Selma Boumerdassi · Samia Bouzefrane
Éric Renault (Eds.)

LNCS 9395

Mobile, Secure, and Programmable Networking

First International Conference, MSPN 2015
Paris, France, June 15–17, 2015
Selected Papers

Springer

Lecture Notes in Computer Science 9395

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, Lancaster, UK
Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
University of Surrey, Guildford, UK
Jon M. Kleinberg
Cornell University, Ithaca, NY, USA
Friedemann Mattern
ETH Zurich, Zürich, Switzerland
John C. Mitchell
Stanford University, Stanford, CA, USA
Moni Naor
Weizmann Institute of Science, Rehovot, Israel
C. Pandu Rangan
Indian Institute of Technology, Madras, India
Bernhard Steffen
TU Dortmund University, Dortmund, Germany
Demetri Terzopoulos
University of California, Los Angeles, CA, USA
Doug Tygar
University of California, Berkeley, CA, USA
Gerhard Weikum
Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at http://www.springer.com/series/7411

Selma Boumerdassi · Samia Bouzefrane
Éric Renault (Eds.)

Mobile, Secure, and Programmable Networking

First International Conference, MSPN 2015
Paris, France, June 15–17, 2015
Selected Papers

Springer

Editors
Selma Boumerdassi
CNAM/CEDRIC
Paris
France

Samia Bouzefrane
CNAM/CEDRIC
Paris
France

Éric Renault
Institut Mines-Télécom -Télécom SudParis
Evry
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-25743-3 ISBN 978-3-319-25744-0 (eBook)
DOI 10.1007/978-3-319-25744-0

Library of Congress Control Number: 2015952766

LNCS Sublibrary: SL5 – Computer Communication Networks and Telecommunications

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2015
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media
(www.springer.com)

Preface

The rapid deployment of new infrastructures based on network virtualization and cloud computing triggers new applications and services that in turn generate new constraints such as security and/or mobility. The International Conference on Mobile, Secure and Programmable Networking (MSPN) aimed at providing a top forum for researchers and practitioners to present and discuss new trends in networking infrastructures, security, services, and applications while focusing on virtualization and cloud computing for networks, network programming, software-defined networks (SDN) and their security. In 2015, MSPN was hosted by CNAM Paris, which is one of the oldest teaching centers in Paris.

The call for papers resulted in a total of 36 submissions from around the world. Every submission was assigned to at least three members of the Program Committee for review. The Program Committee accepted 14 papers, which are from: Algeria, China, Colombia, Denmark, France, Germany, Greece, India, Ireland, Russia, Spain, and Vietnam. One intriguing keynote from Professor Pierre Paradinas complemented the technical program.

We would like to thank all who contributed to the success of this conference, in particular the members of the Program Committee (and the additional reviewers) for carefully reviewing the contributions and selecting a high-quality program. Our special thanks go to the members of the Organizing Committee for their great help. We would like to especially thank Habiba Chelah, Lamia Essalhi, and Lynda Saad for taking care of the local arrangements and many other aspects in the organization of the conference.

We hope that all participants enjoyed this successful conference, made a lot of new contacts, engaged in fruitful discussions, and had a pleasant stay in Paris, France.

June 2015

Selma Boumerdassi
Samia Bouzefrane
Éric Renault

Organization

MSPN 2015 was organized by the CEDRIC laboratory of CNAM Paris and the Wireless Networks and Multimedia Services Department of Télécom Sud-Paris (a member of Institut Mines-Télécom) in cooperation with IFIP Working Group 11.2 on Pervasive Systems Security.

General Chairs

Selma Boumerdassi	CNAM, France
Samia Bouzefrane	CNAM, France
Éric Renault	Institut Mines-Télécom – Télécom SudParis, France

Steering Committee

Abdella Battou	NIST, USA
Pierre Paradinas	CNAM, France
Omar Charkaoui	UQAM, Canada
Damien Sauveron	University of Limoges, France

Publicity Chair

Ruggero Donida Labati	Università degli Studi di Milano, Italy

Organizing Committee

Habiba Chelah	CNAM, France
Lamia Essalhi	CNAM, France
Jean-Marc Farinone	CNAM, France
Lynda Saad	CNAM, France
Thinh Le Vinh	CNAM, France

Technical Program Committee

Claudio A. Ardagna	Università degli Studi di Milano, Italy
Ioannis Askoxylakis	FORTH-ICS, Greece
Chakib Bekara	University of Tlemcen, Algeria
André-Luc Beylot	ENSEEIHT, France
Weiwei Chen	Google Inc., USA
Mauro Conti	University of Padua, Italy
Mehammed Daoui	University Mouloud Mammeri of Tizi-Ouzou, Algeria
Yuhui Deng	Jinan University, China
Wassim Drira	Qatar Mobility Innovations Center, Qatar
Hamamache Kheddouci	Université Claude Bernard Lyon 1, France

José María de Fuentes	Carlos III University of Madrid, Spain
Viet Hai Ha	Hue University, Vietnam
M-Tahar Kechadi	University College Dublin, Ireland
Karim Khireddine	Université de Béjaia, Algeria
Andrea Lanzi	University of Milan, Italy
Sjouke Mauw	University of Luxembourg, Luxembourg
Alessio Merlo	University of Genoa, Italy
Ruben Milocco	Universidad Nacional del Comahue, Argentina
Pascale Minet	Inria, France
Paul Muhlethaler	Inria, France
Henrich C. Pöhls	Institute of IT Security and Security Law at the University of Passau, Germany
Christina Pöpper	Ruhr University Bochum, Germany
Laurent Roullet	Alcatel Lucent Bell Labs Villarceaux
Shiju Sathyadevan	Amrita University, Kerala, India
Ahmed Soua	NIST, USA
Xuan-Tu Tran	Vietnam National University, Vietnam
Chao Wang	Arizona State University, USA
Qian Wang	Wuhan University, China
Muhammad Younas	Oxford Brookes University, UK
Stefano Zanero	Politecnico di Milano, Italy
Weishan Zhang	University of Petroleum, China
Rong Zheng	MacMaster University, Canada

Additional Reviewer

Aravinthan Gopalasingham	Alcatel Lucent Bell Labs Villarceaux

Sponsoring Institutions

Conservatoire National des Arts et Métiers, Paris, France
Institut Mines-Télécom – Télécom SudParis, Évry, France

Contents

Adaptive and Flexible Virtual Honeynet

Wenjun Fan[1(✉)], David Fernández[1], and Zhihui Du[2]

[1] Departamento de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain
efan@dit.upm.es

[2] Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract. Honeypots have been largely employed to help securing computer systems and capture malicious activities. At present, virtual honeynets -network scenarios made of virtual honeypots- are frequently used to investigate the adversary's behaviour. The static deploying scheme used traditionally, in which the configuration of the honeynet is determined by security experts beforehand, lacks the capability of dynamically adapting its configuration after deployment. In this paper, a new adaptive and flexible virtual honeynet management system is proposed that dynamically creates, configures and deploys both low-interaction and high-interaction honeypots, emulating multiple operating systems. The results and measurements of the experiments carried out illustrate that new virtual honeynet system is more capable than previous virtual honeynet architectures.

Keywords: Dynamic honeynet · Virtual honeynet · Honeynet configuration

1 Introduction

A honeypot is an information system resource whose value lies in its unauthorized or illicit use [1]. Nowadays, honeypots are widely used in a variety of different situations to address system and network threats, as preventing the production systems from being attacked and other security goals.

First of all, honeypots can be used to detect unknown attacks. For instance, they can help solving the false negative problem of network intrusion detection systems (NIDS), when they cannot identify the signatures of the new and unknown attacks. All of the traffic sent to a honeypot is almost certainly unauthorized traffic, meaning no false positives. Honeypots can identify the suspicious activities by monitoring those network packets that could have been previously marked as non-suspicious by a normal NIDS. Thus, honeypots are able to assist intrusion detection system to reduce the number of false negatives and help in detecting potential attacks.

Secondly, honeypots can also be used to study the adversary's behaviour, discovering the attacking tactics and skills. The case in point is when the intrusion response system (IRS) always diverts the intrusion traffic to honeypots that are located on a virtualization platform. The IRS honeypots can be used to collect and analyse the activities

© Springer International Publishing Switzerland 2015

DOI: 10.1007/978-3-319-25744-0_1

of the community of intruders, capture their keystrokes and attack conversations. Honeypots are an important part of the infrastructure of an IRS because they enable the security expert to react to the attack.

Therefore, honeypot systems are useful and important to secure production systems. However, the usefulness of one single honeypot is limited. In order to provide a better protection, honeynets, defined as a network of honeypots [13], were created. A honeynet can include various honeypots just like a "zoo" has many animals. For example, a honeynet can consist of several low-interaction and high-interaction honeypots as a hybrid system to get a good balance among scalability, fidelity and performance. A honeynet can also be used to clone a target network for security research purposes. Additionally, a honeynet can receive the redirected intrusion traffic to investigate the adversary's behaviour.

Although honeynets have many advantages, deploying a honeynet is not an easy task. On the one hand, it is difficult achieve scalability in honeynet deployment when using physical honeypots, due to physical machine resources cost. On the contrary, the use of virtualized honeypots is a safety and effective way to deploy honeynets. Thus, virtualization technologies are very efficient alternative to deploy honeynets, as one physical machine can run multiple virtual machines simultaneously. Therefore, using virtual machines to run honeypots is cost-efficient and can simplify their management. On the other hand, honeynet configuration is another challenging task. Traditionally, security experts determine the configuration of the honeynet beforehand and reconfigure it manually when the honeynet needs to be redeployed. This traditional static deploying scheme has several shortcomings. First of all, manually configuring a honeynet is generally a complex and costly task. Secondly, a static honeynet deploying scheme is not able to react to an intrusion event. Thirdly, it is not flexible enough to adapt the change of the target network in real time. Thus, it is meaningful to conquer the challenge of dynamic configuration and deploy the honeypots by using virtualization technology.

Although the well-known Honeyd tool [2] can dynamically reconfigure the emulated virtual honeypots and services, it only provides low-interaction honeypots. The availability of dynamic configuration for high-interaction honeypots depends on the ability of the virtualization platform where honeynets are deployed. However, there is not any ready-made virtualized tool that can provide dynamic configuration for high-interaction honeypots deployment. Therefore, in this paper, a novel adaptable virtual honeynet architecture is proposed. This easy to manage adaptable virtual honeynet management tool allows the dynamic reconfiguration of the honeynets deployed and it is also very flexible because it allows deploying multiple honeypot types and even hybrid systems. The tool is designed to allow the easy management of virtual honeynet, hiding the user all the underlying virtualization complex details.

The organization of this paper is as follows: in Sect. 2 related work is described; in Sect. 3 a honeynet are defined and described in detail; in Sect. 4, the new virtual honeynet architecture is proposed; in Sect. 5 some experimental results and measurements are demonstrated; finally, in Sect. 6 some conclusions are presented and some future work is suggested.

2 Related Work

Honeyd [2] provides a virtual honeypots framework, which can simulate multiple honeypots simultaneously following certain network topology. However, it has several shortcomings. First of all, it is a software solution that only focuses on low-interaction honeypots simulation. Secondly, though Honeyd can dynamically reconfigure every template, it cannot change the simulated route configuration, which must be set beforehand. Third, every template only can set one Ethernet with MAC address, but it can bind multiple IPv4 addresses. Thus, Honeyd has its own original limitation. While, because of its lightweight and distributed appearance as well as its dynamic feature, Honeyd still has many applications. For instance, it can be used to clone the target network for research purposes [3]. It also can be employed to study the algorithms for hiding honeypots [4, 5].

With the generalization of the use of virtualization technologies, more and more virtual machine based solutions were proposed to create honeypots. For instance, User-Mode Linux (UML) as the virtualization engine was used in the virtual honeynet architecture [6] to mimic Gen II honeynets. In this case, the hosting machine acts as Honeywall to monitor the virtual honeypots running in the hosting machine. The built-in tty logging mechanism enables the keystroke sent to hosting machine to be silently captured. Hence, using UML virtualization technology, the proposed honeynet architecture is much more portable, easier to setup and more cost effective. The author proposed an alternative approach to deploy honeynet based virtualization technology, which is very inspired for other system administrators. However, this virtual honeynet architecture did not provide the dynamic configuration capability, which is a limitation for meeting the requirement of current honeynet research.

Another similar case in point is the use of VNUML (Virtual Network User Mode Linux) in virtual honeynet deployment [7]. The authors of this work found UML (User-Mode Linux) as a powerful but complex tool: it is generally difficult to manage medium-to-big size scenarios by hand. Thus, they devised a high-level description of the virtual honeynet and build it without dealing with virtualization low-level complex details. VNUML had two components proposed: a simple descriptive XML-based language that can specify a honeynet scenario to be simulated; and an application that interprets the honeynet description and generates and manages the honeynet scenario inside the hosting machine. The authors provided a case using VNUML to mimic a GenII honeynet. However, this work still did not provided the capability of dynamic reconfiguration of the honeynets deployed. The user has to describe the honeynet scenario beforehand manually by using the XML-based language. Very similar work to VNUML at first sight is NoSE (Network Simulation Environment) [14]. VNUML still focus on using UML to build honeynet and there seems to be no effort going beyond UML, while NoSE integrates different virtual machine emulators such as Xen, UML, and QEMU, hence it can support to emulate various operating systems. But the honeynets created by NoSE are still static.

Other virtualization technologies like VMware also can provide the ability to create a specialized network of hosts on a single physical machine. For example, the authors of the work [8] shared their experiences with a Generation III Virtual Honeynet deployment by using VMware server. But the author also did not devise the capability of dynamic configuration for the VMware server. Compared with previous virtual honeynet architectures, this work did not place the Honeywall on the host, but installed it on a single virtual machine. The data capture tool namely Sebek [9] was used to capture the keystroke in virtual honeypots.

Most recently, some new virtualization technologies were proposed to facilitate more capable honeynet architectures. For instance, the decoys based on KVM (Kernel-based Virtual Machine) hypervisor [21] can provide low-level surveillance from outside the guest OS, which can keep the system activity monitor stealthy and the intruder has no way to bypass that surveillance. Besides, LXC (Linux Containers) virtualization provides a lightweight alternative to hypervisor-based virtualization [22]. LXC can create multiple isolated Linux user-space instances by partitioning the resource of the host. Thus, the startup of a LXC based virtual machine is much quicker than that based on KVM, but LXC can only be used to emulate Linux over Linux, but not other operating systems.

3 Honeynet Description

3.1 Basic Concepts

(1) Honeypot. In general, any computer system with no authorized value can act as a honeypot. Additionally, instead of a computer system, the definition implies that a honeypot can also be a digital entity. For example, in Cliff Stoll's book "The Cuckoo's Egg" [10], we learn how he deploys digital files to track and monitor a German hacker. Honeytoken [11] is the terminology for this kind of digital entity acting as honeypot. In summary, a honeypot can be either a computer system or more generally a digital entity.

A physical machine running as a honeypot is named a *physical honeypot*, while a virtual machine running as a honeypot is named a *virtual honeypot*. The main concept we must keep in mind is that a virtual honeypot is simulated or emulated inside another machine that responds to network traffic sent to the virtual honeypot [12]. In addition, the honeypots represented by software solutions are considered virtual honeypots too. Hence, for example, a honeypot emulated by a software solution running on a physical machine is a virtual honeypot instead of physical honeypot.

Honeypots can be classified into low-interaction honeypots and high-interaction honeypots. High-interaction honeypots can provide unlimited functions and has no difference with the conventional information system resource. Thus, high-interaction honeypots can be computer system or honeytoken. On the contrary, low-interaction honeypots are always emulated by honeypot software solutions that only provide a limited or even minimum set of functions.

(2) Honeynet. In a sense, a honeynet is an extension of the honeypot concept. The narrow honeynet term refers to Gen II honeynet that is an interaction type of honeypot solution [13], while the generalized honeynet term means a network of honeypots. In this paper, the honeynet is not limited to its narrow definition. Our proposed definition is that a honeynet is a network of honeypots following certain network topology.

A physical honeynet is made of honeypots running in separate physical machines. A virtual honeynet is a honeynet made of virtual honeypots running over one or more physical machines. By using virtualized tools, all the honeypots are virtually housed in one or more machines, but they still appear to the attacker like being different separate machines.

Some honeypot software solutions have the capacity of generating virtual honeynet, i.e. Honeyd. However, the honeynets created by Honeyd only consists of low-interaction honeypots. High-interaction honeypots running on virtual machines also can form virtual honeynet.

3.2 Honeynet Functionalities

Data Control – Data Control functionality is aimed to mitigate the risk that the adversary uses the compromised honeypot to attack other non-Honeynet systems, such as any system on the Internet. The outbound attack must be controlled in order to protect the non-Honeynet systems, but, at the same time, we have to minimize the attacker's or malicious code chance of detecting it. Thus, the challenge of data control is how to set the threshold of outbound activity. The more you allow the attacker to do, the more you can learn. However, the more you allow the attacker to do, the more harm they can potentially cause. It is a complex task to hunt the balance between how much intrusive data you want to get and how much baleful activity you want to restrict.

Data Capture – The purpose of Data Capture is to log all of the attacker's activity for later investigation. Three critical layers of Data Capture were identified: firewall logs (inbound and outbound connections), network traffic (every packet and its payload as it enters or leaves the honeynet), system activity (attacker's keystroke, system call, modified files, etc.). The more data and the higher quality of the data the honeynet can capture, the better the honeynet is.

Data Collection – Data Collection requirement proposes the secure means of centrally collecting all of the captured data from distributed honeynets. Due to the fact that honeypots are themselves insecure systems, the captured data must be centralized in an external secure system. On the other hand, if the data were distributed stored, it is not an easy way to manage. Thus, Data Collection also provides an easier management of the captured data.

Another requirement is that honeynets should be stealthy. In other words, the Data Control, Data Capture and Data Collection should be hidden or camouflaged to avoid the adversary to be aware of these honeynet activities. The adversary has many ways to detect if he is in a honeynet, for example, by detecting whether he is in an environment set up to record his activity, in which case, the adversary will erase all his tracks and break the connection with the honeynet and then no data would be recorded.

3.3 Honeynet Architecture

(1) Physical Honeynet Architecture. The physical honeynet architecture proposed by the Honeynet Project [13] has evolved across 3 generations.

Generation I – Gen I Honeynet was developed in 1999 by the Honeynet Project. Figure 1 shows a graphical representation of Generation I Honeynet architecture. As can be seen, a firewall separates the network into three different parts: Internet, Honeynet and Administrative Network. This architecture is made up of several components, which are used to facilitate the main honeynet functionality.

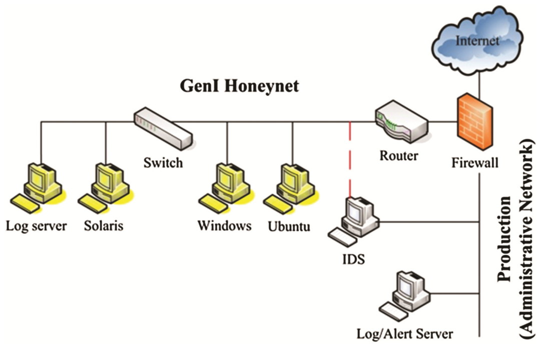


Fig. 1. Gen I honeynet architecture

The firewall keeps the track of any connection that has been made between the honeynet and the Internet. The firewall can block the outbound connections once a defined limit has been reached for the goal of Data Control. On the other hand, the firewall also logs all connections to and from the honeynet for the goal of Data Capture.

The router is located between the firewall and the honeynet. It hides the firewall from the attacker and provides the attacker a more realistic network with a production router in order to keep the attacker from becoming suspicious. Besides, the router also works for Data Control, for example, it only allows packets that have the source IP address of the honeynet to leave in order to protect against attacks such as spoofing, DoS, and ICMP based attacks.

The IDS connects to the honeynet via a physical switch. Thus, the IDS can use a "port monitoring" port enabling it to record all network activity for the goal of Data Capture. In Gen I Honeynet architecture, the IDS is signature based. When a packet matches a signature, an alert with detailed information about the connection will be provided by the IDS though any traffic on a honeynet is considered suspicious.

Finally the attacker reaches the honeynet. The honeypots capture all system activity and the remote log server can provide the centralized Data Collection. If an advanced attacker detects the syslog and even compromises the remote log server, we still have the IDS that act as a backup remote log system.

There are several limitations of Gen I Honeynet. First of all, if the limit of outbound connections is reached and all of the attacker's outbound activity is blocked, the attacker will suspect he is in a honeynet environment. Secondly, the firewall is easy to be identified since all traffic that passes through the firewall has TTL (time to live) decrement. Thirdly, the data such as keystroke and user activity are captured at the network level. IDS such as Snort can capture protocols such as FTP, Telnet or HTTP, which are plaintext. However, the attacker can use encryption technology such as protocol SSH to transfer the data. Thus, it will be failed if we monitor the attacker's connection to capture keystroke and user activity. Last but not least, the deployment is also limited, the Gen I Honeynet must be deployed on an isolated network otherwise the non-honeynet systems will be dangerous.

Generation II – In 2001, Honeynet Project released a honeywall, called eeyore, which allowed for Gen II Honeynet architecture and improved both Data Capture and Data Control capabilities over Gen I Honeynet architecture. Figure 2 illustrates the Generation II Honeynet architecture.

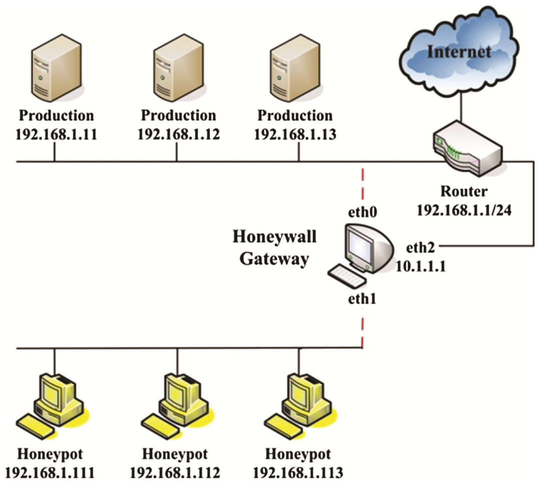


Fig. 2. Gen II honeynet architecture

Honeywall is traditionally a layer 2 bridging device with three interfaces. Two of them (eth0 and eth1) are used to segregate the honeypots traffic from the production network. They are bridged interfaces with no IP stack. The third interface (eth2, which is optional) has an IP stack and is used for remote administration. The honeywall combines the functionality of both the IDS and the firewall in a single system, which can perform attack control and network activity logging. The deployment of Gen II Honeynet is much easier than Gen I Honeynet. There are several advantages of using the honeywall.

Firstly, the honeywall is implemented as a transparent bridge to the attacker, meaning the device should be invisible to anyone interacting with the honeypots. Any traffic going to or from the honeypots must go through the honeywall but there is no routing of packets, no TTL decrement of system hops, thus the honeywall is hard to detect.

Secondly, instead of relying on a layer-three firewall that applies Data Control based on IP headers. Gen II honeynet applies a technology called IDS gateway, which not only block connections based on service, but it also has the intelligence to distinguish between an attack and legitimate activity.

Thirdly, depending on the advantage of the layer 2 interfaces, the honeynet deployment can be part of a production network instead of being on an isolated network. Although in reality all the systems including honeypots and production systems are part of the same network, the honeywall divides the honeynet from the production network at layer two, as opposed to layer three.

In addition, for the purpose of Data Capture of system activity, Sebek [9] kernel modules were developed to modify the system kernel in order to record system activity, especially keystrokes, in a harder way to be detected.

The honeywall is also used for Data Collection. If there are several Gen II honeynets deployed in a distributed environment, the data can be encrypted by some technology such as IPsec tunnels to a central point by the third interface of the honeywall and then all distributed honeynets are managed.

However, Gen II Honeynet architecture still has several limitations: Sebek and eeyore are no longer maintained and Sebek can currently be detected easily.

Generation III – In the summer of 2005, Honeynet Project released a new honeywall namely roo, which enables Gen III Honeynet architecture. Indeed, Gen III has no architectural difference from Gen II, but the roo improved the data model over eeyore.

Firstly, Roo improved Data Capture capability by introducing a new hflow database schema and pcap-api for manipulating packet captures. Secondly, it improved data analysis capability by introducing a new web based analysis tool called walleye. And thirdly, it improved installation, operation, customization and the user interface.

However, the Gen III Honeynet architecture still has some limitations. The problem of system activity capture is still unresolved. The honeynet deploying on physical machine is difficult for dynamic configuration.

(2) Virtual Honeynet Architecture. As stated, a virtual honeynet is a solution that allows running multiple honeypots simultaneously over a single physical machine by using virtualization software. Virtual honeynets can be broken into two categories, Self-contained and Hybrid virtual honeynet.

Self-contained – A self-contained virtual honeynet is an entire honeynet network deployed over a single computer. The left part of Fig. 3 presents an overview of self-contained virtual honeynet architecture.

The physical machine running the host operating system includes the whole honeynet architecture that consists of data control and data capture tools and the virtual honeypots running separately different guest operating systems. Thus, this virtual honeynet architecture is very portable and cost effective. Another advantage

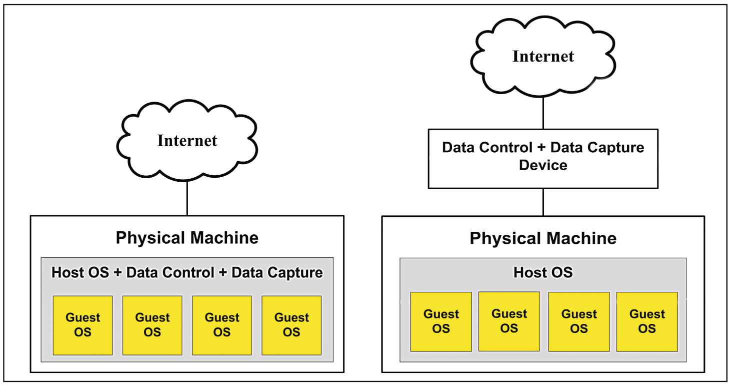


Fig. 3. Self-contained virtual honeynet and hybrid virtual honeynet

is the convenience of Data Collection because it is not necessary to use encrypted tunnel to collect data but we can use some system technology to log the data on the virtual honeypots.

However, the main drawback of this virtual architecture is that since all the services run on one physical machine, if that machine fails or it is compromised then the whole honeynet will break down. Furthermore, service performance is another problem. If the hardware and the virtual software provide a limited service performance, the attacker may easily detect the virtual honeynet environment.

Hybrid – A hybrid virtual honeynet is a combination of the data control and data capture implemented in physically separate machine and the virtual honeypots running on another computer. The right part of Fig. 4 exhibits an overview of hybrid virtual honeynet architecture.

This isolation of the data control and data capture can reduce the risk of honeynet compromise. Thus, this virtual honeynet architecture is much more robust than the self-contained virtual honeynet architecture. Moreover, this virtual honeynet architecture can get a better service performance profit from the physically separate deployment.

Nevertheless, the hybrid virtual honeynet architecture still has several shortcomings. Firstly, this virtual honeynet architecture is not as portable as the self-contained virtual honeynet architecture because there are at least two machines. Secondly, due to the physically separate deployment the cost is not as efficient as the self-contained virtual honeynet architecture.

4 New Virtual Honeynet Architecture

In this section, a new virtual honeynet architecture is proposed. The creation of complex honeynets results in the need of specialized tools to manage them. They should facilitate their definition – topology, addresses, and types of systems, among others–, their deployment, their monitoring, and their security, hiding all the complexity of the underlying virtualization platforms to the user. These specialized tools also must achieve the goal of data control and data capture. In our case, the following tools were selected:

VNX [16] as high-interaction honeypots and virtual network generation tool; Honeyd [2] as low-interaction honeypots creation tool; Nitro [17] as system data capture tool; Snort as intrusion detection tool; and Honeybrid gateway [18] as the network data capture and control tool. Figure 4 presents an overview of our adaptive and flexible virtual honeynet architecture.

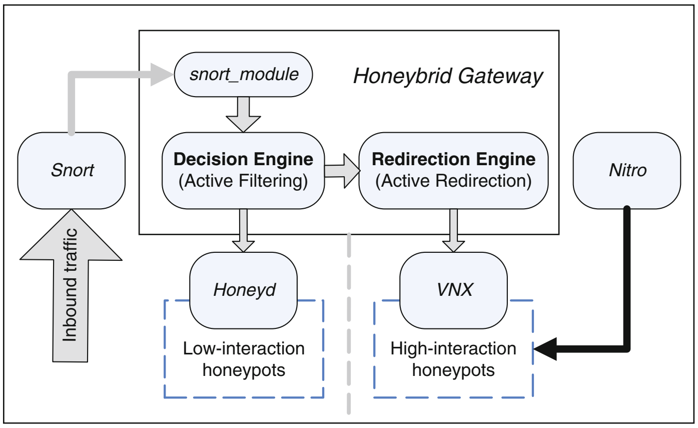

snort_module
Honeybrid Gateway
Snort
Decision Engine (Active Filtering)
Redirection Engine (Active Redirection)
Nitro
Inbound traffic
Honeyd
VNX
Low-interaction honeypots
High-interaction honeypots

Fig. 4. An overview of the new virtual honeynet architecture

We use these specialized tools as the components for hosting the new adaptive and flexible virtual honeynet, but our ideas are not limited to these tools. For the future research, other advanced specialized tools can replace the current tools based on the architecture.

Currently virtual honeynet does not mimic Gen III honeynet, but similar to other virtualization software, VNX also can host a Gen III honeynet. Indeed, our virtual honeynet improves the Gen III honeynet. Firstly, our virtual honeynet can bridge into a production network or deploy in an isolated environment, thus the deployment strategy is flexible. Secondly, the combination of low-interaction honeypots and high-interaction honeypots provides a good balance among scalability, performance and fidelity. Thirdly, we use the out-of-the-box approach to capture the system activity which is much stealthier. Fourthly, the capability of dynamic configuration allows the virtual honeynet to adapt itself to the real-time network environment. The following discussion depicts each tool in detail.

VNX – There are several tools available to create and manage virtual network scenarios, such as VNX (Virtual Networks over linuX), Netkit, MLN (Manage Large Networks), and vBET (VM-Based Emulation Testbed). Among them, VNX emerges as the most powerful solution due to (i) its scalability in creating very complex high interaction honeynets, due to the ability to deploy virtual network scenarios over a cluster of servers; (ii) its ability to automatize the execution of commends inside virtual machines; and (iii), its support for multiple virtualization platforms like KVM, which allows emulating various operating systems for high-interaction honeypots on demand.

