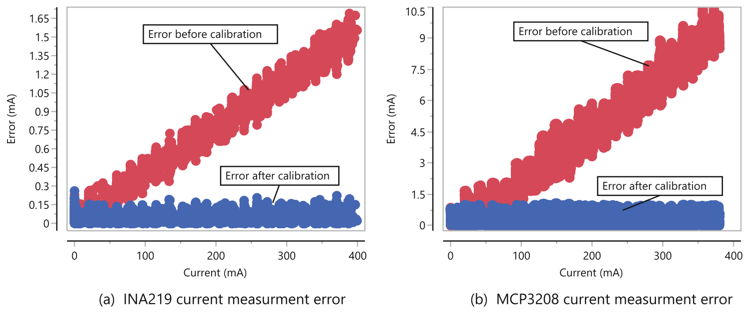


Fig. 4. Current calibrations using ProCal. Experimental results for (a) INA219, and (b) MCP3208 calibrations. These results show current measurement errors before and after calibration. Errors are significantly reduced after calibration.

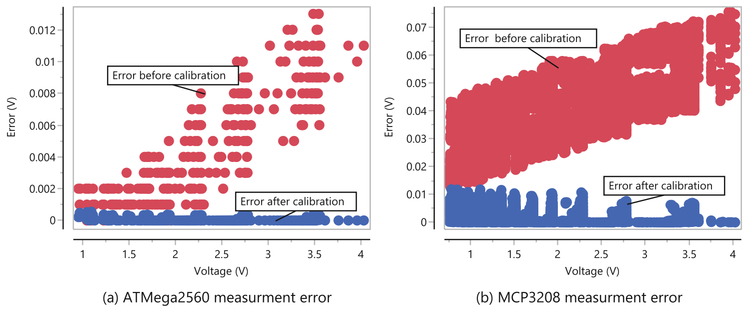


Fig. 5. Voltage calibrations using ProCal. Experimental results for (a) ATMega2560, and (b) MCP3208 calibrations. These results show voltage measurement error before and after calibration.

Table 2. Comparison of measurement errors before and after calibration

ADC	Measurement type	%Error (before)	%Error (after)
INA219	Current	0.42%	0.02%
MCP3208	Current	2.58%	0.01%
MCP3208	Voltage	5.29%	0.01%
ATMega2560	Voltage	0.2%	0.01%

Table 2 presents the calibration results, and confirms the significant effect of ProCal calibration on measurement accuracy. These results demonstrate that by providing an extensive calibration range, the errors are reduced to nearly 0.01%. For example, in the MCP voltage case, the ADC error reduces from 5.29% to 0.01%.

4 Related Work

Data conversion accuracy is one of the main requirements of IoT applications. However, there are various factors affecting the precision of data conversion. For example, device characteristic, process technology, resistance of the circuit, temperature variations, and finite gain, are among the parameters that affect accuracy. Standard methods to minimize the effects of these parameters are *trimming* and *calibration*. Trimming is performed during the production phase. After measuring and correcting the parameters of the device at a controlled condition, the trimmed values are programmed into the device. Trimming values, however, cannot be changed after manufacturing. Therefore, the subsequent drift due to device aging, temperature, or system-level noise, cannot be corrected.

Calibration can be performed multiple times after the device is fabricated. Therefore, it can be used to compensate for changes that occur over time and those influenced by environmental conditions. There are two types of calibrations: *self-calibration*, and *external calibrations*. Self-calibration performs the measurement-correction process within the device itself and does not require an external device. The calibration process that interrupts the ordinary data conversion process and is performed during device power up is referred to as foreground calibration or power up calibration [3,10,11,29]. Although this technique results in significant error reduction, it cannot cope with temperature drift and other sources of inaccuracy such as sensor-to-ADC path loss. Alternatively, data converter can be calibrated while it is performing the regular conversion, and the result of conversion would not be affected by the calibration process. This is referred to as background calibration or runtime calibration [14,30]. Background calibration encompasses temperature effect, which provides a higher absolute accuracy than foreground calibration.

However, self-calibration is only as accurate as the onboard reference voltage. It can drift slightly over time, and thereby, external calibration is still essential. For example, although the 12-bit SAR ADC of STM32F205 includes a background calibration feature, Hartung et al. [8] used two resistors to perform external calibration, and they showed that voltage and current errors are reduced from 2.36% and 2.21% to 0.87% and 1.56%, respectively. Similarly, Zhou et al. [31] relies on external calibration, although TI MSP430F2618 has a background calibration capability. The common approaches of external calibration include using: (i) fixed resistors [9,13], (ii) mechanical potentiometer [7,31], and (iii) commercial power analyzer [12,15]. However, as we mentioned in Sect. 1, these calibration mechanisms pose several limitations. Resistors and potentiometers cannot provide quality calibrations due to limited range or unreliable resistance. Furthermore, although commercial power analyzers can provide quality results, they are costly. The proposed ProCal tool provides a cost-efficient solution without imposing any of the limitations of the aforementioned solutions.

5 Conclusion

Calibration is an important step towards building reliable IoT systems. For example, accurate sensor reading requires ADC calibration, and power monitoring chips must be calibrated before being used for measuring the energy consumption of IoT devices. In this paper, we presented a power calibration tool, called ProCal, which is a programmable and scalable tool. ProCal is a low-cost programmable platform that provides dynamic voltage and current output for calibration. The basic idea is to use a digital potentiometer connected to a parallel resistor network controlled through digital switches. The resistance and output frequency of ProCal is controlled by a software communicating with the board through SPI and GPIO interfaces. We incorporated the concept of Fibonacci sequence into our mathematical model to prove that ProCal can be extended to other and wider ranges. Our extensive experimental case studies demonstrated that ProCal can reduce measurement errors significantly. In addition to using ProCal for calibration, it can be employed for power emulation. Specifically, instead of using an IoT board to generate sudden variations in current, ProCal can be used to emulate the operation of an IoT device under various conditions. ProCal can also be used, for example, to test the resilience of a power harvesting system under various types of loads.

References

1. Bennett, W.R.: Spectra of quantized signals. Bell Labs Tech. J. **27**(3), 446–472 (1948)
2. Bychkovskiy, V., Megerian, S., Estrin, D., Potkonjak, M.: A collaborative approach to in-place sensor calibration. In: Zhao, F., Guibas, L. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 301–316. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36978-3_20
3. Cao, J., Meng, X., Temes, G.C., Yu, W.: Power-on digital calibration method for delta-sigma ADCs. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2002–2005. IEEE (2016)
4. Creech, J., Rice, D.: Digital potentiometers vs. mechanical potentiometers: Important design considerations to maximize system performance. Analog Devices, MA, USA, Technical article (2015)
5. Dezfouli, B., Amirtharaj, I., Li, C.C.: EMPIOT: An energy measurement platform for wireless IoT devices. arXiv preprint arXiv:1804.04794 (2018)
6. Dezfouli, B., Radi, M., Chipara, O.: REWIMO: a real-time and reliable low-power wireless mobile network. ACM Trans. Sensor Netw. (TOSN) **13**(3), 17 (2017)
7. Haratcherev, I., Halkes, G., Parker, T., Visser, O., Langendoen, K.: Powerbench: A scalable testbed infrastructure for benchmarking power consumption. In: International Workshop on Sensor Network Engineering (IWSNE), pp. 37–44 (2008)
8. Hartung, R., Kulau, U., Wolf, L.: Distributed energy measurement in WSNs for outdoor applications. In: 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), pp. 1–9. IEEE (2016)
9. Jiang, X., Dutta, P., Culler, D., Stoica, I.: Micro power meter for energy monitoring of wireless sensor networks at scale. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks, pp. 186–195. ACM (2007)

10. Karanicolas, A.N., Lee, H.S., Barcrania, K.L.: A 15-b 1-Msample/s digitally self-calibrated pipeline ADC. IEEE J. Solid-State Circuits **28**(12), 1207–1215 (12 1993)
11. Lee, H.S., Hodges, D.A., Gray, P.R.: A self-calibrating 15 bit CMOS A/D converter. IEEE J. Solid-State Circuits **19**(6), 813–819 (1984)
12. Lim, R., Ferrari, F., Zimmerling, M., Walser, C., Sommer, P., Beutel, J.: Flocklab: a testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In: Proceedings of the 12th International Conference on Information Processing in Sensor Networks, pp. 153–166. ACM (2013)
13. Milenkovic, A., Milenkovic, M., Jovanov, E., Hite, D., Raskovic, D.: An environment for runtime power monitoring of wireless sensor network platforms. In: Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory (SSST), pp. 406–410. IEEE (2005)
14. Moon, U.K., Song, B.S.: Background digital calibration techniques for pipelined ADCs. IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process. **44**(2), 102–109 (1997)
15. Pötsch, A., Berger, A., Springer, A.: Efficient analysis of power consumption behaviour of embedded wireless IoT systems. In: IEEE International on Instrumentation and Measurement Technology Conference (I2MTC), pp. 1–6. IEEE (2017)
16. Analog Devices Inc.: 256-Position Digital Potentiometers (2012). http://www.analog.com/media/en/technical-documentation/data-sheets/AD5200.pdf
17. Analog Devices Inc.: Quad SPST Switches (2015). http://www.analog.com/media/en/technical-documentation/data-sheets/ADG1611_1612_1613.pdf
18. Arduino Inc.: ARDUINO MEGA 2560 REV3 (2017). https://store.arduino.cc/usa/arduino-mega-2560-rev3
19. Freescale Semiconductor: Official SPI block guide v03. 06 (2014)
20. IEEE collaboration and others: IEEE Standard for Terminology and Test Methods for Analog-To-Digital Converters. IEEE Std., pp. 1241–2000 (2011)
21. Keysight Technologies: Keysight N6700 Modular Power System (2017). https://www.keysight.com/en/pc-851482/n6700-modular-power-system?pm=SC
22. Microchip Inc.: MCP3204/3208 2.7V 4-Channel/8-Channel 12-Bit A/D Converters (2017). http://ww1.microchip.com/downloads/en/DeviceDoc/21298c.pdf
23. Tektronix, INC.: DMM7510 $7\frac{1}{2}$-Digit Graphical Sampling Multimeter (2017). https://www.tek.com/tektronix-and-keithley-digital-multimeter/dmm7510
24. Texas Instruments: INA219 Zerø -Drift, Bidirectional Current/Power Monitor With I2C Interface (2015). http://www.ti.com/lit/ds/symlink/ina219.pdf
25. Texas Instruments: Selecting an A/D Converter (2015). http://www.ti.com/lit/an/sbaa004a/sbaa004a.pdf
26. Vishay Spectrol: Pot 10K Ω (2014). http://www.vishay.com/docs/57093/860.pdf
27. Suchanek, P., Haasz, V., Slepicka, D.: ADC nonlinearity correction based on INL (n) approximations. In: IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), pp. 137–140. IEEE (2009)
28. Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., Culler, D.: An analysis of a large scale habitat monitoring application. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 214–226. ACM (2004)
29. Taft, R.C., Menkus, C.A., Tursi, M.R., Hidri, O., Pons, V.: A 1.8-V 1.6-Gsample/s 8-b self-calibrating folding ADC with 7.26 ENOB at nyquist frequency. IEEE J. Solid-State Circuits **39**(12), 2107–2115 (2004)

30. Tsang, C., Chiu, Y., Vanderhaegen, J., Hoyos, S., Chen, C., Brodersen, R., Nikolic, B.: Background ADC calibration in digital domain. In: IEEE Custom Integrated Circuits Conference (CICC), pp. 301–304. IEEE (2008)
31. Zhou, R., Xing, G.: Nemo: A high-fidelity noninvasive power meter system for wireless sensor networks. In: ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 141–152. IEEE (2013)

Research on the Design of Human Settlement System and Personal Air Purification System Under the Background of Internet of Things

Yi Su(✉), Gang Liu, and Lei Zhang

Tsinghua University, Beijing, China
bfsuyi@126.com

Abstract. This paper discusses my research on human settlement system and personal air purification system based on mobile internet, internet of things, cloud computing, and big data analysis technologies under the condition that air pollution in China seriously affects people's normal life. The design and development of personal air purification system is mainly designed by optimizing the principle of anion air purification. The design background, design principles, design innovations, and design results of the system are described in detail. The problems encountered in the R&D process and solutions are described in detail in conjunction with actual products. Since the product was introduced, this product has been widely recognized by the market and has achieved a lot of honor and research patent achievements.

Keywords: Haze · Air purifier · Human settlement system · Anion
Internet of things · Big data · Wearable device

1 Research Background

1.1 The Perception of PM2.5 in Chinese Cities

Air pollution, water pollution, noise pollution and even light pollution have long been known and valued. The word "Haze" has gradually been recognized by us since 2011 because of the continuous release of PM2.5 readings by the US embassy.

Nowadays, people attach great importance to haze, and people will always pay attention to the numerical changes of PM2.5. The Chinese government has managed and improved the air quality index through various methods, but we cannot improve the over-all environment quickly. Recently, China's market for purifiers is booming because aforementioned problems. Air pollution can cause respiratory diseases and cardiovascular diseases, and some sensitive people may have obvious discomfort or complications in a short period of time. The simplest protection is to wear masks with anti-haze effects, so the local mask market can even be sold out when the severe smog comes, but people can't wear masks sustaining. Most people have a misconception that indoor air is better than outside, so it's safe indoors. In fact, according to the experiment, even if the doors and Windows are closed, indoor PM2.5 concentration is between 1/3 and 2/3 of the outdoor concentration. So people spend a lot of time indoors, and the amount of smog they breathe is higher than in outdoor activities.

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 106–118, 2018.
https://doi.org/10.1007/978-3-319-94370-1_8

1.2 Usage Scenario Analysis for Personal Air Purification System

Before designing the system, we need to analyze the user's usage scenarios first. It is divided into three main scenarios: Home, Outdoor and Public Place. The main feature of the first scene "Home" is the space fixed, the environment is relatively stable, the activity range is small, high frequency of use at night. The second scene is mainly "Outdoor" and "Open Space". Its characteristics are mainly in the outdoor, moving, the surrounding environment changes fast and so on. The third public place is mainly characterized by large indoor space, tremendous flow of passengers. In addition to public building space, it also includes public transportation such as buses, subways, high-speed trains and taxis. How to solve the multi-scene air purification problem at the same time and how to provide a healthy air environment to the user all-weather is the biggest challenge of the system. The wearable device is a good direction. After many attempts, we finally chose the anion technology to introduce a purifier that can be carried around. Its main features are wearable, low energy consumption, zero noise, zero pollution and excellent purification effect.

The personal air purifying system provides air purification and monitoring service for users in various usage scenarios through anion technology and Internet of things technology. There are already a lot of purifiers in the market, but there are few solutions for a full range of personal air purification systems. The architecture of personal air purification systems based on technology such as the Internet of things, cloud computing and big data analysis. (see Fig. 1).

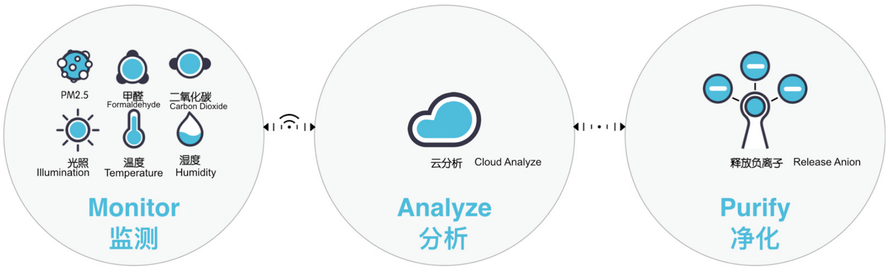


Fig. 1. The framework of the personal air purification system

2 Innovation of Personal Air Purification System Under the Background of Internet of Things

What is about the personal air purification system? In simple terms, there are mainly air purification, air monitoring, data transmission, cloud analysis, remote control, automatic work and data analysis.

The development of new technology and product system should be based on the relevant infrastructure construction and the development of relevant technology. For example, the important reason for the rapid development of the "ofo" and "mobike" sharing bikes in the Chinese market is the improvement of the environment of mobile Internet technology and mobile payment system. The development of smart home is also based on the huge development of Internet of Things and cloud-computing technology, so the data generated by these smart home products can be stored and analyzed.

The living environment system (see Figs. 2 and 3) can satisfy both managers and ordinary users to check the relevant data of air quality. The manager can log on to the management system and check the air quality of multiple rooms in real time. Ordinary users can check the air quality of the current room through mobile devices such as

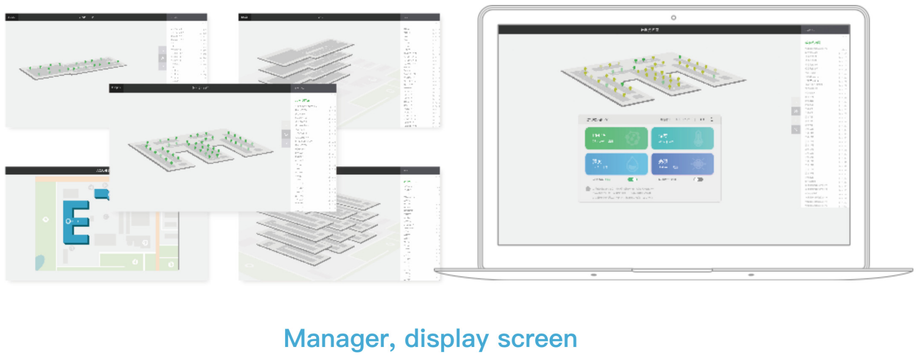


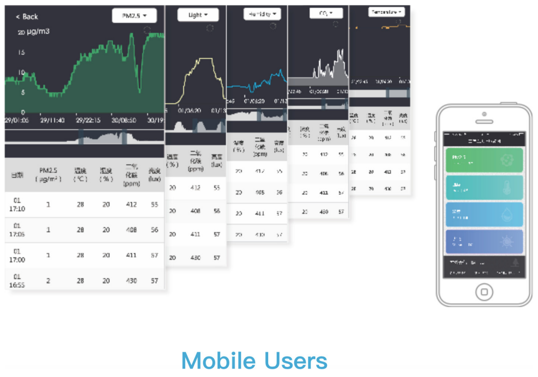
Fig. 2. The living environment system

mobile phones.

Why isn't the smart home market moving faster than sharing a bike? Because of the connection cost of Internet of Things, network security, user demand, business model etc., the Internet of Things technology is implemented slowly. Without complete infrastructure construction and mature technology application scale, it is difficult to make household products really intelligent. There is no guarantee that users' data and privacy security is the biggest threat. If the home intelligence security camera was hacked, then personal privacy and property security will be seriously threatened.

3 Design Principle of Air Purification System

First of all, we will compare and analyze the technical principle of existing air purification. At present, the widely used purification technology in the market is mainly three kinds: fiber filtration, electrostatic adsorption and anion. According to the new technology trend and market environment, it is easy to figure out which technology is best suited for the intelligent air purification system according to the following comparative analysis.



日期	PM2.5 (μg/m³)	温度 (℃)	湿度 (%)	二氧化碳 (ppm)	光照 (lux)
01 17:10	1	28	20	412	55
01 17:05	1	28	20	408	56
01 17:00	1	28	20	411	57
01 16:55	2	28	20	430	57

Fig. 3. The App of the living environment system

3.1 Fiber Filtration

This technology is the most widely used and the least technically difficult technology in the market. We call it air purification 1.0 technology. Its principle is in air to the machine, through internal filters and filter physics will air dirty things such as separate layers of filtration, finally, the clean air out of the machine outside so reciprocating cycle. The disadvantage of this technique is that it is noisy in use, and it is necessary to replace the filter and filter element frequently. If the filter element is not replaced in time, it will cause secondary pollution, leading to the high maintenance cost.

3.2 Electrostatic Adsorption

The purification principle of this technology is different from fiber filtration purification. We call it air purification 2.0 technology. The principle is to inhale the air through a fan, and the dust in air positively charged by the positive wire. Then it goes into the anion- charged multi-layer metal dust collector. Thus, the positively charged particles will rapidly adsorb on the dust-collecting device with negative power to achieve the purpose of dust removal and purification. Its advantage is that it does not need to consume material, purify efficiency is high, the disadvantage is if control was not good discharge voltage and so on condition can produce ozone, and this kind of principle air purifier general price is high.

This kind of electrostatic adsorption purification technology is mostly applied to the new wind system. Although the new wind system has been widely used, the main problem of the new wind system is that the installation is very complicated. Moreover, the fresh air system can cause secondary pollution of the air if the ventilation ducts and improper use are not regularly cleaned.

3.3 Anion

This air purification technology is the latest and most effective new technology that we call air purification 3.0. Its technical principle is to release negatively charged ions into the air through an anion generator, which binds the negatively charged ions to the airborne particles that are less than PM2.5. The small particles accumulate and form large particles and are eventually settled by gravity, thus achieving the effect of air purification (see Fig. 4), its advantages are that there are no consumables, no radiation, no noise, low energy consumption, sterilization, and sleep improvement. The disadvantage is that it is difficult to control the release of ozone. But through our purification system to the whole optimization design, adopted new booster module to the traditional technology upgrading, ensure air purification with excellent results at the same time avoid the secondary pollution of ozone. The interpretation of its working principle is also in the previous period of time in the column of CCTV channel 10 *Approaching Science* to be invited to show, made more people understand and believe that the reliability and practicability of this technology.

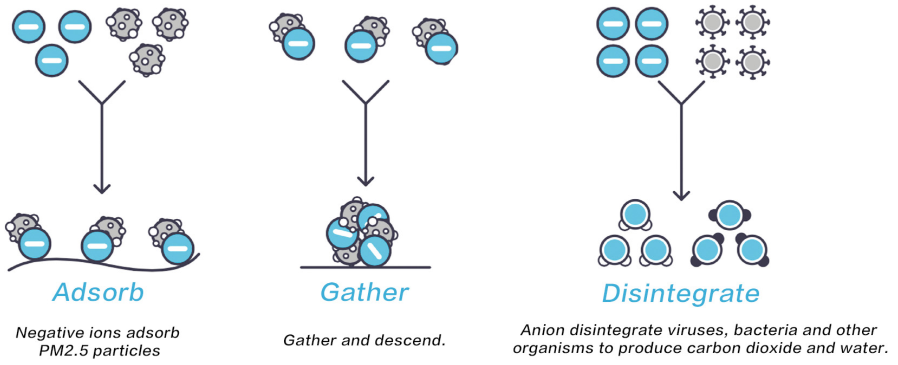

Adsorb
Negative ions adsorb PM2.5 particles
Gather
Gather and descend.
Disintegrate
Anion disintegrate viruses, bacteria and other organisms to produce carbon dioxide and water.

Fig. 4. Principle of anion purification

After the above analysis, the anionic air purification principle is finally selected. After technical improvement and optimization, professionals and authorities have also detected the purification effect. As is shown in the Fig. 5, various agencies also provide authoritative inspection reports. Because the anion generator module we developed is small enough to design as a wearable device. With this portable air purifier as the starting point, a whole set of personal respiratory purification systems has been developed to improve our living environment.

After using the new purification system, the indoor air environment can be improved obviously and the purification effect can be viewed remotely and in real time. As is shown in the Fig. 6, there is a huge difference in air between inside and outside the classroom and the data is updated in real time.