Last but not least, the new version of VNX has included the capability of dynamic configuration. In other words, the user can reconfigure the scenario on the fly. The user only needs to write a reconfiguration file based on XML syntax, and later he can use VNX to automatically reconfigure the scenario by processing the reconfiguration file. One part of the running scenario can be redeployed on the fly without impacting on the rest part of the on-going honeynet. Figure 5 shows the state diagram of a high-interaction virtual honeypot emulated by VNX.

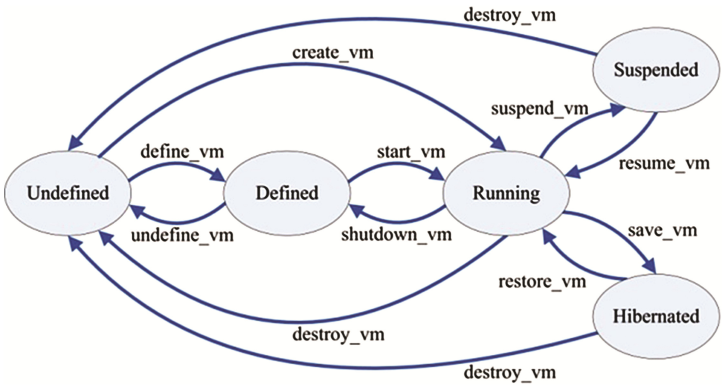


Fig. 5. State diagram of a virtual machine emulated by VNX

A virtual machine emulated by VNX has five states. They are undefined, defined, running, suspended and hibernated. This state diagram follows the states model defined in libvirt and practically uses the same terminologies. VNX has developed functions that can switch a virtual honeypot from one state to any other state.

The undefined state is the unborn state of virtual honeypot. It means the virtual honeypot node is not defined in the honeynet scenario. The defined state is the initial state of virtual honeypot, which indicates that the virtual honeypot exists in the honeynet scenario but the system is power-off. The running state is the normal working state of virtual honeypot. The virtual honeypot on this state can provide services. The suspended state is the power saving state. This method cuts power to most parts of the machine aside from the RAM, which is required to restore the machine's state. The hibernated state is resource saving state. This approach saves the machine's state into swap space and completely powers off the machine. When the machine is powered on, the state is restored. Until then, there is zero power consumption. The virtual honeypot can spin up much quicker from suspended state than from hibernated state, but the hibernated state is much more energy efficiency than suspended state.

Therefore, VNX can host the adaptive and flexible high-interaction honeynet.

Honeyd – Honeyd is a well-known low-interaction virtual honeypot framework. It can quickly deploy a virtual honeynet integrating into a target network or standing alongside a target network. It can emulate fingerprints of various operating systems.

Honeyd has a doorway called Honeydctl to communicate the inner workings of Honeyd. Honeydctl can be used to reconfigure the templates on the fly. All commands used in the regular configuration file of Honeyd template can be interactively used after "honeydctl>" prompt, however, this is also the shortcoming of Honeydctl. The user has to input the commands interactively, one command by one. But, instead of interactively interacting with Honeydctl, our system requires an automatic capability of reconfiguration. Fortunately, the author of Honeyd leaves us a UNIX socket located in/var/run/honeyd.sock. Using this socket, we can create a client socket to communicate with the inner workings of Honeyd, in other words, we can reconfigure the Honeyd templates by the client socket with a script consists of Honeyd commands. Every template has two states, created (online) and deleted (offline). We can use Honeyd commands two spin up and down any template and even any service of a template freely.

In summary, Honeyd can provide the adaptive and flexible low-interaction honeynet.

Nitro – In the Gen II and Gen III honeynet, Sebek/Qebek [15] is the monitor tool used to track the system activity. The main drawback of this kind of kernel module is that it must be installed inside the virtual honeypot to do use an in-the-box monitor approach. However, this monitoring approach can be easily be detected by the adversary. Another monitor method using the idea called "out-of-the-box" [19] is proposed in order to prevent the monitor tool from being detected. In this paper, we employ the tool Nitro to monitor the high-interaction honeypots from out-of-the-box, since it focus on track the system call of the KVM based virtual machine.

Thus, Nitro is applied to capture the system activity but the potential monitor tool is not limited to it.

Honeybrid Gateway – Honeybrid consists of three parts: a gateway, a set of low-interaction honeypots (front-end) and a set of high-interaction honeypots (back-end). The Honeybrid gateway is the central part including the Decision Engine and the Redirection Engine in charge of orchestrating the filtering and the redirection between front-end and back-end. The Decision Engine is used to select interesting traffics and the Redirection Engine is used to transparently redirect the traffic from low-interaction honeypots to the farm of high-interaction honeypots. A contribution of Honeybrid is that a simple classification of interesting attacks is proposed: attacks matching a specific fingerprint (well-known attacks); attacks presenting an original content that was never seen before (0day attacks); attacks sending commands that are not implemented in the low-interaction honeypots (i.e., not implements in Honeyd). Using this classification, Honeybrid can effectively redirect the attacks worth of further investigation to high-interaction honeypots.

The main drawback of Honeybrid is that it is not able to distinguish the malware automated attacks and human manual attacks. More accurately speaking, it neglects the human manual attacks. In the current design, a pair of low-interaction honeypots and high-interaction honeypots has different IP addresses in a honeynet deployment, and the Honeybrid gateway can transparently redirect the traffic from the low-interaction honeypot to the corresponding high-interaction honeypot. It is useful to catch the malware automated attacks but if the attack is from an intelligent human adversary, he can easily detect the traffic redirection by simply checking the IP address of the final compromised system. If the final IP address of the compromised system is different from his original attacking destination IP address, the adversary can convince that he is immersed in a honeypot. As shown in Fig. 5, the Honeybrid gateway is not only a data controller, but also a function controller. When it decides to redirect the interesting traffic from the low-interaction honeypot to the high-interaction honeypot, before it replays the connection, it must switch off the low-interaction honeypot and spin up the corresponding high-interaction honeypot.

Therefore, Honeybrid can play the role of network data capture and control.

Snort and Other Honeybrid modules – Snort is used as the NIDS (network intrusion detection system) to monitor the virtual honeynet. In our virtual honeynet, it is an auxiliary tool for Honeybrid. Because Honeybrid is a programmable tool that provides API to develop new modules for its data control. The security expert can develop a Snort module for Honeybrid blocking the well-known attacks. As such, we allow the Snort NIDS mode to send alerts and output to a UNIX socket that the new Honeybrid module can listen on. The Decision Engine can decide to accept the packet or discard it according to the output from Snort. Besides, we also can develop other modules to achieve other data control goals.

5 Experiments and Measurement

In this section, the proposed virtual honeynet is validated by some experiments. We also compare our virtual honeynet architecture with the existing honeynet architectures in term of the standard measurement criteria provided by Honeynet Project.

5.1 Experiments

Our virtual honeynet can be applied to self-contained virtual honeynet or hybrid virtual honeynet. In this paper, we use the self-contained approach to present our virtual honeynet architecture. The testbed architecture is shown in Fig. 6.

The incoming traffic can firstly connect to the low-interaction honeypot through br0. If Snort detects a well-known attack, the Honeybrid gateway can discard the packet. After the traffic penetrates the NIDS, the Honeybrid gateway should decide to filter the packet or redirect the traffic into a high-interaction honeypot. If the decision is redirection, so before the connection replay, the low-interaction should be deleted and the corresponding high-interaction honeypot has to be waked up. As soon as the high-interaction honeypot get into running state, the interesting traffic is redirected into it.

The possible problem is the delay to start a high-interaction honeypot. So, we test the duration to start or wake up a VNX based high-interaction honeypot. The system parameters of the host machine are: CPU, 4 Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz; RAM, 16 GB; OS, Ubuntu 14.04; Kernel, Linux 3.13.0-24-generic.

VNX can deploy KVM based virtual machine and LXC based virtual machine. The delay is very short to start up a LXC based virtual machine. It is less than 1 s. But the virtual machine based on LXC only can emulate the Linux operation system that uses the same Linux kernel with the host. Thus, this method lacks fidelity. While virtual

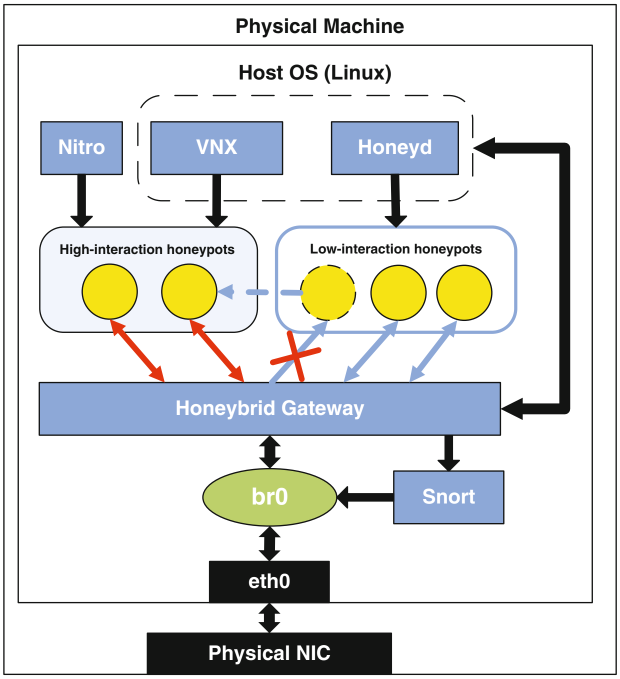


Fig. 6. Testbed Architecture

machine based on KVM can emulate various operating systems. The startup delay of a KVM based virtual machine from different states is demonstrated in Fig. 7.

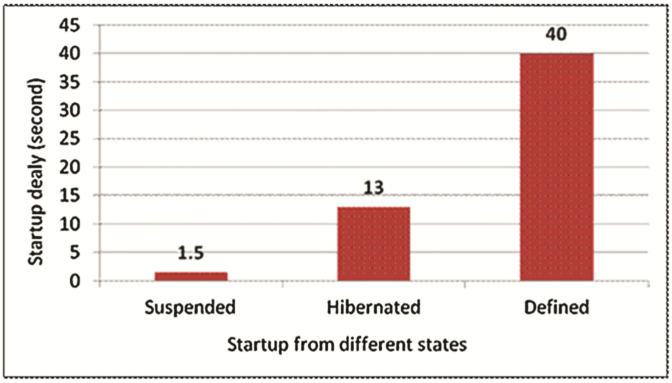


Fig. 7. Startup delay of different states

Thus, to start up or reboot a KVM based virtual machine, the time cost is 40 s. This delay is too long and could result in time out of connection request. To wake up a hibernated virtual machine is not so long, but still need 13 s, thus it is not a good choice either. Lastly, a suspended virtual machine spins up only within 1.5 s. Hence, the suspended is the best state that has the shortest delay to wakeup. Therefore, the approach for KVM based virtual honeypots is to start up a group of virtual honeypots and then keep them into suspended state. When the interesting traffic is decided to be redirected into the high-interaction honeypot, then the corresponding suspended virtual honeypot is wake up to provide service.

5.2 Measurement

Table 1 compares some criteria of measurement provided by the Honeynet Project [20] plus new features namely adaptability provided by us.

Table 1. Comparison of honeynet features

Item	Gen III [8]	UML based [6]	Our work
Portable	No	Yes	Yes
Setup	Involved	Plug and catch	Plug and catch
Cost	High	Low	Low
Flexibility	Very	Limited	Moderated
Security	Secure	Software dependent	Software dependent
Detectability	Reasonable effort	Reasonable effort	Hard to detect
Adaptability	No	No	Yes

The Gen III [8] is the traditional physical virtual honeynet, and the UML based virtual honeynet [6] is the typical self-contained virtual honeynet that is a representation of previous virtual honeynets. Each measure is elaborated below.

Portable is the feature indicates if the honeynet can be moved around. Gen III honeynets are not potable since each components reside on its own physical machine. The self-contained virtual honeynet can be configured and deployed in a single physical machine, thus the UML based virtual honeynet and this virtual honeynet is portable.

Setup is measure of the ease of setting up a new honeynet at a new location. As the physical honeynet, Gen III honeynets have to set up each physical machine individually. However, self-contained virtual honeynets are "plug and catch", which can quickly deploy a whole virtual honeynet without driver conflicts or other hardware related configuration problems

Cost is a measure of hardware cost. Since Gen III honeynets require physical machine to run every component, the hardware cost is high. On the contrary, self-contained virtual honeynets can be setup on a single physical machine therefore hardware cost is low.

Flexibility is a measure of what types of honeynets can be configured. UML is only able to support Linux based operating systems, while KVM can support operating systems that run on the x86 processor architecture. The physical machine can support any kind of operating system with the right hardware.

Security is a measure of how susceptible the honeynet is to be compromised and becomes a threat to systems on the outside. In Gen III honeynet, the security depends on the honeywall. Since the honeywall is tightly configured and access is limited to network based methods, the security of Gen III honeynets is high. However, the security of virtual honeynet depends on the virtualization software. Protection against host-based attacks is provided by software running on the host.

Detectability is a measure that indicates if the honeynet can be detected by the attacker. Kernel module based monitor is easy to be detected by the blackhats currently, thus Gen III honeynets and UML based virtual honeynets are detectable. We use the "out-of-the-box" approach to monitor the system call, thus it is hard to be detected.

Adaptability is a measure of what kinds of honeynets can be configured dynamically. Our virtual honeynet system provides the capability of dynamic configuration and deployment, but the previous work didn't provide this kind of capability.

6 Conclusion and Future Work

In this paper, a new adaptive and flexible virtual honeynet architecture has been proposed. Compared to the previous honeynets, our new honeynet has many advantages, like the dynamic configuration capability, which isn't supported by previous honeynets. Furthermore, as a virtual honeynet, our work overcomes the limitation of operating system supporting. Our virtual honeynet is quite flexible and it can support most of the operating systems that runs over x86 processor architecture. In addition, our virtual honeynet architecture is also flexible, since the security experts can replace the recommended tools with future advanced tools.

Since our virtual honeynet is programmable, for the future work, we will develop much more modules for Honeybrid to provide much better data control. Additionally, we also will design and develop a tool that can manage the whole virtual honeynet system. We hope other security experts can benefit from our work.

Acknowledgement. This research is supported in part by National Natural Science Foundation of China (No. 61440057, 61272087, 61363019 and 61073008), Beijing Natural Science Foundation (No. 4082016 and 4122039), the Sci-Tech Interdisciplinary Innovation and Cooperation Team Program of the Chinese Academy of Sciences, the Specialized Research Fund for State Key Laboratories. It also has been partially funded with support from the Spanish MICINN (project RECLAMO, Virtual and Collaborative Honeynets based on Trust Management and Autonomous Systems applied to Intrusion Management, with codes TIN2011-28287- C02-01 and TIN2011-28287-C02-02) and the European Commission (FEDER/ERDF).

References

1. Spitzner, L.: The Value of Honeypots, Part One: Definitions and Values of Honeypots, 10 Oct 2001. http://www.symantec.com/connect/articles/value-honeypots-part-one-definitions-and-values-honeypots
2. Provos, N.: A virtual honeypot framework. In: SSYM 2004 Proceedings of the 13th Conference on USENIX Security Symposium, vol. 13 (2004)

3. Hecker, C., Hay, B.: Automated honeynet deployment for dynamic network environment. In: 2013 46th Hawaii International Conference on System Sciences (HICSS), pp. 4880–4889, 7–10 Jan 2013
4. Fu, X., Bryan, G., Cheng, D., Bettati, R., Zhao, W.: Camouflaging virtual honeypots. In: Texas A&M University (2005)
5. Wang, H., Chen, Q.: Dynamic deploying distributed low-interaction honeynet. J. Comput. N. Am. **7**, 692–698 (2012)
6. Yan, L.K.: Virtual honeynets revisited. In: Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, IAW 2005. pp. 232–239, 15–17 June 2005
7. Galán, F., Fernández, D.: Use of VNUML in Virtual Honeynets Deployment. IX Reunión Española sobre Criptología y Seguridad de la Información (RECSI), Barcelona (Spain), September 2006. ISBN: 84-9788-502-3
8. Abbasi, F.H., Harris, R.J.: Experiences with a generation III virtual honeynet. In: 2009 Australasian, Telecommunication Networks and Applications Conference (ATNAC), pp. 1–6, 10–12 Nov 2009
9. Honeynet Project. Know Your Enemy: Sebek, A kernel based data capture tool, 17 November 2003. http://old.honeynet.org/papers/sebek.pdf
10. Stoll, C.: The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage. Pocket, New York (1990)
11. Spitzner, L.: Honeytokens: The Other Honeypot, 17 July 2003. http://www.symantec.com/connect/articles/honeytokens-other-honeypot
12. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection, 1st edn. Addison-Wesley Professional, Boston (2007)
13. Honeynet Project. Know Your Enemy: Honeynets, 26 April 2001. http://www.symantec.com/connect/articles/know-your-enemy-honeynets
14. Stumpf, F., Görlach, A., Homann, F., Bruuckner, L.: NoSE - building virtual honeynets made easy. In: Proceedings of the 12th International Linux System Technology Conference, Hamburg, Germany (2005)
15. Honeynet Project. Know Your Tools: Qebek – Conceal the Monitoring, 03 Nov 2010. http://www.honeynet.org/papers/KYT_qebek
16. Fernandez, D., Cordero, A., Somavilla, J., Rodriguez, J., Corchero, A., Tarrafeta, L., Galan, F.: Distributed virtual scenarios over multi-host Linux environments. In: 5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM), pp. 1–8, 24 Oct 2011
17. Pfoh, J., Schneider, C., Eckert, C.: Nitro: hardware-based system call tracing for virtual machines. In: Iwata, T., Nishigaki, M. (eds.) IWSEC 2011. LNCS, vol. 7038, pp. 96–112. Springer, Heidelberg (2011)
18. Berthier, R., Cukier, M.: Honeybrid: a hybrid honeypot architecture. In: USENIX Security Symposium 2008 (2008)
19. Jiang, X., Wang, X.: "Out-of-the-box" monitoring of VM-based high-interaction honeypots. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 198–218. Springer, Heidelberg (2007)
20. Honeynet Project. Know Your Enemy: Defining Virtual Honeynets, 27 January 2001. http://old.honeynet.org/papers/virtual/
21. Capalik, A.: Next-generation honeynet technology with real-time forensics for U.S. defense. In: Military Communications Conference, MILCOM 2007, pp. 1–7. IEEE, 29–31 Oct 2007
22. Memari, N., Hashim, S.J.B., Samsudin, K.B.: Towards virtual honeynet based on LXC virtualization. In: 2014 IEEE Region 10 Symposium, pp. 496– 501, 14–16 April 2014

Analysis of Location Spoofing Identification in Cellular Networks

Yuxin Wei[1] and Dawei Liu[2](✉)

[1] Air Force Engineering University, Xi'an, China
weij0831@126.com
[2] Xi'An JiaoTong-Liverpool University, Suzhou, China
dawei.liu@xjtlu.edu.cn

Abstract. Location spoofing is considered as a serious threat to positioning and location based services in wireless networks. Existing identification methods for location spoofing have focused primarily on wireless sensor networks. These methods may not be applicable in cellular networks due to the following two limitations: (i) relying on accurate distance measurement; (ii) incapable of dealing with bad propagation conditions. To address these two issues, we carry out an analysis of location spoofing based on angle-of-arrival (AOA) and time-difference-of-arrival (TDOA) measurement models, two commonly used signal measurement models in cellular networks, in bad propagation conditions with large measurement errors. Our analysis shows that AOA model is more robust to location spoofing in noisy conditions.

1 Introduction

Location spoofing has attracted much attention during the past ten years because of the development of wireless networking technologies. It refers to an attack carried out by malicious network users for the purpose of misleading a positioning systems. To address this problem, many location identification methods have been developed for wireless sensor networks. The basic idea is to verify the location of a target user with respective to its location. Sastry *et al.* [1] proposed an ECHO protocol for verifying the location claim of a network node based on a challenge-response mechanism. In [2,3], a network node must verify its respective distances to at least three detecting points in order to securely estimate its position. Signals forged in this way will always lead to a consistent localization. Wang [4] pointed out there existed a perfect location spoofing that traditional location spoofing identification methods were unable to deal with. A possible solution is to make use of multiple sensor nodes that can identify with each other [5]. Zhang [6] introduced a mobility-assisted secure positioning scheme and extended the application to Ultra-Wideband (UWB) sensor networks. These identification methods are primarily designed for wireless sensor network (WSN) applications. When applied to cellular networks, these methods would not work properly because of the following two limitations:

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 18–27, 2015.
DOI: 10.1007/978-3-319-25744-0_2

– *Relying on accurate distance measurement.* Existing methods commonly make use of a distance-based identification which requires accurate distance measurement between pairs of sensor nodes. Measuring the distance accurately could be difficult in cellular networks; therefore, identification would become inaccurate.
– *Incapable of dealing with large measurement error.* WSN usually covers a small area. The environmental noises and the signal measurement errors caused could be stable. In contrast, a cellular network can be deployed in a wide and complex area. The signal measurement errors caused by environmental noises could vary significantly over time and place. It is not clear whether existing identification methods can be applied in the error-prone conditions.

In this paper, we address the above mentioned two problems. We first carry out an analysis of location spoofing based the angle-of-arrival (AOA) and time-difference-of-arrival (TDOA) based measurement models which are commonly used in cellular networks. Then we propose a cooperative method to identify location spoofing in bad channel conditions. Our security analysis shows that the AOA model could offer better security and lower hardware requirement when facing location spoofing. The remainder of the paper is organized as follows. Section 2 discusses wireless positioning in AOA and TDOA models. Section 3 presents the cooperative secure positioning method. Section 4 analyzes the security of the two models. Section 5 concludes this paper and gives some directions for further research.

2 AOA and TDOA Models

In this section we present an analysis of location spoofing in AOA and TDOA based models. We propose two methods to identify location spoofing in these two models separately.

A general cellular network is considered. It consists of a mobile station (MS) at X and a group of base stations $(BS_1, \ldots, BS_n)$ at $(X_1, X_2, \ldots, X_n)$. In this section, we assume the magnitude of environmental noise is small. The influence of a large environmental noises will be discussed in the next section.

2.1 AOA Model

AOA measurement model has been widely applied in existing cellular networks [7]. BS equipped with antenna array allows the measurement of arriving angle of radio signals. Let θ_i be the arriving angle of a radio signal sent from the MS and measured by BS_i. The location $X(x, y)$ of the MS can be estimated via the following equation:

$$\theta_i = atan(\frac{y_i - y}{x_i - x}) \tag{1}$$

in which $X_i(x_i, y_i)$ is the location coordinates of BS_i which can be obtained beforehand. Clearly, Eq. (1) represents a line connecting the MS and BS_i. We refer to

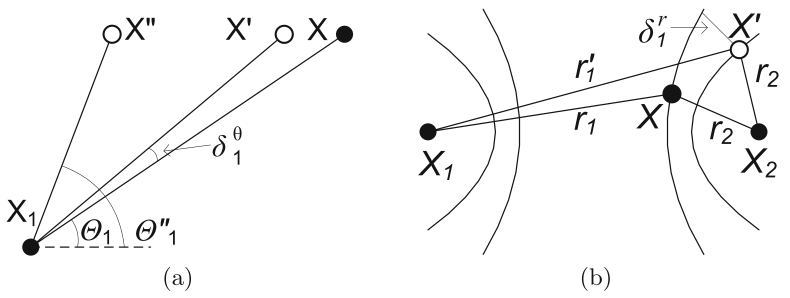


Fig. 1. Inconsistency caused by environmental noises and malicious MS. (a) In AOA model, angle measurement is biased to θ_1' and θ_1'' due to environmental noises and malicious attack respectively. (b) In TDOA model, hyperbola estimation is biased due to distance measurement error in r_1 (biased from r_1 to r_1') caused by environmental noises and/or malicious attack.

this line as a positioning line. If the radio signal sent from the MS can be measured by two or more BSs, X should be a variable satisfying Eq. (1) for all the BSs, or geometrically, be the crossing point of all the positioning lines.

By manipulating its radio signal, a malicious MS can mislead the angle measurement at BS_1 from θ_1 to a significantly biased value θ_1''. Figure 1(a) shows an example. A location estimation that makes use of BS_1 as well as other two beacons in which the angles are measured accurately would very likely to be inconsistent such that there does not exist an X satisfying Eq. (1) with all three BSs. This can be understood geometrically in terms of multiple positioning lines crossing in a region instead of a point. The degree of inconsistency could be closely related to the range of error caused by location spoofing. Liu et al. [8] proposed to measure the degree of inconsistency with the mean square error and use it to identify a location spoofing.

The presence of a malicious MS may not necessarily lead to an inconsistent location estimation [4]. A smart MS could manipulate its radio signal carefully so that all the positioning lines would cross at a point $X'' \neq X$. This means t he location estimation would be consistent; and the MS would not be identified as malicious. This type of location spoofing could be more deceptive compared with the one mentioned above. To deal with this problem, Capkun [9] proposed an identification method that made use of a hidden BS with the location unknown to any MS. According to Eq. (1), it is impossible to determine the positioning line if the BS location $X_i(x_i, y_i)$ is unknown. A malicious MS would be unable to estimate the location of the positioning lines and therefore unable to keep the consistency in location estimation. In the following discussion, we will assume the existence of at least one hidden BS. Therefore, any location spoofing would be associated with inconsistent location estimation. It is worth to mention that, literature [9], as an important contribution to location spoofing in cellular network, has not managed to address the two problems we mentioned in Sect. 1.

Environmental noises can be another cause of inconsistent location estimation. Cellular networks deployed in urban regions may have inaccurate angle measurements at every BS. Similar to the errors caused by a malicious MS, the measurement error caused by the environmental noises could lead to inconsistent location estimation. However, the range of the error is typically small when caused by the environmental noises, such as δ_1^θ shown in Fig. 1(a). This can be explained as follows: if a malicious MS wants to cause a large error in the positioning, it would have to bring a bias larger than the ones caused by environmental noises; otherwise, it would not have any significant impact on positioning result. In order words, there is not need to identify a malicious MS if its influence is in the same level of environmental noise

Based on the above analysis, we propose a location spoofing identification method as follows: (i) estimating X with Eq. (1); (ii) computing the mean square angle error (MSAE) based on the positioning result X; (iii) checking if the MSAE is within the range of the measurement error caused by environmental noises which can be measured beforehand. The MSAE is defined below:

Definition 1. Given the positioning result $X(x, y)$, of an MS and the arriving angles $(\theta_1, \ldots, \theta_n)$ measured at $(BS_1, \ldots, BS_n)$, the square angle error (SAE) of BS$_i$ is defined as

$$\delta_i^\theta = (\theta_i - atan(\frac{y_i - y}{x_i - x}))^2 \tag{2}$$

and MSAE of all BSs is defined as $\varDelta^\theta = \sum_{i=1}^n \frac{\delta_i^\theta}{n}$

In the above method, the MSAE is used as a measure for the degree of inconsistency. The underlying principle is: if the angle measurement is largely biased, the positioning result X would always be accompanied with a large MSAE; since the environmental noises could never cause such a large MSAE, the MS would be identified as malicious. In contrast, if the bias is small, it would be possible to find an X associated with a small MSAE. As mentioned above, a small MSAE is likely to be caused by the environmental noises or a malicious MS that does not have any significant influence on the positioning result. A similar application of MSAE has been discussed in detail in TOA based location spoofing identification in wireless sensor networks [8].

2.2 TDOA Model

TDOA is another measurement model that has been widely applied for location estimation in cellular networks. In this model, the radio signal's propagation distance is measured between the MS and BS_i with $r_i = (t_i - t) \times c$, where t is the time at which the radio signal is sent out from the MS, t_i is the time the radio signal arriving at BS_i, and c is the transmission speed of the radio signal. r_i is referred to as a pseudorange. This is because the clocks at MS and BS_i may not be synchronized. Let ϵ_i be the error caused by the synchronization bias in r_i, the location $X(x, y)$ of the MS can be related to the location $X_i(x_i, y_i)$ of BS_i as following:

$$r_i = \sqrt{(x_i - x)^2 + (y_i - y)^2} + \epsilon_i \tag{3}$$

Unlike the MS, BSs are usually synchronized with each other, meaning ϵ_i would be the same for different i. By subtracting r_1 from r_i, we can obtain the following nonlinear equation

$$\begin{aligned} r_i - r_1 = &\sqrt{(x_i - x)^2 + (y_i - y)^2} \\ &-\sqrt{(x_1 - x)^2 + (y_1 - y)^2} \end{aligned} \tag{4}$$

which relates X to X_i and X_1. Geometrically, Eq. (4) represents a hyperbola consisting of all the possible locations of the MS. If the radio signal of the MS can be measured by 3 or more BSs at the same time, X would be the crossing point of the correspoding hyperbolas determined by pairs of BSs.

The problem of inconsistent location estimation could arise in TODA measurement model if a malicious MS is involved or environmental noises are considered. In the pseudorange measurement, a malicious MS may falsify the time t whereas environmental noises can affect the measurement of t_i, both of which can result in a distorted r_i in Eq. (4). An example is shown in Fig. 1(b). A location estimation carried out by multiple BSs involving such an r_i would very likely to be inconsistent, i.e., there does not exist an X satisfying all set of equations in the form of (4). As we have explained in Sect. 2.1, the degree of inconsistency caused by environmental noises could be assumed small, whereas a large inconsistency should be caused by a malicious MS.

Based on the above analysis, we propose to identify a malicious MS with a three-step method similar to the one proposed in Sect. 2.1. In particular, we modify the parameter in step (ii) from MSAE to the mean square distance error (MSDE), and use it as a measure of consistency in locationing. The MSDE is defined as below:

Definition 2. Given an MS with the TDOA measurements $(r_2 - r_1, r_3 - r_1, \ldots, r_n - r_1)$, the square distance error (SDE) of BS$_i$ is

$$\delta_i^r = (r_i - r_1 - (l_i - l_1))^2 \tag{5}$$

where $l_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$, and the MSDE of all BSs can be obtained by $\varDelta^r = \sum_{i=2}^{n} \frac{\delta_i^r}{n}$

For a benign MS, the ranges of SDE $\varDelta_i^r$ and MSDE $\varDelta^r$ would be small which can be measured in advance. If a malicious MS forges the time t or causes a largely biased distance measurement, $\varDelta^r$ would increase significantly. Generally, the problem of secure location in both the AOA and the TDOA models can be summarized as follows: Determine whether the measured $\varDelta$ ($\varDelta^\theta$ for the AOA model and S^r for the TDOA model) is below the threshold $\varDelta_0$ ($\varDelta_0^\theta$ or $\varDelta_0^r$) measured beforehand.