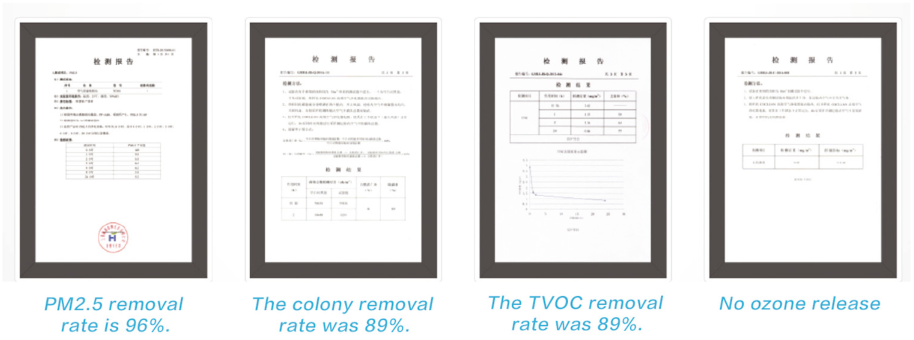

PM2.5 removal rate is 96%.
The colony removal rate was 89%.
The TVOC removal rate was 89%.
No ozone release

Fig. 5. The authoritative inspection reports

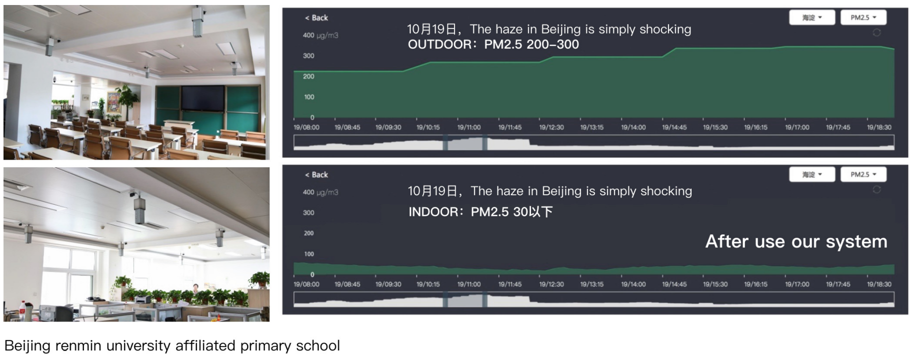
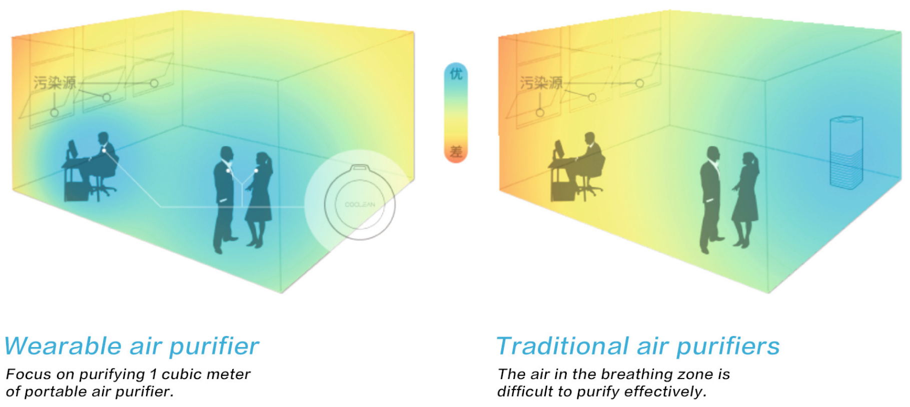


Fig. 6. Indoor and outdoor air quality comparison.

In addition to the technical breakthrough, in the air purification form has also been reconsidered. The core of this personal air purification system is to purify and monitor the air in people’s breathing areas. Other air purifiers clean up all the air in the space in order to complete the purification (see Fig. 7). The process is about 30 to 40 min or more, so it’s less efficient. Such purifying duration may be acceptable to the user, but the noise caused by its high power is unacceptable to most part of users. And if the work is not sustained, indoor PM2.5 levels will rise slowly. Noise can greatly affect people’s sleep quality. So the air that directly purifies the breathing area reflects the idea of efficient purification. After all, the air we care about most is the air that is inhaled into our bodies. Of course, we also developed a large anion air purification system to improve the indoor air quality. The product has high precision environmental monitoring function and can monitor the temperature, humidity and PM2.5 changes. Through the real-time air condition, the device will automatically open the purification or enter the sleep state. At the same time, the data will be feedback to the background environment of service system, so you can learn the same place of indoor and outdoor air condition, to make the building of the whole environment improvement plan has accumulated a large amount of data. When enough indoor and outdoor air data are collected, professional analysis results and recommendations for habitat improvement can be provided.



污染源
优
差
Wearable air purifier
Focus on purifying 1 cubic meter of portable air purifier.
Traditional air purifiers
The air in the breathing zone is difficult to purify effectively.

Fig. 7. Comparison of purification methods

4 Analysis of the Effect of Personal Air Purification System

4.1 Anion Concentration Contrast in Everyday Situations

By optimizing the anion generator, the high active negative oxygen ions can be released under the condition that no ozone is produced. Anion can cause the particles in the air to be charged with particles (PM2.5), smoke, pollen and other particles and then settle down with other uncharged particles around them. Anion can also cause the hydrogen bond of bacteria, viruses and other organisms to break down to the effect of sterilization. The concentration of negative oxygen ions in the 20 cm range is 3–8 million/cm^3. The concept of this number and the anion concentration in the natural environment will be very intuitive (see Fig. 8). Wearing such an air purifier can guarantee the air safety of 1 m^3 in the respiratory area. And it would be better to work with large anion purifiers, because large anion purifiers can release 60 million/cm^3 of highly reactive oxygen ions into the chamber every second.

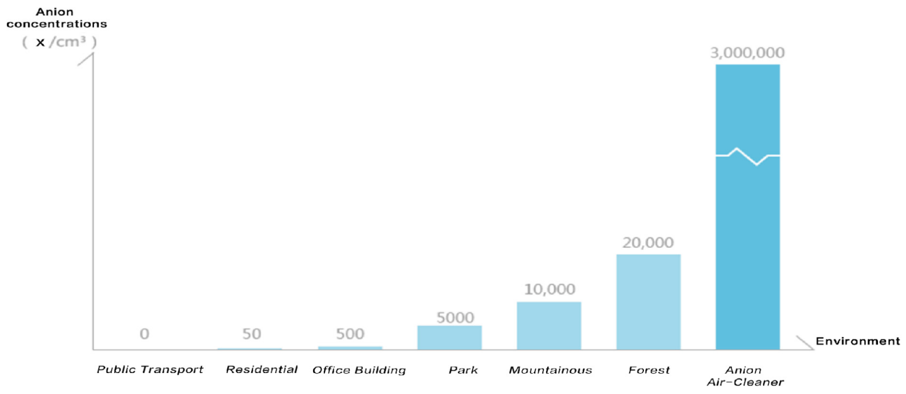


Fig. 8. Anion concentration contrast in everyday situations

4.2 The Advantage of Anion Purification Technology

The advantage of anion purification technology is that it does not produce radiation, noise and consumables. Compared with the noise and consumables produced by traditional air purifiers, we have caused other pollution to our environment. Because there is no noise, we can use it in sleep, work, meetings and study without affecting other people. Also, the portable purification module USES very low energy consumption, and the built-in lithium battery can be fully charged for half an hour and work continuously for 10 h. 10 h can basically meet the need to go out for a day. The appearance of the product is shown in Fig. 9.

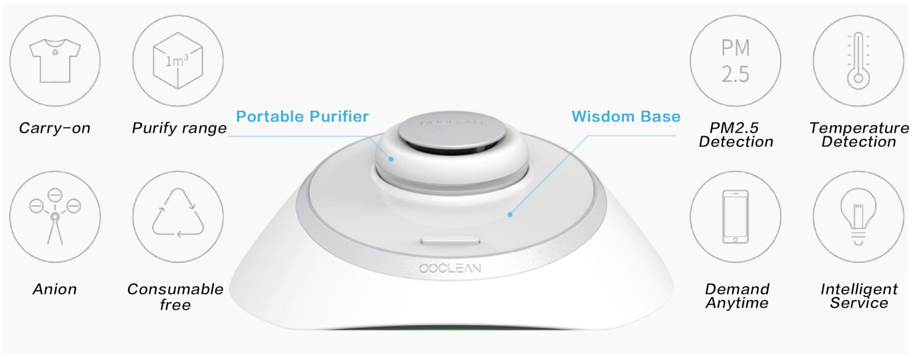


Fig. 9. The product renderings

4.3 The Smart Center Base

The purification system also designed the smart center base of the personal health respiratory system (hereinafter referred to as the base). The base part is the brain of a person's health respiratory system, which can monitor PM2.5 levels, temperature and humidity in the surrounding environment. These data would be uploaded to mobile phones in real time by bounding to WeChat, so that people can master the air quality of the places they care about. Now that the product has been purchased by many cities across the country, we can also learn about the indoor and outdoor environmental data using the product at the same time through the sharing of data. These data are analyzed through our cloud data, which is of great significance for the scientific improvement of human living environment.

The base part not only can monitor and share data, but also can charge the mobile purification module and enhance the purification effect. We use the special five-pin charging interface to achieve the functions of charging, data transmission and positive power export. Since the charging port of the base and the mobile purification module is the same, the charging of two devices can be combined separately. When two modules combine to work together, they can also enhance the purification effect of charging (as shown in Fig. 10) during sleep, if placed near the head of the bed, it can actually create a better sleep environment for users.

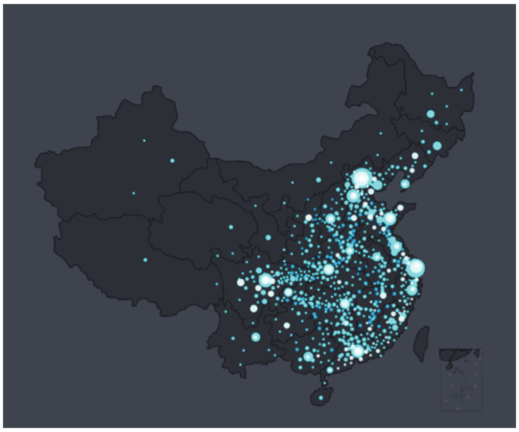


Fig. 10. The user maps

5 Analysis of the Wearing Forms of Personal Air Purifiers

At the beginning of product design, we also studied product positioning and target users. The background of the design is also fully introduced, so what kind of purification products can be made to stand out among the many air purifiers in this context? First of all, we have a patented technology to purify module is small, the core of the first generation product purification module size only 24 mm * 14 mm * 10 mm. The size like a pen is suitable design as a portable product, but the main question is how to wear it and whether to consider multiple ways of wearing it. We did a lot of experimentation and analysis. The core idea after this analyst is that design the purifier wearable to distinguish between traditional large air purifiers.

5.1 The Form of Wristbands and Watches

The most popular wearable device at the time was the sports bracelet, but after research, our products were not suitable for wearing on the wrist. And lead us to give up the main reason is because of the concept of purification module is the most effective purification area is 1 cubic meters, if wear on your wrist will move away from breathing to greatly weaken the purification efficiency.

5.2 Head Wear

This mainly form of product is VR equipment and hat products will be worn by people. Because these products are not suitable for long wear, they do not meet the purpose of clean air at any time.

5.3 The Form of a Brooch or a Badge

The placement of the brooch and the breastplate is very appropriate, but they are very small and very thin products. It is difficult to make air purifiers so thin, so this form is not appropriate.

5.4 In the Form of Arms Package

This type of product is mostly used in the fitness process, with a mobile phone or purse strings attached to the big arm. This form is more suitable for exercise, wearing the scene is still limited.

5.5 In the Form like Necklace Pendant

This form is the way we finally consider how to wear the product (see Fig. 11). First of all, the form is the simplest and most suitable for various scenarios. Second, the product needs to be exposed to human skin for technical reasons. So it's easy to use the form of a necklace. One of the most important reasons is to wear the product's position so that the anions released by the core purification module are close to the human respiratory area. The necklace form can be the most efficient and simple way to solve all the problems; it is the best choice for wearable.

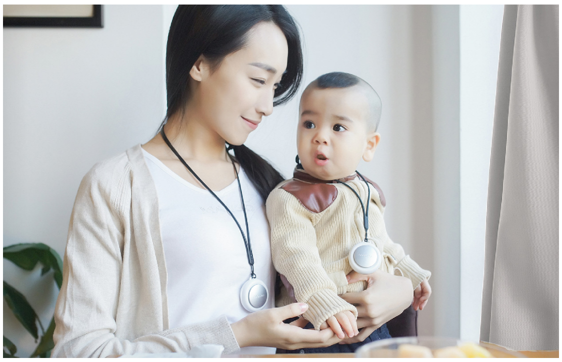


Fig. 11. In the form like necklace pendant

6 The Other Application of Anion Purification System

In order to solve the overall human living environment, we have also made the car version and the high power version after making the portable air purifier.

6.1 Car Version

At the beginning of the design, to facilitate product expansion, the power connector of the product was made into a universal interface. So by using a custom car charger, you can easy to turn a common version of a purifier into a car version. Traditional car purifier in engine condition will not be able to work long and even immediately shut down, and the use of anions produced by the principle of car purifier because of very low energy consumption, the built-in lithium battery sustainable work more than 10 h. Every time the user enters the car, it is already a clean car environment. We call it parking purification. The average driving time of the customer is about 45 min, but the cleaning time is 20 min. If you start cleaning up, half of your driving time is not breathing fresh air. The effect comparison is shown in Fig. 12.

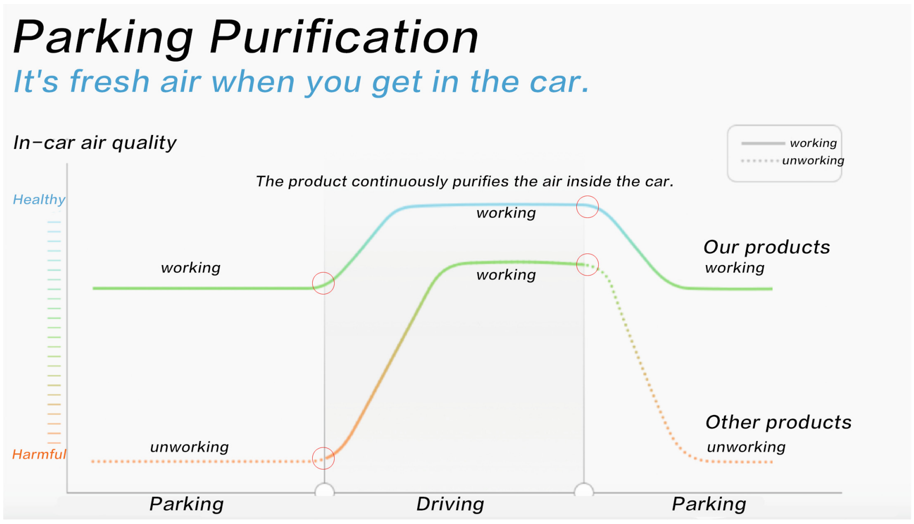


Fig. 12. Parking purification

6.2 High Power Version

With the same technology, we developed a high power version of the indoor anion air purifier. The high power version of the anion purifier has been widely used in families, schools, hospitals, hotels and factories across the country. Product installation and construction is also very simple. The ceiling version is mounted like a chandelier, while the landing version is more like a floor lamp. Therefore, the product does not need to punch holes in the wall or replace the filter element. Compared to the traditional large air purifier and the new air system anion purifier has a great advantage. The anion release concentration of high power purifier is 60 million/cm^3. And the energy consumption is only 3w, and the electricity cost is only 15 Yuan per year. This product has no consumables, noiseless, low energy consumption, convenient construction, intelligent monitoring, etc., which has a strong market competitiveness compared with other products. As shown in the Fig. 13, we collect users' indoor air quality statistics, which can be used to analyze data to obtain users' usage habits and rules of life, so as to make accumulation for intelligent settlement of human settlement environment.

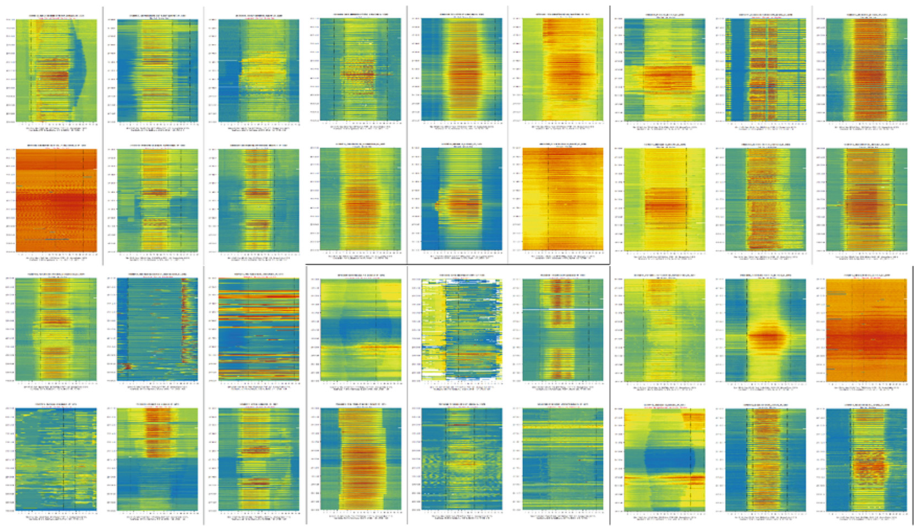


Fig. 13. User's indoor air quality detection data

7 Design Impact and Results

The personal air purification system has achieved mass production, has been enjoyed by a wide range of consumers and has received many honors and patents. Through the wide acceptance of the product by the market, the company is also rated as national high-tech enterprise. A lot of experience and achievements have been accumulated in focusing on human settlements environment and multi-parameter solutions to the Internet of Things. Through some air sensitive actual user's feedback, the personal air purification system improves their living conditions and reduces the discomfort caused by air quality and pollen. And users who insist on wearing products daily report that they have not had a cold or fever for nearly one year, because the industry also has the function of sterilization and disinfection. Large anion air purifier has been widely applied to classrooms, universities, primary and secondary schools, kindergartens and public spaces, and effectively improve the air quality of public space and to reduce the cross infection of bacteria. Users and managers can also check the indoor air quality at any time through smart phones, so that the purification effect can be visualized and monitored.

8 Conclusion

Through the design of personal air purification system, it is found that the solution proposed by combining the Internet of Things can solve the problem of air purification more intelligently and comprehensively. Although the whole system is still imperfect, but it provides a complete solution for the personal air purification system compared with the air purification products in the market. Its convenient and efficient purifying concept is also designed to fully consider the living environment and habits of urban people. After a period of market testing, most of the feedback is good. Some of the return users are sensitive to the air, and feedback that the product has really changed the quality of their lives. Some of the users bought the product specifically to put it on the bedside of the babies, which moved us a lot. We hope that our products can help more people to improve the living environment around us.

Acknowledgements. Thanks for the support of the Beijing Zhongqing Technologies, Co., Ltd. and my tutor, professor Zhang Lei.

References

1. Philip Chen, C.L., Zhang, C.-Y.: Data-intensive application, challenges, techniques and technologies: a survey on big data. Inf. Sci. **275**(11), 314–347 (2014)
2. Hsu, C.-H.: Intelligent big data processing. Future Gener. Comput. Syst. FGCS **36**(3), 16–18 (2014)
3. Grinshpun, S.A., Mainelis, G., Trunov, M., et al.: Evaluation of ionic air purifiers for reducing aerosol exposure in confined indoor spaces. Indoor Air **15**(4), 235–245 (2005)
4. Mayya, Y.S., Sapra, B.K., Khan, A., et al.: Aerosol removal by unipolar ionization in indoor environments. J. Aerosol Sci. **35**(8), 923–941 (2004)

Application and Industry Track

FPGA Hardware Implementation of Smart Home Autonomous System Based on Deep Learning

Basman M. Hasan Alhafidh[1,3](✉), Amar I. Daood[1,3], Mohammed M. Alawad[4], and William Allen[2]

[1] Department of Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL, USA
`bhasan2012@my.fit.edu`
[2] Department of Computer Sciences and Cybersecurity, Florida Institute of Technology, Melbourne, FL, USA
[3] Department of Electrical and Computer Engineering, University of Mosul, Mosul, Iraq
[4] Biomedical Sciences, Engineering, and Computing Group, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract. The use of deep learning algorithms, as a core element of artificial intelligence, has attracted increased attention from industrial and academic institutes recently. One important use of deep learning is to predict the next user action inside an intelligent home environment that is based on Internet of Things (IoT). Recent researcher discusses the benefit of using deep learning based on different datasets to assist their result. However, assuring the best performance to satisfy real-time applications leads us to use a real-world dataset to make sure that the designed system meets the requirements of real-time applications. This paper uses the MavPad dataset which was gathered from distributed sensors and actuators in a real-world environment. The authors use simulation to investigate the performance of a multilayer neural network that predicts future human actions. The authors also present a hardware implementation of the deep learning model on an FPGA. The results showed that the hardware implementation demonstrated similar accuracy with significantly improved performance compared to the software-based implementation due to the exploitation of parallel computing and using optimization techniques to map the designed system into the target device. Additionally, our implementation of FPGA-based neural network system supports its future utilization for other applications.

Keywords: IoT · Smart environment · Middleware
Machine Learning Algorithm · Neural network
Hardware implementation · FPGA · Embedded systems

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 121–133, 2018.
https://doi.org/10.1007/978-3-319-94370-1_9

1 Introduction

Recently, design and implementation of a smart home environment as fundamental elements of smart cities under the principles of IoT network have been obtained a lot of attention for both academic circles and industry world. L.C.D. Silva et al. [3] define smart homes as a "home-like environment that possesses ambient intelligence and automatic control". The definition indicates that the automated design has a complex and intelligent operating system which may use to do on behalf of its users [4].

In addition to the ability to manage the environment using an intelligent system approach, such system should accurately predict the needs of the human occupants, since the people in their nature try to delegate most of their needs to an automation system. The prediction process produced after in-depth studying of the sequence of interaction events between a user and the surrounded environment. The data provided from distributed sensors and actuators in the environment must be collected, filtered, managed to construct the hypotheses and finally generate the model of prediction. Generating the prediction model implies the use of an Artificial Intelligence (AI) technique or Machine Learning Algorithms (MLAs) such as Neural Network, Deep Learning, Support Vector Machine, etc.

It is important to mention that in such applications, the generated model to predict the next user action inside a smart home is entirely different from other models regarding the limited time constraint in a real-time application. In some applications, a 1 s delay could produce a severe problem [8] or even 500 ms as a maximum delay [10]. In other words, the prediction system of a smart home system should be interactive, robust, dynamic, and fast enough to predict the next user action in real-time domain. Therefore, the authors, as shown in Sect. 4, discuss the implementation not only by using Matlab simulation software but also by implementing the experiment on an embedded FPGA hardware kit to assure that our designed system will operate actively under the constraint of a real-time response in a real-world environment. The following paragraph presents different approaches that have been represented by various researchers.

Some of the recent research describes an environment as a smart due to using a smartphone or some remote controlling and monitoring system as in [2,9,12,13]. Other researchers try to enhance such a system design through the use of intelligence exhibited by machine. In other words, they use a more complex architectural model that has a business platform [5]. Other researchers use a cloud-based architecture design for IoT framework which makes the system incapable of remote access, dynamic monitoring, and real-time management with dynamic response [7,11].

This paper presents a new architecture design for a smart home system that tries to connect all the nodes (sensors & actuators) inside home environment through the use of an intelligent agent (IA). The IA locally manage all the events and status of the entire environment based on real-time application principles. Besides, the paper presents a performance analysis for the prediction model via the comparison between Software and Hardware Implementation. The comparison is an essential process to assure a real-time response not only in S.W. implementation experiment but also in a real hardware implementation experiment when the configured system predicts the needs of human occupants in a real-time basis.

The remainder of this paper is organized as follows. Section 2 lists the proposed architectural design of the smart home environment and explain the prediction process using artificial neural network approach that is used in our experiments. Section 4 presents our experiments results using the MavPad dataset. Section 5 discusses the results and highlights the differences due to using software and hardware implementation approaches. In Sect. 6, we sum up our conclusions.

2 Our Proposed Design Approach

Our proposed system as shown in Fig. 1, tries to enhance the design of the smart home domain by introducing a new method which implies the use of a smart agent for each subsystem. Also, it uses a fog computing agent (Storage Agent) which synchronized to cloud computing environment. The cloud system adds some extra services represented by an essential backup database system for the row of data, information, rules, and hypotheses which generated by the designed system. Also, the cloud enriches the BUTLER with a variety of advanced business analysis and insight services capabilities, ending with the central Intelligent Agent (IA), which connects all the components in our architectural design in one unique design approach, that we call the "BUTLER". The idea behind this design is to present a synergistic approach of a personal assistant in the home.

2.1 Architecture Design

Beginning from the lower layers in our design, we can find that there are eight subsystems which interface the same kind of sensors/Actuators (nodes) type. Each subsystem has been connected the central IA via a dedicated agent. The agent plays an essential role in communication with subsystem and IA, filtering of input data and event, and security for communication. The most important part of the presented design is the local IA which responsible for several things as follows:

- Stakeholder interface
- Communication with IA/subsystems
- Learning stakeholder's behavior patterns
- Implementing the predicted actions
- Security and monitoring subsystem
- Storage of data from long-term and short-term activities

In this design, we would like to concentrate on presenting prediction system that depends on local computations which manipulated by the local IA. The key point here is to show that, such a system should locally monitored, controlled,

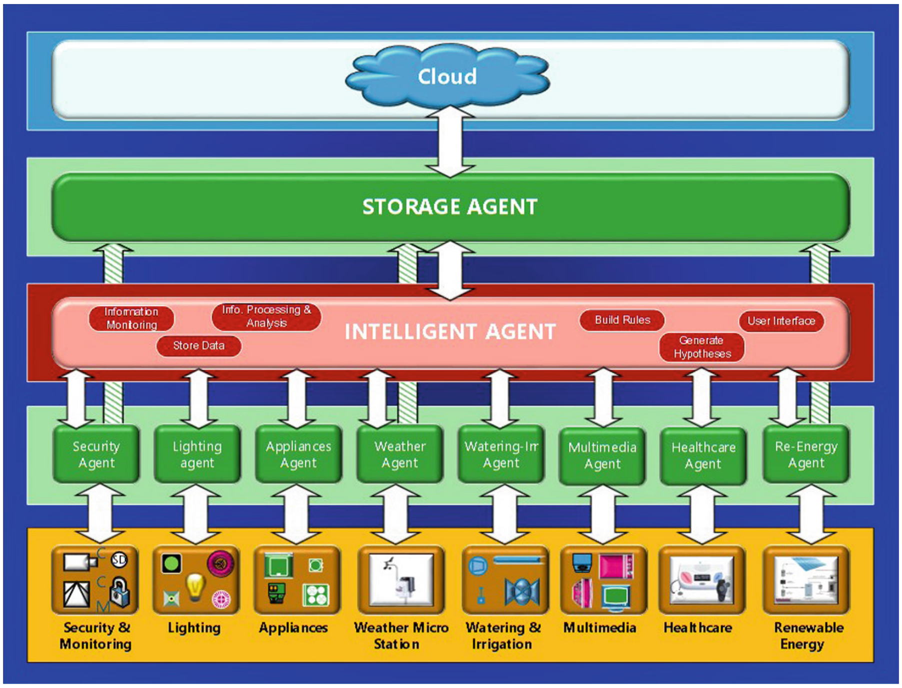


Fig. 1. The proposed architecture design of smart home system

and managed all nodes inside a home environment to predict the next stakeholder actions without the need of an external cloud-based computing system. Cloud system, as presented in recent researches, has many concerns related to its using in real-time application systems. Besides that, we would like to show where the middle-ware is located inside the architecture and how the prediction will take place locally to overcome the mentioned issue as discussed in the next paragraph.

2.2 Prediction Using Automatic Mode of Operation and Machine Learning Algorithms

Beginning with predefined variables and rules which exist in the first phase of the automatic mode of operation as shown in Fig. 2, there are learning phase and action phase. The learning phase starts with collecting the status of the nodes, analyzing and aggregating the features and contexts to establish accurate hypotheses. The generated hypotheses are very necessary to deliver the final decision. The final decision formed in IA using MLAs technique and deep learning based artificial neural network in our case. In the last phase, The predicted action should take place on the targeted actuator. Since the human nature has an inconsistent behavior in a real-life, the BUTLER should be designed to have the ability to adapt to any sudden changes that a stakeholder made himself (manual mode) in case of a stakeholder change his mind.

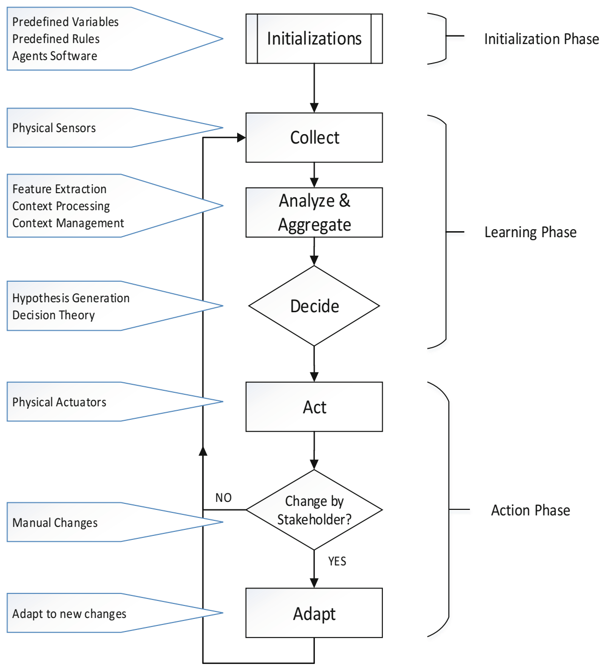


Fig. 2. The development phases for the automatic mode of operation

It is important to mention that a lot of papers present the process to predict a next user action using AI algorithms. The accuracy and time consumption of prediction process is used to choose the best algorithm among MLA techniques. Most of research papers' results describe useable systems. However, these results were collected and aggregated by implementing the MLAs on a software environment using a personal computer, server or even supercomputer. Since such a smart home system is considered to be a real-time operating system, the use of a hardware-based implementation could potentially produce the same accuracy with improved execution time. Thus, this paper attempts to verify whether a hardware implementation will provide the same, or better, results that have been produced by a software implementation. Furthermore, the authors will attempt to demonstrate that the hardware implementation will meet the needs of a real-time applications domain. These experiments use an embedded FPGA hardware development kit for the hardware implementation experiment and make use of deep learning based Neural Network (NN) algorithm, as discussed in Sect. 4.

3 Experimental Design

Before presenting our results in Sect. 4, we describe the dataset that used to implement our experiments. The MavPad dataset contains events that were collected from a real-world interaction between an individual and the home environment in which he resided [14]. The environment consists of several zones represented by a kitchen, restroom, bedroom, and living room zones. The dataset has 127 nodes which contain 86 sensors (input predictors) and 41 actuators (observers) for seven weeks (49 days). Each day has an independent data file. The data syntax for each file represented by date, time, zone number, state, level, and source information. We used MATLAB to pre-process the row of data by extracting the informative details. After filtering the noise, we converted all 49 data files to one matrix file called (OP.mat). The MAT-file has more than 100 K rows or events for the first day alone and more than 4 million rows of information for a sum of seven weeks. Each raw has a status value for each node type which represents a predictor value or an attribute at a specific time. After data conversion process, we applied the machine learning algorithms, listed in Sect. 4, to predict the next user action in the environment.

We factorize a binary observation vector as Xt = (x1t, x2t, . . ., x86t) for the sensors. Actuators are represented by the status of each actuator at a certain time t, so it is denoted with Yt $\in$ 1, ., Q for Q possible states. To investigate the performance of Neural Network algorithm that has been used in our experiments, we chose to study the behavior of a user in one zone (Restroom Zone) in this apartment. Table 1 presents the node's information inside the restroom zone.

Table 1. Sensors and actuators details for restroom zone

No.	Node type	Node label	Node name	Status value
1	Sensor	V21	Motion sensor on ceiling over toilet	0/1
2	Sensor	V22	Motion sensor on ceiling over shower	0/1
3	Sensor	V23	Motion sensor on ceiling over bathroom door	0/1
4	Sensor	S137	Light, east wall. Facing into room	DDI
5	Sensor	S138	Heat, east wall. Facing into room	DDI
6	Sensor	S139	Humidity, east wall. Facing into room	DDI
7	Sensor	S140	Reed switch over door	0/1
8	Actuator	B5	Light over mirror	0/1
9	Actuator	B6	Fan on ceiling	0/1
10	Actuator	B7	Shower light over shower	0/1

Two different combinations of sensors were used to predict the next stakeholder's action. The first case applies all the seven sensors that existed inside the restroom as shown in Table 1. The second case uses all the 86 sensors inside all the zones of the environment. The predicted action represents an actuator with binary outputs (0/1) inside the restroom labeled in (B5). The actuator B5 is used to turn the light over the mirror ON or shut it OFF.

In our experiment, we decide to take the first four weeks (28 days) for training dataset and the fifth week as the test dataset.

4 Experiments Results

4.1 Why We Use a Neural Network (NN)

The Neural Network is a commonly used tool in the machine learning field. The strength of the neural network comes from the mathematical model representation. NN is inspired by the biological nervous system of the human brain. It tries to mimic the way of the human brain processes and learns patterns. A Neural Network consists of interconnected nodes (neurons) that process the input data in a certain way to perform a specific task. Theoretically, the NN can represent many different kinds of complex function. However, a neural network has two major issues. First, the training time which requires high resources availability especially when deep learning is adopted in the network (for more than two hidden layers network). Second, the over-fitting problem due to lack of data which makes the network less generalized to unexpected patterns. Recently, data revolution, parallel architectures, and GPU designs have been developed drastically. Therefore, the neural network has become an efficient tool in the machine learning discipline. Neural network's nature offers very beneficial characteristics such as learning adaptation, self-organization, real-time output, and implementation ease.

4.2 NN Software Implementation Results

In this section, we describe the software implementation of our network. Network configuration, such as network depth (i.e., number of layers) and the number of neurons of each layer, determines computational speed. Although increasing the depth of NN improves the recognition rate (in case of having enough data), it consumes more CPU and memory resources. In this work, network depth, the number of neurons for each layer, and the training window size (i.e., the number of samples used in the training process of each step to update network's parameters) were determined experimentally by maximizing the classification performance using the available resources.

First, we started our implementation with one hidden layer. Then, we increased the number of neurons in the hidden layer to find the best representation experimentally. After that, we increased the depth of the network by adding a second layer. By fixing the number of neurons for the first layer, we increased the number of neurons for the second layer to come up with the best representation. We repeated this approach for the third layer in our network to optimize the number of nodes. Additionally, we used drop-out layers to reduce the effect of the over-fitting problem. We added a drop-out layer between every tow fully connected layers by a factor of 0.5. Removing some units of a network during training prevented excessive parameter updating. This drop-out technique may help reduce overfitting effect. We used two types of zones in our experiments, local zone, and global zone. The local zone uses seven input sensors, as shown in Table 1, to predict the B5 actuator., while the global zone uses all sensors in the environment (86 sensors) to predict the B5 actuator. The results of these experiments are shown in Tables 2 and 3 concurrently.

A Deep Learning technique was used to facilitate the power of neural network via the implementation of the multi-layer approach. As shown in Table 2, the use of the local sensors with only one layer will produce an accuracy of %95.85 with 245 ms needed for the prediction process. Supporting the first layer with a second one can significantly enhance the accuracy performance to be 99.11% with an acceptable time of 387 ms in the scope of real-time application. While adding a third layer to the designed model, doesn't enhance the accuracy. Also, the response time of prediction process using three layers is about the double of what we possess using two layers. The mentioned results using local sensors proves that adding more layers in deep learning model doesn't always assure the best performance: Therefore, layers optimization process needs to be considered when adding further layers to the system architecture.

Table 2. The accuracy and prediction time results using local zone's sensors for multilayer neural network

No. of hidden layers	Accuracy	T (Sec)
One hidden layer	0.9585	0.245
Two hidden layers	0.9911	0.387
Three hidden layers	0.9917	0.768

Table 3. The accuracy and prediction time results using global zone sensors for multilayer neural network

No. of hidden layers	Accuracy	T (Sec)
One hidden layer	0.9017	6.1023
Two hidden layers	0.9536	9.108
Three hidden layers	0.9611	13.706

Similarly, Table 3 discusses the performance of adding a second and a third layer to the prediction system. Since this experiments use 86 sensors, which distributed in the entire environment, we can see that the response time is much higher than what we have in the first experiment that uses only seven sensors. A significant enhancement in accuracy can be noticed when adding a second layer to the deep learning model with 9.1 s of prediction time. A similar result of accuracy values has been seen when adding a third layer. In other words, support the model with a third layer doesn't facilitate better performance in the model; Therefore, the authors decided to consider designing the deep learning model to have the first two layers only which possess a maximum average accuracy and minimum average prediction time.

4.3 NN Hardware Implementation Result Using FPGA

The mathematical operations of neural network models are simple. However, the massive number of operations needs intensive computing resources. Therefore, a fast and efficient realization is required to achieve the benefits of neural models. The FPGA based system allows designers to create digital designs, test them, make modification very quickly, and reduce development time significantly [1]. Also, the Hardware implementation of nonlinear activations, e.g., the sigmoid function, is one of the challenges due to the complexity of implementing division and exponential regarding time and hardware resources. Therefore, the approximate-based approach has been presented to realize sigmoid function efficiently and maintain an acceptable level of accuracy, such as using Lookup Table LUT [15].

In this paper, we utilize an FPGA platform to realize reconfigurable hardware-based neural networks for smart home systems. The proposed architecture is shown in Fig. 3, which can be configured based on the number of hidden layers in a neural network. To achieve high performance, each neuron has one processing unit to make them work in parallel. The configuration as shown in Fig. 3 is a realization of three hidden layers neural network by assigning each set of processing units to a hidden layer. For one hidden layer configuration, all processing units are directly connected to the input buffers and the output layer. The reconfigurable switches in the figure are used for reconfiguration purpose by activating the necessary connections.

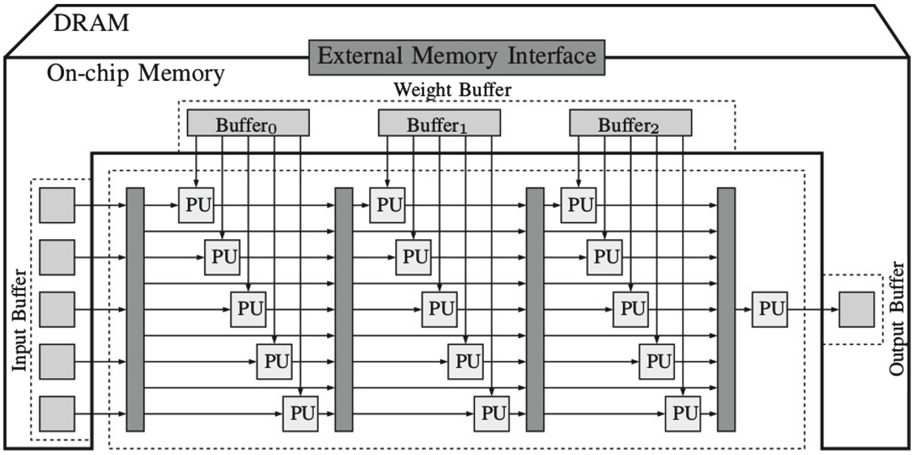


Fig. 3. Reconfigurable neural network architecture, where PUs are processing units and each set of PUs are separated by a reconfigurable switch.

Digital designs are usually modeled using hardware description languages like Very high speed integrated circuit Hardware Description Language (VHDL) and verified by simulation. In this paper, instead of using low-level coding, such as VHDL, we used a high-level programming language, LabVIEW, to realize the neural network. National Instruments LabVIEW FPGA module uses LabVIEW embedded technology to extend LabVIEW graphical development to target FPGAs on NI reconfigurable I/O (RIO) hardware or some Xilinx boards. This module enables users to create custom hardware without low-level hardware description language coding or board design experience. Moreover, it allows a user to executes multiple tasks simultaneously and deterministically, and also expands the functionality of LabVIEW solutions, including unique timing and triggering routines, ultrahigh-speed control, interfacing to digital protocols, digital signal processing (DSP) virtual instruments (VIs).

The implementation of the trained weight data, the synaptic coefficients which are determined off-line in a computing environment, is done using signed fixed-point representation 16-bit total length. Fixed point arithmetic is used for NNs realization, which is implemented as one of the available data types in LabVIEW FPGA module. Therefore, NN coefficients are used in the hardware design without additional pre-calculation. Also, LabVIEW FPGA module provides nonlinear functions, which are used to implement the nonlinear activation functions of each neuron. The other important feature in using LabVIEW software is the ability to import weight coefficients of NNs from Matlab to the NN structure implemented in the FPGA, specifically to the block RAMs that store these coefficients.

Three optimization techniques are used in this paper to optimize and improve the performance of the FPGA-based neural network. The first one is loop pipelining to achieve high throughput by organizing the overlap in the sequence of operation of neural network systems. Single Cycle Timed Loop (SCTL) was used to reduce the required hardware resources and improve the execution speed of our proposed neural network circuit. SCTL is an optimization technique available in LabVIEW FPGA module to eliminate the unnecessary resources exploited by the standard while loop function. Due to limited computation resources in FPGA platforms and the massive computation required in realizing neural network models, loop unrolling has been exploited to efficiently utilize the resources and avoid complex hardware connections. We used this strategy to unroll the independent data and avoid complex connection topologies.

5 Discussion

The hardware implementation of the neural network has been done by LabVIEW FPGA module and downloaded to Xilinx XC3S500E FPGA. The whole neural network system fits in a low-cost Xilinx Spartan-3E FPGA platform and uses block RAMs as on-chip buffers and a DRAM as external storage. The Spartan-3E family of field-programmable gate arrays is specifically designed to meet the needs of high volume, and cost-sensitive consumer hardware digital systems, where the cost must be lower than the general purpose processors. The five-member family offers densities ranging from 100,000 to 1.6 million system gates. The XC3S500E FPGA has 4,656 slices, almost 10,476 logic cells, twenty

Table 4. FPGA-based neural network resource utilization and a comparison with other FPGA implementations

Reference	Platform	Slice LUTs	Slice registers	DSP48Es/ Multipliers	CPS	CPPSL
Gomperts et al. 2011	XC5VSX50T	8043	2243	70	536 M	9.6 M
Zhai et al. 2013	Virtex-4 LX40	4346	N/A	8	1.2 M	0.07 M
Zhai et al. 2016	XC7Z010T	4032	2863	28	72.3 M	10.3 M
Proposed	XC3S500E	3938	2862	20	481.3 M	31.2 M

18×18 hardware multipliers, as well as twenty 18 Kbits modules of dedicated dual-port RAM. In this section, we report the performance of our proposed reconfigurable FPGA-based neural network architecture and compare it with the software implementation. Then, we provide the hardware efficiency compared with the existing FPGA implementations.

Table 5. The prediction time results using local and global zone's sensors for hardware-based multilayer neural network

Zone name	No. of hidden layers	T (Sec)
Local zone's sensors	One hidden layer	0.091
	Two hidden layers	0.199
	Three hidden layers	0.485
Global zone's sensors	One hidden layer	0.329
	Two hidden layers	0.658
	Three hidden layers	1.841

Our proposed FPGA realization of neural network has **the same accuracy results** as what we have in Matlab implementation, and the accuracy is not compromised due to the usage of fixed-point computing units. The characteristic feature of using hardware platforms is performing the mathematical calculations of neural networks in parallel. This feature cannot be achieved with the software-based implementation of neural networks because of the sequential execution of the code. Table 5 illustrates the prediction time results for local and global zone's sensors with different network structures. The results show that the hardware implementation significantly improves the prediction time on average by factors of two times and thirteen times compared with simulation results in the local and global zone respectively. The significant improvement in average prediction time commits substantial evidence that our designed autonomous system applies to be used in such a real-time application paradigm.

The hardware utilization summary is shown in Table 4 and compared with other FPGA realization approaches [6,15,16]. The proposed architecture consumes 2,828 slices out of the available 4,656 slices, which is about 60% of the total number of slices. Specifically, the reconfigurable hardware realization utilizes 3,938 slice LUTs or 42% and 2,862 slice registers or 30%. Connections per second (CPS) metric, the number of operations to be performed in a second, is used to compare hardware-based neural network architectures due to using different FPGA platforms for realization. The second metric is connection primitives per second per LUT (CPPSL), which takes the hardware resource utilization into account. CPPS can be calculated by multiplying CPS by the bit width of inputs and weights. Table 4 shows the performance comparison between our proposed architecture and other implementations using CPS and CPPSL metrics. Our proposed architecture achieves three times more CPPSL compared to the best of existing FPGA implementations.

6 Conclusion

This paper presents the design and implementation of a reconfigurable hardware-based neural network for smart home systems. The proposed architecture outperformed the performance of software-based implementation regarding speed due to exploiting parallel computing and some optimization techniques. LabVIEW software enables developers to implement digital systems without the need for low-level HDL language knowledge and reduces the usage of FPGA hardware resources. It also eliminates the need for pre-processing the neural network weights and maps them to FPGA's storage units. The implementation of FPGA-based neural network system allows its future utilization for other applications.

References

1. Ali, F.H., Mahmood, H.M., Ismael, S.M.B.: LabVIEW FPGA implementation of a PID controller for D.C. motor speed control. In: 2010 1st International Conference on Energy, Power and Control (EPC-IQ), pp. 139–144, November 2010
2. Bian, J., Fan, D., Zhang, J.: The new intelligent home control system based on the dynamic and intelligent gateway. In: 4th IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), 2011, pp. 526–530. IEEE (2011)
3. De Silva, L.C., Morikawa, C., Petra, I.M.: State of the art of smart homes. Eng. Appl. Artif. Intell. **25**(7), 1313–1321 (2012). https://doi.org/10.1016/j.engappai.2012.05.002
4. Galvin, P.B., Gagne, G., Silberschatz, A.: Operating System Concepts. Wiley, New York (2013)

5. Gao, S., Zhu, H., Zhou, X., Liu, Y., Sun, L.: Design and implementation of smart home linkage system based on OSGI and REST architecture. In: Sun, L., Ma, H., Fang, D., Niu, J., Wang, W. (eds.) CWSN 2014. CCIS, vol. 501, pp. 568–582. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46981-1_54
6. Gomperts, A., Ukil, A., Zurfluh, F.: Development and implementation of parameterized FPGA-based general purpose neural networks for online applications. IEEE Trans. Ind. Inf. **7**(1), 78–89 (2011). https://doi.org/10.1109/TII.2010.2085006
7. Han, J., Choi, C.S., Park, W.K., Lee, I., Kim, S.H.: Smart home energy management system including renewable energy based on ZigBee and PLC. IEEE Trans. Consum. Electron. **60**(2), 198–202 (2014)
8. Hu, F.: Cyber-Physical Systems: Integrated Computing and Engineering Design. CRC Press, Boca Raton (2013)
9. Hwang, S., Yu, D.: Remote monitoring and controlling system based on ZigBee networks. Int. J. Softw. Eng. Appl. **6**(3), 35–42 (2012)
10. Koçak, D.: Thinking embedded, designing cyber-physical: Is it possible? In: Suh, S., Tanik, U., Carbone, J., Eroglu, A. (eds.) Applied Cyber-Physical Systems, pp. 241–253. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-7336-7_18
11. Mendes, T.D., Godina, R., Rodrigues, E.M., Matias, J.C., Catalão, J.P.: Smart home communication technologies and applications: wireless protocol assessment for home area network resources. Energies **8**(7), 7279–7311 (2015)
12. Piyare, R.: Internet of things: ubiquitous home control and monitoring system using android based smart phone. Int. J. Internet Things **2**(1), 5–11 (2013)
13. Riaz, M.T., Ahmed, E.M., Durrani, F., Mond, M.A.: Wireless Android Based Home Automation System
14. Youngblood, G.M., Cook, D.J.: Data mining for hierarchical model creation. IEEE Trans. Syst. Man Cybern. Part C (Applications and Reviews) **37**(4), 561–572 (2007)
15. Zhai, X., Ali, A.A.S., Amira, A., Bensaali, F.: MLP neural network based gas classification system on Zynq SoC. IEEE Access **4**, 8138–8146 (2016)
16. Zhai, X., Bensaali, F., Sotudeh, R.: Real-time optical character recognition on field programmable gate array for automatic number plate recognition system. IET Circuits Devices & Syst. **7**(6), 337–344 (2013)

An Empirical Analysis of Smart Connected Home Data

Joseph Bugeja(✉), Andreas Jacobsson, and Paul Davidsson

Internet of Things and People Research Center, Department of Computer Science and Media Technology, Malmö University, Malmö, Sweden
{joseph.bugeja,andreas.jacobsson,paul.davidsson}@mau.com

Abstract. The increasing presence of heterogeneous Internet of Things devices inside the home brings with it added convenience and value to the householders. At the same time, these devices tend to be Internet-connected and continuously monitor and collect data about the residents and their daily lifestyle activities. Such data can be of a sensitive nature, given that the house is the place where privacy is naturally expected. To gain insight into this state of affairs, we empirically investigate the privacy policies of 87 different categories of commercial smart home devices in terms of data being collected. This is done using a combination of manual and data mining techniques. The overall contribution of this work is a model that identifies and categorizes smart connected home data in terms of its collection mode, collection method, and collection phase. Our findings bring up several implications for smart connected home privacy, which include the need for better security controls to safeguard the privacy of the householders.