3 Cooperative Secure Positioning

In this section, we present a cooperative method to support secure positioning in non-line-of-sight (NLOS) propagation conditions. NLOS is known as the primary

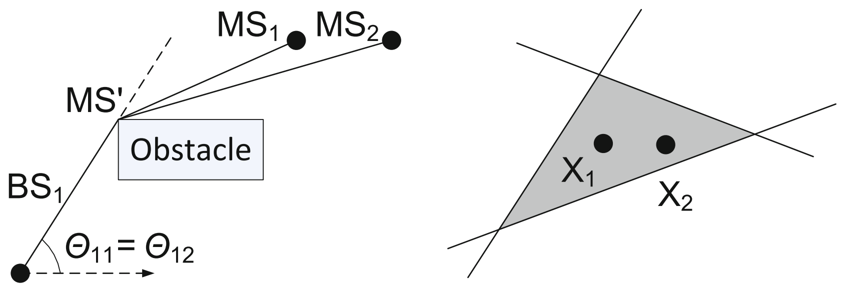


Fig. 2. AOA positioning of two MSs nearby under NLSO propagation conditions. (a) Inaccurate angle measurement; (b) Inconsistent positioning

cause of large positioning errors [10]. It is caused by obstacles blocking the direct propagation path of radio signal. Refer to Fig. 2(a) for an example and let MS_1 be benign. The radio signal's arriving angle measured at BS_1 is θ_{11}. This will lead to a biased position estimation in the form of Eq. (1). If there are two other BSs BS_2 and BS_3 that can measure the radio signal's arriving angle accurately. The positioning of MS_1 carried out by BS_1, BS_2, and BS_3 would be inconsistent as the corresponding positioning lines would cross in an area rather than a point, such the situation shown in Fig. 2(b). The same problem could happen to MS_2. Moreover, because of the obstacle, the radio signals sent from MS_1 and MS_2 could have the same arriving angle to BS_1 as if they were sent from an MS' located at a corner of the block.

In the case of a large obstacle located between MS_1 and BS_1, the degree of the inconsistency in the positioning of MS_1, measured with $\varDelta_1$, would become large. Consequently, the validity of the three-step secure positioning methods proposed in Sect. 2 would be questionable: if we set a small threshold $\varDelta_0$ as before, the benign MS_1 would be identified as malicious; if we adjust the threshold $\varDelta_0$ to a large value, a malicious MS could be identified as benign.

```
1: record $\varDelta(\varDelta_1, \ldots, \varDelta_m)$ for each MS
2: $for$ each $\varDelta_i$
3:    $if$ there is a $\varDelta_{j \neq i}$ satisfying $|\varDelta_i - \varDelta_j| < \varDelta_0$
4:       identify $MS_i$ as benign
5:    $else$
6:       identify $MS_i$ as malicious
```

To address this problem, let us consider again the MS_2 in Fig. 2(a). Since the inconsistency in the positioning of MS_2 and MS_1 is caused by the same obstacle, the degree of inconsistency, measured with $\varDelta_2$, could be similar to $\varDelta_1$, and the similarity can be measured based on the relative position of MS_1 and MS_2. Specifically, the arriving angle measured at BS_1 is the same for MS_1 and MS_2, and the difference between $\varDelta_1$ and $\varDelta_2$ is determined by the other two BSs, BS_2

and BS_3. In the above, we assumed accurate measurement at BS_2 and BS_3. This means the difference is determined by the relative position of MS_1 and MS_2. In particular, if the distance between MS_1 and MS_2 is within the range of the positioning error caused by channel noise in good channel conditions, the value of $|\Delta_1 - \Delta_2|$ would be smaller than the threshold Δ_0 measured in good channel conditions. Based on this, we propose an identification method as above.

It is easy to see that, in the TDOA positioning model, the above-proposed method can be applied directly for the identification of location spoofing, and the analysis of the threshold will hold well. It is worth mentioning that we assumed in the above analysis that majority of the MSs are benign. This assumption would be valid for most cellular networks. For some WSNs, such as those deployed in hostile environments, this assumption may not be valid, consequently, the out method may not be able to work properly.

4 Security Analysis

4.1 AOA Model

In the AOA positioning model, the measurement of arriving angle θ does not need the coordination from the MS. As a result, an attacker will not be able to mislead the BS about θ by manipulating its radio signal. In contrast, an inaccurate measurement of θ can be caused by NLOS propagation. In this situation, the method proposed in Sect. 3 can be used for secure positioning.

In the presence of multiple attackers, one attacker, say A, can compromise the measurement of θ at one or several BSs by coordinating with another attacker in the follower manner: A does not communicate directly with the BSs around, instead, it sends and receives radio signal to another attack A' which acts as an agent for communication. In this situation, BSs may not be able to measure θ of A accurately. In order to mislead the positioning carried out by BS_1 and BS_2, A has to be cautious about its radio signal strength and its position regard to the BSs. Specifically, a too strong signal of A would be detected by the BSs and lead to the inconsistency between θ_1 and θ_1'. In order to avoid the inconsistency, A has to control its transmitting power carefully or use directional antennas. In practice, controlling the transmitting power without being detected is difficult since BSs do not release their positions in AOA positioning. Moreover, it A is inside the convex hull of the BSs [11], it would be impossible for A to carry out such an attack.

Another problem for the proposed the cooperative secure positioning comes with the presence of multiple attackers. Recall that we identify an attacker by testing its inconsistency Δ with other MSs. If an pair of MSs can be found with a similar Δ, both of them could be identified as benign. This identification method may not be robust against multiple attackers. Consider Fig. 2 for an example, and let A and A' be two close attackers. By coordinating with each other, A and A' could cause inconsistencies similar to each other, consequently, both of them would be identified as benign. A solution is to modify the step 3 of the method proposed in Sect. 3 as follows: "if majority of Δ satisfying $|\Delta_i - \Delta| < \Delta_0$". As long as the majority of MSs are benign, which is a reasonable assumption for cellular networks, all the attackers can be identified even if they had the same Δ.

4.2 TDOA Model

Distance-based positioning requires coordination between each BS and MS. An attacker A may compromise BSs by manipulating its radio signal. For example, A may report a false local time t to a BS, resulting in an error in the measurement of r. Consequently, the positioning based on Eq. (3) would be inaccurate. However, this type of attack will not affect the TDOA positioning. Recall that the biases δ_1 and δ_i are removed by subtracting r_1 from r_i. After this, the Eq. (4) will not contain the variable t. As long as the same signal (with the same t) is observed by BS_i and BS_1, a manipulated t would not have any influence on TDOA positioning.

On the other hand, the attacker may affect the TDOA positioning by reporting to BS_i and BS_1, respectively, two different t, and the positioning using Eq. (4) would be inaccurate. Figure 1(b) illustrates an example, where r_2 is accurate and r_1' is altered because t sent to MS_2 is different from the one sent to MS_1. This attack can be carried out by one or multiple attackers. In order to convince all the BSs that they are observing the same signal, A needs to avoid different signals with different t being observed by the same BSs. Based the analysis in Sect. 4.1, this requires the position information all BSs.

In the TDOA model, positioning could be carried out in the uplink and downlink. In the uplink positioning, such information is not released, therefore, the positioning system would be secure against the attack. In the donwlink positioning, such information is released to every MS, consequently, an attacker outside the convex hull region of the BSs can compromise the positioning by manipulating its radio signal.

Remark. Based on the analysis above, the angle-based model outperforms the TDOA distance-based model in: (i) It require less BSs for a secure positioning. (ii) It does not suffer from some attacks. The detail comparison can be found in Table 1.

Table 1. Comparison of two positioning models.

	AOA model	TDOA model
Number of BSs	≥ 2	≥ 3
Physical measurement	Uplink	Uplink & downlink
Reveal BS coordinates	No	Yes*
Number of attacker	≥ 2	≥ 1
Coordination required	Yes	No
Attacker's position	Restricted	Anywhere

*BS coordinates are broadcast in downlink positioning

5 Conclusions

Existing location spoofing identification methods are mostly inadequate for cellular networks. One major reason is that they rely on accurate distance for positioning, which is difficult to achieve in cellular networks. Another reason is that they cannot work properly in non-line-or-sight (NLOS) propagation conditions. In this paper, we present an identification method based on the angle-of-arrival (AOA) model and the time-difference-of-arrival (TDOA) model, which have been applied widely in cellular networks. This method can identify a malicious mobile station (MS) when the positioning error caused by environmental noises is small. We also propose a cooperative secure positioning method to deal with the problem of NLOS propagation conditions which can cause a large positioning error. The underlying principle is that a malicious MS can be identified by analyzing the inconsistency in the positioning of the MSs located nearby. Our security analysis shows that compared with the TDOA model, AOA model could be more robust, as it requires only 2 BSs for a secure positioning. Our future work includes the combination of the AOA and TDOA models, and the applicability of the proposed identification method when an MS is lack of large number of neighbors.

Acknowledgement. This work was supported in part by the Natural Science Foundation of Jiangsu Province under Grant BK20140404, by the Jiangsu University Natural Science Research Programme under Grant 13KJB510035, and by the Suzhou Science and Technology Development Plan under Grant SYG201405.

References

1. Sastry, N., Shankar, U., Wagner, D.: Secure verification of location claims. In: Proceedings of ACM workshop on Wireless Security, pp. 1–10 (2003)
2. Capkun, S., Hubaux, J.: Secure positioning of wireless devices with application to sensor networks. Proc. IEEE INFOCOM **3**, 1917–1928 (2005)
3. Anjum, F., Pandey, S., Agrawal, P.: Secure localization in sensor networks using transmission range variation. In: Proceedings of IEEE MASS, pp. 195–203 (2005)
4. Wang, T., Yang, Y.: Analysis on perfect location spoofing attacks using beamforming. In: Proceedings of IEEE INFOCOM, pp. 2778–2786 (2013)
5. Liu, D.: Identifying malicious attacks to wireless localization in bad channel conditions. In: Proceedings of IEEE International Workshop on Mission-Oriented Wireless Sensor Networking (MiSeNet), pp. 636–641 (2014)
6. Zhang, Y., Liu, W., Fang, Y., Wu, D.: Secure localization and authentication in ultra-wideband sensor networks. IEEE J. Sel. Areas Commun. **24**(4), 829–835 (2006)
7. Cong, L., Zhuang, W.: Hybrid TDOA/AOA mobile user location for wideband CDMA cellularsystems. IEEE Trans. Wirel. Commun. **1**(3), 439–447 (2002)
8. Liu, D., Ning, P., Du, W.: Attack-resistant location estimation in sensor networks. In: Proceedings of ACM/IEEE IPSN 2005, pp, 99–106 (2005)
9. Capkun, S., Rasmussen, K., Cagalj, M., Srivastava, M.: Secure location verification with hidden and mobile base stations. IEEE Trans. Mob. Comput. **7**(4), 470–483 (2008)

10. Sayed, A., Tarighat, A., Khajehnouri, N.: Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. IEEE Sig. Process. Mag. **22**(4), 24–40 (2005)
11. Liu, D., Lee, M.C., Pun, C.M., Liu, H.: Analysis of wireless localization in nonline-of-sight conditions. IEEE Trans. Veh. Technol. **62**(4), 1484–1492 (2013)

Leveraging Event Structure for Adaptive Machine Learning on Big Data Landscapes

Amir Azodi(✉), Marian Gawron, Andrey Sapegin, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam, 14482 Potsdam, Germany
{amir.azodi,marian.gawron,andrey.sapegin, feng.cheng,christoph.meinel}@hpi.de

Abstract. Modern machine learning techniques have been applied to many aspects of network analytics in order to discover patterns that can clarify or better demonstrate the behavior of users and systems within a given network. Often the information to be processed has to be converted to a different type in order for machine learning algorithms to be able to process them. To accurately process the information generated by systems within a network, the true intention and meaning behind the information must be observed. In this paper we propose different approaches for mapping network information such as IP addresses to integer values that attempts to keep the relation present in the original format of the information intact. With one exception, all of the proposed mappings result in (at most) 64 bit long outputs in order to allow atomic operations using CPUs with 64 bit registers. The mapping output size is restricted in the interest of performance. Additionally we demonstrate the benefits of the new mappings for one specific machine learning algorithm (k-means) and compare the algorithm's results for datasets with and without the proposed transformations.

Keywords: Machine learning · Network monitoring · Traffic classification · Event normalization

1 Introduction

Machine Learning is a data driven approach to solving problems [5]. It allows the analytical process of an algorithm to change in accordance to changes in the data set being processed. Pattern detection, self modification and intelligent logic are some of the properties that make up machine learning algorithms. Machine learning can be very powerful since it can learn from data and use the acquired knowledge to make future decisions. Here we look at one problem facing the application of machine learning algorithms to computer network landscapes for the purpose of monitoring them. The aim of such a system would be to better recognize security threats lurking within a network. One obstacle causing inaccuracies in the results of machine learning algorithms is the lack of a representative data type for the information being processed. This becomes more

© Springer International Publishing Switzerland 2015

DOI: 10.1007/978-3-319-25744-0_3

apparent when applying machine learning to network traffic, since in many cases the events are treated as generic strings as opposed to integers, booleans, etc. In this paper we distinguish some of the most vital fields in network events and look for effective ways of casting them to improved types.

1.1 Real-Time Event Analytics and Monitoring System - REAMS

Network events are often logged and written out to log files. Large IT landscapes often posses central systems tasked with storing the events recorded by the systems inside of their respective networks. The main approach is for the hosts to forward the events they produce to this central log repository. In many countries this can be a mandatory function of large networks for IT security auditing compliance. For the purposes of the experiments described in this paper, we have utilized a research system [1,2] designed to process and normalize network events in real-time. Using this system is beneficial as it allows us to extract the specific sections of the events that are of interest. Examples of such fields include *Internet Protocol Addresses* for hosts, *Port Numbers*, *Hostnames* and *Usernames*.

1.2 Events of Interest

Not all events produced across a network can be used for security analytics purposes. In many cases the relevant information contained can be very limited. In order to minimize and focus the data being processed by machine learning algorithms, the feed to the function can be filtered. The filtration can vary based on a number of different factors such as the software systems used throughout the network. Some examples of events that hold valuable information are firewall events, authentication events and routing events.

1.3 Article Layout

Some interesting research works on machine learning related data transformations are described in the Background Work section. Additionally, data normalization is looked at and evaluated as it relates to information mappings. Security domain specific data transformations and mappings are examined in the Data Transformation section. We present the findings of our experiments in the Results section. Some data visualizations are provided in the Visualization section. The paper's conclusions are shown in the Conclusions section and some suggestions for future directions are noted in the Future Works section.

2 Background Work

In our demonstrations, we focus on a specific machine learning algorithm (k-means) in order to accurately provide measured gains made due to the optimizations made to the data points. Pre-normalized data sets are used during the tests. Object Log Format was chosen as the standard for the normalization.

2.1 Object Log Format

System events are often recorded and logged in their textual form. However, textual representation of numbers and their mappings causes complications for analytical tasks. Object Log Format [3] is an event representation model that preserves the data types of the fields and values present in an event. This allows analytical systems to do their calculations in a more effective way. We use OLF to extract access specific fields of an event. Additionally, we can derive the meaning behind the specific fields (e.g. Source vs Destination Addresses) and use them to map the values in a meaningful way.

2.2 Machine Learning

k-means. k-means [11] is an algorithm which is normally used to perform a cluster analysis of data. This method can deal with multiple features of the data to identify correlations or patterns. The generated clusters divide the data into partitions which contain data with similar characteristics for the specified features. Therefore k-means creates clusters in the way that the difference between the center of the cluster and all the data points that belong to the cluster is minimal for each feature. If this minimal distance between a cluster center and a data point is achieved, the data point is connected to this cluster. This relation is similar to Voronoi diagrams [12], where each Voronoi cell is an area around a center. For all points in this area, the distance to all other centers exceeds the distance to the center of the related cell. A picture of a Voronoi diagram can be found at Fig. 1 where we will show different cells with different color and the center of each cell is indicated by a black dot.

Furthermore Voronoi diagrams can be used to visualize the results of a k-means computation as was also shown in [14]. Although this visualization is more suited to visualize two dimensional data, it shows the general idea of k-means clustering. For illustration of k-means, the Voronoi cells will represent the different clusters and the centers of each cell are related to the centroids of

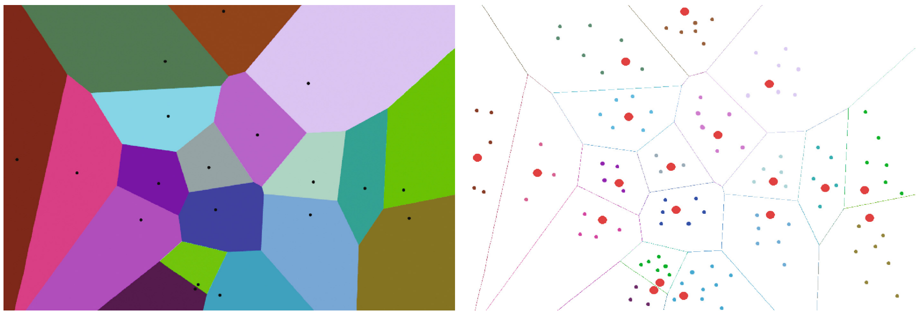

(a) Voronoi (b) corresponding k-means

Fig. 1. Voronoi visualization in comparison to k-means clustering (Color figure online)

each cluster. This correlation is illustrated with the second picture of Fig. 1. The areas directly map to the different clusters and the centers of each Voronoi cell are replaced by red markers for the centroid of the cluster.

In the following equation we regard x as the data point in a f-dimensional vector with f as the number of features, C_i as the current cluster and μ_i as the mean point of this cluster, which was earlier referenced as the center. Then for each of the clusters the sum of all the distances of data point to center should be minimal which is illustrated in Eq. 1.

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i|| \tag{1}$$

This problem is still known as NP-hard which means we can only achieve an approximative solution for clustering. Nevertheless the result could be enhanced by using a round-based approach with the following individual steps. In step 1 the initial centers are randomly chosen. Then for step 2 the associated points, which have the lowest distance to the current cluster, are identified and marked as related to this cluster. Afterwards in step 3 the centroid of each of this clusters is computed. This means that the mean of all of the associated data points will become the new center of the cluster. Then step 2 and 3 are repeated until no more changes arise for the centroids. In this case the optimal position for each of the cluster centers was determined. For further observations we will refer to the k-means algorithm of Hartigan and Wong [11].

Anomaly Detection. The detection of anomalous behavior is an interesting approach to detect malicious actions in a complex system. Most of the time malicious users exploit unusual functionality or unchecked user input to produce unpredicted system behavior. This could lead to problems, such as critical failures of the service or the system. Since often the system behavior is unwanted, it could be seen as anomalous, since under normal conditions the system would react differently. For our implementation we chose an anomaly detection which uses the Poisson distribution [7] as a reference to normal user behavior. This implementation of anomaly detection lies in the field of supervised machine learning. The major difference to unsupervised machine learning is that for our approach we need training data. This training of our model is used to identify normal behavior for future comparisons. With the Poisson distribution we can get the probability of a given number of unrelated events in a fixed time period if we know the average rate of those events in this time period. Equation 2 illustrates this problem where k is the given number of events and λ is the average rate of those events in the period of time.

$$Pr(X = k) = \frac{\lambda^k \times e^{-\lambda}}{k!} \tag{2}$$

To solve Eq. 2 we need to obtain k, the number of events which were identified, and λ, the rate of events in the period of time. Thus we have to train our algorithm first to identify the normal behavior of the investigated system which is represented by the average rate of events in the period of time. So the training of the algorithm is used to specify λ. The number of occurrences of the events (k) will be captured by the event logs in real-time. Thus it is possible to give information about anomalies at each time after the training is completed.

SVM. Support Vector Machines [6] also come from the field of supervised machine learning. For this method the data is transformed to a set of hyperplanes with a large number of dimensions. Afterwards different types of objects should be separated. The requirement to create a good separation is that the distance between every data point and the separator is maximized. Sometimes the data could not be divided with a linear separator in this multi-dimensional space. In these cases the data points have to be translated into feature space first and afterwards a proper division could be applied. Nevertheless the initial creation of separation functions requires additional checks to ensure the desired results. Another possibility to use this approach is to define a separation of usual and unusual data. This separation could result in a set of thresholds for desired features that should never exceed a maximum. Therefore one could integrate unusual data or log events of attacks into the training data. The creation of the separator to divide normal events from attack related events would be the next step.

After this training is done the separation functions are applied to the real data. The idea is that this real data could be separated in the same way. This results in the automatic classification of data points to formerly specified groups. Thus we could immediately identify special log events or even find events which are very similar to attacks.

3 Data Transformation

By observing events occurring throughout a network it becomes evident that the majority of the event records are stored and transmitted in text form. Given that in machine learning different data types can lead to widely differing results, the right data type is needed to produce accurate analysis of the data set. Additionally some fields can present more challenges. For example IPv4 addresses – in many cases – are recorded using four octets separated by dots. Each octet is represented using an integer range between 0 and 255 (e.g. 192.168.178.2). When machine learning algorithms such as k-means clustering are used to analyze IPv4 addresses in such a form, the quasi random data points used as cluster centers will have meaningless distance relations between themselves and the rest of the data points. This is evident in the following example case; the IP addresses 19.168.0.1 and 192.168.0.1 in their text form would be placed into the same cluster since the only textual difference between them is the character ‘2’. However the integer representations (i.e. 329,777,153 and 3,232,235,521 respectively) of them are so far apart that it would be highly unlikely that they would end up in the same cluster. Figure 2 presents a simplified view of the case when shorter

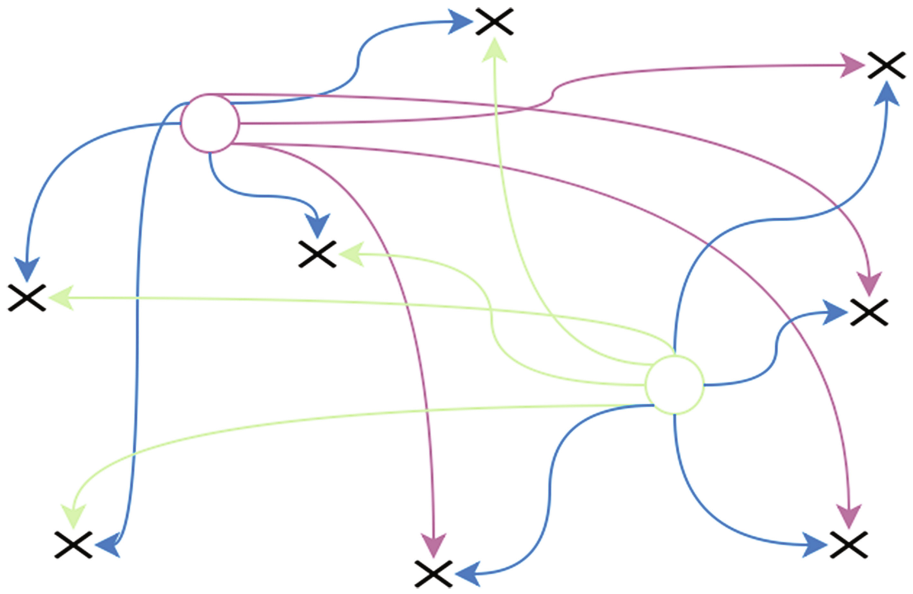


Fig. 2. Distance based selection model (e.g. k-means Clustering)

distances between cluster points and data points are selected. The method for calculating the distances between the nodes can differ as it depends on the data type and other factors involved.

Though highly effective, in many instances it is difficult to find a meaningful mapping between a textual representation of an attribute and its intended value. We demonstrate alternative mappings for the following network traffic relevant fields. IP (v4 and v6), Hostnames, Usernames, and Port numbers.

3.1 Internet Protocol: IPv4

The mapping of IP addresses requires additional normalization steps, since we also have to be able to compute distances between different addresses. The algorithm which should be used to perform the clustering is only able to work with numerical inputs. This input is later directly used for the distance calculation. The first and simple approach would be to create a dictionary and assign the first IP to 1, the next IP to 2 and further on. This method satisfies the requirement that the machine learning algorithm can use numeric input and is able to identify different IP addresses, since those addresses will get different indexes. The problem of this naive approach is that the relation between different IP addresses is completely lost. This approach would allow a scenario where an IP address like 192.168.2.4 which was found first will receive index 1. Another IP address like 192.168.2.1 which was found on position 100 will receive the index 100. This means that the two IP addresses will be perceived to be very different from each other, although they do not differ that much in the first place. This behavior could lead to problems since the k-means algorithm will then use the indexes and evaluate the two IP addresses with a high distance from each other, although the algorithm should treat them as similar.

Additionally, the only remaining information after having applied this normalization approach to IP addresses is that they are not the same address. However, it is not possible to investigate which addresses are *more* different than others.

In contrast to the naive approach, we propose a mechanism whereby IP addresses are mapped and normalized into 32 bit integer values that represent them and preserve their relation and distance to other IP addresses. In this scenario, we regard IP addresses with four blocks (octets) of numbers (0 to 255) farther apart from each other if the difference is located in a leading block as opposed to a later block. It is more likely that the IP addresses originate from different subnetworks if the difference is in one of the leading blocks.

We use the following simple formula to convert an IPv4 address to its decimal representation where $\mathrm{O}_{1\text{-}4}$ represent the 4 octets of an IP address from left to right.

$$(O_1 * 256^3) + (O_2 * 256^2) + (O_3 * 256) + (O_4).$$

3.2 Internet Protocol: IPv6

Similarly to IPv4, it would be logical to transform IPv6 to its integer representation for comparisons. The major problem with doing so is that IPv6 addresses are 128 bits long which would not fit in an integer comparison. Many programming environments allow for mathematical operations on very large (>64 bits) integers but do so using software. Since they can not take full advantage of the hardware, the operations can be considerably slower. In [8], the author demonstrates the performance penalty resulting from calculations on large integers. The relation emerging from their tests is a linear polynomial $f(x) = a_0 + a_1x$ where $a_0 = 2.55216e - 07$ *and* $a_1 = 8.14e - 09$. Figure 3 demonstrates how this performance penalty increases as the integer grows.

In the interest of better performance, we can make certain assumptions about the input values. Given that the input is always going to be IPv6 addresses, we make the following weak assumption.

Assumption: IPv6 addresses that are more than 64 bits apart from each other (in the integer realm) can safely be excluded from being placed into the same cluster.

We use binary operations for calculating the difference between two very large numbers. Given that N_1 and N_2 are two positive 128bit (at most) numbers and $N_1 >= N_2$ we calculate the difference between them using $R = N_1 + ({\sim}N_2) + 1$ where $({\sim}N_2)$ is the ones' complement (bitwise NOT) of N_2. By adding one to it we calculate the two's complement. The result (R) of adding N_1 to the two's complement of N_2 is in turn the difference between the two numbers. Given our assumption, if the binary length of R is greater than 64 we ignore the relation

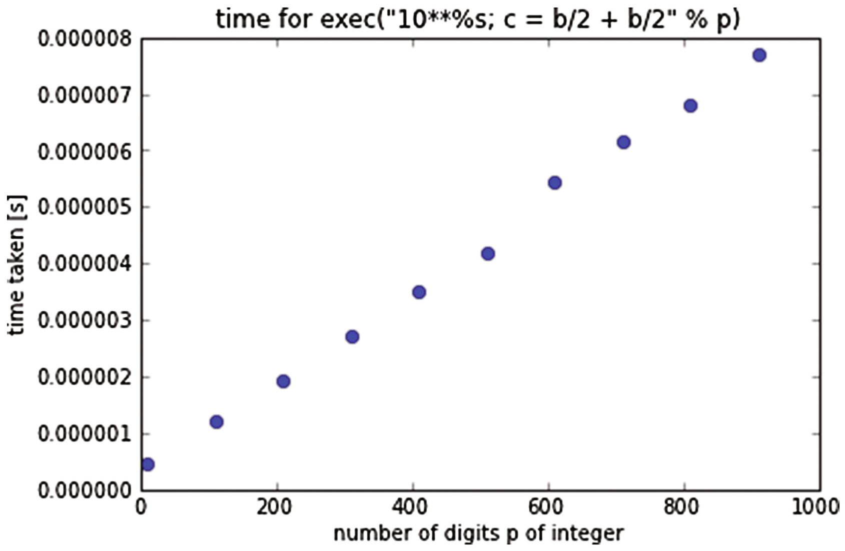


Fig. 3. Integer operations performance in Python. Reprinted from Ref. [8] with permission.

as the two values are two far apart to have a relation. However if R is 64 bits (or less) long, we can use normal hardware accelerated arithmetics to calculate their closeness to the Center Point.

3.3 Hostnames and Usernames

Hostname and username attributes are diverse by nature. Small differences in usernames can mean they cannot be clustered together for most meaningful analysis. In contracts, subnets in IP addresses provide a good base for event clustering. In order to map these fields for high performing comparisons, we convert them to 64 bit unsigned integer representations. However one caveat exists. Given that hostnames/usernames can be longer than the 8 characters (ASCII) – which a single 64 bit CPU register can hold –, a fixed length function would have to be used. In our experiments we used *SipHash* [4] to convert longer than 64 bit input to a fixed 64 bit output. Other hash functions such as CityHash.64 [9] (non-cryptographic) and Farmhash64 [10] can also be used with small differences in performance vs. accuracy (collision resistance) indicators.

3.4 Port Numbers

Mapping port numbers to representative values is somewhat more difficult. Since port numbers are at most 16 bits long, they require no compression to enable atomic operations on modern CPUs. However they also provide very little substantive information. In our approach we limit port number analysis to all well known ports. This approach may be flawed for some calculations where non-standard ports are used but is the most reliable way to map the meaning behind a port number barring more intrusive service detection mechanisms. Instead of mapping the port numbers directly, we can map their classifications instead. The meaning behind the ports is categorized based on the *type* of service commonly associated with that port. In [13], the author presents an approach for TCP/IP traffic classification based on port numbers. For example, port 80 and 443 could be bundled into a single group of ports titled *WWW*. Table 1 contains (not exhaustive) some of the mappings used in this process.

Table 1. IP traffic classification using port numbers

Classification	IP port numbers				
Web	80	443	8080	8443	
VPN	1194	1723	500	4500	
FTP	20	21	990		
Mail	25	110	993	587	465
Management	22				
Torrent	6881–6999				
Version control	9418	3690	2401		
VoIP	5060	5061			

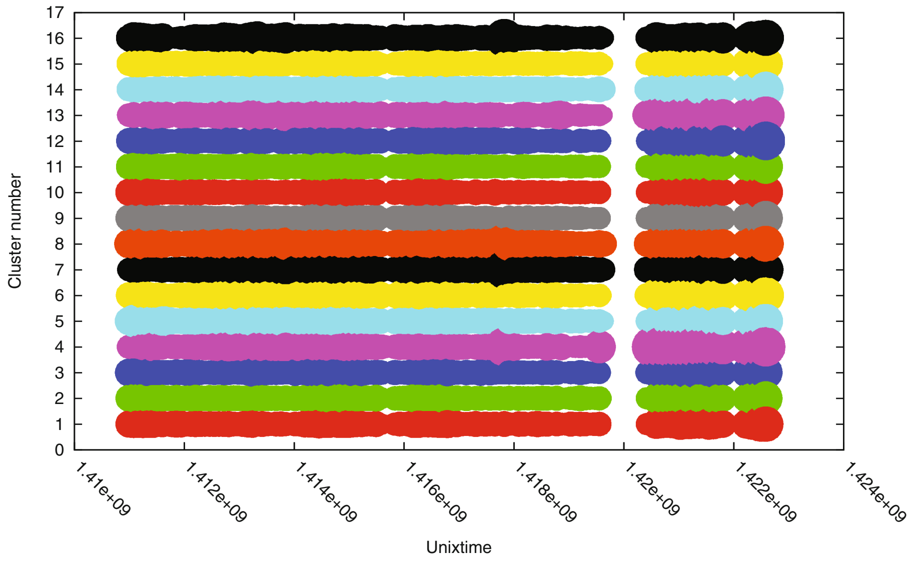
4 Results

In this section we present the results of active mappings applied to the Internet Protocol Address (v4) values after running them through k-means. The dataset used consists of over 30 million events and contains over $170,000$ distinct IP addresses. Through analyzing and experimenting with the dataset, an optimal number of clusters were chosen. Figures 4a and b show the distribution of events from the dataset within the k-means clusters, with and without mapping.

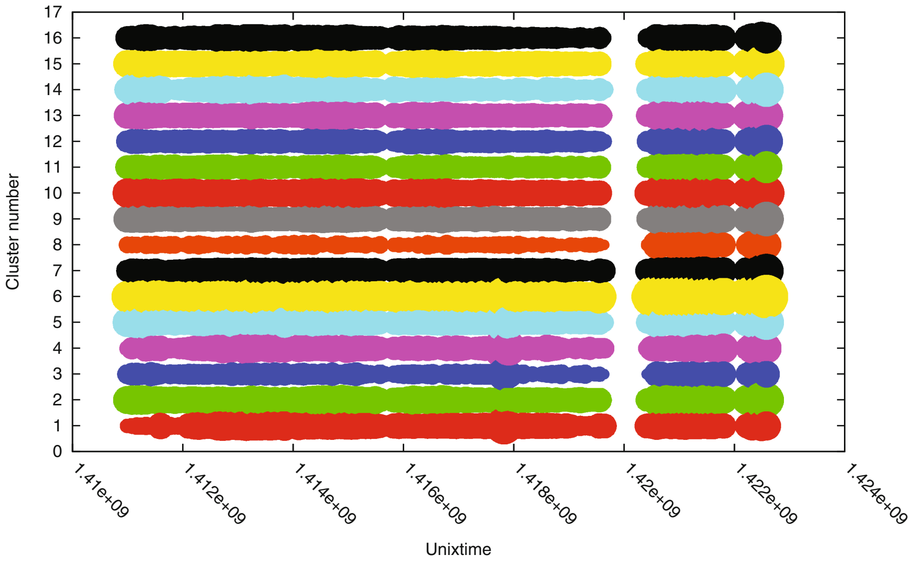
We observe that the distributions have changed in a number of the clusters. Looking at clusters 1, 3 and 16, we observe large changes at $1.418e + 09$ (Mon, 08 Dec 2014 00:53:20 GMT).

Figures 5a and b show the results of running k-means on the same dataset with the first octet of the IP addresses as the X axis against the cluster number on the Y axis. As a result of the mappings, we observe a far more accurate representation of the dataset analyzed by k-means.

Additionally, we observe more density in the manner in which the clusters are formed against the X axis. By overlaying the findings against the realities in the dataset, we find a close relationship between the clusters and the subnets within the network. The number on center points in k-means effects the accuracy. The closer the number of center points to the number of subnets used, the more accurate the output of the algorithm.

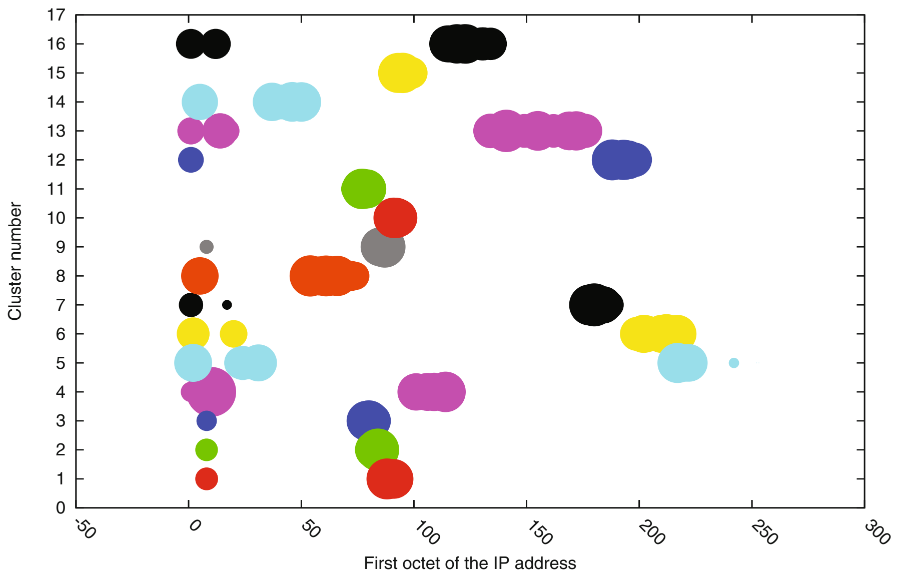


(a) Without IP Mapping

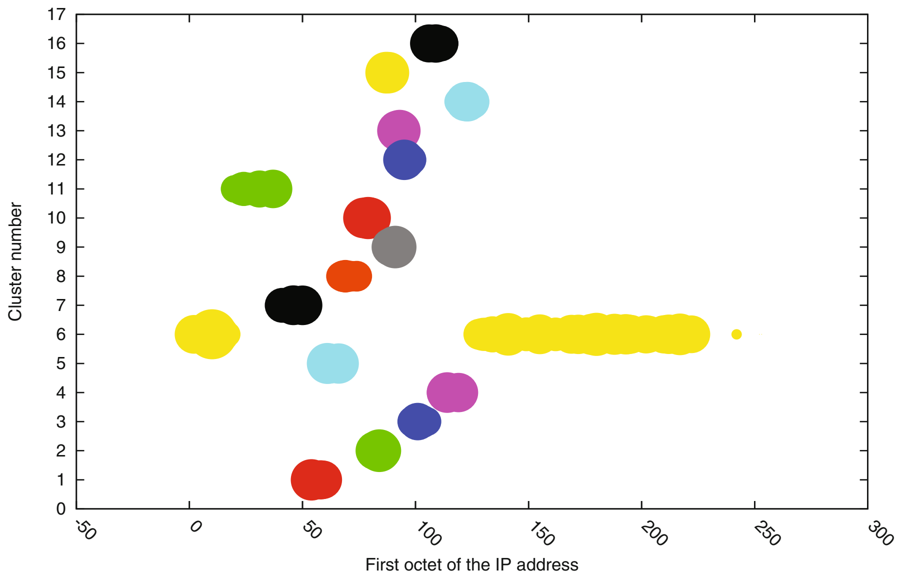


(b) With IP Mapping

Fig. 4. Distribution of events through time



(a) Pre IP Mapping



(b) Post IP Mapping

Fig. 5. k-means clusters pre and post IP mapping

5 Conclusion

By analyzing the results of the k-means algorithm (run on the dataset with the proposed mappings) we find that the IP addresses are clustered far more accurately. While the same algorithm run on a dataset without the mappings was flawed with inaccurate (meaningless) clusters, the proposed changes resulted in clusters which greatly reflected different IP subnets and the distances between the data points were accurately reflecting the distance between the original IP addresses (and subnets) and not the distance between the simple unique integer representation of the IP addresses.

6 Future Work

Given that the mappings proposed in this paper are specific to network environments, we would consider the same types of mappings applied to host information. Mappings that retain the essence of the information being mapped as well as its relation to other relevant information could lead to highly accurate machine learning algorithms with improved detection and statistical capabilities in regards to computer networks and systems. Such systems could aid greatly in the discovery of anomalies and patterns that could lead to more efficient and accurate networks systems.

References

1. Cheng, F., Meinel, C., Azodi, A., Jaeger, D.: A new approach to building a multi-tier direct access knowledgebase for ids/siem systems. In: Proceedings of the 11th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC2013), Chengdu, China, 12 2013. IEEE CS (2013)
2. Cheng, F., Meinel, C., Azodi, A., Jaeger, D.: Pushing the limits in event normalisation to improve attack detection in ids/siem systems. In: Proceedings of the 1st International Conference on Advanced Cloud and Big Data, Nanjing, China, 12 2013. IEEE CS (2013)
3. Azodi, A., Gawron, M., Cheng, F., Meinel, C., Sapegin, A., Jaeger, D.: Hierarchical object log format for normalization of security events. In: Proceedings of the 9th International Conference on Information Assurance and Security (IAS 2013), Tunis, Tunisia, 12 2013. IEEE CS (2013)
4. Aumasson, J.-P., Bernstein, D.J.: Siphash: a fast short-input prf, Jan 2015. https://131002.net/siphash/
5. Brink, H., Richards, J.: Real-world machine learning. In: MEAP, pp. 1–22 (2014)
6. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. **2**(2), 121–167 (1998)
7. Consul, P.C., Famoye, F.: Generalized poisson distribution. In: Lagrangian Probability Distributions, pp. 165–190 (2006)
8. Fangohr, H.: Performance of python's long data type, Jan 2013. http://www.southampton.ac.uk/~fangohr/blog/performance-of-pythons-long-data-type.html
9. Google Inc. Cityhash provides hash functions for strings, Jan 2010. https://code.google.com/p/cityhash/

10. Google Inc. The farmhash family of hash functions, Jan 2015. https://code.google.com/p/farmhash/
11. Hartigan, J.A., Wong, M.A.: Algorithm as 136: a k-means clustering algorithm. Appl. Stat. **28**, 100–108 (1979)
12. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, vol. 501. Wiley, New York (2009)
13. Schneider, P.: Tcp/ip traffic classification based on port numbers. Division of Applied Sciences, Cambridge, MA, 2138 (1996)
14. Schreiber, T.: A voronoi diagram based adaptive k-means-type clustering algorithm for multidimensional weighted data. In: Bieri, H., Noltemeier, H. (eds.) CG-WS 1991. LNCS, vol. 553, pp. 265–275. Springer, Heidelberg (1991)

Graph- and Cloud-Based Tools for Computer Science Education

Victor N. Kasyanov[✉] and Elena V. Kasyanova

Institute of Informatics Systems, Novosibirsk State University,
Novosibirsk 630090, Russia
kvn@iis.nsk.su

Abstract. In the paper, the graph- and cloud-based tools being under development at Institute of Informatics Systems in Novosibirsk for supporting of computer science education are presented. The Web-systems Wiki GRAPP and WEGA intended to help in teaching and research in graph theory, graph algorithms and their applications to computer science are described. The visual systems Higres and Visual Graph for graphs and graph algorithms are presented. The cloud system CSS aimed at supporting of functional and parallel programming is considered.

Keywords: Cloud computing · Computer science education · Graphs · Graph algorithms · Information visualization · Parallel programming · Web-systems

1 Introduction

Graph is the most common "abstract" structure encountered in computer science. Any system that consists of discrete states (or sites) and connections between them can be modeled by a graph. Graphs are used almost everywhere in computer science and are routinely included in many courses in undergraduate and graduate programs.

Intricate nature of software systems and computer education systems can, and in our opinion should, be represented by graph models. Their wide applicability is due to the fact that graphs are a very natural way of explaining complex situations on an intuitive level. We are entirely convinced the future is "visual", and the graph models are the best formalism for visual presentation of information of complex and intricate nature. The visualization of conceptual structure is a key component of support tools for complex applications in science and engineering [1–3]. Moreover, the information that is interested us in computer science and computer education is nonquantitative, but rather, of structural and relational nature. For this reason many graph visualization systems, graph editors and libraries of graph algorithms have been developed in recent years. Examples of these tools include VCG, daVinci, Graphlet, GLT&GET, yEd and aiSee.

Graph drawing is a useful way of representation of graph models, and visualization of graphs is used in many applications for the design and analysis of communication networks, related documents, as well as static and dynamic structures of programs. However, systems of related objects frequently are dynamic. For example, relations between objects or properties of objects can be changed. If transformation processes

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 41–54, 2015.
DOI: 10.1007/978-3-319-25744-0_4

can be formalized and presented in the algorithmic form then it is useful to create a graphical representation of transformations. Animating an algorithm allows for better understanding of the inner workings of the algorithm, furthermore it makes apparent its deficiencies and advantages thus allowing for further optimization [4].

The size of the graph model to view is a key issue in graph visualization [2, 3]. Large graphs pose several difficult problems. If the number of graph elements is large it can compromise performance or even reach the limits of the viewing platform. Even if it is possible to layout and display all the graph elements, the issue of viewability or usability arises, because it will become impossible to discern between nodes and edges of graph model.

Academic research and engineering challenge both require high performance computing, which can be achieved through parallel programming. The existing curricula of most universities do not properly address the major transition from single-core to multicore systems and sequential to parallel programming. They focus on applying application program interface (API) libraries and open multiprocessing (OpenMP), message passing interface (MPI), and compute unified device architecture (CUDA)/GPU techniques. This approach misses the goal of developing students' long-term ability to solve real-life problems by "thinking in parallel".

We see that the history of computing has shown shifts from explicit to implicit programming. In the early days, computers were programmed in assembly language, mostly with the purpose of utilizing the available memory space as effectively as possible. This came at the cost of obscure, machine-dependent, hard to maintain programs, which were designed with high programming effort. High-level languages were introduced to make programming more implicit, portable and less machine-dependent. With the advent of massively parallel computers and their promise of hundreds of gigaflops, we have seen a return to the explicit programming paradigm. Using these languages with explicit message passing library routines as "machine languages", people attempt to utilize the available processing power to the largest extent, again at the cost of high programming effort, machine-dependent, and hard to maintain code. A compiler for an implicitly parallel programming language alleviates the programmer from the task of partitioning program and data over the massively parallel machine.

Functional programming [5] is a programming paradigm, which is entirely different from the conventional model: a functional program can be recursively defined as a composition of functions where each function can itself be another composition of functions or a primitive operator (such as arithmetic operators, etc.). The programmer need not be concerned with explicit specification of parallel processes since independent functions are activated by the predecessor functions and the data dependencies of the program. This also means that control can be distributed. Further, no central memory system is inherent to the model since data is not "written" in by any instruction but is "passed from" one function to the next.

In the paper, we describe the graph- and cloud-based tools being under development at Institute of Informatics Systems and aimed at supporting of computer science education. The Web-systems Wiki GRAPP and WEGA intended to help in teaching and research in graph theory, graph algorithms and their applications to computer science are described. The visual systems Higres and Visual Graph for graph algorithms and graph models are presented. The CSS system based on hierarchical graph representations of functional and parallel programs and aimed at supporting of cloud supercomputing is considered.

2 Systems for Supporting of Graph Models and Methods

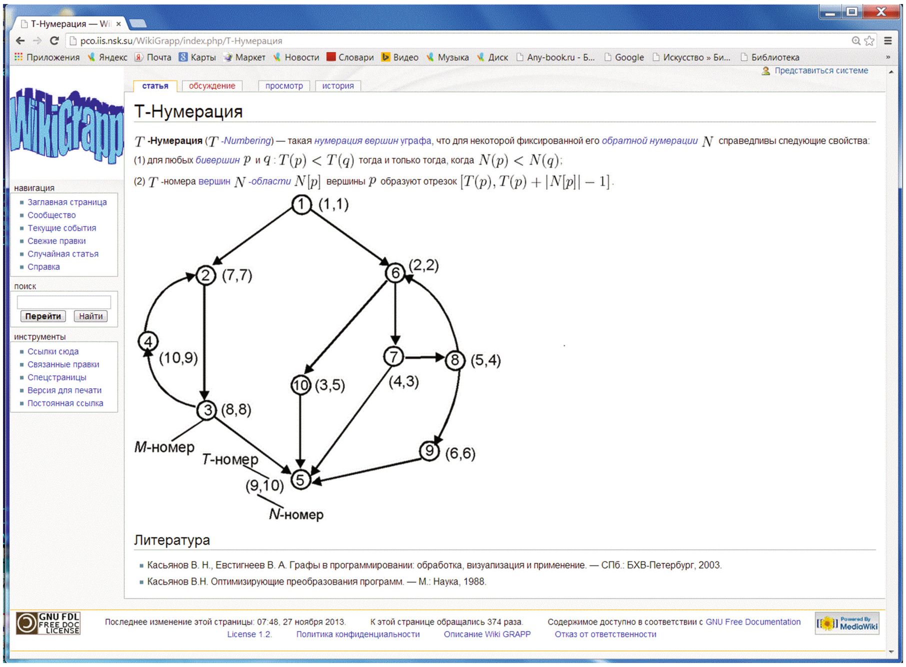
2.1 The Wiki GRAPP System

The problem of terminology is one of the main problems in application of graph methods to programming and computer science.

In 1999 our dictionary [6] was published, which covered main graph-related terms from monographs in Russian. It was the first dictionary of graphs in computing and it aroused a great interest of readers.

Our new dictionary [7] is an extended dictionary of 1999 and it includes more than 1000 new terms from journal articles whose abstracts were published in Abstract Journal "Mathematics" in section "Graph Theory", as well as from volumes of annual conferences "Graph-Theoretic Concepts in Computer Science" and book series "Graph Theory Notes of New York". The dictionary contains more than 2500 graph-related terms with their clear and succinct definitions.

We are developing the Wiki GRAPP system [8] as an on-line "edition" of the dictionaries of graph-theory and its applications to computer science and programming (See Fig. 1).



Т-Нумерация

T **-Нумерация** (T *-Numbering*) — такая *нумерация вершин* уграфа, что для некоторой фиксированной его *обратной нумерации* N справедливы следующие свойства:

(1) для любых *бивершин* p и q : $T(p) < T(q)$ тогда и только тогда, когда $N(p) < N(q)$;

(2) T -номера вершин N *-области* $N[p]$ вершины p образуют отрезок $[T(p), T(p) + |N[p]| - 1]$.

Литература

- Касьянов В. Н., Евстигнеев В. А. Графы в программировании: обработка, визуализация и применение. — СПб.: БХВ-Петербург, 2003.
- Касьянов В.Н. Оптимизирующие преобразования программ. — М.: Наука, 1988.

Последнее изменение этой страницы: 07:48, 27 ноября 2013. К этой странице обращались 374 раза. Содержимое доступно в соответствии с GNU Free Documentation License 1.2. Политика конфиденциальности Описание Wiki GRAPP Отказ от ответственности

Fig. 1. The Wiki GRAPP system.

Like Wikipedia the Wiki GRAPP system runs on the MediaWiki system, but now only certain editors are able to modify it. At present it includes definitions of about 4500 terms and covers all terms from our dictionaries.

2.2 The WEGA System

The Web-Encyclopedia of Graph Algorithms (WEGA) [8] being under development on the basis of the books [9, 10] and the MediaWiki system is aimed to be not only a reference manual on graph algorithms but also an introduction to the graph theory and its applications to computer science.

In contrast to Donald Knuth who used the assembly language of the so-called MIX computer in his fundamental books "The art of computer programming", we decided to use a high-level and language-independent representation of graph algorithms in our books and system (See Fig. 2).

In our view, such an approach is preferential, as it allows us to describe algorithms in a form that admits direct analysis of their correctness and complexity, as well as a simple translation of algorithms to high-level programming languages without disturbance of their correctness and complexity. Besides, the described approach allows the readers to understand an algorithm at the informative level, to evaluate its applicability to a specific problem, and to make all its modifications needed for correct application of the algorithm.

We also believe that visualization could be very helpful for readers in understanding graph algorithms, and we are embedding capabilities of interactive animation of graph algorithms into the WEGA system.

For constructing of graph algorithm visualizations we use the Higres system (see below) and a subsystem ALVIS.

The ALVIS system has been created on the base of a model of interactive visualization of graph algorithms, providing the capability to build the algorithm visualization with the help of a flexible system of visual effects and using the algorithm as an input parameter [11].

The ALVIS system implements visualization in following two steps.

First, the algorithm text is transformed into a program ready for execution; after that the program is executed with the given graph as a parameter. The result is a log of items, each of which contains information about changes in the graph model state.

Second, the visualizer receives the input graph, the original algorithm text, the log of execution and visual effects settings. As a result, the visualization system works out a sequence of images corresponding to the graph model of intermediate states of the algorithm.

2.3 The Higres System

The Higres system was developed as a visualization tool and an editor for graphs and as a platform for execution and animation of graph algorithms implemented via so-called external modules [12].

Problem

Objects. A directed graph G. Each vertex of this graph is either labelled or unlabelled.

Operations. For any vertex p of the graph G, the procedure LABEL (p) is executable if p is unlabelled and its execution turns p into a labelled vertex; the predicate UNLABELLED(p) is true if p is unlabelled.

Given. A vertex p_0 such that any vertex of the graph G reachable from p_0 is unlabelled.

Required. Label all vertices of the graph G reachable from p_0.

Remark. It is assumed that the traversal of the graph is undertaken to perform certain operations over the data content of the vertex p and that these operations are performed in the course of the execution of the procedure LABEL (p).

Solution

```
procedure FOREST($p_0$: vertex) =
    $S$: set of arcs = $\varnothing$;
    $q$: vertex = $p_0$;
    $L$: begin LABEL($q$);
              $S$ <= OUTARCS($q$);
              while $S \neq \varnothing$ do
                  $q$ :=TAIL(any from $S$);
                  if UNLABELLED($q$) then repeat $L$ end
              end
         end
end.
```

Properties

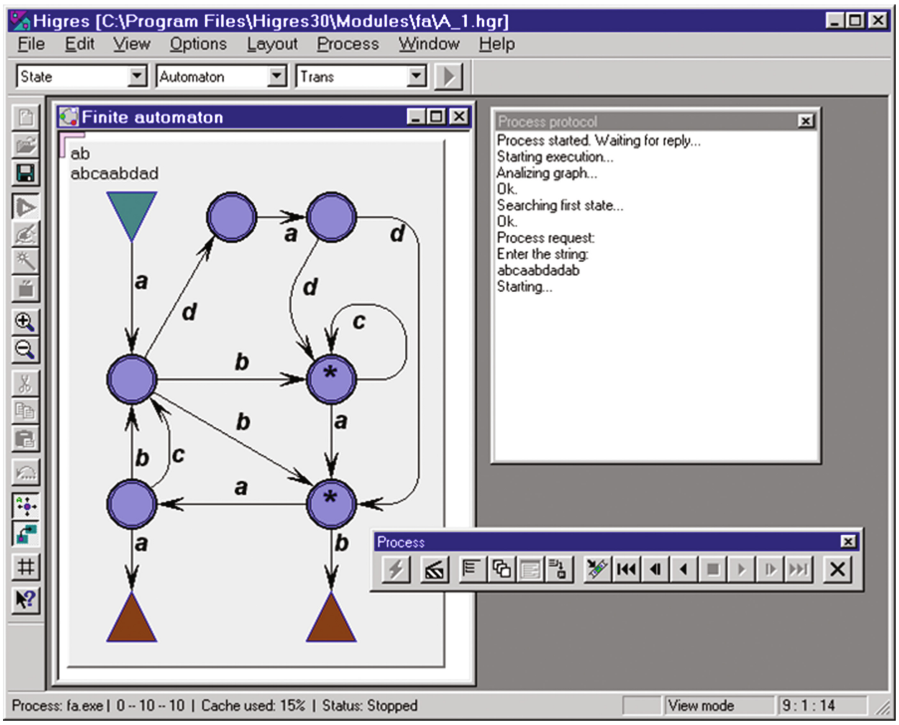
1. The time of the algorithm is O(k), where k is the number of arcs of the graph G reachable from the vertex p_0

2. The algorithm turns the state of an unlabelled vertex q if and only if it is reachable from p.

3. If S is a stack, then the procedure labels the vertices of the graph G reachable from p_0 in the order of depth-first search and if S is a queue, then the vertices are labelled in the order of breadth-first search.

Fig. 2. General algorithm for arc-based traversals of graph

The Higres system can be used for the run-time visualization of graph algorithm (See Fig. 3). It also caches samples for the repeated and backward animation. A set of parameters can be defined inside a module. These parameters can be changed by the user at any execution step. The module can ask user to input strings and numbers. It can also send any textual information to the protocol that is shown in the process window.



Higres [C:\Program Files\Higres30\Modules\fa\A_1.hgr]

State | Automaton | Trans

ab
abcaabdad

Process protocol

```
Process started. Waiting for reply...
Starting execution...
Analizing graph...
Ok.
Searching first state...
Ok.
Process request:
Enter the string:
abcaabdadab
Starting...
```

Process: fa.exe | 0 -- 10 -- 10 | Cache used: 15% | Status: Stopped | View mode | 9 : 1 : 14

Fig. 3. The Higres system: finite automaton emulation.

The animation feature can be used for algorithm debugging, educational purposes and exploration of iteration processes such as force methods in graph drawing.

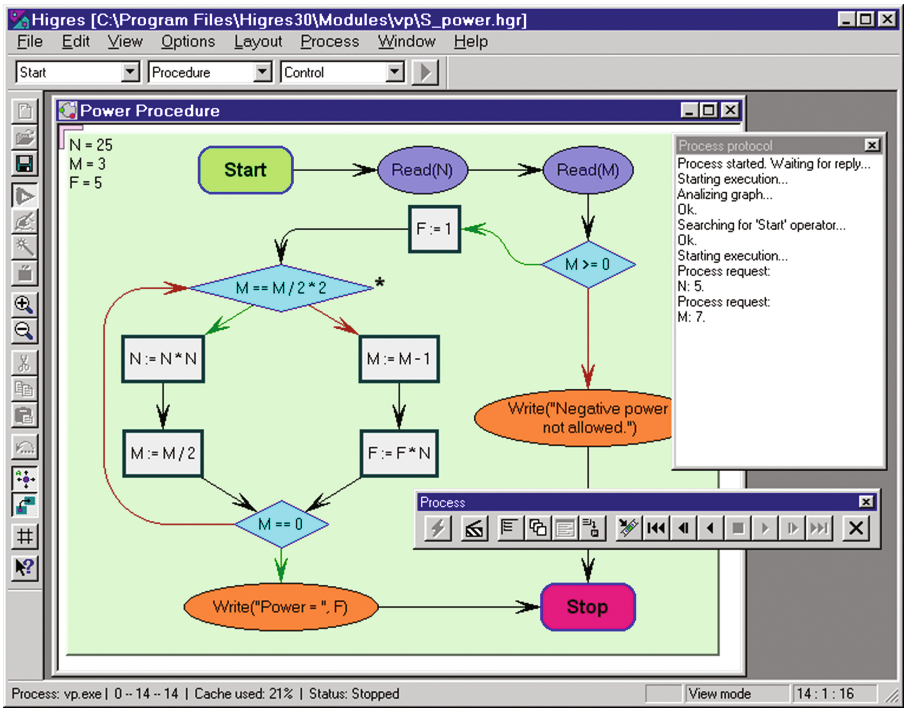
We provide a special API that can be used to create external modules. The API includes functions for graph modification and functions that provide interaction with the Higres system. It is unnecessary for programmer, who uses this API, to know the details of the internal representation of graphs and system/module communication interface. Hence, the creation of new modules is a rather simple work.

At present we have external modules for animation of various kinds of graph algorithms: from graph layout methods, that deal only with drawing of a graph in a plane, to complex semantic processing of graph models such as Petri net simulation or block-scheme program execution (See Fig. 4).

2.4 The Visual Graph System

The Visual Graph system is intended to visualize and explore large and complex abstract forms of information represented by large graphs, e.g. graphs automatically generated by optimizing compilers or other applications.

The system reads a textual and human-readable GraphML-specification [13] of an attributed hierarchical graph [11, 14] and automatically calculates its customizable



```
Process started. Waiting for reply...
Starting execution...
Analizing graph...
Ok.
Searching for 'Start' operator...
Ok.
Starting execution...
Process request:
N: 5.
Process request:
M: 7.
```

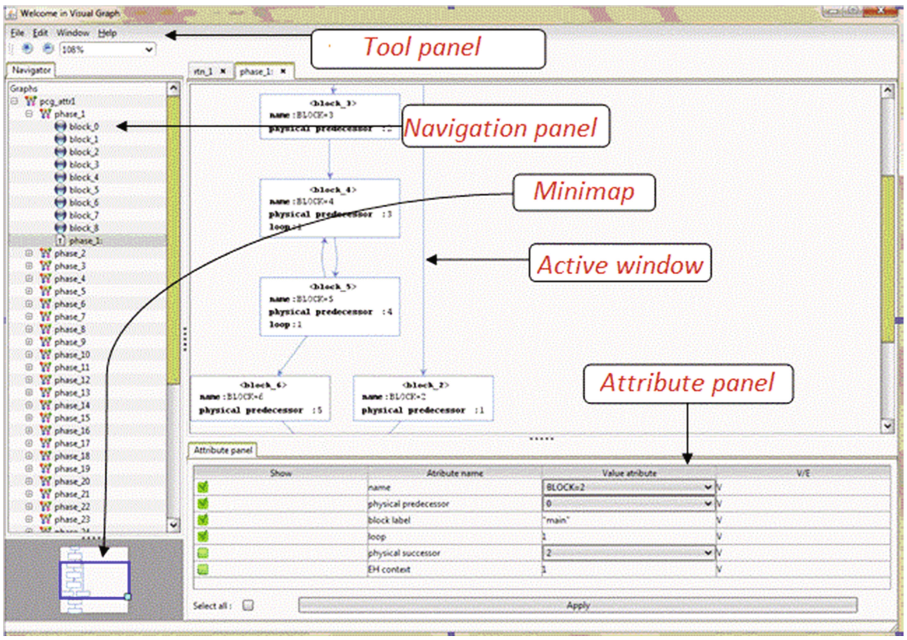
Fig. 4. The Higres system: block-scheme program execution.

multi-aspect layout (drawing). This layout is then displayed, and can be interactively explored, extended and analyzed by user.