Keywords: Smart home · IoT · Data model · Privacy policies

1 Introduction

An increasing number of consumers install Internet of Things (IoT) devices in their homes. This benefits the householders offering an improved ability to control and automate relevant aspects of their house and daily home chores. According to Gartner the number of IoT devices is expected to exceed 20 billion by 2020[1]. IoT consumer devices found in private homes include learning thermostats, energy tracking switches, IP-based video doorbells, smart speakers, and more. These devices tend to use embedded sensors and the Internet to collect and exchange data with each other and their users, integrating the digital with the physical environment inside the home.

As these IoT devices become more widespread in homes so is the volume, granularity, and diversity of data collected by these devices. For instance, smart televisions, may be equipped with microphones and cameras allowing for interaction through voice commands and gestures; and a smart thermostat may capture the temperature, humidity, and activities inside the home to predict and adjust the room temperature automatically. While this brings added convenience and value to the householders, the home being the

[1] https://www.gartner.com/newsroom/id/3165317 [accessed May 06, 2018].

natural place where private conservations are held, and where there is an implicit trust between the householders and their house, makes the investigation of data being collected by smart home devices fundamental in understanding privacy concerns. Furthermore, this is useful to realize what is at stake if a device is compromised [1].

Researchers have studied the privacy challenges of smart connected homes and proposed different mitigations to protect user privacy (e.g., [2]), but no study has looked in detail into the actual data collection practices of commercial smart home manufacturers. To different extents, this limits the applicability of previous studies when it comes to applying them to the complexity of real-world setups. Such setups tend to involve heterogeneous devices and data, ranging from low-power devices to high-performance nodes supporting multiple input and output data sources. The closest research work in this regard, concern mostly technical studies that have investigated data exfiltration, oftentimes indirectly. This is typically done by assessing and exploiting vulnerabilities of smart home devices through experiments. While experiments are likely to lead to more accurate results, the number of investigated devices tends to be relatively small (e.g., laboratory setup with 7 device types [3]), is focused on a subset of devices (e.g., web cameras [4]), and targeting specific data states (e.g., concerning stored data [5]). Our goal in this work is to identify all the main categories of data that are collected by smart home systems. This is needed to ground the conversation especially about smart home privacy with empirical evidence on data collected.

In conducting this study, we analyze the privacy policies concerning 87 different type of devices issued from 64 manufacturers of commercial smart home devices. Each manufacturer should have a privacy policy or a similar document detailing how data are captured by a device [6]. Policies are investigated using a hybrid approach combining manual analysis with data mining techniques. Overall, the main contributions of this work are: (i) identification of data types being collected by a smart connected home system; (ii) a categorization of smart connected home data types according to their data source; and (iii) a data model grouping the different data types according to their associated data collection mode, collection method, and collection phase. Our findings bring up several implications for smart home privacy, which include the need for better security controls to safeguard the privacy of the householders.

The rest of this paper is structured as follows. In Sect. 2, we describe data flows in a smart connected system in a generic way. Next, in Sect. 3, we formalize the specification of privacy policies. Following that, in Sect. 4, we review and group existing work related to privacy policy analysis according to the adopted approach. The utilized research design methodology is discussed in Sect. 5. Next, the policy analysis results, identified data types, application of the data categorization to the analyzed policies, and the data collection model are presented in Sect. 6. Finally, in Sect. 7, we discuss some implications of our findings, and conclude this paper.

2 Smart Connected Home Data Flows

A smart connected home system is composed of a number of physical and virtual entities, and data flows occurring between them. At an abstract level, a smart home system is made of three entities:

1. *Smart device:* Hardware components such as Internet-connected household devices, networked appliances, or wearable technologies. These collect data about the householders and the environment and use it to communicate with other devices and users. An example of a smart device is Amazon Echo – an 'intelligent' smart speaker – that uses voice detection technology to detect and respond to tasks such as streaming music. Smart devices may feature integrated sensors, actuators, and processors.
2. *Service:* Software components such as mobile applications, web applications, and Application Program Interfaces (APIs). These use, retain or transfer data from smart devices to implement the householders' desired goals, e.g., enhanced security and safety. Smart home services are commonly deployed over a cloud infrastructure, gateway devices, or as native services inside a device.
3. *User:* Householders, service providers, network providers, and other stakeholders, that together create and enable the smart home ecosystem. The householders tend to be the main subject of data collection by the smart devices surrounding them.

Any smart connected home system involves the exchange of data to enable its function. This is done through a communication infrastructure, consisting of protocols, typically IP-based, and networking components such as router, bridges, and hubs. This data can range from non-personal data (e.g., room temperature), personal data (e.g., resident names), sensitive data (e.g., health conditions), and more.

Based on the work of Ziegeldorf et al. [7] but adapted for the smart connected home environment, we identify four main data flows occurring between the entities. We refer to these flows as: interaction, collection, dissemination, and presentation (see Fig. 1). In the interaction phase, the user, actively or passively, interacts with the smart device. Here, as an example, the householder might switch on an IP-enabled lightbulb. In the collection phase, the smart device, e.g., the connected-lightbulb, gathers the information, e.g., data about the bulb state (on/off), and delegates that to the corresponding service, e.g., embedded software in the connected-bulb. Services then process the data to provide desired function (e.g., turning on the room light) and may initiate further actions on their own. Once that is done, in the dissemination phase the service relays the information towards the data subject or a third-party. This is mediated through a smart device, which can be the same as the one used for interaction phase or a different one. Such device then implements an actuation response (e.g., adjusting the room light level) or provides a notification (e.g., mobile notification displaying the current room light level) to the user. We refer to the final stage, as the presentation phase.

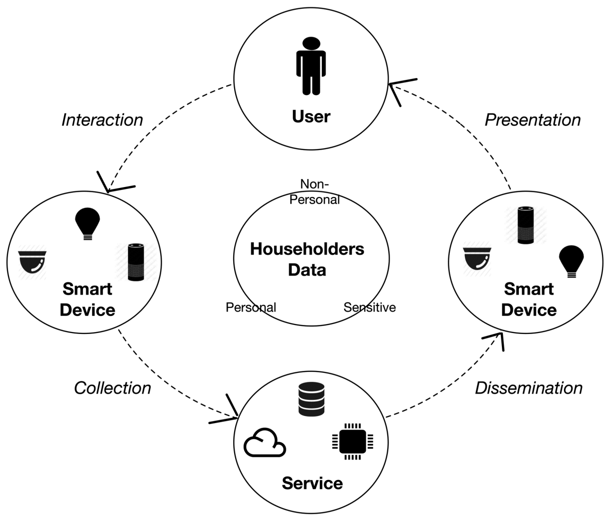


Fig. 1. A smart connected home system entities, data, and data flows. Data in a smart home system tends to initially flow from the user (*interaction*), typically the householder. Consequently, this is received (*collection*) by the smart home device, and then processed by a software service(s). In turn, this is used to send (*dissemination*) a notification to the user (*presentation*) or to change an environment parameter, e.g., temperature.

Although not enforced by law, companies, especially those operating in the US, are required to post notices of data practices, especially concerning their privacy management procedures [8]. In the US, regulators such as data protection authorities or the US Federal Trade Commission leverage companies' privacy policies to assess and enforce regulatory compliance [9]. In the past, privacy policies were mostly the target for web-based systems, however nowadays especially with new regulations (e.g., the European General Data Protection Regulation[2]) these are becoming key also for IoT-based applications, such as smart connected homes. This is especially to meet the individual's right to restrict data processing.

3 Privacy Policies

Privacy policies aim to answer important questions, such as: what data are collected, for what purposes is this data used, and with whom is the data shared with, amongst other things. Based on the privacy requirements specification proposed by Breaux et al. [10] but applying it to the smart connected home, at an abstract level, policy requirements can be formally specified as: <*S, Ac, A, D, P*> where:

- *S,* the policy scope,
- *Ac*, a set of actors among whom data are shared,

[2] https://www.eugdpr.org [accessed May 06, 2018].

- A, a set of actions that are performed on the data,
- D, a set of data elements on which actions are performed,
- P, a set of purposes for which data may be acted upon.

In our case, $S \in$ *{device, website, service, all}*, indicating that a privacy policy may cover the smart device, webpage a user may interact with, services such as mobile applications, and all of these. *Ac* indicates data recipients, typically the service providers. $A \in$ *{interaction, collection, dissemination, presentation}*, corresponding to the data flows defined earlier in Sect. 2. D represents data about the smart home environment. P specifies the data collection and usage reasons as provided by the vendor.

For the scope of this paper, we are mainly interested in finding D where $S =$ *{all}*, and $A =$ *{collection}*. This is the first logical phase where data from the user is received by a smart home device. In a sense, this is arguably the entry point whereby which the householders' privacy can get compromised.

4 Related Work

Despite efforts to make privacy policies machine-readable, e.g., with P3P, or formalize them, e.g., using EPAL, policies are oftentimes written in natural language [11]. Given this, we observe three distinct approaches concerning the extraction of data types and semantics from privacy policies: Natural Language Processing (NLP), machine learning (ML), and hybrid approaches.

4.1 NLP Approaches

Alohaly et al. [12] used NLP techniques to extract data types from privacy policies. This was done by first locating text fragments that were relevant to the data collection practice. Then, noun phrases were extracted from the retrieved sections keeping only those that were present in the Information Type Lexicon [13]. These represented actual data types. The Information Type Lexicon is a dictionary based on privacy policy annotations. This was created through manual, human annotations, and with the help of an entity extractor based on part-of-speech (POS) tagging. The entity extractor succeeds at finding 92% annotations (from 15 policies). In our work, we use some of the data elements and categories present in the Information Type Lexicon, as guidance for retrieving and grouping potential data items. Nonetheless, its main limitation is that it has a saturation limit of 31–78%. Therefore, its overall effectiveness is limited for previously unseen policies or domains.

Bhatia et al. [14] used constituency parse trees from POS tagged sentences to automate the extraction of hyponyms found in privacy policies. Hyponyms are specific phrases that are sub-ordinate to another more general phrase, e.g., a GPS location is a kind of real-time location. The authors manually identified a set of hyponyms among 15 privacy policies and then formalized the patterns using a tree regular expression language (Tregex). An important conclusion of this study is that only 17% of the data types found in the dataset appeared in WordNet (a popular lexical database). Alas, this study involved policies that were not connected to the IoT domain.

Costante et al. [15] leveraged Information Extraction (IE) techniques to extract a website's data collection practices from its privacy policy. The proposed approach leverages the semantics of the text leading to consistently accurate results. However, the extraction rules are manually created for specific scenarios. This thereby limits their reusability, e.g., for identifying data items collected by a smart home system.

4.2 ML Approaches

Costante et al. [16] proposed a supervised learning approach to determine which data practice categories (e.g., data sharing) are covered in a privacy policy. The privacy categories are extracted from privacy regulations, while text classification and machine learning techniques are used to verify the categories that are covered by a policy. This study is however focusing on the evaluation of privacy policies for their completeness. This is different from our study, where we are interested in the extraction of data types.

Liu et al. [17] explores the problem of aligning or grouping segments of policies based on the privacy issues they address. This is done by using clustering and hidden Markov model based alignment methods. In this study a corpus of 1,010 collected policies was used. One conclusion of this work is that unsupervised methods reach far better agreement with the consensus of crowd workers than originally estimated. Despite this, the applicability of this study for our purposes is restricted especially as it is not directly the scope of the cited study to extract data types.

4.3 Hybrid Approaches

Zimmeck and Bellovin [18] introduced an architecture for analyzing privacy policies using rule and ML classification with crowdsourcing. Policies were classified into different categories as derived from privacy legislation: collection, encryption, ad tracking, limited retention, profiling, and ad disclosure. Here, features (similar to data types) are extracted using bigrams represented as regular expressions. While the feature extraction part is interesting for our study, the main focus of the referenced work is on classifying policies for privacy factors. In our work, we employ a similar data extraction method but also consider unigrams as potential data types.

The Usable Privacy Project [19] uses natural language processing, machine learning, privacy preference modeling, crowdsourcing, and formal methods to semi-automatically annotate privacy policies. This project has annotated over 7,000 privacy policies using machine learning classifiers. Additionally, 115 privacy policies (the OPP-115 Privacy Policy Corpus) were manually annotated by law students. Given the amount of policies reviewed, achieved results, and online availability of this project, we utilize the obtained data categorization here as guidance for creating a data categorization for the smart connected home.

4.4 Main Observations

In reviewing the existing work, we draw three main observations. First, we observe that the majority of the existing work involves to different extents manual analysis as part of the preprocessing stage of policies. This facilitates tackling problems that remain hard to solve with automated methods by leveraging human intelligence especially to resolve ambiguities in parsing policies. Typically, this utilizes crowdsourcing, and commonly uses the Amazon Mechanical Turk platform[3]. Second, we note limitations, especially in the pure NLP approaches, when it comes to reusing the data types, obtained lexicon, and data extraction rules to other domains that were not initially considered. Third, we observe that the most cited, current, and rather generic approaches tend to use a hybrid approach leveraging manual and semi-automated methods. However, it is interesting to note that in the hybrid but also in the ML approaches the extraction of data types is usually not the main focus of the underlying study.

For these reasons, we adopt a hybrid approach combining manual with data mining to identify and categorize collected data types in a smart connected home system. However, different from previous studies we focus on manufacturers of smart connected home systems, and leverage both unigrams and bigrams to extract these.

5 Research Methodology

In this section we describe the research approach that was adopted to identify and categorize the data types found in smart connected home privacy policies.

5.1 Data Collection

As a precursor for identifying privacy policies, an IoT product collection platform was consulted first to identify smart home devices and their corresponding manufacturer. In this regard, there are two main databases, namely, *iotlist.co* (IoTList) and *smarthomedb.com* (SmartHomeDB) that can be used.

IoTList is an online directory listing IoT devices available on the market. SmartHomeDB is an open community-supported database focusing specifically on smart connected home devices. In our case, we relied on SmartHomeDB, as in comparison to IoTList, this database targets consumer devices intended for the smart home, and also has a product ranking system. The ranking system was used to select the top reviewed devices.

At the time of our evaluation (as of January 2018) SmartHomeDB identified a total of 87 different categories of devices (e.g., lighting system, scale, voice command device, etc.). For each category, the most reviewed device within that was selected. In case, a category contained multiple devices that had the same ranking the most specific device was selected. Example, if two devices within the '*Sensor: Door*' category had the same number of reviews but one featured multiple sensor types, and one functioned as a standalone, then the latter was chosen. This was done to avoid having technologies already covered by other devices affecting the data categorization.

[3] https://www.mturk.com [accessed May 06, 2018].

After the 87 device types were logged, a Python script that interfaced with *Google Search API*[4] was created to retrieve and download the privacy policy of each device. Here, the manufacturer name appended to the specific product name were used as search queries. After the corresponding policy was downloaded it was automatically converted from HTML or PDF format to text in preparation for the data preprocessing stage.

5.2 Data Preprocessing

After the policies were downloaded, the resultant corpus was first manually inspected. Here, the policies were investigated and sorted according to their scope. In our case, we were interested in policies that covered the entire spectrum of possible input data sources, i.e., policies where $S = \{all\}$ (cf. Sect. 3).

At this stage, policies were also inspected to ensure that they identified the collection phase. In case, policies were only not available in English then these were translated using *Google Translate*. Furthermore, if a manufacturer produced more than one device and that was covered by the same policy (or a supplement in the same policy) then only one version of the policy was kept. Since policies covered other actions besides the collection phase, we manually removed other sections from the policies not pertaining to that.

Following the manual preprocessing stage, the policy was further preprocessed in memory using *R*. Here, numbers, punctuation marks, extraneous white spaces, and stop words (using *tm* package) were removed from the corpus as these did not contribute to data types. Additionally, text was converted to lower case, and stemmed using Porter stemming algorithm (using *SnowballC* package).

5.3 Data Type Identification and Categorization

To identify possible data types we transformed the preprocessed policy documents (the corpus) into a term-document matrix. Essentially, a term document matrix is a two-dimensional matrix whose rows represent the terms and columns represent the documents. In our case, the terms represent possible data types and the documents represent privacy policies.

As a method for locating terms in privacy policies, we employed a n-gram tokenizer (using *RWeka* package) to find both unigrams and bigrams. Unigrams are needed to detect data types, such as "gps", and bigrams are needed to find other instances, e.g., "ip address" or data categories, e.g., "contact information".

Consequently, the resultant list was saved and scanned manually for data types and possible categories. In doing so, and in grouping the different elements, we were guided by the "Information Type Lexicon" [13] and the "Usable Privacy Project" [19].

[4] https://github.com/abenassi/Google-Search-API [accessed May 06, 2018].

6 Results

In this section, we present the results obtained after executing the research methodology described in the previous section.

6.1 Privacy Policy Analysis

From the collected dataset, the number of reviews varied significantly with mean ($\mu = 2{,}439$) and standard deviation ($\sigma = 6{,}098$). Effectively, the most reviewed device (34,372 reviews) was a media player, and the least reviewed devices consisted mostly of sensor devices.

There were multiple 12 manufacturers producing multiple device types ranging from 2 up to 4 most reviewed products. In total, covering 87 different devices types, there were 64 manufacturers, out of which 3 had no privacy policy, and one had a policy that was available only in Chinese.

Different policies differed in terms of their scope. In Table 1, we summarize the different types of policies available from different smart connected home manufacturers, including the number of documents covering each scope, and the number of device types covered by each policy scope.

Table 1. Distribution of smart connected home privacy policies.

Privacy policy scope	Corpus size	Number of device types
Website	27	34
Service	4	4
Website and service	7	7
Website, service, and device	23	39

All policies contained information about data collected, however since we were interested in the entire smart connected home environment then we focused the rest of this work on the policies that covered all components (i.e., the website, service, and device). In total, this included 39 different device types manufactured by 23 different manufacturers.

6.2 Smart Connected Home Data Categorization

The results of our analysis indicate that there are two main entities that are subject to data collection by smart home stakeholders:

User. This represents the householders who are the end-users of the smart home system. However, this also may include visitors, guests, or other physical entities who are interacting directly or indirectly with the smart home system. Data categories in this dimension include:

- *Contact information.* Information that can be used to communicate or correspond with users. Examples of data types here are: email, phone number, address, contact list, and friend user IDs. Typically, this information is captured during system setup.
- *Personal and account details.* Data that can be used to identify and authenticate a user. This includes names, login identifiers, social networking services account information, passwords, security codes, profile pictures, job titles, gender, birth date, and body metrics data (e.g., weight). All the surveyed device types require this type of information in order to use the services being offered by the smart home system.
- *User activity data.* Data that corresponds to a measurement of user physical activities or interactions. Examples found in this category include: interaction data, e.g., voice commands, browser data, webpages accessed, service used, features accessed, activity time and duration, and inputted search query terms. The type of user activity data captured and utilized is dependent of the type of device.
- *Configuration settings.* Customization or personalization information related to the smart home system. Examples of data types here include: device and service specific settings, operating schedules, contact preferences, cookie settings, and language preferences. Commonly, this data are stored on the mobile applications but may also be stored in the device and cloud infrastructure.
- *Location information.* Position data related to the current geographic location of the user. In a smart home system this is provided either directly by the user or is derived automatically, e.g., from the IP address, or through location-based technologies (e.g., by capturing a smartphone GPS' signal). Such data may be used, e.g., to provide location-based services, e.g., weather forecasts.
- *Financial information.* Payment related information required to purchase additional services/products, e.g., extra cloud storage space. Data types here, include credit card data such as card numbers, expiration dates, and related security codes. Commonly, this information is captured during the smart home setup phase.
- *User-generated content.* Data submitted voluntarily by the user, typically as part of a survey or in relation to technical issues. Examples of data types here include: feedback, opinions, reviews, comments, uploaded files, interests, demographic data, and other information furnished by the user e.g., for due diligence process, technical support, and marketing research. Typically, this data are collected over the web interface.
- *Offline data.* Data that may be used to identity the user but is captured indirectly, e.g., by visiting a service provider premises or by talking to service personnel. Examples of data types captured here include: CCTV footage, phone conversations, and offline interactions.

Device. IoT devices that are present inside the home environment (e.g., IP camera, smart TV, network-enabled washing machine), and end-user devices such as smartphones, tablets, and smart watches that can be used to monitor and control the smart home system. Data categories in this dimension include:

- *Device information.* Technical information describing a hardware device. Data types here include: device identifiers (e.g., IP address, MAC address, and IMEI number), performance information, network/connectivity-related information (e.g., Wi-Fi

status and Bluetooth data), firmware and software versions, sensor status, battery charge level, diagnostic information, and consumption data (e.g., energy consumed).
- *Environmental data.* Technical data describing the smart home environment including its surroundings. Typically, data types here feature parameters that are captured by sensors. Examples of these found in the reviewed policies include: motion, humidity, ambient light, CO2 concentration, and rain level.

6.3 Smart Connected Home Devices and Data

As an application of the previously defined data categorization to the smart connected home, in this section we review each of the 39 devices identified in Sect. 6.1 in terms of their collected data. The results of this mapping is shown in Table 2.

Table 2. Matrix showing different device types alongside their collected data types: CI, contact information; DI, device information; PAD, personal and account details; UAD, user activity data; CS, configuration settings; LI, location information; FI, financial information; ED, environmental data; UGC, user-generated content; OD, offline data.

Device type	CI	DI	PAD	UAD	CS	LI	FI	ED	UGC	OD
Music player, gateway/hub	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Door bell, audio speaker, TV, irrigation controller	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
Scale, plug, light switch, light bulb, wireless signal extender	✓	✓	✓	✓	✓	✓	✓	–	✓	–
Switch, power outlet, oven, clothes dryer	✓	✓	✓	✓	✓	✓	–	✓	–	–
Vacuum cleaner, floor mopper, floor scrubber, gutter cleaner	✓	✓	✓	✓	✓	✓	✓	–	–	–
Tracker	✓	✓	✓	✓	✓	✓	–	✓	✓	–
Blood pressure monitor, temperature sensor	✓	✓	✓	✓	✓	✓	–	✓	–	–
Remote control, light strip, cooker	✓	✓	✓	✓	–	✓	✓	✓	✓	✓
Air quality sensor, rain sensor, CO2 sensor, wind speed sensor	✓	✓	✓	✓	–	✓	✓	–	✓	–
Siren	✓	✓	✓	✓	–	✓	✓	✓	✓	–
Cloud camera	✓	✓	✓	✓	–	–	✓	✓	✓	–
Shower head water meter	✓	✓	✓	–	✓	✓	✓	✓	✓	–
Bar code scanner	✓	✓	✓	✓	✓	✓	✓	✓	–	–
Thermostat, Smoke detector	✓	✓	✓	✓	✓	✓	–	✓	–	–
Cloud camera	✓	✓	✓	✓	–	–	✓	✓	✓	–
Door lock	✓	✓	✓	✓	–	✓	✓	–	–	–
Accelerometer sensor	–	–	✓	✓	–	✓	✓	✓	✓	–

"✓" = data type is captured; "—" = data type is not specified to be collected.

Here, it can be noted that the devices types that collect all data categories are a music player, and a gateway/hub device. On the contrary, the device types that collect the least amount of data are a door lock and accelerometer sensor. All investigated device types require some personal and account details from users.

6.4 Smart Connected Home Data Collection Model

It can be observed that the different data categories are collected either explicitly, i.e., the user manually provides the information directly to the system, or implicitly by the system. In the latter case, the data are collected automatically without involving the end-user's explicit awareness and consent. The data source for the explicit collection mode tends to be the user, in particular the householders, whereas the data source for the implicit collection mode tends to be the smart home device. In Table 3, we map each data category to its corresponding collection mode, and identify methods and phases during which this data are captured.

Table 3. Smart connected home data collection model.

Data category	Collection mode	Collection method	Collection phase
Contact information	Explicit	Website form, service	System setup
Device information	Implicit, explicit	Smart home device, end-user device	System operation, sync process
Personal and account details	Explicit	Website form, service	System setup
User activity data	Implicit, explicit	Sensors, service	System operation, sync process
Configuration settings	Explicit	Website form, cookies, service	System setup
Location information	Implicit, explicit	Smart home device, end-user device	System operation
Financial information	Explicit	Website form, service	Purchase process
Environmental data	Implicit	Sensors	System operation
User-generated content	Explicit	Surveys, feedback form, support ticket	System operation, troubleshooting process
Offline data	Implicit, explicit	Paper, digital	Offline

When it comes to the data provided explicitly by the user this correlates with the interaction phase of a smart connected system. Here, the user is typically setting up or registering a smart home device for the first time or customizing it to cater for new conditions. This type of data are typically provided through a mobile application furnished by the smart home device manufacturer or service provider. In some cases, such input is provided through a website or service managed by the service provider or third-parties (e.g., PayPal). Commonly, this type of data can be opted-out although doing so may sometimes hinder the smart home system performance or stability.