Visual Graph was designed to visualize and explore large graphs that consist of many hundreds of thousands of elements. In order to reduce time for work with a graph, it is possible to avoid computing the layout of its parts that are currently not of interest. Interactive exploring of a graph is based on step by step construction of its multi-aspect layout being a set of drawings of some its subgraphs. For presentation of multi-aspects layout a set of windows which includes a separate window for visualization of each considered subgraph is used. At each step of layout algorithm it is applied to a subgraph being interested to user at this step. Multi-aspect layout of a graph makes every visible part of the graph smaller, thus enabling the layout to be calculated faster and the quality of the layout to be improved.

Visual Graph offers several tools for navigating through a graph model: minimap, navigator, attribute panel, filter, search panel, notebook (See Fig. 5). It provides also tools for analyzing graph structures. This analysis means solving advanced questions that relate to a graph structure, for instance, determining for a graph all its beconnected components or dominance relation between its nodes.

Simple possibilities to extend the functionality of Visual Graph (e. g. to add a new layout, search, analysis or navigating algorithm, a new tool for processing information associated with elements of graph models and so on) are provided.



Show	Atribute name	Value atribute	V/E
✓	name	BLOCK=2	V
✓	physical predecessor	0	V
✓	block label	"main"	V
✓	loop	1	V
☐	physical successor	2	V
☐	EH context	1	V

Select all : ☐ Apply

Fig. 5. The Visual Graph system.

3 Cloud System for Supporting of Functional and Parallel Programming

3.1 The CSS System

The advancement of computer technology and the increasing complexity of research problems are creating the need to teach parallel programming in higher education more effectively. Programming massively-parallel machine is a daunting task for any human programmer and parallelization may even be impossible for any compiler. Instead, the functional programming paradigm may prove to be an ideal solution by providing an implicitly parallel interface to the programmer.

The CSS system is based on hierarchical graph representations of functional and parallel programs and is intended to provide a general-purpose user interface for a wide range of parallel processing platforms (See Fig. 6). In our conception, cloud interface gives transparent ability to execute programs on arbitrary environment. JavaScript client does not demand installation; small educational programs can be executed on client devices (computers or smart phones). V8 server allows the language parser and some optimizations to be used at both client and server sides.

The CSS system uses a functional language Cloud Sisal and includes five big parts: interface, interpreter, graphic visualization/debugging subsystem, optimizing cross compiler, cluster runtime. The interpreter is available on web via browser; it translates Cloud-Sisal-program to the first internal representation (IR1) and runs it without

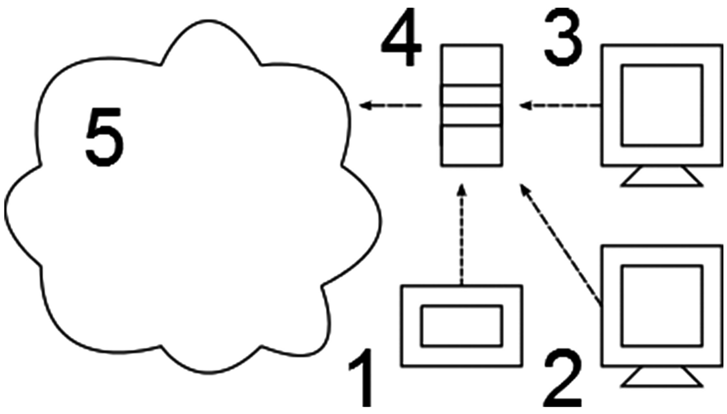


Fig. 6. Cloud service structure: 1, 2 and 3 – clients, 4 – cloud access server, 5 – execution environment

making actual low-level code. It is useful because in this case a user can get any debugging information in visual forms of hierarchical graphs [11, 14].

Web interface also contain some usual parts like syntax highlighting, persistent storage for program code, authorization and so on.

The CSS system provides means to write and debug Cloud-Sisal-programs on low-cost devices as well as to translate and execute them in clouds. The CSS system can open the world of parallel and functional programming to all students and scientists without requiring a large investment in new, top-end computer systems.

3.2 Cloud Sisal

Functional language Sisal (Steams and Iterations in a Single Assignment Language) is considered as an alternative to FORTRAN language for supercomputers [15, 16]. Compared with imperative languages (like FORTRAN), functional languages, such as Sisal, simplifies programmer's work. He has only to specify a result of calculations and it is a compiler that is responsible for mapping an algorithm to certain calculator architecture. In contrast with other functional languages, Sisal supports data types and operators typical for scientific calculations (See Fig. 7).

At present, there are implementations of the Sisal 1.2 language [17] for many supercomputers (e. g., SGI, Sequent, Encore Multimax, Cray X-MP, Cray 2, etc.). The Sisal 90 language definition [18] increases the language's utility for scientific programming. It includes language level support for complex values, array and vector operations, higher order functions, rectangular arrays, and an explicit interface to other languages like FORTRAN and C.

The Sisal 3.2 language [19, 20] is a version of Sisal which integrates features of Sisal 90 and Sisal 2.0 [21] and includes language level support for module design, mixed language programming, preprocessing. The Sisal 3.2 language supports also so-called annotated programming [22] and includes optimizing annotations in the form

```
type OneDim = array[ double_real ];
type TwoDim = array[ OneDim ];
function Main (A,B: TwoDim;
               M, N, L : integer
               returns TwoDim)
     for I in 1, M cross J in 1, L
         S := for K in 1, N
                      R := A[I, K] * B[K, J]
                      returns value of sum R
              end for
         returns array of S
     end for

  end function
```

Fig. 7. Sisal-program for matrix multiplication.

of formalized comments being predicate constraints on admissible properties of program fragments or states of computations.

The Cloud Sisal language that has been designed as the input language of the CSS system is based on the Sisal 3.2 and increases the language's utility for supporting of cloud scientific computations and cloud parallel programming.

3.3 Internal Representations

The CSS system uses three internal representations of Cloud-Sisal-programs: IR1, IR2 and IR3.

IR1 is a language of hierarchical graphs [11, 14] made up simple and compound computation nodes, edges, ports and types (See Fig. 8). Nodes correspond to computations. Simple nodes are vertices and denote operations such as add or divide. Compound nodes are subgraphs and represent compound constructions such as structured expressions and loops. Ports are vertices that are used for input values and results of compound nodes. Edges show the transmission of data between simple nodes and ports; types are associated with the data transmitted on edges. IR1-program represents data dependencies, with control left implicit; e. g. iteration is represented as a compound node with subgraphs describing generation of index values, the body of the loop, and the packaging of results.

IR2 is an extension of IR1 but is not applicative. It introduces operations that explicitly allocate and manipulate memory and also introduces a new class of operations, which are similar to IR1 nodes except that they are told where in memory to construct their results. Also, artificial dependence edges are added to define additional synchronization constraints where they may be useful. Finally, data edges can be decorated with pragmas to specify access rights to the data they transmit and to allow operations to modify their inputs.

IR3 is a classical three-address code representation with hierarchical blocks.

```
function foo(A: array[integer];
             k: integer;
             N: integer
             returns integer)
  let s := 0 in
      for i in 1,100 repeat
          s := old s + k * A[N]
          returns value of s
      end for
  end let
  end function
```

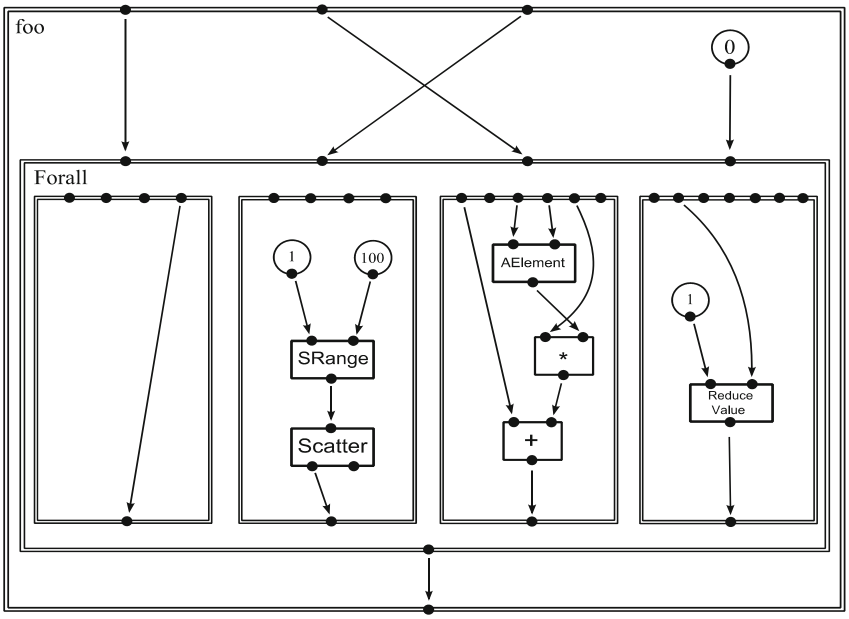


Fig. 8. A function *foo* and its IR1-representation.

3.4 Compiler

The compiler of the CSS system consists of two main parts: front-end and back-end compilers (See Fig. 9).

The front-end compiler translates Cloud-Sisal-modules into a monolithic IR1-program which is used also by the interpreter and the graphic visualization/debugging subsystem.

The back-end compiler begins with R2Gen which produces a semantically equivalent program in IR2. Then the IR2Opt subsystem performs many machine-independent optimizations on the monolith to produce a semantically equivalent, but faster program. After completion of the machine-independent optimizations,

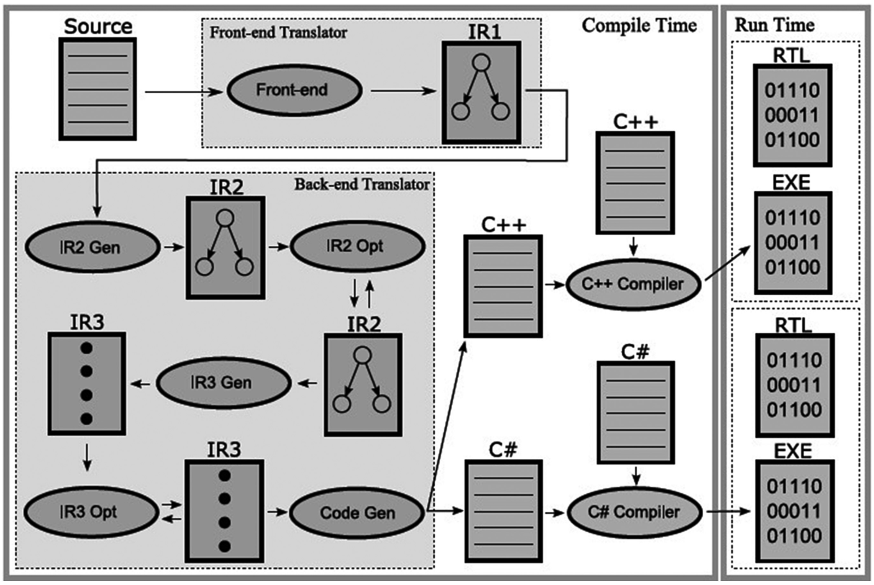


Fig. 9. The Cloud-Sisal-compiler and run time support.

the IR3Gen subsystem preallocates array storage where compile time analysis or compiler generated expressions executed at run time can calculate the final size of an array. The result of this phase is the production of a semantically equivalent program in IR3. The next phase (IR3Opt) performs update-in-place analysis and restructures some graphs to help identify at compile tune those operations that can execute in-place and to improve chances for in-place operation at run time when analysis fails. It performs also some machine-dependent optimizations and defines the desired granularity of parallelism based on an estimate of computational cost and various parameters that tune analysis. After parallelization, CodeGen generates C ++ or C# code, and the compilation can be completed using the target machine's C ++ or C# compiler.

The current target platform for the Cloud-Sisal-compiler is .NET. The compiler generates the C# code. It allows the users to perform the experimental execution of Cloud-Sisal-programs and examine the effectiveness of optimizing transformations applied by the compiler.

4 Conclusion

The graph- and cloud-based tools being under development at Institute of Informatics Systems in Novosibirsk for supporting of computer science education were presented.

The Web-systems WikiGRAPP and WEGA intended to help in teaching and research in graph theory, graph algorithms and their applications to computer science were described. The WikiGRAPP system is a wiki-dictionary of graphs in computer science. The WEGA system is a wiki-encyclopedia of graph algorithms which is aimed to be not only a reference manual on graph algorithms but also an introduction to the graph theory and its applications to computer science.

The visual systems Higres and Visual Graph for graph models and graph algorithms were presented. The Higres system is a graph editor and a platform for execution and animation of graph algorithms. The Visual Graph system is intended to visualize and explore large and complex abstract forms of information represented by large attributed hierarchical graphs automatically generated by compilers and other applications.

The cloud system CSS aimed at supporting of functional and parallel programming for scientific computations was considered. The CSS system provides means to write and debug functional programs regardless target architectures on low-cost devices as well as to translate them into optimized parallel programs, appropriate to the target execution platforms, and then execute on high performance parallel computers without extensive rewriting and debugging.

Acknowledgement. The authors are thankful to all colleagues taking part in the projects described. This research was supported in part by the Russian Foundation for Basic Research under grant RFBR 15-07-02029.

References

1. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, New Jersey (1999)
2. Herman, I., Melançon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: a survey. IEEE Trans. Vis. Comput. Graph. **6**, 24–43 (2000)
3. Kasyanov, V.N., Kasyanova, E.V.: Visualization of Graphs and Graph Models. Siberian Scientific Publishing House, Novosibirsk (2010). (in Russian)
4. Kerren, A., Stasko, J.T.: Algorithm animation. In: Diehl, S. (ed.) Software Visualization. LNCS, vol. 2269, pp. 1–15. Springer, Heidelberg (2002)
5. Backus, J.: Can programming be liberated from the von Neumann style? Commun. ACM **21** (8), 613–641 (1978)
6. Evstigneev, V.N., Kasyanov, V.N.: Explanatory Dictionary of Graph Theory in Computer Science and Programming. Nauka Publ., Novosibirsk (1999). (in Russian)
7. Evstigneev, V.N., Kasyanov, V.N.: Dictionary of Graphs in Computer Science. Siberian Scientific Publishing House, Novosibirsk (2009)
8. Kasyanov, V.N.: Support tools for graphs in computer science, In: Proceedings of the 15th ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2010), p. 315. ACM Press, New York (2010)
9. Kasyanov, V.N., Evstigneev, V.N.: Graph Theory for Programmers: Algorithms for Processing Trees. Kluwer Academic Publishers, Dordrecht/Boston/London (2000)
10. Kasyanov, V.N., Evstigneev, V.N.: Graphs in Programming: Processing, Visualization and Application, BHV-Petersburg, St. Petersburg (2003). (In Russian)
11. Kasyanov, V.N.: Methods and tools for structural information visualization. WSEAS Trans. Comput. **12**(7), 349–359 (2013)
12. Lisitsyn, I.A., Kasyanov, V.N.: Higres - visualization system for clustered graphs and graph algorithms. In: Kratochvíl, J. (ed.) GD 1999. LNCS, vol. 1731, pp. 82–89. Springer, Heidelberg (1999)

13. Brandes, U., Marshall, M.S., North, S.C.: Graph data format workshop report. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 407–409. Springer, Heidelberg (2001)
14. Kasyanov, V.N., Kasyanova, E.V.: Information visualization based on graph models. Enterp. Inf. Syst. **7**(2), 187–197 (2013)
15. Cann, D.C.: Retire Fortran?: a debate rekindled. Commun. ACM **34**(8), 81–89 (1992)
16. Gaudiot, J.-L., DeBoni, T., Feo, J., et al.: The Sisal project: real world functional programming. In: Pande, S., Agrawal, D.P. (eds.) Compiler Optimizations for Scalable Parallel Systems. LNCS, vol. 1808. Springer, Heidelberg (2001)
17. McGraw, J., Skedzielewski, S., Allan, S., Grit, D., Oldehoeft, R., Glauert, J., Dobes, I., Hohensee, P.: SISAL-Streams and Iterations in a Single Assignment Language, Language Reference Manual: Version 1.2. Technical Report TR M-146, University of California, Lawrence Livermore Laboratory, March (1985)
18. Feo, J.T., Miller, P.J., Skedzielewski, S.K., Denton, S. M., Solomon, C. J.: SISAL 90. In: Proceedings of High Performance Functional Computing. pp. 35–47, Denver (1995)
19. Kasyanov, V.N., Stasenko, A.P.: Sisal 3.2 programming language. In: Kasyanov, V.N. (ed.) Tools and Techniques of Program Construction, pp. 56–134, IIS, Novosibirsk (2007) (in Russian)
20. Kasyanov, V.N.: Sisal 3.2: functional language for scientific parallel programming. Enterp. Inf. Syst. **7**(2), 227–236 (2013)
21. Cann, D.C., Feo, J.T., Böhm, A.P.W., et al.: Sisal Reference Manual: Language Version 2.0. Technical report Lawrence Livermore National Laboratory, UCRL-MA-109098, Livermore, CA (1991)
22. Kasyanov, V.N.: Transformational approach to program concretization. Theor. Comput. Sci. **90**(1), 37–46 (1991)

ERP Costumer Assistance Using Ubiquitous Agents

Nardjes Bouchemal[1,2(✉)], Ramdane Maamri[2], and Zaidi Sahnoun[2]

[1] University Center of Mila, Mila, Algeria
[2] LIRE Laboratory of Constantine2, Constantine, Algeria
{nardjes.bouchemal, rmaamri}@yahoo.fr

Abstract. Enterprise Resource Planning (ERP) was an important step taken by many firms and provision of services through mobile and ubiquitous technologies is now inevitable. ERP users (managers, customers, employees, etc.) expect data to be made available to them via the most widely used communication mobile device. But generally, distributed computing through a handheld/mobile device has to be considered with carefulness because of the limited capabilities on these devices. In this paper, we present a new approach based on ubiquitous agents to assist ERP costumer surrounded by many mobile devices, not only any-time and any-where but also on any-device.

Keywords: ERP · Customer assistance · Ubiquitous mobile agent

1 Introduction

Enterprise Resource Planning (ERP) systems have significantly contributed to progressing in enterprise-wide information system integration. ERP provides an integrated view of core business processes, often in real-time, using common databases maintained by a database management system.

An ERP system covers several functional areas. In many ERP systems these are called and grouped together as ERP modules such as: Financial accounting, Management accounting, Human resources, Manufacturing, Order Processing, Supply chain, Project management, Customer relationship management (including sales and marketing, commissions, service, customer contact, call center support), (see Fig. 1).

15–20 years ago, things were relatively simple. An enterprise had to ensure better performance for each business function. But today the customer is at the heart of the business model and he is more and more demanding. He does not want only a low price, but shorter details and highest quality.

These is due to the emergence of sophisticated mobile devices, such as smartphones or pocket PCs and had added a new element into the enterprise, we talk about mobile ERP.

In the simplest approach, mobile ERP is about having access to software that allows a mobile device to be connected to the ERP system of an organization through a mobile net.

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 55–62, 2015.
DOI: 10.1007/978-3-319-25744-0_5

Fig. 1. ERP modules.

In the context of hardware, Roger Kay said in his article appeared in Forbes [7]: "*As soon as you have more than one device in your life, they must necessarily point to you. Cloud services increasingly coordinate devices for us so that our "state" — the exact condition of all our stuff at any one moment — migrates seamlessly from one device to another other. A good example would be reading an eBook. If you stop reading on a certain page on your laptop, you should be able to open your eReader and be on the same page. Ditto for your phone. This idea that "state" follows you around puts you at the center of your own universe. The devices are all around you.*"

Now, if we talk about the software side, to manage these devices in an intelligent and ubiquitous way, we rightly need intelligent applications that follow users, identify their needs and preferences and act autonomously to their requests.

Therefore, the multi-agent systems paradigm appears to provide an excellent platform for developing integrative business information systems generally and ERP systems specifically in ubiquitous environments.

Indeed, literatures show that a multi-agent system is similar to an integrated enterprise because it is essentially a community of autonomous and problem solving agents who interact through network in a coordinated manner to achieve their own and shared goals, thereby contributing to the achievement of system level goals.

Today, customers are surrounded by many mobile and ubiquitous devices, we want to take advantage of this diversity to propose an approach based on a ubiquitous assistant agent that recognizes the client devices; with the ability to migrate from one device to another.

The rest of paper is organized as follow: Sect. 2 summarizes some works based on the use of agents in ERP systems. We present in section three our contribution, then the implementation on JADE-LEAP. We terminate the paper with a conclusion.

2 Related Work

Intelligent agents have the ability to make decisions defined by their inherent properties, which are: reactivity, pro-activity and sociability. These properties are intended to enable the agent to meet the objectives for which they were designed, following rules of behavior that enable them to communicate with their environment [13]. These principles have tempted many researchers in the field of ERP.

Dice the beginning of 90, Pan and Tenenbaum [12] propose an Intelligent Agent based framework to integrate people and computer systems in large geographically dispersed manufacturing enterprise. In the framework, each agent supports a clearly discernible task or job function, and interacts with each other via messages through a shared, distributed knowledge base.

The framework is built by dividing complex enterprise operations into a collection of elementary tasks, and each task is modeled in cognitive terms and entrusted to an IA for execution. Their preliminary experimental results indicate that agent-based systems are a practical way to integrate a complex enterprise.

Lin et al. [9] define an enterprise to be a collection of business entities working toward delivering a product or service to the customer. Each entity performs its independent processes organized in a hierarchy. A high-level process consists of a set of low-level processes. The authors propose a multi agent information system (MAIS) for the supply chain network for capturing both the structure and the processes of an enterprise.

Jennings et al. [5] suggest a community of negotiating agents, and propose the Advanced Decision Environment for Process Tasks (ADEPT) for conceptualizing, designing, and implementing business process management systems.

In the framework, various functions of the business process are delegated to a number of autonomous problem-solving agents, and these agents interact and negotiate with each other in order to coordinate their actions and to buy in the services they require.

Another example of multi-agent system related to ERP is presented by Kehagias [8]. He mentioned the connection with the ERP database designing a new functional layer. This layer corresponds to a MAS architecture which control sales orders and give a recommendation on how could be fulfilled. This system consists of five agents: the client orders agent, the recommender agent, the customer profile identifier agent, the inventory profile identifier agent and the supplier profile identifier agent.

Jingrong [6] refers to the use of a MAS in order to identify the standards used by the flows of processes regarding an already implemented ERP system, since the use of agents allows that each functional unit are defined as an agent than could interacts with other agents in order to identify the exchange of knowledge and how they should handle coordination of functional units.

We can easily remark that over the time, ERP adapt to new technologies. Indeed, there is several works on ERP that introduce and take advantage of new technologies.

Comprise the work of [11] on the use of cloud by the review of development of Low cost ERP solution to Indian industries on Mobile, using latest technologies such as Mobile computing, SaaS, Cloud Computing.

Authors in [10] define: Cloud ERP is an approach to enterprise resource planning (ERP) that makes use of cloud computing platforms and services to provide a business with more flexible business process transformation.

Cacciagrano et al. [2] focus their interest on the use of ontology, web semantic and ubiquitous computing. They propose architecture based on an expandable 'Business Intelligence 2.0' Enterprise Resource Planning (ERP) prototype, with the aim to lead Public Administration toward Business Intelligence and information maturity. Distributed and heterogeneous knowledge through semantic-driven GUI (Graphical User Interface)-based components is integrated on a common semantic knowledge model and embedded in a Cloud-based middleware.

Even if these architectures are promising by the use of agent paradigm, cloud technology, ubiquitous computing, etc., there are many limitations often discourage companies from fully exploiting ERP solutions, restricting their use to trivial operations.

Indeed, communication between distributed locations is often unreliable. Thus when performing a resource scheduling across given network, the big number of agents (or other mobile applications), limited bandwidth, unreliable nature of the Internet and widespread volume of exchanging messages, will slow down the whole process.

Furthermore, it is difficult for a customer to deal directly with different production resources.

He usually relies on procurement having different expertise, different focuses, different locations or different partnerships.

3 Contribution: Ubiquitous Assistant Agent

ERP costumers, like most people are today equipped with handheld devices (smart phones, PDA, Laptops, etc.) that provide them access to Enterprise data. Having positioned a service infrastructure, the communication between the consumer (in this case the application residing on the handheld) and the services is expected to happen over HTTP protocol.

The process may operate in geographies they may not have connectivity, a situation authors in [1] refer to as 'Occasionally Connected Systems. Below are the three distinct characteristics that are expected in an occasionally connected system:

- Offline computing capability. (Provide application accessibility at times when connectivity does not exist.)
- Ability to access a cloud service. (From a mobility perspective, services hosted on/off premise can be termed as a cloud service.)
- Ability to synchronize data with the back office as and when connectivity is restored. (Push and pull of information from the device.)

Our idea is to connect the ERP database with the mobile client, any-where and any-time, through an ubiquitous agent that can migrate from one device to another and can work offline.

We propose to use an ubiquitous agent embedded in one of the customer devices (ex. smart phone) and knows all other devices surrounding him (ex. Tablet, laptop, smart

TV, etc.). The proposed agent is mobile, can migrate on all customer devices and takes advantages from all of them especially in case of energy insufficiency, limited storage space or failure calculating capabilities. So, even if a device is off or has no connection, ubiquitous agent can complete its tasks, assists the customer (offline mode) and communicate with enterprise servers when the connection is established (online mode).

At the enterprise level, we planned to install coordinator agents. Their aim is to provide communication between mobile ubiquitous agents and different ERP modules. Thus, in case of any update, coordinator agents inform ubiquitous agents on the client devices (see Fig. 2).

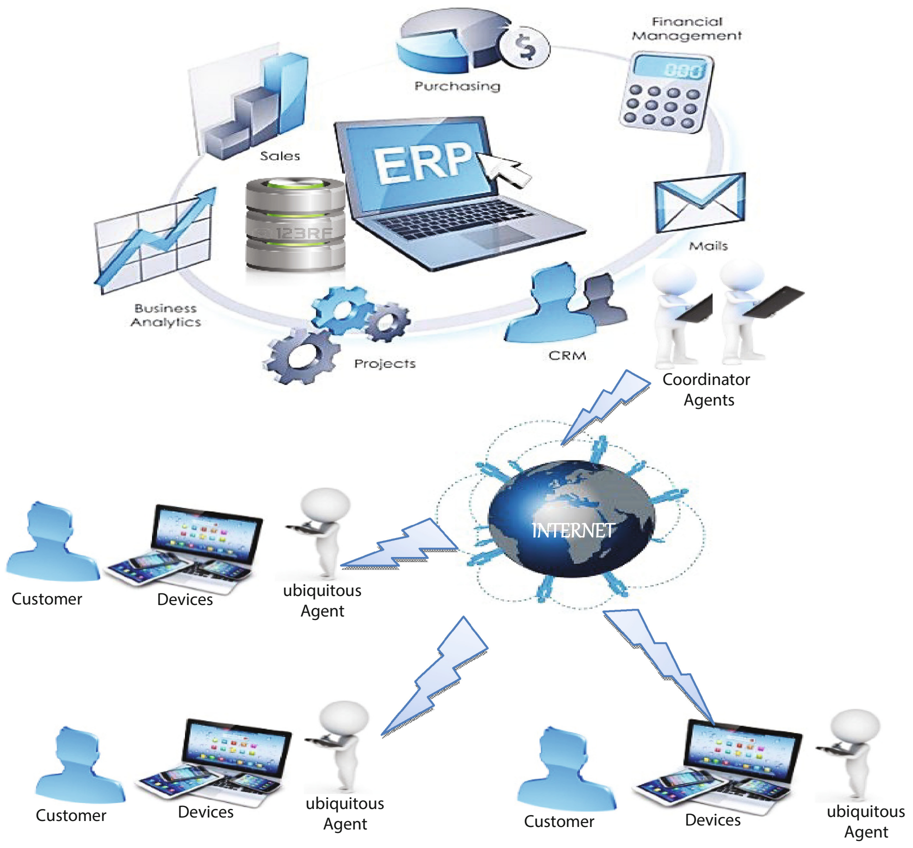


Fig. 2. The proposed architecture: ubiquitous assistance agents and coordinator agents

3.1 Agent for Each Device?

We can suggest running multiple agents on each device simultaneously. But it's embarrassing solution for the customer.

Our solution consists to embed agent to device chosen by the customer, when registering to enterprise *(I rather use my smart phone more than my PC).* We can also ask the client to establish an ordered list of devices according the frequency of use.

3.2 How Assistant Ubiquitous Agent Knows the Client Devices?

The most obvious solution is to have a predefined ordered list of all client devices, and establish a wireless network including all these devices.

In this case, it is necessary that the customer declares to enterprise adding or suspending devices.

This solution is not very pleasant and affects privacy. We propose that the user does not notify the enterprise, but the ubiquitous agent to update the list of devices with the help of the owner.

4 Implementation Using JADE-LEAP

JADE is a set of Java classes that allow a developer to build a FIPA-compliant multi-agent system quite easily, [3]. A few years ago an add-on to JADE, called LEAP, was released. This add-on replaces some parts of the JADE kernel, creating a modified environment called JADE-LEAP [4], which allows the implementation of agents in mobile devices with limited resources.

Figure 3 shows an example of the JADE-LEAP execution environment. In this example there is a multi-agent system with six agents. Two of them are running on the main container of the platform, which is connected through Internet with a container on a PC that holds another two agents.