To collect data automatically, a smart home system, tends to employ different technologies depending on the system and input channels being used. For instance, when interacting with a system via the web interface, cookies are typically used as a mechanism to gather information about the householders' preferences. On the other hand, it is also common to collect information from third-party applications (e.g., Facebook) or from specific APIs. Additionally, smart home devices may employ sensors that automatically collect environmental data. While some data categories, e.g., in the case of cookies, can be opted-out by the user, there are data categories, in particular environmental data that cannot be opted-out.

7 Discussion and Conclusions

The growth and heterogeneity of IoT devices present in the smart connected home raises the importance of an analysis of data being collected by these devices. This is needed to explore what is at stake if a device is compromised, and as a precursor for conducting a privacy impact assessment.

In our study, we have analyzed privacy policies of 64 manufacturers, out of which we identified 23 policies that cover all the smart home components. We observe that no scholarly work has looked specifically into data collection stage of smart home systems. The exception here are a few security vulnerability studies that assess data being disclosed (sometimes indirectly) through lab experiments. While such experiments reveal actual data being collected, rather than declared types (as in the case of policies), they are dependent on the adopted test cases, utilized software/hardware tools, and the physical location of the assessor (or penetration tester). In reality, for many reasons, e.g., costs, time, and expertise required, it is often the case that only certain software components, device types, and data states are targeted. One example is examining built-in webservers embedded in IoT cameras for unauthorized transmission of data. Therefore, this limits the effectiveness of experiments when used as the sole method for identifying data being collected by smart home devices.

Privacy policies have been studied considerably by many researchers over the years as we have mentioned when reviewing the related work. However, we observe that even the most cited publications focus on the investigation of commercial entities operating typically in the traditional web-based systems domain and not on IoT systems such as smart connected homes. While we used some existing work (e.g., [19]), for guidance, we expanded on this domain with data categories that are suitable for investigating IoT-based systems, in particular smart connected homes. For instance, we added the 'Environmental data' to capture information measured by sensors, and 'User activity data' to encapsulate not only online activities, such as browsing websites of service provider, but other interaction channels as well, e.g., voice. Voice input is commonly used to interact with smart home devices.

We have identified 10 different data categories of smart home data, that correspond to data being generated by the user, typically the householder, and data being generated by the device. These categories were empirically derived by analyzing privacy policies through a hybrid approach combing manual analysis with data mining. Our findings reveal that certain data types, in particular 'Device information' and 'Environmental data', are passively and potentially continuously, being collected by smart home devices. A consequence of this is that the system may be automatically monitoring, building detailed user profiles, and automatically inferring user activities without the householders' awareness. Such activities may include sensitive ones, ranging from indicating whether the residents are away to medical diagnosis of an individual. Data collection may be done even when actions are not initiated or given explicit consent by the user. This is especially as IoT-based systems may leverage data mining or artificial intelligence techniques that automatically learn, adjust their services, perform actions, and automatically reach conclusions based on the collected data [20].

Achieving privacy is an inherent trade-off in IoT systems, because IoT devices cannot provide their services and add value without collecting data. However, since such devices tend to be Internet-connected, have a tendency to be 'always-on', and are present in houses where privacy is naturally expected, this stresses the need for smart home developers to create better data security controls to safeguard the privacy of householders. Likewise, this raises the importance to have better methods for informing the householders of potential risks when purchasing and operating smart home technologies.

Smart home stakeholders may collect certain categories of data, related to both personal and non-personal data. Personal data is essentially data that can be used to identify an individual person; whereas non-personal data do not have the capability (on their own) to identify an individual person. Typically, personal data are collected by smart devices, e.g., to provide adapted responses to a user's current need with the fewest explicit information provided by the user [21]. In this study, we have noted that all surveyed manufacturers collect instances of personal data. This may include body metrics (and other physiological data) that are arguably the most sensitive data type. While one may assume a secure and trustworthy entity processing data according to its privacy policy, entities may be targeted by malicious threat agents, such as hackers. A consequence of this is that private information may be lost, sold to third-parties, and used inappropriately for commercial or malicious purposes [21].

As part of future work, we intend to expand this study to investigate the privacy practices of smart home service providers. One way of doing this is by extracting such information from privacy policies. Another avenue we seek to investigate is to complement this study with a lab experiment. In particular, it would be interesting to investigate some prominent gateways devices. Especially, this is as these tend to be the components; as is also evident in this study; that have access to the most data types, and thus an important point where security should be bolstered. Finally, we plan to develop controls to allow householders to be notified about surrounding IoT devices collecting personal information, and to control these collection practices.

Acknowledgments. This work has been carried out within the research profile "Internet of Things and People", funded by the Knowledge Foundation and Malmö University in collaboration with 10 industrial partners.

References

1. Bugeja, J., Jacobsson, A., Davidsson, P.: On privacy and security challenges in smart connected homes. In: Proceedings of the IEEE Intelligence and Security Informatics Conference (EISIC), pp. 172–175 (2016)
2. Ahlam, A., Laila, B., Slimane, B.: An overview of privacy preserving techniques in smart home wireless sensor networks. In: Proceedings of the IEEE 10th International Conference on Intelligent Systems Theories and Applications (SITA), pp. 1–4 (2015)
3. Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., Feamster, N.: Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic (2017). arXiv preprint arXiv: 1702.03681
4. Seralathan, Y., Oh, T.T., Jadhav, S., Myers, J., Jeong, J.P., Kim, Y.H., Kim, J.N.: IoT security vulnerability: a case study of a web camera. In: Proceedings of the IEEE 20th International Conference on Advanced Communications Technology (ICACT), pp. 172–177 (2018)
5. Boztas, A., Riethoven, A.R.J., Roeloffs, M.: Smart TV forensics: digital traces on televisions. Digital Invest. **12**, S72–S80 (2015)
6. Anscombe, T.: IoT and Privacy By Design in the Smart Home. https://www.welivesecurity.com/wp-content/uploads/2018/02/ESET_MWC2018_IoT_SmartHome.pdf. Accessed 06 May 2017
7. Ziegeldorf, J.H., Morchon, O.G., Wehrle, K.: Privacy in the Internet of Things: threats and challenges. Secur. Commun. Netw. **7**(12), 2728–2742 (2014)
8. Massey, A.K., Eisenstein, J., Anton, A.I., Swire, P.P.: Automated text mining for requirements analysis of policy documents. In: Proceedings of the Requirements Engineering Conference (RE) (2013)
9. Schaub, F., Balebako, R., Cranor, L.F.: Designing effective privacy notices and controls. IEEE Internet Comput. **21**(3), 70–77 (2017)
10. Breaux, T.D., Hibshi, H., Rao, A.: Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. Requirements Eng. **19**, 281–307 (2013)
11. Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N., Bellovin, S.M., Reidenberg, J.: Automated analysis of privacy requirements for mobile apps. In: Proceedings of the Network and Distributed System Security (NDSS) Symposium (2017)
12. Alohaly, M., Takabi, H.: Better privacy indicators: a new approach to quantification of privacy policies. In: Proceedings of the WPI SOUPS (2016)
13. Bhatia, J., Breaux, T.D.: Towards an information type lexicon for privacy policies. In: Proceedings of the IEEE Eighth International Workshop on Requirements Engineering and Law (RELAW) (2015)
14. Bhatia, J., Evans, M.C., Wadkar, S., Breaux, T.D.: Automated extraction of regulated information types using hyponymy relations. In: Proceedings of the IEEE 8th International Requirements Engineering Conference Workshops (REW), pp. 19–25 (2016)
15. Costante, E., Hartog, den, J., Petkovic, M.: What websites know about you★ privacy policy analysis using information extraction. In: Data Privacy Management and Autonomous Spontaneous Security, pp. 146–159 (2013)
16. Costante, E., Sun, Y., Petkovic, M., Hartog, den, J.: A machine learning solution to assess privacy policy completeness. In: Proceedings of the ACM Workshop on Privacy in the Electronic Society (2012)
17. Liu, F., Ramanath, R., Sadeh, N., Smith, N.A.: A step towards usable privacy policy: automatic alignment of privacy statements. In: Proceedings of the 25th International Conference on Computational Linguistics (COLING) (2014)

18. Zimmeck, S., Bellovin, S.M.: Privee: an architecture for automatically analyzing web privacy policies. In: Proceedings of the USENIX Security Symposium (2014)
19. Sadeh, N., Acquisti, A., Breaux, T.D., Cranor, L.F., McDonald, A.M., Reidenberg, J.R., Smith, N.A., Liu, F., Russell, N.C., Schaub, F., Wilson, S.: The Usable Privacy Policy Project: Combining Crowdsourcing, Machine Learning and Natural Language Processing to Semi-Automatically Answer Those Privacy Questions Users Care About. Carnegie Mellon University (2013)
20. Zhang, L.-J., Li, C.: Internet of Things Solutions. Services Transactions on Internet of Things (STIOT) **1**, 1–22 (2017)
21. Habegger, B., Hasan, O., Brunie, L., Bennani, N., Kosch, H., Damiani, E.: Personalization vs. privacy in big data analysis. Int. J. Big Data (IJBD) **1**, 25–35 (2014)

Using Blockchain for IOT Access Control and Authentication Management

Abdallah Zoubir Ourad[1(✉)], Boutheyna Belgacem[1(✉)], and Khaled Salah[2(✉)]

[1] Universität Passau, Innstraße 43, 94032 Passau, Germany
abdallah.ourad@uni-passau.de, be@sec.uni-passau.de
[2] Khalifa University, Al Zafranah, Abu Dhabi, United Arab Emirates
khaled.salah@kustar.ac.ae
https://web.sec.uni-passau.de
https://www.kustar.ac.ae

Abstract. Securing Access to IOT devices is a challenging task as IoT devices are resource-constrained devices in terms of processing, storage, and networking capacity. Because of their fast spreading and deployment, significant disadvantages are seen in today's authentication and access control schemes. This paper proposes a blockchain-based solution which allows for authentication and secure communication to IOT devices. Our solution benefits greatly from the intrinsic features of blockchain and also builds on existing authentication schemes. Specifically, our proposed blockchain-based solution, architecture, and design allow for accountability, integrity, and traceability with tamper-proof logs. The paper provides overall system design and architecture, and details on testing and implementation of a realistic scenario as a proof of concept.

Keywords: Internet-Of-Things · Blockchain · Smart contract
Authentication · Access control

1 Introduction

Internet-Of-Things concept can be defined as a system of connected computing devices in various fields and forms that can be deployed everywhere. These devices can communicate with other devices, gather, share, and process information to deliver a certain service [12]. According to experts from CISCO, Ericsson and other companies, by the year 2020, we will have over 50 billion devices connected to the INTERNET [3,18]. IOT devices are functioning in all fields nowadays: house appliances, medical area, personal accessories, etc. To allow such functionality, these devices must have certain characteristics. They should posses the abilities of operating using low energy and communicating with other heterogeneous devices. Also, They should maintain stable connection with the back-end if existed and be able to receive patches when necessary.

Authentication and access control are key concepts to securely manage computer resources and networks. Based on the previously mentioned characteristics, these concepts should be redefined in the context of IOT. Authentication

D. Georgakopoulos and L.-J. Zhang (Eds.): ICIOT 2018, LNCS 10972, pp. 150–164, 2018.
https://doi.org/10.1007/978-3-319-94370-1_11

techniques and access control policies need to take the limited resources drawback into consideration. Likewise, when it comes to IOT conditions, classical approaches for access control like **ABAC** (Attribute-based access control) and **RBAC** (Role-based access control) are proven to be inflexible, unscalable and difficult to upgrade [6, 13].

In addition, the centralized perception of authentication where all devices have to contact a certain entity creates a major drawback. Building a system that depends on a trusted third party implies the assumption of a TTP that is always available and authentic. This creates a bottleneck around the trusted party which affects availability. Also, the model fails when the centralized entity is compromised. Additionally, the TTP can tamper records without accountability. Such disadvantages in the IOT design can be overcome using the blockchain technology.

Blockchain is known as the underlying technology for **Bitcoin** [9]. It can be defined as a growing chain of records. By design, the blockchain inherits effective characteristics in which the blocks of records are decentralized, tamper-proof and can be accessed by all nodes equally. This concept can be extended to all types of applications that require a trusted third party to validate records or transactions. Blockchain made it possible to replace the trusted third party with a transparent untampered block of records that is available via a distributed form, hence, the trust is moved from a single entity to a decentralized community of blockchain nodes.

An effective method that utilizes blockchain concept is the smart-contract. Smart-Contract was first defined in 1996 as a self-operating or self-executing program [16]. This method was reintroduced in Ethereum blockchain to facilitate the development of blockchain automated applications.

Events and logs are Ethereum blockchain features. An event is a reply (returned value) from the smart contract to the user interface interacting with it. The main goal of using events and logs is to ease the communication between smart contracts and programs communicating with them.

Figure 1 demonstrates a sample scenario for an Ethereum Smart Contract Application. First, the client requests access for a certain resource/asset from the smart contract. Second, the smart contract checks if the asset is free then books the fee from the client. For this example, the client is paying via Ethereum crypto-currency. Third, the smart contract reserves the resource for the current client. Fourth, The client uses the resource as approved. Finally, if everything went according to the contract rules, the smart contract charges the client as agreed. It is important to note that the smart contract acts fully autonomously and the owner is not involved in any steps.

The rest of the paper is organized as follows. Section 2 will review various implementations of IOT authentication solutions. Then, Sect. 3 will explain the detailed design of the proposed solution. Section 4 will clarify the implementation phases, elements, technologies and techniques. After that, Sect. 5 will illustrate the tests performed on this paper's approach, and the outcome results and

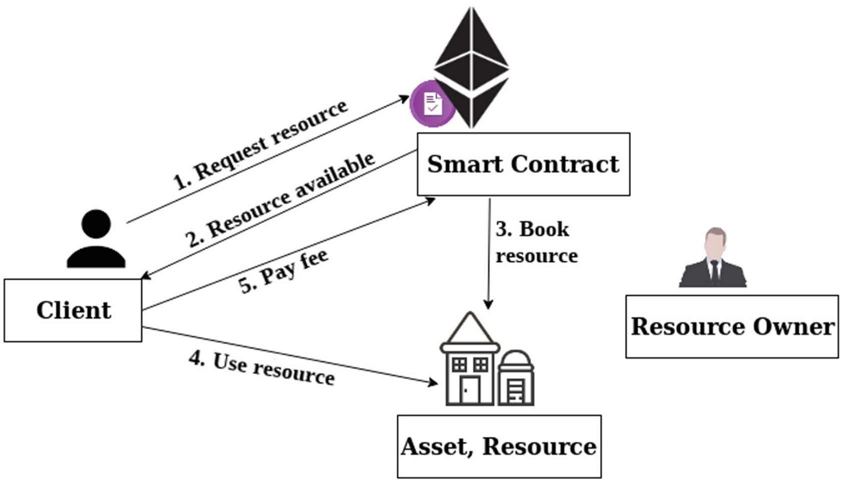


Fig. 1. A sample Ethereum smart contract scenario.

evaluation. Finally, the paper is concluded with the lessons learned and future plans that will help improving this solution in Sect. 6.

2 Related Work

This section presents different existing approaches to solve the problem of authentication management and access control in IOT devices. Section 2.1 will discuss traditional approaches used for authentication and access control in addition to their advantages and disadvantages. Solutions presented in Sect. 2.2 are distinguished by the fact that they all use blockchain as a backbone for their ideas.

2.1 IOT Authentication Traditional Models

A basic approach is to authenticate to each device directly using a combination of (username, password). This method provides decent access control because each authenticated user has his roles and permissions specified and stored on the device by the admin (owner). However, since the user must authenticate to each machine independently, this method produces an overhead and doesn't scale. This technique can be seen in classic IOT devices like IP cameras and Internet accessed home utilities.

Authenticating using single-sign-on protocols is a more advanced solution. For instance, when **oauth2** is deployed as an authentication method, users try to access a device by authenticating to a trusted oauth2 provider. This trusted third party can be GOOGLE, FACEBOOK, etc. If they successfully authenticate and have the required permissions, the trusted entity grants access. All devices managed by the same individual can be accessed by authenticating to the trusted entity [14]. Figure 2 illustrates an abstract of the Oauth2 protocol flow in the context of IOT. First, when the user tries to access the IOT device resources, the device act as the oauth2 client and sends an authorization request to the user. Second, the user grants the client the authority to communicate with the authorization server i.e. the oauth2 provider. Then, the IOT device contacts the oauth2 provider to access the user's information in order to check if such user is allowed to use its resources [14].

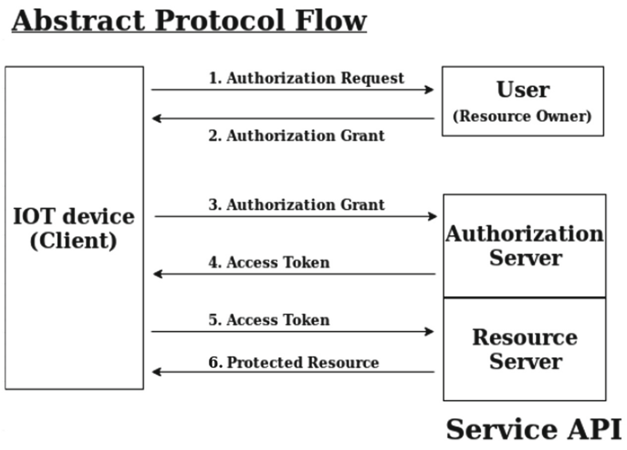


Fig. 2. An abstract for Oauth2 flow.

Using such approach saves time and effort since the user accesses multiple entities by authenticating to a single entity. Also, the oauth2 provider is usually a trusted third party with a high reputation which eases the integration of such solution [14].

On the other hand, trusting a centralized entity increases the threat of having a single point of failure which threatens the availability of the presented approach. Moreover, if the user's account or the centralized entity were compromised, the whole system is affected. An essential attack vector that may lead to this model's failure is phishing which has a high successful rate. In addition, spear phishing campaigns are getting more intensive, precise and sophisticated nowadays which may even trick the educated users [2]. According to Symantec Latest Intelligence Report for June 2017, 76% of organizations came forward of being phishing victims in 2016 [11].

The approaches discussed so far present a valid implementation for IOT authentication. However, they suffer from some drawbacks that may affect scalability, performance and availability. The following approaches have the potential for an IOT authentication management and access control solution that uses blockchain technology.

2.2 Blockchain Based Authentication Models

Auth0 introduced a method to authenticate to a server via Ethereum blockchain using a challenge-response method. The drawback discovered in auth0 approach is the need of a 3rd party authentication server. It is a hybrid solution that combines the decentralization of blockchain with the centralization of a trusted third party. The problem of "Single Point of Failure" or "Single Point of Trust" will reappear with this approach. According to **Auth0**, the centralized server holds an essential role since it is involved in 62.5% of the whole operation. This increases the dependency on a centralized entity which contradicts the benefit of using blockchain technology [10].

Blockstack is presenting the concept of a new decentralized INTERNET. This network contains applications for authentication and storage. The system utilizes a user's keypair to authenticate in a similar manner to PKI. When a users is authenticated correctly, a unique JSON Web Token is created. The JWT permits access to all pre-authorized resources via the one authentication process validated previously.

On the downside, blockstack has many requirements to operate. The goal of this system is to shift to a newly decentralized network therefore, the first requirement is to use a blockstack browser. If not, then blockstack app should be installed on the user's machine. In addition, the system built an additional two layers on top of blockchain **- peer network layer and storage layer -** which increases the operational overhead. Finally, the entities interaction is based on a new domain name protocol called the **Blockchain Name System** as a replacement for traditional DNS [7]. BNS is one of many novel implementations that are aiming to replace classic domain name system. More examples include NameCoin and Ethereum Name Server [1].

3 Proposed Systems Design and Architecture

The paper offers a blockchain based solution with distinct system architecture. It addresses the drawbacks of current solutions. Also, it should be portable and run on any network with minimum dependencies unlike **blockstack**. It is targeting IOT devices that lack strong processing power. It is also presenting the idea of Oauth implementation via smart contract to login once and control all the authorized devices without the need to login separately for each IOT device. In addition, the IOT devices can run smart contracts to become self profiting devices.

Section 5.1 will discuss the costs of this solution. Section 5.2 will go over the tests operated on a running prototype in addition to the test outcome. Tests will include performance experiments and crafted attacks against them. Then, Sect. 5.3 will evaluate in terms of availability, scalability, decentralization, tamper proof and performance advantages and drawback.

There are many advantages for using Ethereum blockchain as a platform for this solution. Ethereum has a stable development framework with existing incentive for miners to solve challenge hashes. Also, Ethereum light client protocol can run on IOT devices with low processing power and memory which is essential for the proposed solution [8].

One Time Authentication: Authenticate once Directly to the Blockchain then Access the Resource Using Smart Contract Tokens: In this scenario, the user authenticates to the smart contract which verifies his/her identity. The smart contract then determine if the user is allowed to access the resources. The authentication is performed in an isolated phase. When succeeded, the user can interact with the device in any preferred method e.g., ssh, http, https, etc. This accomplishes decentralized authentication in an efficient way. The evaluation of this solution will be discussed in Sect. 5.

3.1 Assumptions

To implement such solution, the following assumptions must be taken into consideration:

– The user owns one or more IOT devices.
– The user's Ethereum keystore is not compromised i.e. the private key is protected.
– The user has an Ethereum account.
– The user and the IOT device are connected to the Ethereum blockchain
– The user will deploy his smart contract.

The system can budge the last assumption with full functionality. A centralized smart contract that authenticate users to their respective IOT devices can be created. However, since one of the goals of this paper is to avoid depending on a central entity. It is more suitable to ask users to deploy their own smart contracts so they can achieve full control of their own systems. This will shape a system of decentralized smart contracts running on a decentralized blockchain where each user owns his/her smart contract.

3.2 System Architecture

The message sequence diagram shown in Fig. 3 illustrates the steps of the authentication process as follows:

1. The user Authenticates to the smart contract using his Ethereum wallet address.
2. If the user is valid, the smart contract broadcasts an Access token and the sender's ethereum address. The user and the IOT device receive the broadcasted information from the smart contract.
3. The user crafts a package that contains (Access token, User IP, Access duration and the ethereum public key). This package is signed using the ethereum private key then sent with the corresponding public key. The package can be encrypted if wanted. However, It is not required for the protocol to operate. Integrity is what matters in this scenario hence the signing of the message.

4. When the IOT device receives the package, it verifies its content. If succeeded, the device grants access to the user from the sender's IP for the duration specified. Otherwise, if any of those checks fails, the request is dropped. The verification phase is described in Sect. 4.2.

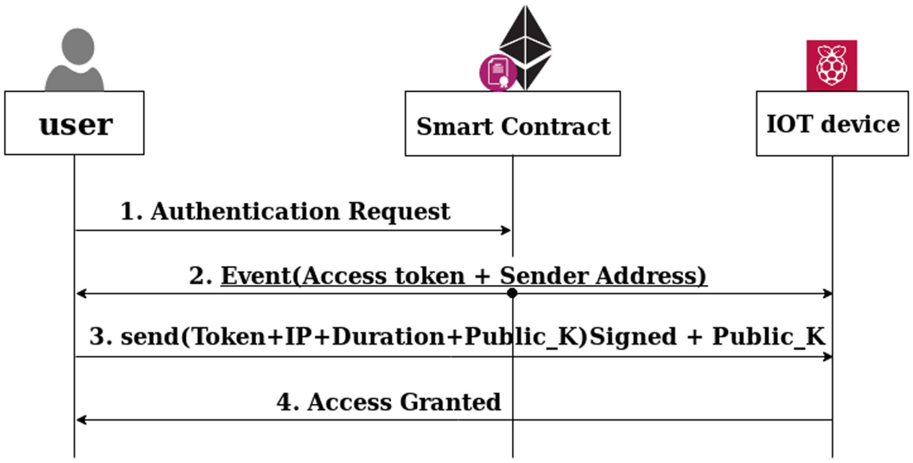

user
Smart Contract
IOT device
1. Authentication Request
2. Event(Access token + Sender Address)
3. send(Token+IP+Duration+Public_K)Signed + Public_K
4. Access Granted

Fig. 3. The second solution authentication scenario.