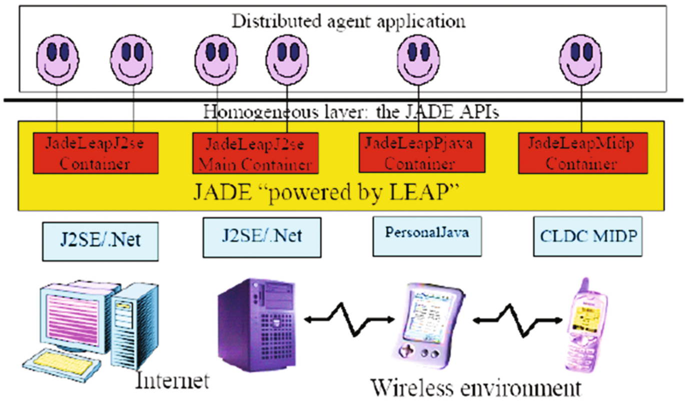
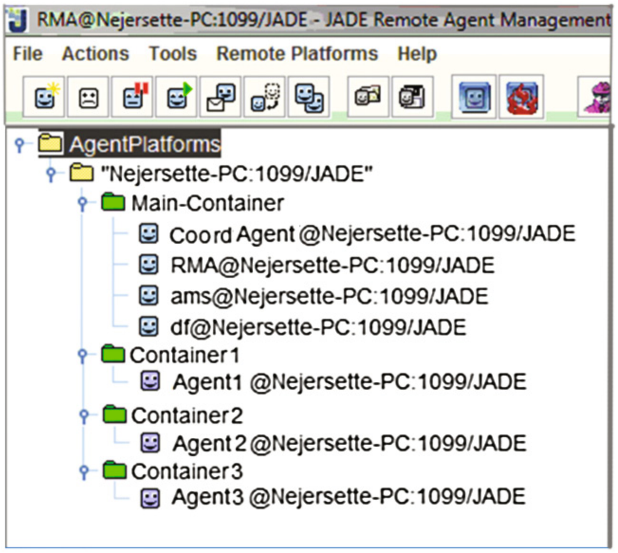


Fig. 3. JADE-LEAP execution environment

These four agents can communicate wirelessly with two agents running in mobile devices (one on a PDA and one on a mobile phone). Note that there is one container in each mobile device in this example.

For our proposal, we use three costumers equipped by various devices where JADE-LEAP is installed. We create three agents for the management of costumers' devices: Agent1, Agent2 and Agent3. In Fig. 4, each container comprises ubiquitous agent representing a set of devices. The main container represents the enterprise. We suggest one coordinator agent (CoordAgent in Fig. 4).



```
AgentPlatforms
  "Nejersette-PC:1099/JADE"
    Main-Container
      CoordAgent@Nejersette-PC:1099/JADE
      RMA@Nejersette-PC:1099/JADE
      ams@Nejersette-PC:1099/JADE
      df@Nejersette-PC:1099/JADE
    Container1
      Agent1 @Nejersette-PC:1099/JADE
    Container2
      Agent2@Nejersette-PC:1099/JADE
    Container3
      Agent3 @Nejersette-PC:1099/JADE
```

Fig. 4. Agents and containers

So, communication between ERP system and costumers is done by the communication between coordinator agent and ubiquitous agents.

We did some experiments using three different devices previously known by the ubiquitous agent (smart phone, tablet and laptop).

At first, it is on the smartphone for establishing a connection with the coordinator in order to receive updates from the ERP database.

Ubiquitous agent is completely related to the device and has the ability to survey energy, storage and computing capacity.

We tested the disconnection due to energy failure or the offline mode. For this, we programmed ubiquitous agent to migrate toward another client device, perform its task and notify the customer. We also proposed that the ubiquitous agent inform the costumer before moving: "*Your smart phone will shut down, go on your tablet.*" Or, "*There is no longer connecting to your Laptop, find me on your smart phone*".

We also considered the case where the agent is embedded on the smart phone and because of limited computing capacity; it cannot decrypt messages and data received from coordinator agent. It will choose another device (sorted by computing power), migrates on it, and notify the customer.

5 Conclusion

We proposed in this paper an approach based on ubiquitous agents in order to assist customers connected to an ERP system. Being mobile, the customer hands several mobile, ubiquitous and restricted devices. Our proposal is to take advantage of all devices surrounding the customer by allowing the assistant ubiquitous agent to recognize them and to migrate on, in order to complete its task.

At enterprise level, coordinators agents are responsible for communicating and informing ubiquitous agents of any update, or to respond to customer queries. We have done a preliminary test of our proposal using JADE-LEAP framework.

In future works, we would like to focus more on CRM: Costumer Relation Management, and test our proposal in real enterprises.

References

1. Alur, S.J.: Enterprise integration and distributed computing: a ubiquitous phenomenon. Microsoft Corporation (2008). http://msdn.microsoft.com/en-us/library/cc949110.aspx
2. Cacciagrano, D.: Semantics on the cloud: toward an ubiquitous business intelligence 2.0 ERP desktop. In: SEMAPRO 2012: The Sixth International Conference on Advances in Semantic Processing (2012)
3. Caire, G.: JADE tutorial-JADE programming for beginners. TILAB (2001)
4. JADE-LEAP. http://sharon.cslet.it/project/jade
5. Jennings, N.R., Norman, T.J., Faratin, P., O'Brien, P., Odgers, B.: Autonomous agents for business process management. J. Appl. Artif. Intel. **14**(2), 145–189 (2000)
6. Jingrong, Y.: Research on reengineering of ERP system based on data mining and MAS, pp. 180–184 (2008)
7. Kay, R.: Surrounded by devices, We inhabit a world of increasing user-centricity. Forbes (2013). http://www.forbes.com/sites/rogerkay/2013/01/04/surrounded-by-devices-we-inhabit-a-world-of-increasing-user-centricity/. Accessed 04 June 2013
8. Kehagias, D.: Information agents cooperating with heterogenous data sources for customer-order management. In: Proceedings of the 2004 ACM symposium (2004)
9. Lin, F.R., Tan, G.W., Shaw, M.J.: Multi agent enterprise modeling. J. Organ. Comput. Electron. Commer. **9**(1), 7–32 (1999)
10. McKendrick, J.: When ERP goes to the cloud, you know things are getting serious. Forbes (2014). http://www.forbes.com/sites/joemckendrick/2014/06/18/when-erp-goes-to-the-cloud-you-know-things-are-getting-serious/. Accessed 18 June 2014
11. Saini, S.L.: Cloud and ERP. In: Proceedings of the World Congress on Engineering (2011)
12. Pan, J.Y.C., Tenenbaum, J.M.: An intelligent agent framework for enterprise integration. IEEE Trans. Syst. Man Cybern. **21**(6), 1391–1408 (1991)
13. Wooldridge, M.: An Introduction too Multi agents Systems. Wiley, Sussez (2002)

A Hybrid Algorithm for DAG Application Scheduling on Computational Grids

Lyes Bouali[1], Karima Oukfif[2,3], Samia Bouzefrane[4](✉), and Fatima Oulebsir-Boumghar[3]

[1] LARI Laboratory, UMMTO, Tizi Ouzou, Algeria
bouali.lyes@gmail.com
[2] Compute Science Department, UMMTO, Tizi Ouzou, Algeria
karima.oukfif@gmail.com
[3] LRPE Laboratory, USTHB, Bab Ezzouar, Algeria
fboumghar@usthb.dz
[4] CEDRIC Laboratory, CNAM, Paris, France
samia.bouzefrane@cnam.fr

Abstract. In the late three decades, grid computing has emerged as a new field providing a high computing performance to solve larger scale computational demands. Because *Directed Acyclic Graph* (DAG) application scheduling in a distributed environment is a NP-Complete problem, meta-heuristics are introduced to solve this issue. In this paper, we propose to hybridize two well-known heuristics. The first one is the *Heterogeneous Earliest Finish Time* (HEFT) heuristic which determines a static scheduling for a DAG in a heterogeneous environment. The second one is *Particle Swarm Optimization* (PSO) which is a stochastic meta-heuristic used to solve optimization problems. This hybridization aims to minimize the makespan (i.e., overall completion time) of all the tasks within the DAG. The experimental results that have been conducted under hybridization show that this approach improves the scheduling in terms of completion time compared to existing algorithms such as HEFT.

Keywords: Grid computing · Task scheduling · Directed acyclic graph · Heterogeneous earliest finish time algorithm · Particle swarm optimization algorithm · Makespan

1 Introduction

The deployment of high-speed networks and powerful computers has involved to new computing paradigms. Hence, while current hardware infrastructures are distributed in nature such as in grid computing, the underlying applications are composed of tasks distributed on different nodes. In fact, a grid computing is a set of geographically remote resources deployed across multiple nodes allowing their computational power and storage space to be shared. Grid resources are heterogeneous, dynamic and owned by various administrative organizations under locally defined policies. Grids are used in a variety of scientific applications such as in astronomy, geophysics and bioinformatics where a single and powerful parallel super computer [1] cannot resolve the large-scale application issues.

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 63–77, 2015.
DOI: 10.1007/978-3-319-25744-0_6

To take advantage of the potentials of grid computing, efficient scheduling algorithms are fundamentally important [2]. The task-scheduling problem refers to the mapping of the application tasks to a set of distributed resources or nodes. Because this problem is NP-Complete, various algorithms are proposed in the literature with different criteria in order to schedule efficiently application tasks.

Our contribution in this paper is twofold: firstly, we propose a scheduling approach based on the hybridization of two scheduling algorithms like HEFT and an adapted DPSO, called DPSO*, for the dependent-tasks scheduling problem. Secondly, we undertake some measurements that show that the hybridization approach improves the performances in terms of makespan. Makespan is the time difference between the start and finish of a sequence of tasks.

The remainder of this paper is organized as follows. Related works are presented in Sect. 2. Section 3 formalizes the scheduling problem. Section 4 describes the HEFT, our adapted DPSO* algorithms and their hybridization. Then, performance tests are discussed in Sect. 5. Finally, Sect. 6 concludes the paper with some perspectives.

2 Related Works

In [3], Casavant and Kuhl have proposed a taxonomy of scheduling algorithms for general-purpose parallel and distributed computing systems. Since grid computing has specific features, scheduling algorithms for grid computing fall into a subset of this taxonomy [2]. In fact, in [2], the authors classified grid scheduling algorithms depending on whether the grid scheduling algorithm is static or dynamic, distributed or centralized, cooperative or non-cooperative.

Due to the NP-Complete property of the scheduling algorithms and the difficulty to prove the optimality of a given solution, researchers tried to find sub-optimal solutions through heuristic approaches. When the relationship between the tasks within the grid application is considered, scheduling algorithms are dichotomized into independent and dependent task scheduling. Hence, in [4], a comparison between eleven heuristics used to schedule independent tasks is discussed. Among these heuristics, we can find *Opportunistic Load Balancing*, *Minimum Execution Time*, *Minimum Completion Time,* etc. Each of them aims to assign a task to a resource with an optimal completion time.

In the case of dependent task scheduling, also called workflow scheduling, a task precedence graph called *Directed Acyclic Graph* (DAG) is usually used to model the application scheduling. The nodes of the DAG represent the tasks and the directed edges represent the execution dependencies and the data communication between tasks [5]. There are two major types of scheduling, best-effort based and QoS constraint based scheduling. Supporting QoS scheduling algorithms are based on either deadline (time) or budget (cost) constraints and are at a very preliminary stage [6]. Best-effort based scheduling attempts to minimize the makespan using different approaches. These approaches can be classified into different heuristics such as list-scheduling, clustering, duplication-based algorithms, and meta-heuristics (guided random search methods) approaches.

List-scheduling heuristics are based on two steps: in the prioritizing phase, tasks are ordered in a list by assigning a priority for each task, while in there resource selection phase each selected task is scheduled on the resource that minimizes a predefined cost function [7]. Various research works have been proposed in the literature under this type of heuristics such as HEFT and CPOP [7], FCP [8], DCP [9], DLS [10], and xDCP [11].

While the clustering approaches (DSC [12], CASS II [13], EZ [9], CTHP [14]) assign a group of inter-communicated tasks to the same cluster hence to the same resource, the duplication based-scheduling approaches (DSH [15], CPFD [16], TDS [17], BTDH [18], THAN [19]) duplicate tasks to assign them to idle-time slots within the resource, thus avoiding the data communication overhead.

Besides, meta-heuristics are stochastic algorithms dedicated to solve optimization problems. Using meta-heuristic approach, there is no guarantee to find a global optimum but it provides an approximation of this optimum in a reasonable time. Genetic Algorithms (GA) [20–24] are examples of meta-heuristics that are widely used, for the good solutions they provide. To overcome the high execution time taken by GA, Kennedy and Eberhart introduced the Particle Swarm Optimization (PSO) methodology in [25]. In the context of grid computing, PSO has been used by the authors of [26] to schedule independent tasks by transforming the continuous values of particles into discrete values thanks to the *Smallest Position Value* (SPV) rule. In [27], Liu et al. designed a fuzzy scheme based on discrete PSO to solve the independent job scheduling problem on computational grids. Izakian et al. [28] have proposed a version of discrete PSO for grid independent job scheduling.

In this paper, our aim is to build an optimal scheduling algorithm by adapting the PSO algorithm and then by hybridizing it with HEFT heuristic, in order to schedule DAG tasks in the context of grid computing. In fact, according to the PSO principle, since the particles are initialized randomly, our idea is to inject a particular particle initialized thanks to HEFT algorithm that is considered as a high-quality solution, in order to enhance PSO technique and hence to reduce significantly the convergence time. Before describing this hybridization, we formalize the scheduling problem in the next section.

3 Scheduling-Problem Formalization

A scheduling system is generally modeled thanks to an application, a platform and a scheduling-performance criterion. In our case, the criterion is the makespan. In the next sub-sections, we formalize each part of the scheduling model.

3.1 Application Formalization

Each application is modeled by a Direct Acyclic Graph $G = (V, E)$, where V is a set of v vertices representing tasks (jobs) T_i $(1 \leq i \leq v)$, and E a set of directed edges. An edge $(i, j) \in E$, corresponds to a dependence constraint between task T_i and T_j. T_i is an immediate parent task of T_j, and T_j the immediate child task of T_i. A child task cannot be executed until all of its parent tasks are completed. A task with no parent tasks is called an *entry task* and a task with no children tasks is called an *exit task*. We assume that only one entry and one exit tasks exist in the graph.

Data matrix, with $v \times v$ dimensions, represents the data volume exchanged between tasks.

3.2 Platform Formalization

The target computing environment is made up of a set of q heterogeneous compute resources completely interconnected. We assume that the communication between compute resources is performed without contention and can be overlapped with computation. We define two distinct matrices.

Computation_time matrix, with $v \times q$ dimensions, represents the execution time of tasks on compute resources. $Computation\_time_{i,\ j}$ is the estimated execution time of the task T_i on the compute resource PC_j.

The average execution time of a task T_i is:

$$\overline{Computation\_time_i} = \frac{\sum_{j=1}^{q} Computation\_time_{i,j}}{q} \tag{1}$$

Transfer_rate matrix, with $q \times q$ dimensions, represents the data transfer rate (bandwidth) between compute resources.

The communication time of an edge $(i, j) \in E$ in the DAG, which is the time taken to transfer data from task T_i (executed on PC_p) to task T_j (executed on PC_k), is defined as in the following:

$$Communication\_time_{i,j} = \frac{data_{i,j}}{Transfer\_rata_{p,k}} \tag{2}$$

When tasks T_i and T_j are executed on the same compute resource, we have Communication_time$_{i,j}$ equal to zero. Consequently, the average communication time of an edge (i, j) is given in formula 3.

$$\overline{Communication\_time_{i,j}} = \frac{data_{i,j}}{\overline{Transfer\_rate}} \tag{3}$$

Where $\overline{Transfer\_rate}$ is the average of transfer rates between all the compute resources.

3.3 Makespan Formalization

To define the makespan, we use two attributes as defined in [7]:

1. *Earliest execution Start Time* (EST) of a task T_i assigned to a compute resource PC_j. EST is the earliest time during which a task T_i is started. As shown in the following,

the EST of a task *T* depends not only on the end of execution of the parent tasks of *T* but also on the data communication time, except when *T* is an entry task in which case EST is equal to zero.

$$\mathrm{EST}(T_{Entry}, PC_j) = 0 \tag{4}$$

$$\mathrm{EST}\left(T_i,\, PC_j\right) = \max\left\{avail[j],\, \max_{T_m \in pred(T_i)}(AFT(T_m) + Communication\_time_{m,i})\right\}. \tag{5}$$

Where:

– *pred*(T_i) is the set of immediate parent tasks of T_i.
– *avail*[*j*] is the earliest time at which the compute resource PC_j is available to execute a task, and
– *AFT* is the *Actual Finish Time* of a task as described here after.

2. *Earliest execution Finish Time* (EFT) of a task T_i on a compute resource PC_j corresponds to the time at which T_i ends its execution, that is, the starting time *EST* of T_i added to its execution time.

$$EFT(T_i, PC_j) = Computation\_time_{i,j} + EST(T_i, PC_j). \tag{6}$$

After task T_i is actually scheduled on the compute resource PC_j, *Actual Start Time* of the T_i is calculated as $AST(T_i) = EST(T_i,\ PC_j)$. In addition, *Actual Finish Time* of task T_i is defined as: $AFT\ (T_i) = EFT(T_i,\ PC_j)$.

When all the DAG tasks are scheduled, the completion time of the application is simply the *AFT* of the exit task.

$$\mathrm{Makespan} = AFT(T_{exit}) \tag{7}$$

The objective of any scheduling algorithm is to find an assignment of tasks on the compute resources, that minimizes the makespan among other criteria. The next section deals with our contribution that aims to hybridize two heuristics in order to minimize makespan when scheduling tasks in a grid computing.

4 HEFT/DPSO* Hybridization

Our objective in this research work is twofold:

• First, we aim to adapt the basic DPSO [28] for the dependent tasks scheduling problem. In our proposed solution (DPSO*), after their assignment, tasks are ordered in such a way that the dependencies constraints are satisfied.

- Second, we aim to further improve the performances of our solution, i.e. the DPSO*, by combining it with the HEFT algorithm.

Our purpose here is to adapt the DPSO algorithm so that we can minimize the completion time of tasks when scheduled on a grid computing. Since the particles used by PSO algorithm are initialized randomly, our idea is to inject a particular particle initialized thanks to HEFT algorithm that is considered as a high-quality solution, in order to reduce significantly the convergence time. Before describing our proposed hybridization approach, we first recall the features of HEFT and describe our proposed DPSO* approach.

4.1 Heft

HEFT (Heterogeneous Earliest-Finish-Time) [7] is one of the most widespread scheduling-list algorithms. It determines a static scheduling of a DAG on a heterogeneous environment so as to minimize the makespan. As described in the following, HEFT has two execution steps.

1. **Task-prioritizing phase:** HEFT uses the *upward rank* attribute to order the tasks of the DAG. It is recursively defined by:

$$\mathrm{rank_u}(\mathrm{T_i}) = \overline{Computation\_time_i} + \max_{T_j \in succ(T_i)} \left(\overline{Communication\_time_{i,j}} + \mathrm{rank_u}(T_j)\right) \quad (8)$$

   Where $succ(T_i)$ is the set of immediate children of task T_i. The rank is calculated starting from the exit task.

$$\mathrm{rank_u}(\mathrm{T_{exit}}) = \overline{Computation\_time_{exit}}. \quad (9)$$

2. **Compute-resource selection phase:** Tasks are mapped to the adequate compute resources that minimize the EFT like in the formula 6.
   A variant of HEFT is the **Duplication based HEFT (DHEFT)** [29] that is based on task duplication. By duplicating dependent tasks and assigning them within the compute resources that host their children tasks, the communication overhead is reduced, hence improving the makespan of the application.

4.2 Adapted Discrete PSO (DPSO*)

PSO is an adaptive population-based search method inspired by social behavior patterns such as bird flocking and fish schooling. It can be implemented easily to solve various function optimization problems. Its main advantage is its fast convergence. Initially, PSO was used to solve continuous problems. However, a discrete binary version of PSO was introduced to solve discrete optimization problems in [30].

To solve scheduling problems, various versions of PSO were used like fuzzy PSO in [27] or Discrete PSO in [28]. DPSO deals with scheduling independent jobs in the grid environment. Since we are interested in this variant of PSO, we propose to adapt it to the dependent-tasks problem then hybridize our adapted DPSO, called DPSO*, with HEFT. Here, we explain its principle. In fact, our DPSO* algorithm initially generates randomly a swarm of particles. A particle is analogous to a bird flying through a search space. Each particle has a position *X*, a velocity *V*, and a fitness value. Particle's position is seen as a potential solution to the problem. Positions are evaluated by a fitness function to be optimized. Also, each particle knows its best past position it has reached (*pbest*) and the best position ever reached by any particle in the swarm (*gbest*). The movement of particles is influenced by its actual position and its velocity. Particle's velocity represents the direction and the magnitude of the next movement. It is calculated by considering its actual velocity, *pbest* and *gbest*. The next paragraphs discuss the features that characterize DPSO* algorithm.

Particle's Position. A particle's position represents a potential scheduling solution. We use the direct representation [28] to encode the scheduling solutions. The position (solution) is a vector X of v elements where v is the number of jobs. The elements of the vector are natural numbers included in range [0, q] where q is the number of compute resources in the grid. Hence, X[j] is the index of the computer resource where job j is assigned by the scheduler. For example, a solution of scheduling problem with 4 jobs to schedule and 2 available compute resources is represented by a vector of 4 elements X = (1, 0, 0, 1). So, jobs 1 and 4 are assigned to the compute resource indexed by 1 and jobs 2 and 3 are assigned to compute resource indexed by 0.

Particle's Velocity. The velocity is a $q \times v$ matrix called V where q is the number of available compute resources and v the number of jobs, as expressed in the following:

$$\mathrm{V[i,j]} \in [-v_{max}, V_{max}], \forall \mathrm{i} \in \{1, 2, \ldots, q\}\, \mathrm{et} \forall \mathrm{j} \in \{1, 2, \ldots, v\}$$

Initially, position's vectors and velocity's matrixes of particles are randomly generated as stated in Sect. 2.

Fitness Function. In general, the fitness function used to evaluate the particles is the makespan. Because makespan refers to the efficiency of the tasks-compute resources mapping, we have chosen to use it as a criterion to minimize.

$$\mathrm{Finess} = \mathrm{makespan}. \tag{10}$$

Movement. The movement is realized by firstly updating the matrix velocity and then the vector position of each particle. After each particle is moved, the *pbest* and *gbest* must be updated by checking the performance of each particle using the fitness function. The movement of the particles through the search space is described by the following algorithm.

Particles movement algorithm:

```
For each particle k = 1, ..., P do
    // Matrix velocity updating.
    For each job j = 1, ..., n do
       q =$X_k^t$[j];
       z =$pbest_k^t$[j];
       s = $pbest_k^t$[j];
       if q ≠z then
         $V_k^t$[q, j] =w.$V_k^t$[q, j] - c1   r1;
         $V_k^t$[z, j] w.$V_k^t$[z, j] + c1   r1;
       end if
       if q ≠s then
          $V_k^t$[q, j] =w.$V_k^t$[q, j] – c2   r2;
          $V_k^t$[s, j] =w.$V_k^t$[s, j] + c2   r2;
       end if
     end for

 // Vector position updating.
    For each job j = 1, ..., n do
     if $V_k^t$[ , j] = max {$V_k^t$[i, j]}∀i∈ (1, 2, ..., m)then
        $X_k^t$[j]=  ;
     end if
    end for
end for
```

w, c1 and *c2* are the PSO parameters, *w* is the inertia weight, *c1* is the coefficient of the self-recognition component and *c2* is the coefficient of the social component, *r1* and *r2* are random numbers used to maintain the diversity of the swarm.

Dependency Constraint Supported. To take into consideration the dependency constraints between tasks, we characterize each task with the following parameters: task number, EST, EFT, and a tag value that indicates if the task is scheduled or not.

Before scheduling, the AST and AFT folders of each task are initially unknown, and all the tasks are tagged as not scheduled. Once DPSO* is applied, EST and EFT folders of each task are known and all the tasks are tagged as scheduled.

To get the final scheduling, our approach operates according to the following steps:

1. All the tasks are assigned to compute resources on which they will run. This step is similar to the basic DPSO that is suitable for independent tasks as described in [28]. However at this step, the order in which each task will start and finish its execution on a given compute resource is not known yet due to the task dependencies. Consequently, a second step is necessary as in the following.
2. In this second step, the start and finish execution time of each task will be defined on each compute resource. To do so, the DAG must be traveled downwards starting from the entry task. First, because entry task has no parent, it is tagged as scheduled and its EST is set to 0 (see formula 4) and its EFT is calculated using formula 5.

Second, the other tasks will wait until all their immediate parents are scheduled, in other terms, their ESTs and EFTs are calculated according respectively to the formulas 5 and 6.

DPSO* Algorithm. Before the start of the DPSO* execution some parameter values must be set. Then, particles are generated and initialized randomly. After that, they explore the search space trying to find a satisfactory solution for the problem until the maximum number of iteration is reached. A pseudo-code of the DPSO* algorithm is shown in the following.

4.3 The HEFT/DPSO* Hybridization

As depicted in Fig. 1, after generating the initial swarm, instead of randomly initializing all the particles, in our proposed DPSO* algorithm, one particle is initialized with the solution given by HEFT and other particles are randomly initialized. In this way, this step is optimized.

```
Initialize PSO parameter (swarm size, $max_{iter}$ ,w , c1, c2)
Generate initial swarm
Initialize particles positions and velocities randomly
While ($iter<max_{iter}$) do
   For each particle k = 1, . . ., P do
     if Fitness ($X_k$) > Fitness ($pbest_k$) then
       $pbest_k= X_k$;
     end if
     if Fitness ($pbest_k$) > Fitness ($gbest_k$) then
       $gbest=pbest_k$;
     end if
   end for
   For each particle k = 1, . . ., P do
     Movement of the particle k;
   end for
   iter =iter+1;
end while
```

5 Experimental Results

To evaluate the performance of our proposed algorithm, we have conducted some experiments and compared the resulting tests of our hybrid HEFT/DPSO* algorithm with HEFT and DPSO* algorithms regarding the makespan parameter. We have used a DAG generator called *RandomTaskGraphGenerator* to generate our DAGs that represent the applications to schedule.

The grid environment that we considered here is composed of several heterogeneous compute resources which are connected by heterogeneous links.

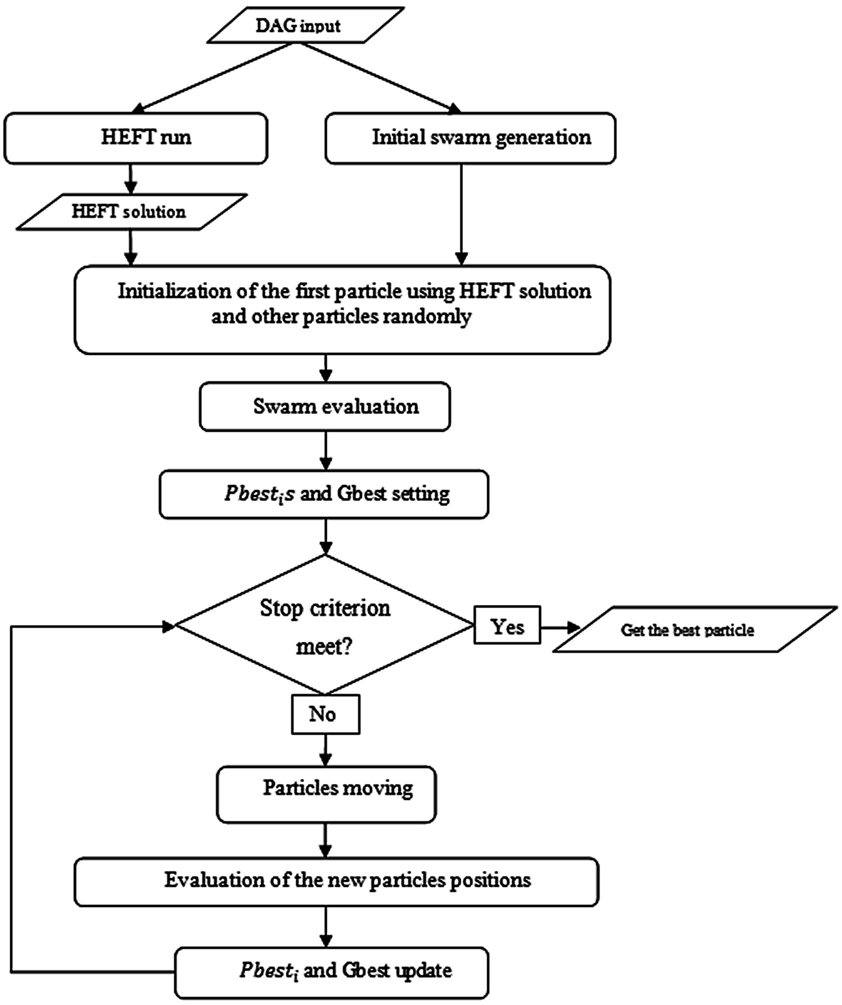


Fig. 1. Flow chart of HEFT/DPSO* hybridization

We assume that the computation time of each task on each compute resource, the data volume exchanged between tasks and the data transfer rate between compute resources are known.

For our simulation, the platform is described within a configuration file that contains the following information: the number of tasks composing each application, the number of compute resources of the platform and three matrices. The first is the computational cost matrix with “Number of tasks × Number of compute resources” dimension (line index represents the number of tasks and column index represents the number of compute resources). The second is the data transfer speed matrix with “Number of compute resources × Number of compute resources” dimension (line and column indexes represent compute resource numbers). The value ‘0’ means there is no transfer between a compute resource and another; besides the transfer speed between a compute resource and itself is null. The third one is the data matrix that contains the data transferred between tasks. This matrix has “Number of tasks × Number of tasks” dimension (the line and column indexes represent task numbers). The value ‘−1’ means that there is no data exchanged between tasks. A value ‘0’ means that a task has another kind of dependence, other than data transfer, with another task. Concerning the positive values of the data matrix, each value corresponds to the volume of data transfer between the two corresponding tasks.

Based on the configuration file, we conducted our measurements using five applications with different number of tasks on a grid environment with different number of available compute resources.

Since the results of our DPSO* and our hybrid algorithm are stochastic (due to the DPSO behavior), we repeated each experiment 10 times and recorded the makespan value of the best solution obtained. For HEFT, it is executed only once since it is a deterministic algorithm.

Specific parameter settings used by our hybrid HEFT/DPSO* and our DPSO* are described in Table 1.

In our experiments, we measured the makespan criterion by varying the number of compute resources in one side, and the number of tasks in the other side.

Table 1. Parameter settings of DPSO.

DPSO* parameter	Value
Size of swarm	50
Maximum iteration	1000
Self-recognition coefficient c1	2
Social coefficient c2	1

5.1 First Performance Study

We used an application of 40 tasks which we run on a grid environment with different number of available compute resources (2, 3, 5, 7 and 10), and we measured the makespan as in Table 2.