The smart contract completes its authentication task in the first step. It is noted that the IOT device needs to perform many verification steps. However, as discussed in Sect. 5, this solution runs successfully on a standard raspberry Pie 3 Model B. More details on the technical specifications of the smart contract and the IOT server are explained in Sect. 4.

4 Implementation and Deployment

Implementing the proposed solution can be compartmentalized to the separate phases that can be built individually then integrated accordingly. This will facilitate the developing process. Also, debugging will be easier once challenges are faced. The presented solution will be explained in two subsections. The first subsection will explain the Smart contract functionality and how it performs authentication as the first phase. The second subsection discusses the authentication interaction between the user and the IOT device after the first phase is successfully executed.

4.1 Phase 1: Smart Contract

As mentioned in Sect. 3, the first phase occurs when the user authenticates to the smart contract to prove he/she is a legitimate user. Pseudo-code below describes the source code of the smart contract. This version of the smart contract only considers the admin user as a legitimate user. In other words, the user who deployed this smart contract on the blockchain is the admin user. Adding more users can be achieved by adding an *addUser()* method to the smart contract.

The Smart Contract of the Proposed Solution:

```
contract Login2
Declare Private owner, hash, token_raw, random_number
//begin constructor:
owner = msg.sender;
End Constructor

Private Function login_admin()
IF msg.sender == owner
   set random_number = random(1,100);
   set hash = keccak256(msg.sender,now,random_number);
   trigger even LoginAttempt(msg.sender, hash);
Endif
End Function
End Class
```

(Source code is available on: https://github.com/sasukeourad/OTA)

When users want to authenticate, they use their Ethereum client to call method *login_admin*. The *login_admin* function requires no parameters and it is protected from public usage i.e. only authorized users can call it because a modifier performs the verification of the sender's Ethereum address. If a verified user calls this function, *login_admin* creates a random hash using function *rand*. Then, *login_admin* creates a token by hashing the user's ethereum address, block time and the random hash created in the latest step. After that, an event is initiated to send the token and the authenticated user's address back to the IOT device and user to proceed with phase 2.

4.2 Phase 2: User-IOT Interaction

When phase 1 is completed successfully, the user and the IOT device receive an authentication token and the Ethereum address of the authorized user. Phase 2 connects the two entities together. Note that this solution assumes that the user knows the address - ip or domain name - of the IOT device. In case this wasn't true, the device address can be sent by event LoginAttempt.

User Side Implementation Flow: First, by using **nodejs** and **web3 Ethereum client**, the user's script connects to the Ethereum blockchain listening for events coming from the deployed smart contract in phase 1. When the event arrives, the user's script accesses the keystore directory that contains the user's secret keys and extracts the private key. Note that the script needs the key pass in order to perform such action. Then, the script extracts the public key from the private key. The public key is hashed using **keccak256** algorithm. The last 40 bytes of the resulting hash is the Ethereum address of the user. This Ethereum address is compared with the one received from the smart contract event. This is the formal method to obtain an Ethereum address from a keystore directory. Note that the public key usage is minimized and replaced by the Ethereum address since the address is shorter and easier to use. This is achieved using **keythereum** to access the private key and **elliptic** to derive the public key. Those nodejs libraries can be found online [4,17]. Figure 4 shows a message sequence diagram for the process of extracting Ethereum addresses from keystores [4,17].

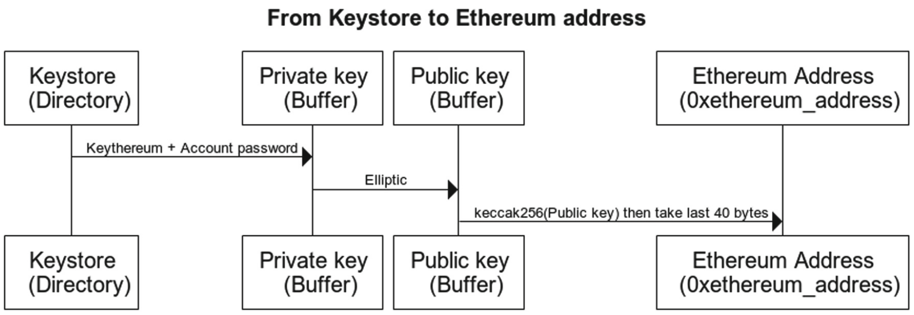

From Keystore to Ethereum address
Keystore (Directory)
Private key (Buffer)
Public key (Buffer)
Ethereum Address (0xethereum_address)
Keythereum + Account password
Elliptic
keccak256(Public key) then take last 40 bytes
Keystore (Directory)
Private key (Buffer)
Public key (Buffer)
Ethereum Address (0xethereum_address)

Fig. 4. The process of deriving the Ethereum address from the user's ethereum keystore

After assuring that the Ethereum address received from smart contract is the user's address. The script starts crafting the authentication message. The message can be described as following:

$$message = [token + src\_ip + Auth\_dur + Pub_K] \tag{1}$$

Where:

- **token:** is the token received from the smart contract.
- **src_ip:** is the ip address the user will connect from.
- **Auth_dur:** is the duration for the authentication validity before it is revoked and another authentication is required.
- **Pubk:** is the public key of the user's ethereum account.

Finally, the message is signed using the private key of the user's ethereum account. The following authentication package is sent to the IOT device:

$$message + Signature + Pub_K \tag{2}$$

IOT Side Implementation Flow: The IOT device script starts in a similar manner to the user script. It connects to the smart contract deployed in phase 1 and listens for the desired event. When the event occurs, the script gets the authentication token and the Ethereum address of the authenticated user. The script waits for the user's authentication package to arrive. If the package arrived, the verification phase starts. If any of the verification steps fail to succeed, the authentication package is dropped to minimize the workload on the IOT device processing power. The process moves to the next step only if the current step is verified.

Verification Phase Steps:

1. Is the authentication package and message in the correct format as shown in Eqs. 1 and 2?
2. Is the message signature valid? This is checked using the public key in Eq. 2
3. Is the public key in the authentication package in Eq. 2 similar to the one in the message in Eq. 1?
4. Is the token in the message similar to the token from the smart contract?
5. Is the source ip in the message similar to the source ip of the authentication package sender?
6. By hashing the public key in the message and taking the last 40 bytes ... Is the result similar to the Ethereum address from the smart contract?

If all those phases were verified, the user is authenticated. Otherwise, the authentication package is dropped. According to BigO standards, the IF-statement adds a linear execution complexity to the program depending on the input. It is denoted as:

$$O(kn) = O(n) \tag{3}$$

Where k is a constant.

5 Testing and Evaluation

After implementing the prototype of the proposed solution, this section discusses the tests performed to assure their security and functionality. Also, the system is evaluated against the other solutions discussed in Sect. 2. In addition, the cost required to establish the communication between the user and the smart contract is covered. Note that generally, only "setter" functions will cost the user since they require miners to modify the blockchain where the "getter" functions don't cost any time or money.

To modify smart contract attributes, a transaction should be made in the Ethereum blockchain. The fee is calculated in GAS and paid in Ether. When the smart contract is deployed, the owner has the choice to set the value of the amount of gas required. A higher gas price means this transaction is mined first. It is a trade-off between priority and cost. In December 2017 the price of "21K" gas used is \$0.01. This gas amount translates to **2 gwei** which is the standard speed to perform a transaction. This cost is arguably acceptable since it provides a decentralized authentication platform in addition to a tamper-proof block of records. Otherwise, the alternative solution would be to trust a centralized entity with your data where the threats of single point of failure and losing sensitive data increase. A third option is to manage a self-owned decentralized database which costs more for maintenance and management.

The fluctuating value of Ethereum price can raise a challenge for smart contract users. Since the deployment test of this paper's smart contract and according to Fig. 5, Ethereum value has peaked for certain interval then returned to the normal price. The fluctuating price of cryptocurrencies is a drawback that affects smart contract usage stability. However, this is planned to be solved in the future consensus algorithms proposed by Ethereum.

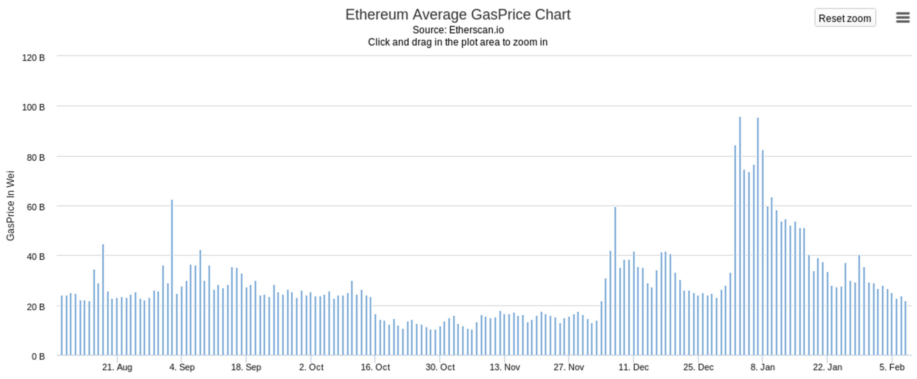


Fig. 5. Ethereum average gas price

5.1 Costs

Table 1 illustrate the transaction cost, execution cost and the equivalent price in US dollars for deploying and using the smart contract.

Table 1. Gas cost for running functions

Function	Transaction cost	Execution cost	Total cost	Price in USD
Deployment	275153	170457	445610	$0.212
login_admin	64089	42817	106906	$0.051

First rows show the price of deploying the smart contract which is only done once. It is clear that deploying the smart contract is more expensive since it writes it on the blockchain. On the other side, the login functionality is cheaper.

The login function can be optimized to cost less. The reason for current price is that it generates the authentication token by hashing, generating a random number and retrieving the block hash. These prices represent the cost of using the traditional proof-of-work consensus protocol. With Ethereum moving to standardizing the proof-of-authority protocol, these costs will decrease noticeably since the miners' work load will decrease too.

5.2 Testing

The testing phase was divided into different subsections. First, manual tests are performed against the proposed solution to assure its robustness, security and performance. Manual tests include malicious scenarios in additional to the ideal test cases. Second, static analysis tools are used to perform automated security assessment for the smart contracts.

Manual Testing: After running the solution prototype, the ideal scenario was tested first. A legitimate user calls the smart contract function: *login_admin* using his/her MIST Ethereum client. The smart contract sends the authentication token and the user's Ethereum address to the user and IOT device simultaneously. According to the test, the first phase was completed in less than 4 s on a private blockchain. Then, the user connects to the IOT device by sending the authentication package explained in Eq. 2.

Bypassing the verification steps in the IOT authentication script were tested by performing few malicious attacks as explained below:

- A replay attack failed since the attacker's source IP needs to be identical to the source IP in the signed authentication message.
- Modifying the signed authentication message failed since the script verifies the message signature.
- Injecting the attacker's own authentication package failed since the public key should lead to the Ethereum address of the legitimate user.
- A man-in-middle may be able to sniff outgoing authentication packages. However, integrity is protected since he/she cannot modify the signed authentication message.

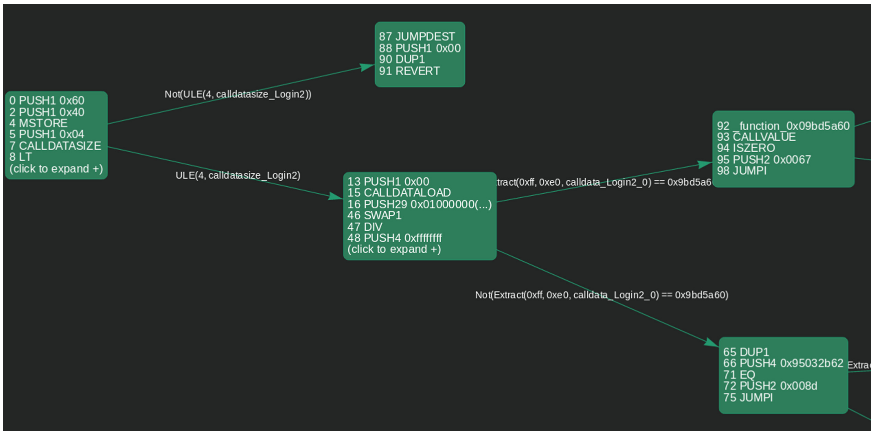
The outcome of the test phase proves that this solution is secure as long as the user's keypairs are not compromised. The authentication tokens should be invalidated and replaced once the authentication is successfully completed. The next steps are outside the scope of this paper.

When testing the current solution, the test environment varies. First, most tests are conducted in a private ethereum blockchain. This eases the process of mining and validating transactions. After that, when the smart contract is tested in the public Ethereum blockchain test-net, it is recommended to use **Rinkeby** instead of **Ropsten** since it uses Proof-of-Authority instead of Proof-Of-Work which is used in **Ropsten**. This will also ease the public testing process.

The proof of work concept is the current method used in Ethereum main network and Ropsten network to confirm transactions. A miner solves a mathematical puzzle to validate the transaction for an incentive. This approach requires a lot of processing power to execute. On the other hand, the proof of authority implemented in the Rinkeby test-net depends on a set of explicitly authorized nodes instead of miners solving mathematical problems therefore, it is considered the future of mining techniques where mining doesn't require as much processing power [19].

Automated Testing: Static Security Analysis: Security assessment via static analysis was performed using **mythril** by ConsenSys. It uses concolic analysis to detect security issues in smart contracts. It can operate in both whitebox and blackbox testing scenarios. Since the smart contract source code is available, whitebox testing was performed on it. Mythril returned no issues.

In addition, a control flow graph is created for the smart contract source code to assure all possible paths are checked. The graph is created using Ethereum Laser Symbolic Virtual Machine. Figure 6 shows the starting point of the smart contract control flow diagram.



```
0 PUSH1 0x60
2 PUSH1 0x40
4 MSTORE
5 PUSH1 0x04
7 CALLDATASIZE
8 LT
(click to expand +)
```

```
87 JUMPDEST
88 PUSH1 0x00
90 DUP1
91 REVERT
```

```
13 PUSH1 0x00
15 CALLDATALOAD
16 PUSH29 0x01000000(...)
46 SWAP1
47 DIV
48 PUSH4 0xffffffff
(click to expand +)
```

```
92 _function_0x09bd5a60
93 CALLVALUE
94 ISZERO
95 PUSH2 0x0067
98 JUMPI
```

```
65 DUP1
66 PUSH4 0x95032b62
71 EQ
72 PUSH2 0x008d
75 JUMPI
```

Not(ULE(4, calldatasize_Login2))
ULE(4, calldatasize_Login2)
tract(0xff, 0xe0, calldata_Login2_0) == 0x9bd5a6
Not(Extract(0xff, 0xe0, calldata_Login2_0) == 0x9bd5a60)
Extra

Fig. 6. Starting point of Login2 CFG

Finally, another metric that can be proposed for future testing is performing formal method tests on the smart contract to verify that all possible execution paths are covered and anticipated. Current efforts are documented in [5,15].

5.3 Evaluation

To assure the quality of the proposed solution, the next step is to compare it to previous solutions presented in Sect. 2. The evaluation metric is based on

Table 2. Comparing and evaluating authentication solutions

	Auth/device	Oauth2	Auth0	Blockstack	Paper sol.
Availability	✓	**X**	**X**	**X**	✓
Scalability	**X**	**X**	✓	✓	✓
Decentralization	✓	**X**	**X**	✓	✓
Tamper proof	**X**	**X**	**X**	✓	✓

whether the offered authentication scheme solved problems occurring in the other authentication mechanisms proposed for the IOT devices.

Table 2 shows a comparison between the proposed solution based on Availability, Scalability, decentralization and tamper proof. For this comparison, the evaluation metrics is defined more specifically. Availability is described as removing the bottleneck and functioning without a single point of failure. Scalability is used here to explain the added overhead to the usage of the application when more devices are added. Decentralization is the ability for the authentication application to run without depending on a central entity that may break the system if taken down. Tamper proof is the assurance that saved data and transactions cannot be tampered once registered in the logs of the system.

6 Conclusion

In this paper, we have proposed a blockchain-based solution to proved authenticate users to access securely IoT devices. We demonstrated how our approach overcomes the shortcomings of existing authentication schemes. We showed that our blockchain based solutions, using Ethereum smart contracts, can provide tamper proof records and decentralization to improve current approaches. We designed and implemented our solution considering real life scenarios using available IoT devices and technologies. Specifically, we showed how to successfully authenticate legitimate users to access their IOT devices. Also, we showed that our approach withstood crafted attacks that were attempting to hijack legitimate sessions and brute force credentials. As a future work, we plan to extend our the proposed approach with a massive scale access and authentication to include huge number of IoT devices and end users. We also plan to test the approach on real Ethereum blockchain network and measure performance in terms of cost (or gas) consumption and scalability. We also considering monetization aspects related to IoT devices and their data, whereby usage is paid through crypto-token of ether.

References

1. Al-Bassam, M.: SCPKI: a smart contract-based PKI and Identity System (2017)
2. Amadeo, R.: Don't trust OAuth: why the "Google Docs" worm was so convincing. https://arstechnica.com/security/2017/05/dont-trust-oauth-why-the-google-docs-worm-was-so-convincing/
3. Evans, D.: The Internet of Things. How the Next Evolution of the Internet Is Changing Everything (2011)
4. George V.: A Next-Generation Smart Contract and Decentralized Application Platform (2018)
5. Hirai, Y.: Formal Verification of Ethereum Contracts (2018)
6. Liu, J., Xiao, Y., Chen, C.: Authentication and access control in the Internet of Things. In: 2012 32nd International Conference on Distributed Computing Systems Workshops, pp. 588–592 (2012)
7. Ali, M., Shea, R., Nelson, J.: Blockstack: A New Internet for Decentralized Applications (2017)
8. McKinney, J.: Light client protocol (2017)
9. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
10. Peyrott, S.: An Introduction to Ethereum and Smart Contracts: an Authentication Solution. https://auth0.com/blog/an-introduction-to-ethereum-and-smart-contracts-part-3/
11. Symantec: Latest Intelligence for June 2017. https://www.symantec.com/connect/blogs/latest-intelligence-june-2017
12. Minerva, R., Biru, A., Rotondi, D.: Towards a definition of the Internet of Things (IoT) (2015)
13. Gusmeroli, S., Piccione, S., Rotondi, D.: IoT access control issues: a capability based approach. In: Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (2012)
14. Sandoval, K.: Why OAuth 2.0 Is Vital to IoT Security. https://nordicapis.com/why-oauth-2-0-is-vital-to-iot-security/
15. Swamy, N., Hriţcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., Bhargavan, K., Fournet, C., Strub, P., Kohlweiss, M., Zinzindohoue, J., Zanella-Béguelin, S.: Dependent types and multi-monadic effects in F*. SIGPLAN Not. **51**, 256–270 (2016)
16. Szabo, N.: Smart Contracts: Building Blocks for Digital Markets (1996)
17. theethereum: Accounts, Addresses, Public and Private Keys, and Tokens. https://theethereum.wiki/w/index.php/Accounts,_Addresses,_Public_And_Private_Keys,_And_Tokens
18. Thomson, D.: IoT and the problem of identity. https://www.symantec.com/connect/blogs/iot-and-problem-identity
19. Tosh, D., Shetty, S., Liang, X., Kamhoua, C., Njilla, L.: Consensus protocols for blockchain-based data provenance: Challenges and opportunities. In: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), pp. 469–474. IEEE (2017)

A Middleware Mediated Application Layer Protocol to Decouple Provider-Consumer Relationship in Web Services Orchestration and Its Application in Novel IoT Integration

Devanathan Venkatesan(✉), Othiyappan Pandithurai, and Sundaramoorthy Sridhar

Anna University Chennai, Chennai, India
d.venkatesan@aubit.edu.in

Abstract. At present URI based web service orchestration poses a fundamental limitation on the nature of resulting web service applications restricting them to client-server paradigm and the partnership determination as a static design time activity. This naïve simplicity limits the scope and capability of orchestration technology that shows up sorely in our inability orchestrate IoT devices using application level web service protocols/technology. This work suggests and demonstrates a middleware (named here as Open Interaction Middleware Services-OIMS) based web service orchestration approach that unfetter the client-server nature of orchestration and introduces run time establishment of provider-consumer relationship. OIMS mediated orchestration permits speech-act based specification and orchestration of partners of the collaboration. OIMS functionality deserves to be a part of the WS infrastructure eventually. Decoupled service orchestration permit creation of novel application layer level web services based protocols and applications – that has the merits of both bus and broker based protocols combined - such as in the case of Internet of Things (IoT) monitoring & integration. This work illustrates the proposed approach using a case study in integrating IoT devices belonging to multi class IP network that provides several important quality attributes such as loose coupling and scalability to the resulting environment.

Keywords: Speech acts · Middleware based web service orchestration Application integration · Loose coupling · Decoupled orchestration Provider-consumer · Case study · WS infrastructure

1 Introduction

Explicit client-server nature of web service composition standards warrant determination and specification of WS provider address at the design time, resulting in applications that does not possess several important quality attributes such as loose coupling and separation of concern in a distributed computing environment (Venkatesan and Sridhar 2016, 2017). This deficiency shows up sorely in situations such as integration of IoT devices using web services standards where consumer cannot statically specify the provider URI addresses at the design time. This deficiency has forced IoT technology to groom kludges and low level ad-hoc protocol techniques to accomplish integration (Al-Fuqaha et al. 2015). This work presents a middleware based orchestration technology (that breaks off client-server nature of orchestration) and expand on the idea presented in Venkatesan and Sridhar (2016, 2017). A case based (Yin 2003) illustration is a good way to present and evaluate the outcome of the novel ideas and so this work adopts the case of IoT device integration. It should be noted the proposed protocol and middleware should be capable of handling integration with smart objects/small devices (that are power/resource constrained) that may have difficulties in dealing with protocols and SOA technologies designed for full-powered computers. Results obtained in this case may be extended subsequently to a more general situation and WS standards may be suitably amended.

This work is organized into 6 sections. Section 1 introduces the problem and outlines an approach and context of the solution. Section 2 describes over all architectural layout for message based orchestration (IEEE-FIPA 2018) using intelligent middleware namely OIMS. It describes the consumer application namely Intelligent Open Interaction Application Framework (IOIAF), messaging format used by IOIAF, its manner of interaction with OIMS and algorithm used by OIMS to determine and invoke WS providers. Section 3 describes the application layer level interaction protocol patterns, and implementation details of a working prototype (GitHub hosted) that uses Microsoft Windows Communication Foundation (WCF) technology (Sharp 2017). Section 4 analyses the results and makes a comparison with other popular IoT integration protocols. Section 5 provides a review of related literature. Section 6 provides a conclusion and proposal for further work. The suggested middleware technology is inspired by the works of Okouya et al. (2013).

2 The Open Interaction System Middleware

Venkatesan and Sridhar (2016) and Chap. 8 of Venkatesan (2018) propose an application layer protocol for integration of IoT devices using speech act messages and WS technology (as depicted in Fig. 1). This application involve a centralized web services consumer namely Intelligent Open Interaction Application Framework (IOIAF) based controller/dashboard, an agent oriented middleware service namely OIMS that provides facility to receive and transmit (IEEE-FIPA standard like speech act messages that carry information to carry out WS orchestration on providers using SOAP messages) across an ecosystem of IoT compute nodes connected to it. OIMS services act as intermediaries in relaying the messages to appropriate WS provider destination. The deployment level block diagram of this environment is depicted in Fig. 1.

A schematic of the proposed integration environment is provided in the Fig. 2. This environment utilizes Microsoft Windows WCF ver. 4.5 technology at OIMS to realize discovery of partnerlink services, enumerate them, enumerate services interfaces/input and output parameters and bind with them to invoke. Interaction between IOIAF service and OIMSs take place using Microsoft Message Queue (MSMQ) technology where the messages are of FIPA-ACL kind as described in subsequent section. For light-weight cases where performance and simplicity is the criteria, the interaction between IOIAF and

OIMS is alternatively realized designed using WCF itself as it is the case between OIMS and IoT leaf nodes. A IOIAF server shall connect with one or more OIMS middleware servers using user supplied configuration information. OIMS in turn connect with other OIMS or IoT leaf nodes to execute the contract as indicated by IOIAF application.