Table 2. Makespans comparison according to the number of compute resources.

Compute resource number	HEFT/DPSO*	DPSO*	HEFT
2	733	736	922
3	560	580	620
5	425	437	441
7	383	392	399
10	296	329	308

Figure 2 shows the makespan as measured for each scheduling algorithm when varying the number of compute resources. According to these measurements, we notice that in respect to makespan, our hybrid HEFT/DPSO* algorithm offers better results than DPSO* and HEFT algorithms.

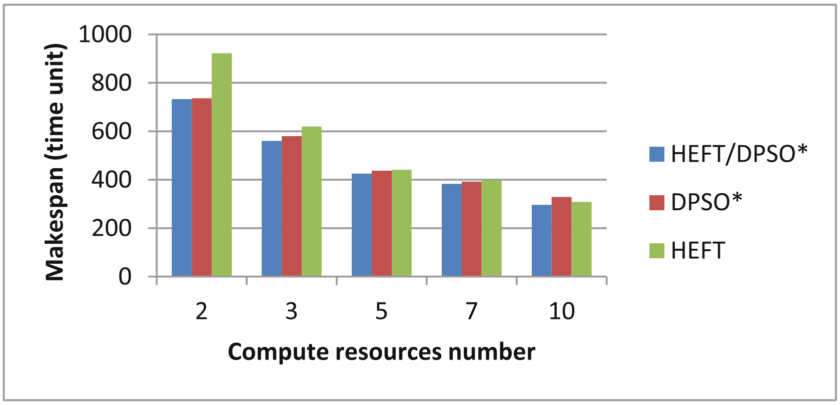


Fig. 2. Makespan comparison according to the number of compute resources

5.2 Second Performance Study

To measure the makespan criterion when varying the number of tasks, we considered five applications with different number of tasks (10, 20, 40, 60 and 100) that we run on a grid environment with 5 available compute resources. We compared the results of our hybrid solution with those obtained with the following scheduling algorithms: DPSO* and HEFT. Table 3 shows the values of the makespan as obtained in the different situations.

Table 3. Makespan comparison according to the number of tasks.

Task number	HEFT/DPSO*	DPSO*	HEFT
10	71	71	84
20	225	228	299
40	431	411	441
60	446	511	654
80	434	573	523
100	681	849	701

Figure 3 depicts the makespan comparison between different scheduling algorithms when varying the number of tasks. Even when varying the number of tasks, the makespan seems to be better especially when the number of tasks is relatively important.

According to Figs. 2 and 3, we can notice that the performances of the hybrid HEFT/DPSO* solution and those of the DPSO* are better than HEFT heuristic regarding the makespan criterion. Moreover, in the majority of cases, the hybrid solution provides better makespans than the DPSO*.

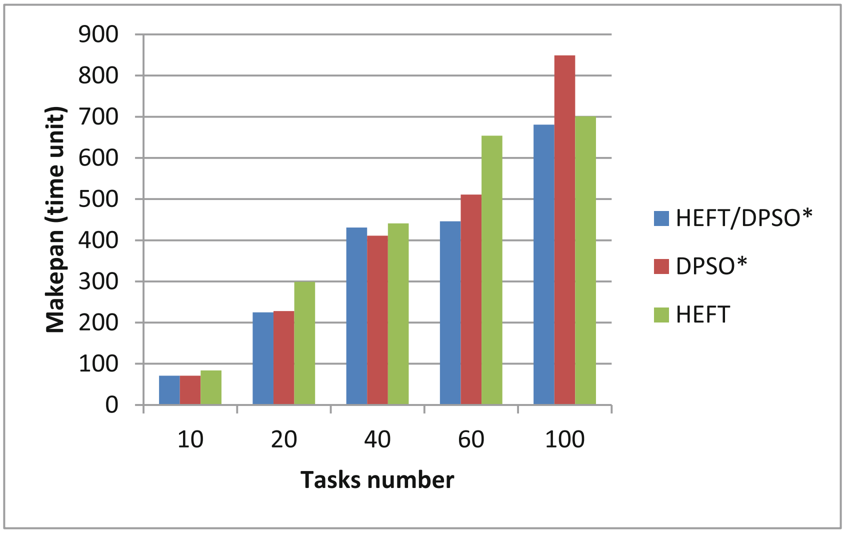


Fig. 3. Makespan comparison according to the number of tasks

6 Conclusion and Future Work

In this paper, a dependent task scheduling algorithm for computational grid has been proposed based on hybridization of two heuristics. The first one is a list-scheduling heuristic which is the well-known HEFT used to schedule DAGs. The second one is based on a meta-heuristic called Particle Swarm Optimization (PSO). A discrete version of PSO has been adapted to handle the scheduling of DAGs. Our objective was to minimize the makespan of applications that are executed on a grid environment. However, we plan to measure other criteria like energy consuming in our future work.

We showed in this article that our proposed scheduling approach gives better results in term of completion time than HEFT and our DPSO*.

At last, we aim to complete our current research work by hybridizing the DHEFT algorithm with a PSO technique to support the duplication of tasks in one side, and in the other side to measure the impact of the clustering approach that groups inter-dependent tasks into meta-tasks by using a genetic algorithm and then by scheduling them using HEFT algorithm.

References

1. Cafaro, M., Aloisio, G.: Grids, Clouds, and Virtualization. 1st edn., Spring (2011). ISBN 978-0-85729-049-6
2. Dong, F., Akl, S.G.: Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical report No. 2006-504. School of Computing, Queen's University, Kingston, Ontario

3. Casavant, T., Kuhl, J.: A taxonomie of scheduling in general-purpose distributed computing systems. IEEE Trans. Softw. Eng. **14**(2), 141–154 (1988)
4. Braun, R., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., Hensgen, D., Freund, R.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. **61**(6), 810–837 (2001)
5. Kwok, Y.K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. **31**(4), 406–471 (1999)
6. Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Distributed Computing Environments. Studies in Computational Intelligence, vol. 146, pp. 173–214. Springer, Heidelberg (2008)
7. Topcuoglu, H., Hariri, S., Wu, M.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
8. Radulescu, A., van Gemund, A.J.C.: On the complexity of list scheduling algorithms for distributed-memory systems. In: Technical report No. 1-68340-44(1999)02, January 1999
9. Kwok, Y., Ahmad, I.: Dynamic critical-path scheduling: an effective technique for allocating task graphs to muliprocessors. IEEE Trans. Parallel Distrib. Syst. **7**(5), 506–521 (1996)
10. Sih, G.C., Lee, E.A.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans. Parallel Distrib. Syst. **4**(2), 75–87 (1993)
11. Ma, T., Buyya, R.: Critical-path and priority based algorithms for scheduling workflows with parameter sweep tasks on global grids. In: IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2005) (2005)
12. Yang, T., Gerasoulis, A.: DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans. Parallel Distrib. Syst. **5**(9), 951–967 (1994)
13. Liou, J., Palis, M.A.: An efficient clustering heuristic for scheduling DAGs on multiprocessors. In: Proceedings of the Symposium Parallel and Distributed Processing (1996)
14. Boeres, C., Filho, J.V., Rebello, V.E.F: A cluster-based strategy for scheduling task on heterogeneous processors. In: IEEE Symposium on Computer Architecture and High Performance Computing, pp. 214–221, October 2004
15. Kruatrachue, B., Lewis, T.: Grain size determination for parallel processing. IEEE Softw. **5**, 23–32 (1988)
16. Ahmad, I., Kwok, Y.-K.: A new approach to scheduling parallel programs using task duplication. In: IEEE International Conference on Parallel Processing, vol. 2 (1994)
17. Darbha, S., Agrawal, D.P.: Optimal scheduling algorithm for distributed-memory machines. IEEE Trans. Parallel Distrib. Syst. **9**(1), 87–95 (1998)
18. Chung, Y.-C., Ranka, S.: Application and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed-memory multiprocessors. In: Proceedings of the Supercomputing, pp. 512–521 (1992)
19. Bajaj, R., Agrawal, D.P.: Improving scheduling of tasks in a heterogeneous environment. IEEE Trans. Parallel Distrib. Syst. **15**(2), 107–118 (2004)
20. Wang, L., Siegel, H.J., Roychowdhury, V.P., Maciejewski, A.A.: Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. J. Parallel Distrib. Comput. **47**(1), 8–22 (1997)
21. Martino, V.D., Mililotti, M.: Sub optimal scheduling in a grid using genetic algorithms. Parallel Comput. **30**, 553–565 (2004)

22. Gao, Y., Rong, H., Huang, J.Z.: Adaptive grid job scheduling with genetic algorithms. Future Gener. Comput. Syst. **2**, 151–161 (2005)
23. Aggarwal, M., Kent, R.D., Ngom, A.: Genetic algorithm based scheduler for computational grids. In: Proceedings of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS 2005), May 2005
24. Song, S., Kwok, Y., Hwang, K.: Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling. In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005), April 2005
25. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ (1995 in press)
26. Zhang, L., Chen, Y., Sun, R., Jing, S., Yang, B.: A task scheduling algorithm based on PSO for grid computing. Int. J. Comput. Intell. Res. **4**(1), 37–43 (2008)
27. Liu, H., Abraham, A., Hassanien, A.E.: Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. Future Gener. Comput. Syst. **26**, 1336–1343 (2010)
28. Izakian, H., Ladani, B.T., Abraham, A., Snasel, V.: A discrete particle swarm optimization approach for grid job scheduling. Int. J. Innovative Comput. Inf. Control **6**(9), 4219–4233 (2010)
29. Zhang, Y.-Y., Inoguchi, Y., Shen, H.: A dynamic task scheduling algorithm for grid computing system. In: Cao, J., Yang, L.T., Guo, M., Lau, F. (eds.) ISPA 2004. LNCS, vol. 3358, pp. 578–583. Springer, Heidelberg (2004)
30. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, vol. 5 (1997)

Toward Scheduling I/O Request of Mapreduce Tasks Based on Markov Model

Sonia Ikken[1,2](✉), Éric Renault[1,2], M. Tahar Kechadi[3], and Abdelkamel Tari[4]

[1] Institut Mines-Télécom – Télécom SudParis, Évry, France
[2] Laboratoire Samovar UMR CNRS 5157, Évry, France
{sonia.ikken,eric.renault}@telecom-sudparis.eu
[3] UCD School of Computer Science and Informatics, Dublin, Irlande
[4] University of Abdarahmane Mira, Bejaia, Algeria
tahar.kechadi@ucd.ie, tarikamal59@gmail.com

Abstract. In Cloud storage of multiple CPU cores, many Mapreduce applications may run in parallel on each compute node and collocate with local Disks storage. These Disks storage are shared by multiple applications that use full CPU power of the node. Each application tends to issue contiguous I/O requests in parallel to the same Disk; however if large number of Mapreduce tasks enters the I/O phase at the same time, the requests from the same task may be interrupted by the requests of other tasks. Then, the I/O nodes receive these requests as non-contiguous way under I/O contention. This interleaved access pattern causes performance degradation for Mapreduce application, this is particularly important when writing intermediate files by multiple tasks in parallel to the shared Disk storage. In order to overcome this problem, we have proposed approach for optimizing write access for Mapreduce application. The contributions of this paper are: (1) analyze the open issues on scheduling access request of Mapreduce workload; (2) propose framework for scheduling and predicting I/O request of Mapreduce application; (3) describe each role of component that intervenes in the scheduling theses I/O request on Block-level of storage server to provide contiguous access.

Keywords: Mapreduce · Cloud storage · Disk I/O · Markov model · Scheduling algorithm

1 Introduction

Cloud computing has become a viable, mainstream solution for data processing, storage and distribution. A cloud environment allows sharing of distributed resources, like CPU, Network and Disk based on virtual machine, these resources must be used efficiently for performing data intensive application. Cloud storage systems use modern server nodes based on Disk storage that contains multiple Disk drives and buffers memory. These typically have many CPU cores; it is

© Springer International Publishing Switzerland 2015

DOI: 10.1007/978-3-319-25744-0_7

desirable to use increased level of parallelism on each node to be able to use the full CPU power of the node. In distributed computing such as those running in Mapreduce [1] and using its open source implementation Hadoop [2] or similar model are often dominated by I/O bound, particularly reading and writing operations conducted on Disk storage that are limited than the CPU resources, theses Disk storage are shared by multiple processes or number of parallel tasks (or job) and alternate between computation and I/O phases.

A Mapreduce application consists of many maps and reduces tasks that read and write data on distributed file system and local Disks. Each map task has a memory buffer that it writes the output to. The buffer is 100 MB by default, it is a size which can be tuned by changing the Hadoop parameters. Each time the memory buffer reaches a certain threshold, then a new intermediate file is created.

In order to use multi-core systems, Hadoop schedules multiple tasks to run simultaneously on each node. In this case, the I/O resources of node are divided between these tasks and are exposed to I/O contention. Particularly, this situation affect write request of parallel tasks which remains on each Buffer befor reaching the threshold to the shared Disk storage. For example, parallel tasks slot often write intermediate data from each buffer to the Disk, each task tends to issue contiguous write I/O requests, but if multiple tasks issue many I/O requests managed by buffers simultaneously in one node, the requests are handled by local Disk in a non-contiguous way. This interleaved access pattern increases read I/O cost and I/O bandwidth falls, and can cause intermediate data fragmentation in many local file system. This means that Hadoop does not have a control over the allocation of physical blocks for intermediate files, and it can not effectively manage all buffers of multiple tasks, but is dependent on the kernel I/O scheduler and allocation strategy used by file system driver.

Particularly those that do not support optimization used at the OS kernel (Buffer memory policy, scheduling queue Disk) [3,4] cannot take advantage of the properties and behaviors of each particular application and are therefore not able to address the overall efficiency of the system. As the size of the system continues to increase, planners must have a global view of I/O operation needs of all applications in order to make appropriate allocation decision.

In order to solve this problem, we have proposed a framework for scheduling I/O request of parallel Mapreduce task, and Markov model [5] for predicting non contiguous write access to improve I/O access of Mapreduce task on shared Disks.

The remainder of the paper is organized as follows. In Sect. 2, we outline the related work and open issues that directs our research. In Sect. 3, we describe the proposed approach for scheduling write access of Mapreduce application, and the role of each component in this framework. Finally, in Sect. 4 we conclude and summarize our plans for future work.

2 Motivation Example

In this section we show an illustrative example for the restraints of the problem of Disk I/O contention during Mapreduce workload processing. The effect of I/O contention is dependent on the workflow and I/O patterns used by the Mapreduce application under I/O phases. During Mapreduce workload execution, map and reduce tasks go through a number of phases of execution. If multiple tasks are running at the same time, it is not guaranteed that all those tasks are executing the same phase at the same time as long as each task execute his own I/O phase and form a life cycle throughout the process Mapreduce. However, the tasks are not always executing the same phase is that tasks are typically not identical in duration, but even for tasks that are normally very similar in execution time this situation can occur because of the variance in task execution time caused by I/O contention that are observed in Fig. 2.

During the shuffle phase, reduce task input is divided into segments, with one segment being read from the output files of each map task spread over the entire cluster. While each segment itself is read sequentially, the set of segments for a particular reduce task is not stored sequentially and not guaranteed to be read in any particular order. This situation occurs when writing intermediate data to the shared Disk. For the spill phase if large number of tasks enters the I/O phase at a same time, the requests from the same task may be interrupted by the requests of other tasks. Then I/O nodes receive these requests as non-contiguous way under I/O contention.

3 Background

3.1 Mapreduce Programming Model

Mapreduce [1] uses a divided-and-conquer approach in which input data are divided into fixed size units processed independently and in parallel by tasks, which are executed distributedly across the nodes in the cluster.

Mapreduce applications consists of a map function and a reduce function. As shown in Fig. 1, the input to an application is organized in records, each of which is a $< \text{k1, v1} >$ pair. The map function processes all records one by one,

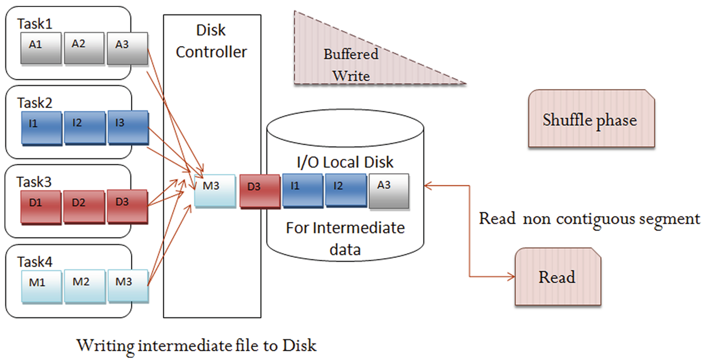


Fig. 1. Disk sharing issue

and for each record outputs a list of zero or more < k2, v2 > records called intermediate data. The map output is stored in an in-memory buffer; when this buffer is almost full then it start (in parallel) the spilling phase in order to remove data from it to the Disk. Then all intermediate data with < k2, v2 > records are collected and reorganized so that records with the same keys (k2) are put together into and shuffled a < k2, list (v2) > record. These < k2, list (v2) > records are then processed by the reduce function one by one, and for each record the reduce function outputs a < k2, v3 > pair. All < k2, v3 > pairs together coalesce into the final result. Map and reduce functions can be summarized in the following equations:

$$\text{map}(< \text{k1}, \text{v1} >) \rightarrow \text{list}(< \text{k2}, \text{v2} >) \ (1)$$
$$\text{reduce}(< \text{k2}, \text{list}(\text{v2}) >) \rightarrow < \text{k2}, \text{v3} > \ (2)$$

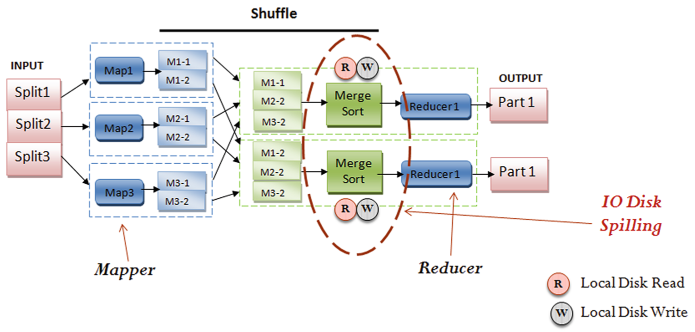


Fig. 2. Structure of map and reduce task in Hadoop

3.2 Linux Disk Scheduler

Disk scheduler are typically work-conserving, since they select a request for services as soon as (or before) the previous request has completed. Since 2.6 version Linux provides four non-conserving Disk I/O schedulers: deadline, anticipatory, noop, and completely fair queuing (CFQ), along with an option to select one of these four at boot time or runtime. The selection is based on a priori knowledge of the workload, file system, and I/O system hardware configuration, among other factors. The anticipatory scheduler (AS) is the default for 2.6 version. Now in our context, we consider process issuing Disk read request synchronously which each process issues a new request shortly after its previous request has finished, and thus maintains at most one outstanding request at any time. "Deceptive idleness" is a situation where a process appears to be finished reading from the Disk when it is actually processing data in preparation of the next read operation. This will cause a normal work-conserving I/O scheduler to switch to servicing I/O from an unrelated process. This situation is detrimental to the throughput of synchronous reads, as it degenerates into a seeking workload. In this situation the Anticipatory scheduler [8] performs well, and it is based on two assumptions: (1) synchronous Disk requests are issued by individual processes and, thus, anticipation occurs only with respect to the process that issued the last request; and (2) for anticipation to work properly, the anticipated process must be alive; if the anticipated process dies, there is no further anticipation for requests to nearby sectors. Instead, any request that arrives at the scheduler is scheduled for dispatch, irrespective of the requested head position and the current head position.

4 Related Work and Open Issues

There are lots of work on Modeling access behavior and scheduling Mapreduce applications in Cloud environment. We classify the current trends and exiting approaches with respect to traditional scheduling algorithm, and presents open issues that direct our research.

4.1 Modeling I/O Behavior of Mapreduce Workload Without I/O Contention

Although in recent years, there has been an increasing amount of work in modeling the I/O behavior of Mapreduce workload. In [9–14] the authors focusing to reduce the total amount of I/O performed by the application. In [14], the authors model the read I/O behavior of map tasks. In [12], the authors propose a statistical model to evaluate the effect of various configuration parameters of Hadoop-Mapreduce job.

4.2 Scheduling Task, Job and VM Under I/O Contention

The authors propose techniques to address the I/O stream contention in Mapreduce tasks. In [15], they propose to limit the number of concurrent I/O streams, and alleviate the I/O contention by orders the I/O streams in accordance to job priority. In [16], they propose competing applications' I/O by interposing HDFS I/O and use an SFQ-based proportional-sharing algorithm. In [17–19] the authors characterize and predict I/O performance under I/O contention, which focuses primarily on the hardware environment. In [20], they propose a scheduling system that takes I/O contention into account, but it applies to VM scheduling rather than I/O access scheduler.

4.3 Scheduling I/O Access Disk Without I/O Contention

Each I/O scheduler algorithm is performed at the top level, which the I/O request reordering and merging is performed by the filesystem driver, and at the lower level, which the I/O requests are reordered by the I/O device. Disk scheduler plays main role in the service of I/O operation, in [7,21] they trie to optimize the movement of the Disk head to specific goals under limit condition, but each has difficulties of predicting these movements. Recently, non-conservation works of Disk schedulers, such as CFQ scheduler [7] and Anticipatory scheduler [8] were designed to record the spatial location with concurrent services of interleaved requests from multiple processes. This strategy takes the Disk head slowed after serving a request from a process until the next request from the same process happens or waiting threshold expires.

4.4 Open Issues

Our researches are focuses to analyze the problem of scheduling intermediate data for parallel Mapreduce tasks under I/O contention. Our work differ from others existing approaches, especially in modeling access behavior and scheduling Mapreduce application on block level of shared Disks nodes.

(1) The works we've cited above, make effort on modeling the I/O performance of Mapreduce, the impact of I/O contention on access request is not considered.

(2) The works presented above about mitigating I/O contention focuse to coordinate Mapreduce application at high level, rather than at block level.

(3) Unlike traditional Disk scheduler problems, scheduling I/O workload for Mapreduce application is even more complicated, and it should meet the algorithms of access requests to the Disk, and they take into account information related to the type of workload behavior and traditional Disk scheduler policy.

Table 1, depicts the summarize of the analysis and comparison of the related work based on the criteria that characterizes our orientation.

Table 1. Feature comparison of related work

Related work	Mapreduce workload	Modeling access behavior	I/O contention	Block level scheduler	High level scheduler
[9–14]	X	X			
[15, 16, 19, 20]	X	X			X
[7, 8]				X	
Our focus research	X	X	X	X	

5 Proposed Approach

We have proposed an approach that defines a framework to anticipate non-contiguous write request of Mapreduce workload on shared Disk storage. Mapreduce phases are sequential write-only and read-only subtask, and write access is done in parallel using a separate thread from in memory buffer to local Disk in round-robin fashion. We develop a methodology to characterize the interleaved

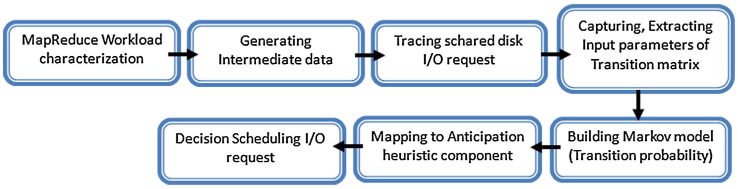

MapReduce Workload characterization
Generating Intermediate data
Tracing schared disk I/O request
Capturing, Extracting Input parameters of Transition matrix
Building Markov model (Transition probability)
Mapping to Anticipation heuristic component
Decision Scheduling I/O request

Fig. 3. A Framework for Scheduling write access of parallel MapReduce Tasks

write access submitted from parallel Mapreduce task to the shared Disk, and uses anticipation support to decided when these I/O request will be allocated on the Disk to avoid non-contiguous access, Fig. 1 show the proposed approach. On our study, we need to modeling write access at block-level to make out the non-contiguous write I/O for guiding future scheduler decision to improve access of Mapreduce tasks.

5.1 Markov Model Prediction

To capture non-contiguous write I/O requests, we have model parallel write access streams for Mapreduce tasks under IO contention using Markov model; the choice of model description is critical to its predictive ability. One must determine what application behavior corresponds to a state s, the total number of states N that can be fully described by its transition probability matrix P, and the allowed observations. Given the present state and all past states, if the future state of the system depends only on the present state, the system is said to have the Markov property. In the context of Disk access request for Mapreduce task slot, one can build a transition probability matrix by sampling the state of the Disk system at regular intervals during Mapreduce job processing. We generate I/O trace from Mapreduce workload processed on same Disk storage. We construct a Markov model where each state corresponds to write I/O request at time t. Each state can have one or multiple part of non-contiguous I/O request related to one task; that means the state can be contiguous or non-contiguous write I/O request of a single task. This is because; request can be interrupted by the request from other task in Mapreduce sub phases. The non-contiguous write I/O requests are sequential access on block regions of variable size. Ideally, write request size is chosen to correspond to a split size from HDFS (Hadoop

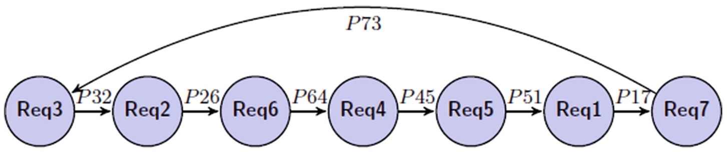


Fig. 4. Markov chain of order m describing contiguous Write I/O request

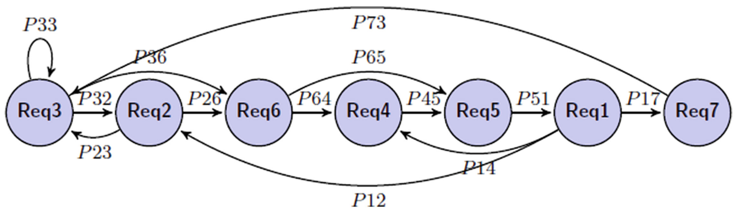


Fig. 5. Markov chain of order m describing Interleaved Write request under I/O contention

Distributed FileSystem). In the context of this paper, we assume that there is only one phase cycle of Mapreduce, then a request size may be more than or equal to 64 Mbyte, and the Hadoop parameters are fixed for each training set of action-sequences. Therefore, the state number is the number of write I/O request of parallel task slot, and If one tasks slot has 64 Mbyte intermediate data size, access to this data region could be modeled by a 64 states Markov model with different sizes of each access.

Observations are write request under I/O contention that changes the current state with some probability to a new state that reflects a new current non-contiguous write request; this is the interleaved write access of each task slot. We process a trace of I/O request so those contiguous write requests that access sequentially the current block region on Disk does incur reflexive transition. Figure 4 illustrates the write access pattern which is deterministic; each task slot tends to issue contiguous write I/O access, but Fig. 5 illustrates a pattern where there is an interleaved write request of two or more task slot since multiple task issue many write I/O request simultaneously on Datanode. This interleaved access pattern can cause intermediate data fragmentation, increases read IO overhead (Fig. 5).

5.2 Scheduling Non-contiguous Write Access Based on Markov Model

Many Disk scheduler algorithm have been proposed that achieved performance applications by taking into account information about characteristic of each individual I/O request and the current state of Disk subsystem [6,7]. In this paper, we use the Anticipatory Disk scheduler [8]. The idea behind this algorithm is to anticipate which streams are most likely to make their deadlines for synchronous read and which are not, based on the estimated supply and demand of time slots in the future. In some situation, the scheduler can cause undesirable delays to IO requests in other environments, particularly when there is update or delayed write like Mapreduce phases. Therefore, fast phase read response is disrupted by interfering writes request in Mapreduce phases. For this situation our idea is having control of non-contiguous write request to improve read access from Mapreduce phases, and use Markov model for predicting these non-contiguous write request.

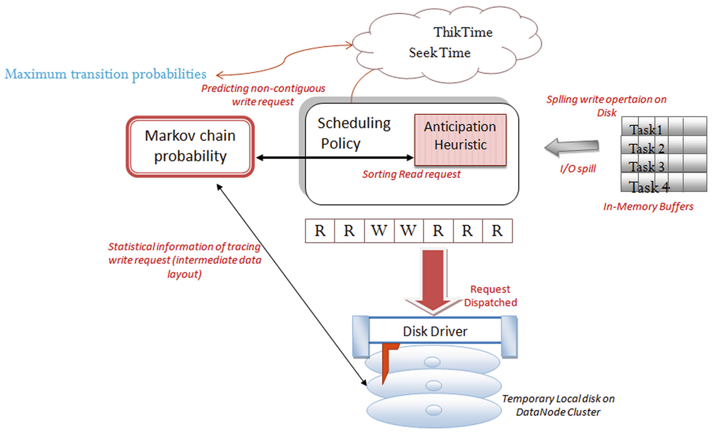

ThikTime
SeekTime
Maximum transition probabilities
Predicting non-contiguous write request
Markov chain probability
Scheduling Policy
Anticipation Heuristic
Sorting Read request
Splling write opertaion on Disk
I/O spill
Task1
Task 2
Task 3
Task 4
In-Memory Buffers
R R W W R R R
Statistical information of tracing write request (intermediate data layout)
Request Dispatched
Disk Driver
Temporary Local disk on DataNode Cluster

Fig. 6. Adaptive anticipatory scheduler based on Markov model

Then, we propose an adaptive Anticipatory scheduler based on Markov model. The challenge of our heuristic is to know that already serviced requests will have other sub-requests and how to estimate the think time for each read and write requests of all task. Given a write I/O access Markov model, it still does not want to delay write requests indefinitely, however, the buffer ensure that data is eventually written to Disk to prevent in-memory buffers from growing too large or reached a threshold. Also, there is I/O access which is contiguous or proximate requests, and it is not necessary to wait for these requests to serve immediately. This optimizes the overall think time for the scheduler and decrease the overlapping of parallel write request from buffers when allocating on shared Disk storage. Markov model parameters which are the transition probability matrix used in a simulated anticipatory algorithm to find a permutation of the buffered write that substantially does not penalize the future Read operation and reduces expected seek distance of IO request (Fig. 6).