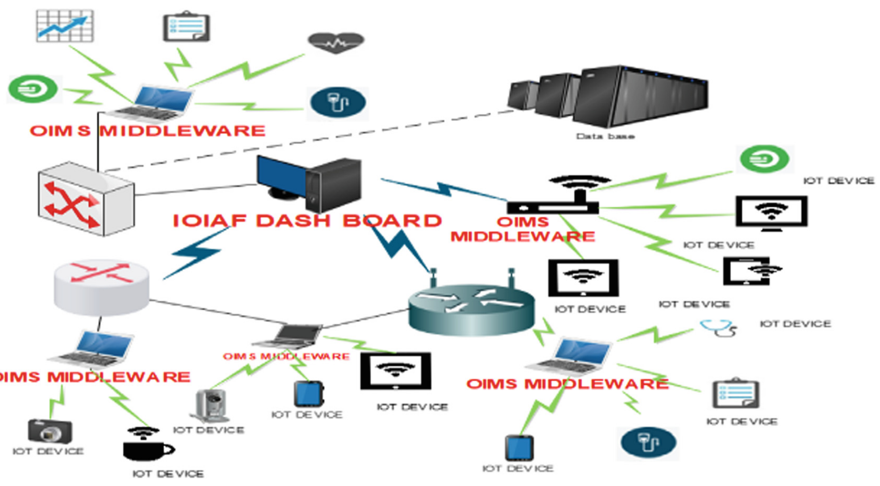


Fig. 1. Block diagram of IoT integration environment

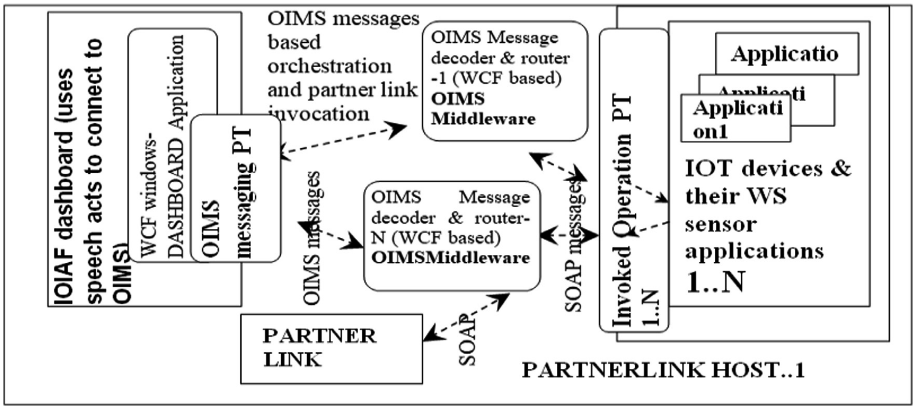

IOIAF dashboard (uses speech acts to connect to OIMS)
WCF windows-DASHBOARD Application
OIMS messaging PT
OIMS messages based orchestration and partner link invocation
OIMS Message decoder & router -1 (WCF based) OIMS Middleware
OIMS messages
OIMS Message decoder & router-N (WCF based) OIMSMiddleware
SOAP messages
Invoked Operation PT 1..N
Applicatio
Applicati
Applicati on1
IOT devices & their WS sensor applications 1..N
PARTNER LINK
SOAP
PARTNERLINK HOST..1

Fig. 2. Schematic of IOIAF orchestrator, OIMS middleware and IOT device (WS-Partner Link) leaf nodes

The operational algorithm of IOIAF service is described in the Fig. 3. The designer and programmer decide on the syntax and semantics of speech act messages based on nature of requirement demanded by the end user or business case. IOIAF service sends this message to various connected OIMS middleware service nodes. OIMS middleware service receives the IOIAF messages and decodes them to understand the nature of partner link invocation to be performed by it (for example asking for a specific set of sensor data from IoT node service). Based on the SA it communicates with various IoT nodes connected to it and consolidate the result and reply back to IOIAF server. The SA based interaction models (requirement scenarios) also provides hints that can be translated into system level functional test cases.

```
Algorithm IOIAF_Node_IoT_Data_Gatherer (input LIST oimsMWNodes)
loop ForEvery("HH:MM:SS")
{   iotData = NULL;
    LIST oimsNd = RefreshListOfConnectedOims (oimsMWNodes);
    SA qry= Generate IoTNodeQuery(input UsrIP);
    for each [oimsNd]
    {  MQ mq_channel;
       mq_channel  = Open Message Queue (Destination oimsNd)
       Send_SA_Msg_to_oimsNd (qry , oimsNd, mq_channel , qry);
       iotData = { iotData } U  receiveIoTDataFromOIMS_Middleware_Nd
       (oimsNd);
    } Report to consumer { iotData}
}
```

Fig. 3. IOIAF application (WS-Consumer) node "Processing" logic

The algorithmic description of OIMS capability is depicted in Fig. 4. Typically the IoT integration specific OIMS network shall have one or more controller or dashboard, each subscribing to an unspecified number of leaf IoT nodes and collect/control information and process (in accordance with the type of collaboration as listed in Table 1).

```
Algorithm OIMS_Middleware_Process (INPUT  IOIAF_iotSA)

   LIST NTWRK_Interfaces =Enumerate_NTWRKInterfaces_In_OIMSNode( );
   LIST OimsMsgQueue= RefreshOimsMsg_Queue();

   for each "NwIntrfc" item in [NW_Interfaces]
       LIST OimsSrvrs= Enumerate_attached_OIMS();

   Relay_Selectively_Msgs_to_Other_OIMS_And_FlushQueue (OimsMsgQueue );
   DATA_RECORDS iotData =NULL, ndData=NULL;
   for each "NwIntrfc" item in [NW_Interfaces]
   {
       LIST iotNd  = Enumerate ConnectedIoTNodes (NwIntrfc);
       //iotNode consists of IP addresses in the same Local Area Network
       for each [iotNd ]
       {
           soapMsg = Decode_SA_specific_to_node (IOIAF_iotSA);
         if (  needToInteractWithIoTNode (iotNd )==TRUE )
           {
           ndData = invoke_IoTNd_PL (iotNd, soapMsg);
           iotData = iotData U ndData;
           }
       }// all IoT nodes of a particular Network Interface contacted
   }// all IoT nodes of a particular Network Interface contacted
```

Fig. 4. Processing control loop in OIMS

While the centralized dashboard (such as closed loop video surveillance network) shall use "ask" and "tell" speech acts through OIMS and other network gateways on the IP network, a local dashboard application of an operation theater instruments shall use subscription to selective IoT devices without intervention of gateways or centralized servers but with the help of OIMS alone).

The OIMS makes use of WS-Addressing technology (for synchronous real-time data processing/Message queues (for asynchronous data processing) along with WS-Eventing technology to realize much of its infrastructure based loosely coupled integration capability. It may be noted OIMS and IOIAF support scalability like *per-instance, per-session or per-call semantics* based service provisioning to support state based, configuration based, service rendering in the case of multiple IOIAF dashboards on the same shared physical network infrastructure – say a server farm - as in the case of, for example a hospital network that has separate dashboards (IOIAF applications)

Table 1. Patterns of interaction between IOIAF and OIMS

Messaging Pattern Type /(Abbreviation)	**Sample SA Message format (as formatted by IOIAF app)**

Point-to-Point (PTP)	{
	sa_type : "ask"; sender_id : "192.168.1.2"; receiver_id : "192.168.2.12"
	command_query: {
	NODE-ID=ROOM203-DEV1 &&
	TYPE=SENSOR-DATA &&
	SENSOR-ID =SENSID001 &&
	REPEAT-TIMES=100 &&
	REPEAT-FREQ=:01:00:00:00 //format DD:HH:MM:SS:MSEC }
	}
	//This message is interpreted by OIMS; This message means to gather data from a specific IoT node with Id. ROOM203-DEV1, for a specific sensor Id. SENSID001, for every I second for next 100 times. OIMS will keep the state info. And query the IoT node for data next 100 times every second and report the datum to IOIAF dash board.

Point-to-Everybody (PTE)	{sa_type : "ask"; sender_id : "192.168.1.2"; receiver_id : "*";
	command_query: {SENSOR-DATA }
	}
	//This message is interpreted by OIMS; it means to gather data from all connected IoT node covering all connected sensors.

Broadcast (BBB)	{
	sa_type : "tell";sender_id : "192.168.1.2"; receiver_id : "*"
	command_query: {SLEEP }
	}
	//This message is interpreted by OIMS, meaning to inform intermediary OIMS to desist from contacting connected IoT nodes; they may cease to function until further instruction from Dashboard controller. This may be used to save IoT node battery and address lull in need to gather data.

Point-to-Multipoint (PTM)	{sa_type : "ask" sender_id : "192.168.1.2" receiver_id : "*" command_query: {SENSOR-DATA }} //This message is interpreted by OIMS, means to gather data from all connected IoT node covering all connected sensors.

for video surveillance, visitor service, patient service, hospital housekeeping, patient monitoring etc.… that all still uses same OIMS middleware – say that are running on the network intermediary hardware – capable of supporting and sustaining multiple conceptual logical IoT networks using same WS executable using scalability (per-session invocation) logic.

The number of OIMS middleware attached to IOIAF server can be handled using different approaches. A straightforward approach is to provide a configuration file to IOIAF server that lists the IP addresses of the attached OIMS middleware servers. This work uses *ioiafconfig.ini* (Fig. 5a) to supply a list of OIMS nodes connected to IOIAF server. Alternatively the available list of OIMS nodes connected to an IOIAF server can be determined and discovered by a dynamic discovery protocol for OIMS middleware (like the case of DHCP or DNS server discovery). In turn, OIMS middleware server node is further configured using a *"oimsconfig.ini"* (Fig. 5b) file. This file provides a list of class of connected IoT networks its network-interface. OIMS search periodically over the entire subnet address space (wired or wireless) to discover available leaf nodes (i.e. IoTClient nodes), at the given moment, using node discovery protocols. This work uses a polling scheme for each IP address. Once the connected IoT leaf nodes are determined, OIMS will start interrogating the IoT leaf nodes for data as indicated by OIMS messages delivered to it by IOIAF server.

The IoT sensor nodes can also be configured to inform like the name of the sensors connected to it and the port details of the sensors (or hardware connection Id) (Fig. 5c). All these configuration information should use a shared ontology so that the received "commands" (from IOIAF application and OIMS messages) and reported "data values" are understood without ambiguity. This will help the entire IoT network work in a standardized manner and able to serve various custom queries. In cases where no standard compliance is available in IoI node, the IOIAF application need to specialize its messages taking into account specific terminology adopted by custom IoT device class. This research work uses a configuration file namely *Iotnodesvc.ini* (Fig. 5c) to supply configuration information of (*simulated)* IoT leaf node. This research assumes each IoT node runs a WS to permit data acquisition or else has at least provides a proprietary custom WS for data acquisition details of which can be supplied to the interfacing OIMS as customized node specific orchestration information for a specified unique node Id.

```
#File IOIAFconfig.ini…
#list of connected OIMS
[ICURoom208]
192.168.2.1=1
192.168.3.1=1
10.1.1.1=1
[SurveillanceCamera]
192.168.2.1=1
10.1.1.4=1
```
(a)

```
#File OIMSconfig.ini…
#1-use network; 0-dont
[CameraLAN]
10.1.60.*=1
10.1.70.*=1
[ICUR208]
192.168.2.*=1
[ICUR209]
192.168.3.*=1
```
(b)

```
# IoTNodeSvc.ini
#1-use sensor; 0-dont use
[AttachedSensors]
TempSnsr=1
HumiditySnsr=1
 [IOTId]
iotNodeId=auIT01
```
(c)

Fig. 5. (a) IOIAFconfig.ini (b) OIMSconfig.ini (C) IoTNodeSvc.ini

End users will invoke IOIAF application with a process initialization argument (i.e. command-line argument) specifying its application identification. Corresponding application identification must be there in the *ioiafconfig.ini* so that this particular instance of IOIAF service goes on to initialize OIMS service instance belonging to it completing the bootstrap of network of monitoring application chain (to accomplish a given business functionality). Based on the logic of application architecture, a single IOIAF application instance can serve multiple business functionality.

OIMS and IOIAF server instances are controlled as to how these instances are created and destroyed by setting ServiceBehavior's InstanceContextMode along with configuration details to be used at the service deployment/invocation time. The implementation details of IOIAF, OIMS and simulated IoT services are documented and hosted in GitHub for easy exploration and experimentation (SIVAN 2018).

3 The Business Logic Based Patterns of Integration in IOIAF Environment

Various kind of business use case served by this arrangement involves, IOIAF application (WS-consumer) periodically sending messages encoding its intention of gathering IoT data from WS-providers (IoT nodes) as Agent Communication Language (IEEE-FIPA 2018) like-message payload (as illustrated in Table 1). This message can have different level of sophistication in describing the URI of the intended providers starting from syntactic encoding (as implemented in this work) to OWL-S based semantic encoding (Sect. 3 in Venkatesan and Sridhar 2017). OIMS in turn determine choice of WS providers to connect to and the nature of the contract to execute (SIVAN 2018; Venkatesan and Sridhar 2016, 2017). Some examples of the syntactic messages originated by IOIAF server is depicted in Table 1.

The mechanism of the orchestration of IoT services described here gives rise to the possibility of realizing application layer level protocol that permit creation of complex integration applications. This capability is called as Message Based Service Integration (MBSI) protocol in this work. Section 4 provides a survey and comparison of capabilities of various IoT (application level) integration protocols including that of MBSI. Table 2 reveal the nature and technical implication of using these protocols for developing data gathering or monitoring end user applications.

A snapshot of the sample experiment carried out to gather data from an IoT network of nodes involving multiple OIMS is illustrated in Fig. 6 below.

The manner of OIMS based WS orchestration envisaged here is quite a different kind of business workflow compared to regular Process Aware Information Systems (PAIS)/business applications (Aalst 2009). Presently PAIS typically invoke providers using flow based technology (i.e. events, activities, gateways) and is entirely based on design time decisions. The proposal made here is not disruptive and only adds a new layer of features above existing standards and infrastructure. Hence it is a kind of incremental innovation or suggestion.

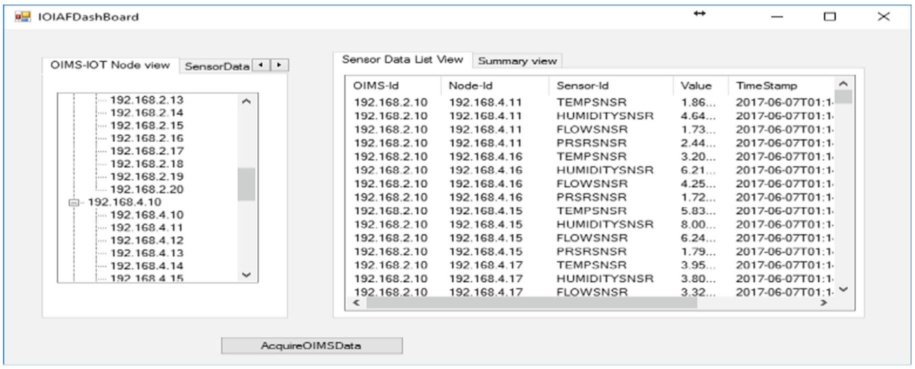
Table 2. Comparison of popular IoT protocols

Protocols vs attributes	MQTT	AMQP	COAP	JMS	DDS	REST	XMPP	MBSI
Applicable standard body	OASIS std.- Version 3.1.1	OASIS std. – version 1.0	IETF std. version-1.2	Java Enterprise edition	Open International data centric connectivity version1.4	IETF version 1.1	IETF version 1.0	Proposed
Protocol architecture type	Broker based	Broker based	Broker based	Broker based	Bus based	Bus based	Bus based	Bus & Broker – Hybrid
Message delivery type	Web based	Message Oriented & middleware	Web based and middleware based	Middleware based	Middleware based	Application based	Application based	Middleware & message oriented
Integration strategies	Publish subscribe, client broker	Publish subscribe, point	Request reply	Publish–subscribe	Peer to peer	Request–reply	Request–reply	Decoupled request-reply based
Connection type	TCP/IP	TCP/IP	UDP	TCP/IP	TCP or UDP	TCP/IP	TCP/IP	ACL + SOAP
Interoperability	Possible	Not possible	Possible	Possible	Possible	Possible	Possible	Possible
Security	TLS/SSL	TLS/SSL	DTLS	TLS/SSL	DTLS	TLS/SSL	DTLS	WS-Security
Message delivery type	Client-server	Peer–peer	Client-server	Client-server	Peer–peer	Client-server	Client-server	Point-to-MultiPoint
Address mode	Unicast	Multicast	Unicast	Unicast	Multicast	Unicast	Unicast	Multi-level Multicast
Implementation complexity	Simple	Complex	Simple	Simple	Simple	Simple	Complex	Simple
Website for more information	mqtt.org	https://www.amqp.org	http://coap.technology	Oracle Java developer site		W3C	https://www.xmpp.org	http://www.openthesis.org/document/view/603477_1.pdf
Foot print	Tiny	Tiny	Tiny	Tiny	Tiny	Tiny	Tiny	Large
Message encoding/representation technology (generic/binary/XML)	Binary encoding	Binary encoding	Binary encoding	Distributed message encoding	Binary encoding	Plain Text, also content encoding e.g. ZIP, compress, deflate etc.…	Plain text	Plain text format; XML schema compliant
Message envelop details (header size and max length)	2 and 5 chars.	4 and 20 chars. typically	8 chars.	JMS does not support the header field transmitted to non-JMS client	Variable length	8 field	5 field	Consumer to OIMS-MQ header; OIMS to IoT WS provider (as per SOAP)
Session handling support	Persistent session	Session can be supported as session modules	Sessions have a window-based flow control model	Session is single thread context	Session Initiation protocol & Session Description Protocol	Server session to maintain a persistent state for a period and allow authenticated	Server depends on the instance and presence session	Provided by OIMS middleware; based on message options;
Flow control of messages	Publisher => central broker => subscriber	Client => server	Publisher-central broker => subscriber	Client => server	Publisher => subscriber	Client => server	Publisher => central broker => subscriber	No need explicit for flow control; OIMS/underlying layers ensure it

(*continued*)

Table 2. (*continued*)

Protocols vs attributes	MQTT	AMQP	COAP	JMS	DDS	REST	XMPP	MBSI
Meta-data support in messaging (content type indicator)	Comprises of identifier (data type & value)	Comprises of representation of resource	Routing key, persistent	Comprises of element (info and pointer)	The meta data can be converted into Scratch Image	Instant and persistent	Instance and presence	Full support available
Performance when connected to 1000s of devices	Poor	Good	Poor	Poor	Poor	Poor	Good	Excellent
Security and QOS	Both	Both	Both	Security	Security	Security	Both	Both
Seamless correlation support	Publisher	Client	Publisher	Client	client	client	Client	App. level and infrastructure based correlation possible
Message ordering support	Re-sends any PUBLISH packets, in order the original PUBLISH packets were sent	All messages arrive in sequence and takes care of retransmission and assembly in correct order	AMQP divides message into header, properties, body and optional footer	JMS defines that messages sent by a session to a destination must be received in the order they were sent	Messaging is 'federated' in nature. Traffic in federation is arbitrated by a unique network-scheduler	Uses asynchronous messaging based systems fits well with the pipe line message ordering or event based message approach	Message can be addressed to another entity. Sender server uses routing for ordering and delivering messages to client	Any kind of message ordering is possible through conversation/SA identifier of the message between provider and consumer
Tool support	MQTT-spy, MQTT-inspector	Core DX DDS-spy /DDS multiprocessor	Popular Linux variants and PyPi	Any Java EE IDE	Eclipse visual modelling tool, Tuner tool	RESTful API based design tools	Kaiwa, ejabbered	All service technology compliant tools applicable



```
192.168.2.13
192.168.2.14
192.168.2.15
192.168.2.16
192.168.2.17
192.168.2.18
192.168.2.19
192.168.2.20
192.168.4.10
    192.168.4.10
    192.168.4.11
    192.168.4.12
    192.168.4.13
    192.168.4.14
    192.168.4.15
```

OIMS-Id	Node-Id	Sensor-Id	Value	TimeStamp
192.168.2.10	192.168.4.11	TEMPSNSR	1.86...	2017-06-07T01:1
192.168.2.10	192.168.4.11	HUMIDITYSNSR	4.64...	2017-06-07T01:1
192.168.2.10	192.168.4.11	FLOWSNSR	1.73...	2017-06-07T01:1
192.168.2.10	192.168.4.11	PRSRSNSR	2.44...	2017-06-07T01:1
192.168.2.10	192.168.4.16	TEMPSNSR	3.20...	2017-06-07T01:1
192.168.2.10	192.168.4.16	HUMIDITYSNSR	6.21...	2017-06-07T01:1
192.168.2.10	192.168.4.16	FLOWSNSR	4.25...	2017-06-07T01:1
192.168.2.10	192.168.4.16	PRSRSNSR	1.72...	2017-06-07T01:1
192.168.2.10	192.168.4.15	TEMPSNSR	5.83...	2017-06-07T01:1
192.168.2.10	192.168.4.15	HUMIDITYSNSR	8.00...	2017-06-07T01:1
192.168.2.10	192.168.4.15	FLOWSNSR	6.24...	2017-06-07T01:1
192.168.2.10	192.168.4.15	PRSRSNSR	1.79...	2017-06-07T01:1
192.168.2.10	192.168.4.17	TEMPSNSR	3.95...	2017-06-07T01:1
192.168.2.10	192.168.4.17	HUMIDITYSNSR	3.80...	2017-06-07T01:1
192.168.2.10	192.168.4.17	FLOWSNSR	3.32...	2017-06-07T01:1

Fig. 6. A snapshot of IOIAF dashboard gathering sensor data from IoT network

4 Related Work

Higher-level business processes can be created – that are solution to business problems - by composing web services together. Service composition standards (Dustdar and Wolfgang 2005) provide standards-based, an open approach to connect web services with reduced complexity. WS standards (Weerawarana et al. 2005) reduces time and costs, and increase overall efficiency in formulating software for businesses problems. A well-established method of composing web services to create business applications is to use WS-BPEL (WS-BPEL 2007). BPEL is a model based, flow based, XML-based, imperative WS composition language which supports WS technology stack fully. However due to static nature of the orchestration provided in this language, it is not suited for IoT integration unless extensions are implemented to the BPEL engine/language (Venkatesan and Sridhar 2017). Okouya et al. (2013) propose middleware technology for integrating applications using speech-act like messages. This work draws inspiration and hints from that work to formulate the novelty presented here. Blake and Gomaa (2005) provides an early account of agent based cross organizational workflow composition. However it lacks a rigorous specification and description of capabilities of agent oriented middleware. It also did not provide an operational description how to realize the system making it a theoretical treatise.

The hardware limitations of IoT devices make design of software application unique and challenging. Jawad et al. (2017) provides a survey of various issues involved in integrating constrained devices in agricultural field monitoring including IoT integration issues. For example regular application level protocols like HTTP, SOAP may not work if one ignores network topology, packet size limitation and data rates and frequency of packet transaction. Hence software applications should explicitly address these constraints in designing IoT application such as minimize unwanted processing, reduce network data transaction.

Most of the existing application level protocols (given in Table 2) do not anticipate or take into account future changes in network topology, application architectures and IoT software stack evolution. Protocol innovations for IoT integration related can be applied to any layers of the network such as physical layer, data link layer, network layer and application layer. *This work specifically addresses problems of integration at the application level protocol only*. New IoT specific protocols (Al-Fuqaha et al. 2015) tends to reduce data errors, avoid unwanted re-transmission, keep simple flow of data, avoid complex buffering/computation algorithms (saving on RAM/CPU cycles) to reorganize packets, increase reliability and wireless range. But one of the main issues here is difficulty of writing device integration, control and monitoring applications using these protocols. They do not offer satisfying experience for varying technical and practical reasons and offer little room for evolution and capability to coexist with newer developments. Presently bus based and broker based approaches are popular to network these devices. Some popular application level protocols include Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (COAP), Java Message Service (JMS), Data distribution Service (DDS), Representational State Transfer (REST) and Extensible Message and Presence Protocol (XMPP). Table 2 provides a survey and comparison of capabilities of various IoT application level integration protocols including the Message Based Service Integration (MBSI) protocol proposed here. Venkatesan and Sridhar (2019) argues agent metaphor based system modeling and software development result in manageable and intuitive information systems. Venkatesan and Sridhar (2018a) argues agent metaphor based information system modeling anddevelopment result in superior model driven development software environment. Hence this work embraces agent oriented approach to realize the middleware solution.

5 Conclusion

This work presented an agent oriented approach to WS application composition and business logic enactment that fuses technological sophistication of WS technologies/standards and theoretical depth of agent oriented system modelling and engineering. Due to space limitation a comparative and experimental evaluation MBSI protocol with other IoT application level protocol is not carried out here. This work highlighted only a few elementary cases of patterns of interaction between IOIAF, OIMS and IoTNode services. This work did not carry out a comparative experimental evaluation of (possibility of) realization of patterns of interaction in other IoT protocols discussed here. These deficiencies need to be addressed in an elaborate study in the near future. From the conceptual viewpoint, the pattern of interaction described here in purely syntactical. This environment can be extended to permit varieties of application integration such as semantic web technology standard based partner/provider identification (using SPARQL, for example). The results arrived here can be utilized to formulate and extend orchestration standards by incorporating WS infrastructure support for the kind of orchestration approach envisaged here. These extensions remain to be carried out in the future.

References

Aalst, W.M.P.: Process-aware information systems : design, enactment and analysis. In: Wah, B. W. (ed.) Wiley Encyclopedia of Computer Science and Engineering, pp. 2221–2233. Wiley, Chicester (2009). ISBN 978-0-471-38393-2

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: a survey on enabling technologies, protocols and applications. IEEE Commun. Surv. Tutor. (2015). https://doi.org/10.1109/COMST.2015.2444095

Blake, B.M., Gomaa, H.: Agent-oriented compositional approaches to services-based cross-organizational workflow. Decis. Support Syst. **40**(1), 31–50 (2005). https://doi.org/10.1016/j.dss.2004.04.003

Dustdar, S., Wolfgang, S.: A survey on web services composition. Int. J. Web Grid Serv. **1**(1), 1–30 (2005)

IEEE-FIPA: IEEE CS Approved Committee on Foundation for Intelligent Physical Agents (2018). https://www.computer.org/web/standards/fipa

Jawad, H.M., et al.: Energy-efficient wireless sensor networks for precision agriculture: a review. Sensors **17**, 1781 (2017). https://doi.org/10.3390/s17081781

Okouya, D., Fornara, N., Colombetti, M.: An infrastructure for the design and development of open interaction systems. In: Cossentino, M., El Fallah Seghrouchni, A., Winikoff, M. (eds.) EMAS 2013. LNCS (LNAI), vol. 8245, pp. 215–234. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45343-4_12

Sharp, J.: Microsoft Windows Communication Foundation Step by Step. Microsoft Press (2017). ISBN 978-0735623361

SIVAN: IOIAF, OIMS, IoTSvc application code base at GitHub (2018). https://github.com/sivanagent/{IOIAFDashBoard, OIMS, IOTNodesvc}

Venkatesan, D.: A novel agent-based enterprise level system development technology, Ph.D. thesis, Anna University (2018). http://www.openthesis.org/document/view/603477_1.pdf

Venkatesan, D., Sridhar, S.: Promoting Business -IT Alignment through Agent Metaphor Based Software Technology, Int. J. of Inf. Technol. Manag. (2018a). https://doi.org/10.1504/IJITM.2018.10013469

Venkatesan, D., Sridhar, S.: A rationale for the choice of enterprise architecture method and software technology in a software driven enterprise, Int. J. of Bus. Inf. Syst. (2019). https://doi.org/10.1504/IJBIS.2019.10013326

Venkatesan, D., Sridhar, S.: A novel programming framework for architecting next generation enterprise scale information systems. Inf. Syst. E-Bus. Manag. **15**(2), 489–534 (2017). https://doi.org/10.1007/s10257-016-0330-y

Venkatesan, D., Sridhar, S.: A novel method and environment for scalable web service orchestration. In: Proceedings of IEEE 12th 2016 World Congress on Services Computing (SERVICES 2016), San Francisco, USA, pp. 128–129 (2016). https://doi.org/10.1109/SERVICES.2016.27

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WSBPEL, WS-Reliable Messaging and More. Prentice Hall PTR, Upper Saddle River (2005)

WS-BPEL: OASIS Web Services Business Process Execution Language v 2.0 (2007). http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Yin, R.K.: Case Study Research: Design and Methods, 3rd edn. Sage Publication, Thousand Oaks (2003). ISBN 0-7619-2553-8

Short Paper Track

RT-OCF: A Lightweight Device-to-Device Framework for the Internet of Things

Chanhee Lee[✉], Jaehong Jo, Jongsung Lee, Daesung An, Jaehyun Cho, and Rami Jung

Samsung Electronics, Seoul 06765, Korea
{ch2102.lee,jaehong.jo,js126.lee,daesung87.an, jaehyun3.cho,rami.jung}@samsung.com

Abstract. Responding to rapid growth of Internet of Things (IoT) services and devices, many IoT platforms and frameworks are presented for successful IoT realization. Recently, Open Connectivity Foundation (OCF) which consists of a group of industry leaders emerges to create new standards for IoT platform and deliver an open source implementation and a certification program. IoTivity open-source software framework is one of the project sponsored by the OCF which has been developed to provide interoperability among heterogeneous IoT devices. It enables seamless Device-to-Device (D2D) connectivity and targets various application domains such as home and manufacturing automation, health care, and social networks. IoTivity uses, however, rather lots of memory in view of small devices which have limited hardware resources and runs Real-Time Operating System (RTOS). In this paper, we propose a light-weight IoT framework, RT-OCF that optimizes memory consumption and provides a memory tracer to prevent memory leaks. It has a layered architecture which consists of resource layer, messaging layer, and platform adaptation layer together with modules for security and utility. RT-OCF is able to run not only on Linux but also on TizenRT, an open source RTOS platform runnable on ARTIK053 board. The experiment performed on both Linux and TizenRT shows that more than 20% of peak memory is reduced when compared with IoTivity while preserving a packet latency for GET operations.

Keywords: IoT · D2D · Open source · Framework · OCF · TizenRT · Memory

1 Introduction

Responding to rapid growth of Internet of Things (IoT) services and devices, many IoT platforms and frameworks are presented for successful IoT realization [1–3]. There are three key factors which needs to be considered for the successful IoT realization [4]. At first, specifications should exist and describe correct mechanisms of major functionalities involving device connections and registrations in a secure way. At next, source code implements the specifications and needs to pass certification. Lastly, the source code is deployed to IoT devices of different manufacturers and is interoperable among the devices.

Recently, Open Connectivity Foundation (OCF) [5] which consists of a group of industry leaders, e.g. Microsoft, Intel, Samsung, Cisco, and LG emerges to create new standards for IoT platform and deliver an open source implementation and a certification program. IoTivity open-source software framework [3] is one of the project sponsored by the OCF which has been developed to provide interoperability among heterogeneous IoT devices. It also enables seamless D2D connectivity and targets various application domains such as home and manufacturing automation, health care, and social networks.

IoTivity uses, however, rather lots of memory in view of small devices which have limited hardware resources, e.g., a low-cost processor and a small memory and run a RTOS. This is because IoTivity targets various common operating systems (OS) such as Linux, Windows, and Android.

In this paper, we propose a light-weight IoT framework, RT-OCF that optimizes memory consumption together with providing a memory tracer to prevent memory leaks. It has a layered architecture which consists of resource layer, messaging layer, and platform adaptation layer together with modules for security and utility. To reduce memory consumption, er-coap [6] is ported into the proposed framework. RT-OCF is not only able to run on Linux but also be compatible with TizenRT [7], an open source RTOS platform runnable on ARTIK053 board [8].

The main contributions of RT-OCF can be summarized as follows.

- RT-OCF enables easy use of OCF-based D2D communications on both a real ARTIK board running TizenRT and a Linux machine as open source software.
- Practical memory optimization techniques are introduced for the OCF specification implementation while preserving a packet latency for basic operations.
- The code qualities of both RT-OCF and IoTivity are accessed and analyzed in a quantitative manner using a static source code analyzer, SonarQube [9].

The rest of the paper is organized as follows. In Sect. 2, OCF specification on which our framework is based and TizenRT where RT-OCF is implemented are explained. Then representative D2D frameworks are introduced and compared in Sect. 3. Section 4 describes the details of each layer in RT-OCF and experimental setup and results regarding memory consumption are followed in Sect. 5. At last, the paper is concluded with future work in Sect. 6.

2 Background

In this section, OCF specification on which the proposed framework is based and TizenRT, a target RTOS on which the framework runs are described briefly.

2.1 OCF Specification

The latest version of OCF specification is 1.3 which consists of 6 categories; Core Framework, Resource Type, Device, Security, Bridging, and Wi-Fi Easy Setup. In Core Framework specification, the OCF core architecture based on the resource-oriented REpresentational State Transfer (REST) [10] architectural style is specified. It mainly describes functional interactions such as CRUDN, messaging, and discovery as well as core features related to resource models for all OCF resources and their combinations which can be exposed by OCF devices. JavaScript Object Notation (JSON) is used for payload definitions and RESTful API Modeling Language (RAML) [11] is used for the representation for the APIs exposed to the outside of the OCF device by the OCF resources. In Device specification, a set of Device Types for use is defined. Device Types can be either mandatory to be implemented or optional and exposed also. In Security specification, device identity, authentication, and provisioning are defined for the establishment of network credentials and access controls of OCF resources. Finally, Bridging and Wi-Fi Setup specification describe translation functionality such as resource discovery and functional extensions for the capabilities to meet the requirements of Wi-Fi Easy Setup, respectively.

2.2 TizenRT

TizenRT is an open source platform based on a RTOS, called NuttX [12]. The goal of TizenRT is to extend the Tizen platform device coverage to low-end devices typically equipped with Cortex-M/R processors with MPU, less than 2 MB RAM, and less than 16 MB Flash.

It overcomes several limitations of typical RTOS targeting IoT devices. Most of existing RTOS cannot load additional modules at runtime, and support only specific libraries that prevent application developers from using and those from being spread out to various types of IoT devices. To tackle these limitations, TizenRT adopts Linux-style development environments including POSIX API, BSD Socket API, Shell, and Kconfig build configuration. This helps Linux developers build their own business logics easily on top of TizenRT. In addition, TizenRT will adopt the lightweight JavaScript environment, consisting of JerryScript [13] and IoT.js [14].

3 Related Work

There are various IoT frameworks as well as CoAP libraries, the major communication protocol in IoT frameworks. Among them, we summarize and compare two D2D IoT frameworks, i.e., IzoT [15], Thingsquare [16] which are similar to RT-OCF.

IzoT considers three main classes of IoT applications; consumer IoT, two classes of Industrial IoT based on application monitoring and Machine-to-Machine (M2M) communications. To support M2M communication-centric applications, IzoT stack consists of high level protocol services that can run on top of any IPv4 or IPv6 UDP socket interface. The stack supports end-to-end acknowledgements and responses for rapid detection of packet failures as well as fully distributed connections among network nodes to avoid single point of failure. It also provides a duplicate node detection by assigning transaction IDs with a unique mechanism and a priority messaging to allow the easy propagation of emergency messages such as notifications of node failures. Though IzoT provides various useful features for M2M communications as mentioned above, its proprietary stack without CoAP makes it difficult to be adopted widely to low-end IoT devices.

Thingsquare supports M2M communications based on the stack of Contiki OS [17], cloud-based device governance and boot-strapping. It is composed of three parts; a device firmware, a backend stack, and user frontends. A device firmware resides in a wireless chip of a product such as an IoT lamp and a backend stack such as Thingsquare cloud provides database and message control between the wireless device and user frontends. Thingsquare, however, is only distributed as binaries which also hinder the spread itself.

4 Design and Implementation

The software architecture of the proposed framework is shown in Fig. 1. All modules in the architecture are implemented in C. For each following subsection, the detailed design and key features together with memory optimization techniques are described.

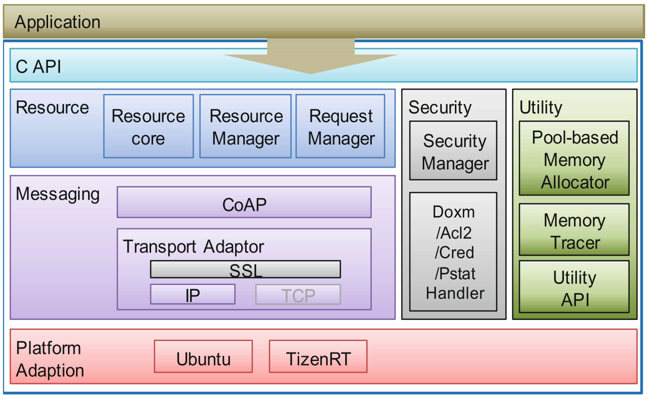


Fig. 1. RT-OCF architecture.

4.1 Architecture

RT-OCF is basically a layered architecture except functions for security and utility which are called in a resource and a messaging layer.

4.2 Resource Layer

A resource layer implements core OCF resources in the OCF specification. For the core resources which are classified with four types, i.e., device, introspection, platform and resource, APIs are provided to application users for the resource creations. In addition, there is a resource manager that operates a queue to process inner requests from other layers or request packets from the outside of the device on which RT-OCF is based. The requests are CRUDN operations mentioned in Sect. 2.1 and in general a callback is registered for each operation in a user application. There is also a request manager to organize a request as a packet to be sent from a controller to a controlee. Before sending a request to a controlee, a controller needs to discover at least one controlee in its local network using Discovery APIs. For example, for an IoT lamp, the lamp is represented as an OCF device and the brightness of the lamp can be one of OCF resources in the OCF device. A user, the owner of the lamp, can increase or decrease the brightness of the lamp through a mechanism such as a GUI panel provided by the controller of the user after discovering and connecting to a controlee. Then the request to configure the brightness value from the controller is sent to the request manager of the discovered controlee, e.g., the lamp, followed by the value modification by the resource manager of the controlee. In addition, when a user wants to get immediate notifications about the status of interesting resources in controlees, e.g., power on/off status or power consumption levels, Observe APIs can be used to set up observers to the controlees and the notifications are sent either in the right moment that the status of the resources are changed or in a periodic manner by Notify APIs provided in this layer.

To reduce run-time memory consumption in this layer, the resource representations based on TinyCBOR [18] are simplified by omitting several wrappings and complex data structures, in contrast to the representation in IoTivity. Instead, we provide Map and Array containers to enable users to build hierarchical resources directly so that internal transformations of the OCF resources required in IoTivity can be skipped.

4.3 Messaging Layer

A message layer is divided more precisely into two layers. The upper layer is implemented with external open sources, i.e., er-coap from Contiki OS. Er-coap is known as the smallest CoAP API until now. We compared er-coap with several CoAP libraries such as libcoap, microcoap, lobaro-coap and some other open source CoAP libraries and chose er-coap with consideration for memory consumption to CRUDN operations, implementation language, and the latest update date. The comparison results are omitted due to page limits. By adopting er-coap much lighter than libcoap used in IoTivity, the run-time memory consumption can also be reduced. And the other layer stands for a transport adaptor and includes Secure Sockets Layer (SSL) and User-defined Protocol (UDP), Transmission Control Protocol (TCP) and Internet Protocol (IP). Various functions such as unicast/multi-cast, normal/secure socket, normal/secure port, and block-wise transfer are implemented in this layer.

4.4 Security Functions

In a target IoT device, all information can be encrypted and then decrypted into one of four types; doxm, acl2, cred, and pstat. Those types cover general information for device identification and description, access authority to each resource, authentication certificate, and cloud provisioning, respectively. A security manager handles initiation and invocation of information transformation to payload of a packet to be transferred and stores the information into a persistent storage in form of encrypted files.

To reduce run-time memory consumption paid for security, we store each type into a separate file and load only the encrypted files of necessary types at run-time instead of storing the whole encrypted information into one large file as IoTivity does. Though this may slightly increase the time complexity at run-time, it can prevent the security manager from loading frequently one large security file which causes the unnecessary accesses to the whole information even in the case that a specific type of the information is needed.

4.5 Utility Functions

The key modules of utility functions are a pool-based memory management module to limit run-time memory consumption and a memory tracer to detect memory leaks. To analyze memory leaks, a specific memory allocation function and a tracer are provided and to log all dynamic memory allocations. In addition, basic data structures such as list and queue as well as string, random, timer, and thread functions are provided as necessary to all the other layers and modules.

5 Experiments

In this section, the performance metrics, i.e., packet latency and run-time memory consumption and the code quality of the proposed framework are evaluated.

5.1 Setup

For fair comparisons with IoTivity v1.3.0, the performance metrics are measured upon both TizenRT and Linux. A network model used when measuring the performance metric on TizenRT are shown in Fig. 2(a). The model consists of two nodes, i.e., a controller and a controlee. As a controller, a simple android application which follows OCF 1.0 specification and can send a group of GET messages to the controlee is developed and used for the experiments. A Galaxy S6 phone is used to execute the application. RT-OCF runs as a controlee upon TizenRT OS flashed into an ARTIK 053 board. Note that the ARTIK 053 board is changed into a desktop PC running Ubuntu 14.04 in case Linux is used rather than TizenRT.

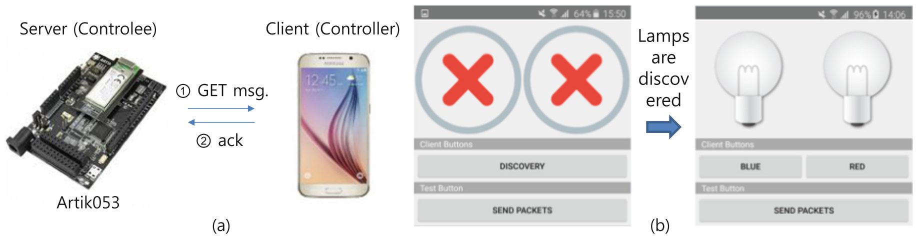


Fig. 2. (a) Network model for performance tests (b) Android application GUI for a controller.

The relevant user interface (UI) of the application for the test with GET operations is shown in Fig. 2(b). At first, a discovery to find the resources on the controlee are performed in the controller after the android application is executed. Once the discovery is finished successfully, GET messages are sent in succession from the controller to the controlee after each ack message of the former packet is received.

5.2 Results

To measure and compare memory consumption, peak memory during a GET operation at run-time is used. It is calculated as the sum of stack and heap memory usage measured by valgrind [19] and heapinfo for Linux and TizenRT, respectively. Heapinfo is an internal memory analysis tool in TizenRT. The results of peak memory normalized with that of RT-OCF when the memory tracer is disabled are shown in Fig. 3. For both Linux and TizenRT, IoTivity consumes 416.7% and 213.8% more memory than RT-OCF, respectively. About 58.5 Kbytes memory reductions of RT-OCF on TizenRT can be achieved by the adaptation of the lighter CoAP library, the simplification to data structures of OCF resources, the reduction of the number of Send/Receive threads in the messaging layer, and the separation of the encrypted information files for each type. The peak memory when the tracer is enabled during the test is also measured to validate the availability of the memory tracer. The tracer increases the memory only by 13.8%, i.e., 7.1Kbytes on TizenRT.

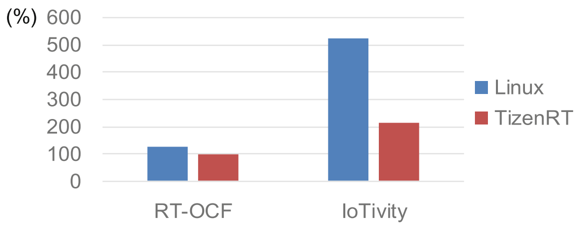


Fig. 3. Peak memory consumption of RT-OCF and IoTivity on Linux and TizenRT.

On the other hand, we also measure an average packet latency during the peak memory measurements to check whether performance degradation is serious or not due to memory optimization techniques. The packet latency is measured with the most representative operation among CRUDN operations, i.e., GET. The results were about 77 ms and 66 ms for RT-OCF and IoTivity on TizenRT, respectively. For Linux, the latencies of both RT-OCF and IoTivity for a GET operation are almost similar as about 32 ms meaning that the performance degradation in RT-OCF is insignificant.

Lastly, the code quality of our open source framework is evaluated using SonarQube. It can estimate code quality metrics such as Lines of Code (LoC), code complexity, duplication lines, and maintainability which mean the sum of total lines of source files, degree of the branch usages such as if, switch and while, duplicated source codes except function calls, and modification efforts to follow standards such as coding rules. The results for both RT-OCF and IoTivity are shown in Table 1. For exact comparison, only

D2D parts of source codes in IoTivity except source codes related to cloud connections are identified and used for this evaluation. The results show that all quality factors are greatly improved than those of IoTivity. Especially for duplicated lines, RT-OCF overwhelms IoTivity. Through code reviews, it is found that the same structures and functions are declared as static and used repetitively in IoTivity. Note that the code quality factors are not proportional to LoC. The LoCs for both frameworks are represented as references for the scale of the frameworks.

Table 1. Comparison with IoTivity framework in code quality

	LoC	Complexity	Duplication lines	Maintainability
RT-OCF	10,887	24.4	68 (0.5%)	31d (1549)
IoTivity	*48,068*	71.5	841 (1.3%)	58d (2355)

6 Conclusion

In this paper, an OCF-based lightweight open source D2D framework, called RT-OCF is proposed. The key contribution of RT-OCF is to enable D2D communication based on an open source RTOS Platform, called TizenRT minimizing run-time memory consumption which is critical to resource-constrained IoT devices. The performance and code quality are also evaluated upon a real IoT device, i.e., ARTIK 053. The results show that the proposed framework reduces the memory consumption efficiently as well as achieves better code quality than the case of IoTivity. The future work is to improve the memory management method using buddy system to minimize memory fragmentation and measure additional performance metrics such as power consumption with more complex usage scenarios. And functionalities such as onboarding when an IoT device is in out-of-box state, TCP communication, Secure Socket Layer (SSL) based on certificate, and provisioning to clouds will be expanded.

References

1. Perera, C., Jayaraman, P.P., Zaslavsky, A., Georgakopoulos, D., Christen, P.: Mosden: An internet of things middleware for resource constrained mobile devices. In: 47th HICSS, pp. 1053–1062. IEEE, Waikoloa (2014)
2. Razzaque, M.A., Milojevic-Jevric, M., Palade, A., Clarke, S.: Middleware for internet of things: a survey. IEEE Internet Things J. **3**(1), 70–95 (2016)
3. Dang, T.-B., Tran, M.-H., Le, D.-T., Choo, H.: On evaluating IoTivity cloud platform. In: Gervasi, O., Murgante, B., Misra, S., Borruso, G., Torre, Carmelo M., Rocha, Ana Maria A.C., Taniar, D., Apduhan, Bernady O., Stankova, E., Cuzzocrea, A. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 137–147. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_10
4. Derhamy, H., Eliassan, J., Delsing, J., Priller, P.: A survey of commercial frameworks for the Internet of Things. In: ETFA 2015. IEEE, Luxembourg (2015)
5. Park, S.: OCF: A new open IoT consortium. In: 31st International Conference on Advanced Information Networking and Applications Workshops, pp. 356–359. IEEE, Taipei (2017)

6. Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power CoAP for Contiki. In: 8th ICMASS. IEEE, Valencia (2011)
7. TizenRT RTOS. https://source.tizen.org/documentation/tizen-rt. Accessed 30 Mar 2018
8. ARTIK 053. https://www.artik.io/modules/artik-05x/. Accessed 30 Mar 2018
9. Ann Campbell, G., Papapetrou, P.P.: SonarQube in Action, 1st edn. Manning, Greenwich (2013)
10. Feng, X., Shen, J., Fan, Y.: REST: an alternative to RPC for web services architecture. In: 1st ICFIN. IEEE, China (2009)
11. RAML. https://raml.org/. Accessed 30 Mar 2018
12. NuttX. http://www.nuttx.org/. Accessed 30 Mar 2018
13. JerryScript. http://jerryscript.net/. Accessed 30 Mar 2018
14. IoT.js. http://iotjs.net/. Accessed 30 Mar 2018
15. IzoT. https://www.echelon.com/izot-platform. Accessed 30 Mar 2018
16. Thingsquare. https://www.thingsquare.com/. Accessed 30 Mar 2018
17. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: 29th ICLCN, IEEE, Tampa (2004)
18. TinyCBOR. https://github.com/intel/tinycbor. Accessed 30 Mar 2018
19. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. In: 28th PLDI, pp. 89–100. ACM, San Diego (2007)

Author Index