The transition probability matrix by itself is need for predicting non-contiguous write request on Disk device after intermediate data buffered in Round robin fashion. Then, the scheduler calculate seek distance for each available request and it predicts the next state for keeps Disk idle with an estimate think time by using the information maintained by means of the transition probability matrix. For this purpose, the scheduler chooses a sequence of request by repeatedly finding the most likely transition from the current state s to have non-contiguous request from the same task. This approach builds on N-step transition models from Markov theory. The sequence of predicted non-contiguous request stop when a specified number of requests is reached, i.e. when the total probability of the sequence drops below a given threshold. Then, Anticipatory scheduler meets all the non-contiguous sequences found to calculate think time for read and write request giving priority for read operation. It interacts with the transition probability matrix based Anticipatory heuristic to decide if and how long to wait for each task slot. Figure 4 show the interaction between the various component.

5.3 Training Markov Chain

Each application/job has one Markov chain per workload and hardware parameters. These models are trained online by running a Mapreduce workload, and then generating trace of block spill file based on trace tool, which has been linked with a training module. The Markov chain segment size is selected in advance based on the block segment size, and the training module maintains counts of all record block write transition. Most spill file (intermediate data) involves only a single access pattern; in this case, a degenerate Markov chain with one state per block of spill file suffices to model them and training is a trivial calculation. The probabilities for each transition are calculated by dividing the number of occurrences of the transition by the total number of transitions from each block, and only a single training execution is required. The examples in this paper utilize this training algorithm. We implemented Markov chain with the C ++ language and used representative benchmarks hosted on a single node cluster: Terasort and Wordcount benchmarks, we generated 40 GB and 20 GB of intermediate data respectively on Hadoop 1.0.3 version on Ubuntu Linux 14.04 machine. The setting parameters of Hadoop are 4 CPU, 8G RAM and 4 Disks of 1TB) for each training set of sequences with 4 tasks Map and 3 Tasks Reduce in parallel. We stressed a Linux system with blktrace to have Disk I/O of read/write operation during the execution of Mapreduce workload.

6 Conclusion

In this paper, we have proposed framework for scheduling write access of parallel Mapreduce application under I/O contention on block-level. We focused on the issues that occur when multiple applications are running in parallel on a shared node in order to take advantage of multiple CPU cores. To characterize I/O request, we modeled the write access of Mapreduce task based on Markov model to predict the non-contiguous write operation. The model uses knowledge about Mapreduce application by tracing I/O access on lower level from shared Disk, and it is used for making decision to scheduling these accesses on Disk queue. For this purpose, we used Anticipatory scheduler to sort non-contiguous write request from buffers to the Disk using transition probability matrix. This proposed framework is an approach that describe different components that play a key role to scheduling I/O access of intermediate data on block-level, and to decide if using the hinted sequence of transition probability matrix for future scheduler is likely to provide benefit. For future work, we will estimate the prediction accuracy of Markov model, and describe our implementation for possible evaluation.

Acknowledgments. Work funded by the European Commission under the Erasmus Mundus GreenIT project (GreenIT for the benefit of civil society. 3772227-1-2012-ES-ERAMUNDUS-EMA21; Grant Agreement n 2012-2625/001-001-EMA2).

References

1. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
2. Apache Hadoop Core. http://hadoop.apache.org/core
3. Zhang, X., Davis, K., Jiang, S.: Opportunistic data-driven execution of parallel programs for efficient I/O services. In: Proceedings of IPDPS12, pp. 330–341. IEEE (2012)
4. Lofstead, J., Zheng, F., Liu, Q., Klasky, S., Oldfield, R., Kordenbrock, T., Schwan, K., Wolf, M.: Managing variability in the IO performance of petascale storage systems. In: Proceedings of SC10. IEEE Computer Society (2010)
5. Ching, W.-K., Ng, M.K.: Markov Chains: Models Algorithms and Applications. Springer, US (2006)
6. Filip, B., Cyril, G., Qingbo, W., Timothy, T.: Priority IO scheduling in the cloud. In: Proceeding of HotCloud 2013, the 5th USENIX Workshop on Hot Topics in Cloud Computing (2013)
7. Prashant, T., Sushma, S.: A development approach towards self learning schedulers in Linux. Proc. Int. J. Recent Innov. Trends Comput. Commun. **2**(4), 814–819 (2014)
8. Iyer, S., Druschel, P.: Anticipatory scheduling: a disk scheduling framework to overcome deceptive idleness in synchronous I/O. In: ACM Symposium on Operating Systems Principles (SOSP 2001) (2001)
9. Kambatla, K., Pathak, A., Pucha, H.: Towards optimizing hadoop provisioning in the cloud. In: Proceeding of HotCloud. USENIX, Berkeley (2009)
10. Huai, Y., Lee, R., Zhang, S., Xia, C.H., Zhang, X.: DOT: a matrix model for analyzing, optimizing and deploying software for big data analytics in distributed systems. In: Proceeding of SOCC, pp. 4:1–4:14. ACM, New York (2011)
11. Jahani, E., Cafarella, M.J., Ré, C.: Automatic optimization for MapReduce programs. Proc. VLDB Endow **4**(6), 385–396 (2011)
12. Yang, H., Luan, Z., Li, W., Qian, D.: MapReduce workload modeling with statistical approach. J. Grid Comput. **10**, 279–310 (2012). doi:10.1007/s10723-011-9201-4
13. Herodotou, H.: Hadoop performance models, Technical report, Duke University (2010). http://www.cs.duke.edu/starfish/files/hadoop-models.pdf
14. Jindal, A., Quiané-Ruiz, J.-A., Dittrich, J.: Trojan data layouts: right shoes for a running elephant. In: Proceeding of SOCC, pp. 21:121:14. ACM, New York (2011)
15. Siyuan, M., Xian-He, S., Ioan, R.: I/O Throttling and Coordination for MapReduce. Technical Report, Illinois Institute of Technology (2012)
16. Yiqi, X., Adrian, S., Ming, Z.: IBIS: interposed big-data I/O scheduler. In: Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing, pp. 109–110. ACM (2013)
17. Pu, X., Liu, L., Mei, Y., Sivathanu, S., Koh, Y., Pu, C.: Understanding performance interference of I/O workload in virtualized cloud environments. In: Proceeding of CLOUD, pp. 51–58 (2010)
18. Mesnier, M.P., Wachs, M., Sambasivan, R.R., Zheng, A.X., Ganger, G.R.: Modeling the relativetness of storage. In: Proceeding of SIGMETRICS, pp. 37–48. ACM, New York
19. Gulati, A., Shanmuganathan, G., Ahmad, I., Waldspurger, C., Uysal, M.: Pesto: online storage performance management in virtualized datacenters. In: Proceeding of SOCC, pp. 19:1–19:14. ACM, New York (2011)

20. Chiang, R., Huang, H.: TRACON: interference-aware scheduling for data-intensive applications in virtualized environments. In: Proceedings of SC, pp. 1–12 (2011)
21. Celis, J.R., Gonzales, D., Lagda, E., Rutaquio Jr., L.: A comprehensive review for disk scheduling algorithms. Int. J. Comput. Sci. Issues (IJCSI) **11**(1), 74 (2014)

Improving the Reliability and the Performance of CAPE by Using MPI for Data Exchange on Network

Van Long Tran[1(✉)], Éric Renault[1], and Viet Hai Ha[2]

[1] Institut Mines-Telecom – Telecom SudParis, Évry, France
{van_long.tran,eric.renault}@telecom-sudparis.eu
[2] College of Education, Hue University, Hue, Vietnam
haviethai@gmail.com

Abstract. CAPE — which stands for Checkpointing Aided Parallel Execution — has demonstrated to be a high-performance and compliant OpenMP implementation for distributed memory systems. CAPE is based on the use of checkpoints to automatically distribute jobs of OpenMP parallel constructs to distant machines and to automatically collect the calculated results on these machines to the master machine. However, on the current version, the data exchange on networks use manual sockets that require time to establish connections between machines for each parallel construct. Furthermore, this technique is not really reliable due to the risk of conflicts on ports and the problem of data exchange using stream. This paper aims at presenting the impact of using MPI to improve the reliability and the performance of CAPE. Both socket and MPI implementations are analyzed and discussed, and performance evaluations are provided.

Keywords: CAPE · OpenMP · MPI · High-performance computing · Parallel programming

1 Introduction

In order to explore further the capabilities of parallel computing architectures such as grid, cluster, multi-processors and multi-cores, an easy-to-use parallel programming language is an important factor.

MPI [1] (which stands for Message Passing Interface) is the de-facto standard for developing parallel applications on distributed-memory architectures. Essentially, it provides point-to-point communications, collective operations, synchronization, virtual topologies, and other communication facilities for a set of processes in a language-independent way, with a language-specific syntax, plus a small set of language-specific features... Although, it is capable of providing high performance, it is difficult to use. MPI requires the programmers to explicitly distribute the program onto the nodes. Moreover, some operations, like sending and receiving data or the synchronization of processes, must be explicitly specified in the program.

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 90–100, 2015.
DOI: 10.1007/978-3-319-25744-0_8

OpenMP [2] also has become a standard for the development of parallel applications but on shared-memory architectures. It is composed of a set of very simple and powerful directives and functions to generate parallel programs in C, C++ or Fortran. From the programmer's point of view, OpenMP is easy to use as it allows to incrementally express parallelism in sequential programs, i.e. the programmer can start with a sequential version of a program and step by step add OpenMP directives to change it into a parallel version. Moreover, the level of abstraction provided by OpenMP makes the expression of parallelism more implicit where the programmer specifies what is desired rather than how to do it. This has to be compared to message-passing libraries, like Message Passing Interface (MPI) [1], where the programmer specifies how things must be done using explicit send/receive and synchronization calls.

Because of these advantages of OpenMP, there have been some efforts to run OpenMP programs on distributed-memory systems. Among them, CAPE [3,4] is a tool to compile and provide an environment to execute OpenMP programs on distributed-memory architectures. This solution provides both high performance and a compiler that is fully-compatible with the OpenMP standard.

In order to automatically distribute jobs onto slave nodes of a distributed-memory system, CAPE follows the following algorithm: when reaching a parallel section, the master thread is dumped and its checkpoint is sent to slaves; then, each slave executes a different thread of the parallel section; at the end of the parallel section, each slave extracts and returns the list of all modifications that has been locally performed to the master thread; the master then includes these modifications and resumes its execution.

In the current version of CAPE, data exchanged between nodes are computed using DICKPT [3,5] (which stands for Discontinuous Incremental Checkpoints), and are transferred over the network using manual sockets. However, initializing, connecting and listening to sockets at runtime is clearly a waste of time. In addition, this approach is weak in terms of reliability, due to the difficulty to manage the data exchanged over the network.

This paper aims at presenting the approach focusing on the reduction of the checkpoint's transfer time and increasing the reliability of data transfers of CAPE over the network. The remainder of the paper is as follows: first, some related works and the advantages of using MPI to transfer data over the network are listed in Sect. 2. Section 3 discusses and analyzes the current version of CAPE using manual sockets. Section 4 proposes a new method that use MPI instead of manual sockets. Section 5 compares the two methods by presenting an evaluation and some experimental results. At the end, Sect. 6 draws some conclusions and future works.

2 Related Works

Using the MPI framework to transfer data between nodes over the network has been developed and widely applied today. This allows to achieve high reliability, security, portability, integrity, availability and high-performance of the transferred data.

A typical example is the combination of MPI and OpenMP. In this case, the MPI framework is used to send data and code from the master node to all working nodes in the network. At the working node side, the OpenMP framework is used to execute the assigned task in parallel. Finally, the results from the working nodes are sent back to the master node by using explicit MPI codes. Although this way takes time and efforts from the programmer, it takes advantages of the performance and the integrity. In [6], authors show that this method can achieve high efficiency and scalable performance. In [7,8], authors show a reduction of the communication needs and memory consummation, or an improvement of the load-balancing ability.

They are also a lot of works that use advantage of MPI to assume the data exchange between accelerators on clusters. For example, the GPU-aware MPI [9] and CUDA Inter-process Communication [10] use the MPI standard to support data communication form GPU to GPU on clusters. This technique has demonstrated high-performance and portability of the system using MPI. In addition, on cloud, Cloud Cluster Communication [11] and ECC-MPICH2 [12] using a modified MPI framework have shown the validation of the security in terms of authentication, confidentiality, portability, data integrity and availability.

The result above is very important for the orientation of the future development of CAPE using MPI. In this paper, the MPI framework is used by CAPE to transfer checkpoints between nodes. In the future, MPI will bring an even more important contribution to CAPE as the latter aims at supporting GPU and cloud computing infrastructures in the near future.

Note that the use of MPI by CAPE as presented in this paper is completely different from the combination of MPI and OpenMP as mentioned above or from the translation of OpenMP constructs into MPI function calls. In fact, the use of MPI as a support for CAPE does not change the essence of CAPE. CAPE is based on the use of checkpointing technique to implement OpenMP on distributed systems. This implementation is fully compliant with the OpenMP standard and programmers do not need to modify their application program source codes. With CAPE, the role of MPI only consists in transferring checkpoints over the network, while for most other cases programmers need to modify their source codes and, as a consequence, cannot provide a fully-compliant implementation of OpenMP.

3 CAPE Based on Manual Sockets

In CAPE, each node consists in two processes. The first one runs the application program. The second one plays two different roles: the first one as a DICKPT checkpointer and the second one as a communicator between the nodes. As a checkpointer, it catches the signals from the application process and executes appropriate handles to create the DICKPT checkpoint. As a communicator, it ensures the distribution of jobs and the exchange of data between nodes. Figure 1 shows the basic principle of the CAPE organization.

In the current version, the master node is in charge of managing slave nodes and does not execute any application job in the parallel sections.

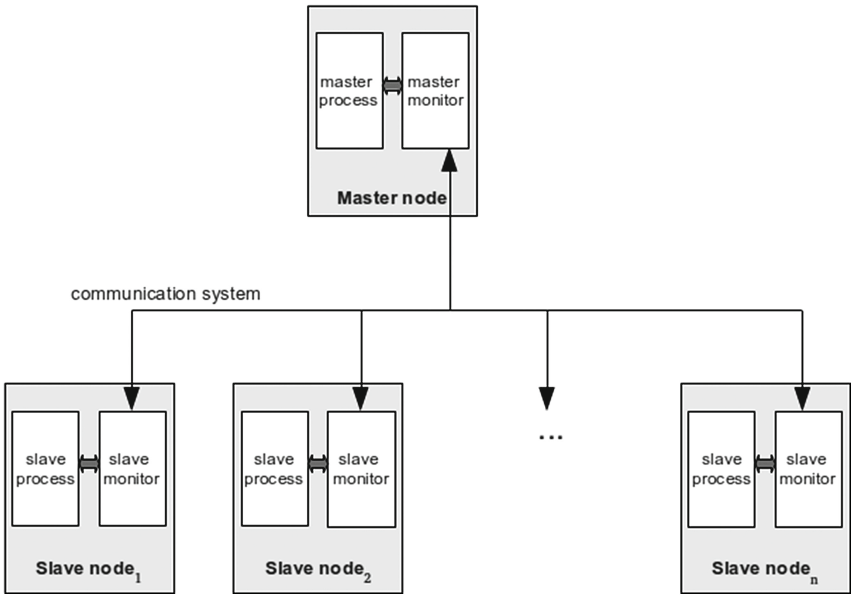


Fig. 1. CAPE organization.

3.1 Execution Model

CAPE is an alternative approach to allow the execution of OpenMP programs on distributed-memory systems. CAPE is based on a process as a parallel unit, which is different from the traditional implementations of OpenMP where the parallel unit is a thread. All the important tasks of the fork-join model are automatically generated by CAPE based on checkpointing techniques, such as task division, reception of results, updating results into the main process, etc. In its first version, CAPE used complete checkpoints so as to prove the concept. However, as the size of complete checkpoints is very large, it takes a lot of traffic on the network to transfer data between processes and involves a high cost for the comparison of the data from the different complete checkpoints to extract the modifications. These factors have significantly reduced the performance and the scalability of our solution. Fortunately, these drawbacks have been overcome in the second version of CAPE based on DICKPT.

Figure 2 describes the execution model of the second version of CAPE using three nodes. At the beginning, the program is initialized on all nodes and the same sequential code block is executed on all nodes. When reaching an OpenMP parallel structure, the master process divides the tasks into several parts and send them to slave processes using DICKPT. Note that these checkpoints are very small in size, typically very few bytes, as they only contain the results of some very simple instructions to make the difference between the threads, which do not change the memory space that much. At each slave node, after receiving a checkpoint, it is injected into the local memory space and initialized

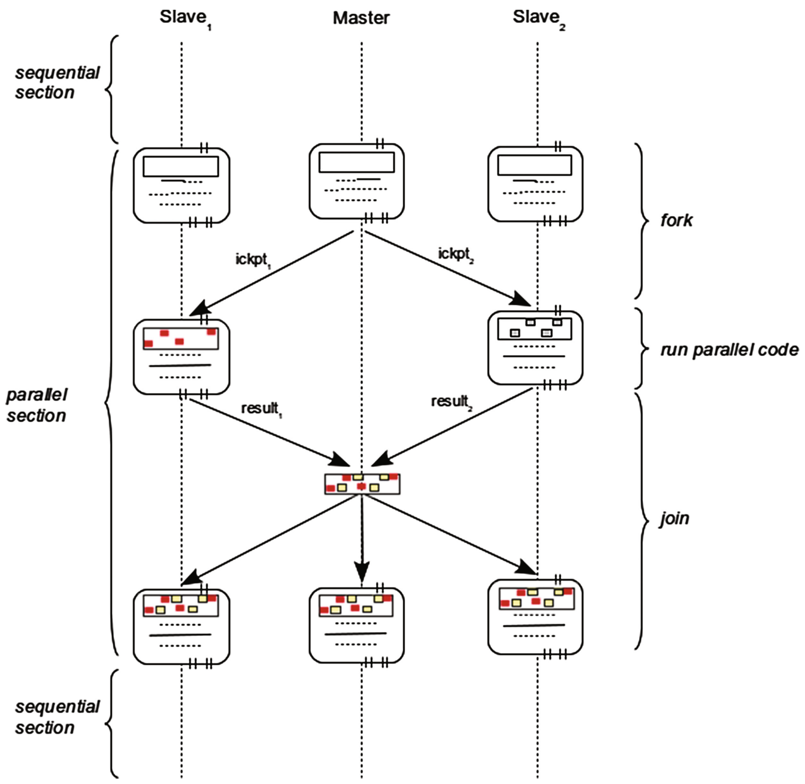


Fig. 2. Data transfer between nodes in CAPE.

for resuming. Then, the slave process executes the assigned task, extracts the result, and creates a resulting checkpoint. This last checkpoint is sent back to the master process. The master process then combines all resulting checkpoints together, injects the result into its memory space and sends it to all the other slave processes to synchronize the memory space of all processes and prepare for the execution of the next instruction of the program.

3.2 Data Transfer

In order to distribute checkpoints to slave nodes, the master node initializes a socket to listen to the connection requests from slaves. After the master accepts a connection request, it sends a checkpoint to the slave node through the established connection. Figure 3 presents the algorithm used to send checkpoints from the master to all slaves.

At the slave node side, a checkpoint must be returned to the master after the execution of the parallel part. The slave node initializes a client socket and tries to connect to the master. After the connection is accepted, the checkpoint is sent to the master.

```
if ( node == MASTER ) {
        initialize a server socket
        foreach ( slave ) {
                wait and accept a connection from a slave
                send a checkpoint to the slave
        }
} else {
        initialize a slave socket
        do {
                request the master for a connection
        } while ( ! connected )
        receive the checkpoint from the master node
        inject the checkpoint into the memory space
}
```

Fig. 3. Master-to-slave transfer using manual sockets.

```
if ( node == MASTER ) {
        foreach ( slave ) {
                wait and accept a connection from a slave
                receive a checkpoint through the socket
                inject the checkpoint into the memory space
        }
} else {
        initialize a client socket
        do {
                request the master for a connection
        } while ( ! connected )
        send the checkpoint through the socket
}
```

Fig. 4. Slave-to-master transfer using manual sockets.

To receive DICKPT checkpoints from the salves, the master initializes a server socket, accepts connections and receives data from the slaves the one after the one. At the other side, each slave always maintains a loop to request a connection to server before receiving data. The algorithm is summarized in Fig. 4.

From the two algorithms presented above, one can see that the use of manual sockets to send and receive data involves a waste of time to initialize and establish the connections between the nodes for each data exchange requirement. Furthermore, in order to request a connection to the master, the slave always performs a polling. This requires resources both on the node and over the network. In addition, transferring data by means of a stream using manual sockets is not reliable as the risk of conflicts on port numbers and data is not packaged.

4 CAPE Based on MPI

Nowadays, parallel programming on clusters have been dominated by message passing, and using MPI [13] has become a de-facto standard. MPI has demonstrated advantages over other systems (see Sect. 2). Moreover, for the case of MPI, data are transfered from the address space of one process to the one of another process through cooperative operations on each process. Simply stated, the goal of MPI is to provide a widely used standard for writing message-passing programs. The interface aims at being practical, portable, efficient and flexible.

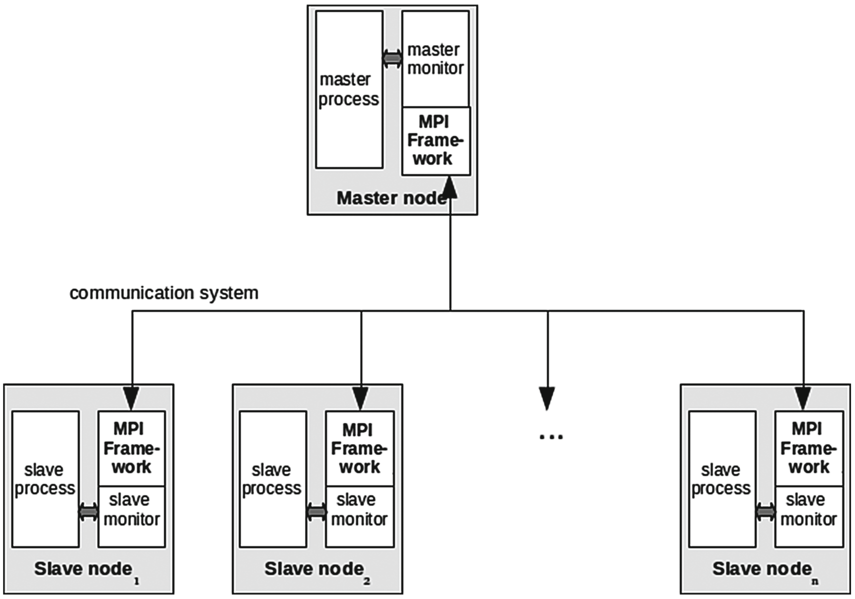


Fig. 5. MPI-based CAPE organization.

In order to take advantage of the MPI benefits, the organization of CAPE has been moved from a socket-based communication system to the MPI framework. The new organization of CAPE is shown in Fig. 5. With this new organization, the monitor process uses the MPI framework to send and receive DICKPT checkpoints. In addition, it also uses MPI routine to reduce the time overhead and improve the global reliably of the system.

4.1 Data Transfer

In order to provide a new version of CAPE on top of MPI, the sending and the receiving of data at both the master and the slaves nodes have been implemented as presented in pseudo-code on Figs. 6 and 7.

For this implementation, the MPI library is loaded by each node at the beginning of the execution, so that it is not necessary to initialize it when the nodes

```
if ( node == MASTER ) {
        current_slave_node ++
        MPI_Send ( current_slave_node, ... , DICKPT, ... )
} else {
         MPI_Recv ( 0, ..., DICKPT, ... )
        inject the checkpoint into the memory space
}
```

Fig. 6. Master-to-slave transfer using MPI.

```
if ( node == MASTER ) {
        foreach ( slave ) {
                MPI_Recv ( i, ..., DICKPT, ... )
                inject the checkpoint into the memory space
        }
} else {
         MPI_Send ( 0, ..., DICKPT, ... )
}
```

Fig. 7. Slave-to-master transfer using MPI.

need to send or receive data. Therefore, the execution time is reduced when compared with the manual-socket implementation. In addition, MPI automatically setup connections between nodes to perform data transfers which means that there is no need for maintaining a loop to request a connection from the slaves to the master. As a result, the use of the CPU and other resources is considerably reduced at this time.

Furthermore, the transfer of data using manual sockets requires the implementation of routines to send and receive data over the network, especially those routines that are very important to distribute and collect data, such as broadcast and reductions [14]. This requires a huge effort in terms of development and ensuring the reliability of such an implementation is not easy. Meanwhile, all these routines have been made available in the MPI framework and after many years of customization they are regarded as highly reliable and efficient [13]. Moreover, for the case of MPI, vendor implementations usually exploit native hardware features to optimize the performance [1]. For all these reasons, using MPI for sending and receiving data over the network is better than using manual sockets, especially when considering reliability and performance.

5 Experimentation and Evaluation

Let t_{comm} be the time to exchange data between the nodes, i.e. the total time for sending and receiving DICKPT checkpoints from the master to all slave nodes and vice versa. Let t_{comp} be the time to execute the application code at both the master and slave nodes. For the two methods mentioned in Sects. 3 and 4, t_{comp} in the same.

According to the execution model of CAPE as presented in Sect. 3, the execution time of a parallel section can be computed using Eq. (1).

$$t = t_{comm} + t_{comp} \tag{1}$$

Let p be the number of slave nodes, $t_{startup}$ be the time to set up a socket, i.e. the time to initialize, connect and prepare to send and receive data of each time when a checkpoint has to be exchanged, and t_{data} be the time to send and received data.

When using manual sockets as presented in Sect. 3, the time required to send and receive DICKPT checkpoints can be computed using Eq. (2).

$$t_{comm_i} = p(t_{startup} + t_{data}) \tag{2}$$

With MPI, the `scatter` operation has been used so that the startup step is executed at the same time on all nodes. As a result, the communication time for the sending or receiving phase can be computed using Eq. (3).

$$t_{comm_i} = t_{startup} + p.t_{data} \tag{3}$$

From Eqs. (2) and (3), one can see that each time a DICKPT checkpoint has to be sent or received, the communication time when using the MPI method is always more efficient than using manual sockets.

In order to verify the above arguments, some performance measurements have been conducted on a real cluster. The plateform is composed of nodes including four 3-GHz Intel(R) Pentium(R) CPUs with 2 GB of RAM, operated by Linux kernel 3.13.0 with the Ubuntu 14.04 flavour and connected by a standard 100 Mbits/s Ethernet. The cluster consists of three nodes, i.e. one master and two slaves. In order to avoid as much as possible any external influences, the entire system was dedicated to the tests during performance measurements.

The program used for tests is the matrix-matrix product for which the size varies from 3,000 × 3,000 to 9,000 × 9,000. Matrices are supposed to be dense and no specific algorithm has been implemented to take into account sparse matrices. Each experiment has been performed at least 10 times and a confidence interval of at least 90 % has always been achieved for the measures. Data reported here are the means of the 10 measures.

Figure 8 shows the total execution time (in seconds) for both MPI and the manual socket implementation. Since the major parts of the program serve for computing works, the time for transferring data between nodes takes a very small scale. Therefore, although there is a significant improvement in the time to send and receive results, the overall execution time of the program remains almost the same.

The details are shown in Fig. 9. During the `Init` step, the DICKPT checkpoints are created and sents to the slave nodes, while during the `Update` step the master waits for the reception of the computed results from the slave nodes and injects them into its memory space.

The DICKPT checkpoints created during the `Init` step are composed of very few bytes of data, so that the time to send these checkpoints is very short. For

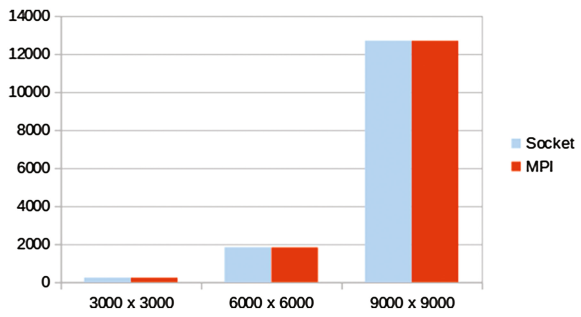


Fig. 8. Total execution time (in seconds) of CAPE using MPI and Socket.

the `Update` step, it takes almost the same amount of time to wait for the result of the computations from the slaves, so that the communication time is not really significant as compared with the overall time of the program. This results in very similar overall times for both methods as shown in Fig. 9.

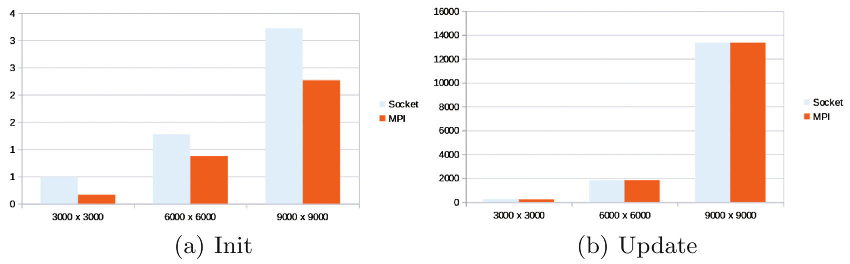


Fig. 9. Execution time (in seconds) for both Init and Update steps.

From the result above, it is clear that using MPI consumes less time than using manual sockets. However, the difference is not significant while comparing the overall time of the program.

6 Conclusion and Future Work

From the analysis and the experiments above, we found that it is interesting to replace the use of manual sockets by use of MPI for data exchange. This helps CAPE achieves higher stability, security and tends to improve performance for the programs using functions supported by MPI, such as broadcast and reductions.

In the near future, we will keep on developing CAPE to support other constructs of OpenMP in order to allow a larger set of algorithms to run on distributed-memory architectures. Moreover, it is also planed to port CAPE on top of other architectures like GPU-based clusters for example.

References

1. MPI: A Message-Passing Interface Standard. Message Passing Interface Forum (2012)
2. OpenMP specification 4.0. OpenMP Architecture Review Board (2013)
3. Ha, V.H., Renault, E.: Design and performance analysis of CAPE based on discontinuous incremental checkpoints. In: Proceedings of the IEEE Conference on Communications, Computers and Signal Processing. Victoria, Canada, August 2011
4. Ha, V.H., Renault, E.: Improving performance of CAPE using discontinuous incremental checkpointing. In: Proceedings of the IEEE International Conference on High Performance and Communications 2011 (HPCC-2011), Banff, Canada, September 2011
5. Ha, V.H., Renault, E.: Discontinuous incremental: a new approach towards extremely checkpoint. In: Proceedings of IEEE International Symposium on Computer Networks and Distributed System (CNDS 2011), Tehran, Iran, February 2011
6. Li, Y., Shen, W., Shi, A.: MPI and OpenMP paradigms on cluster with multicores and its application on FFT. In: Proceedings of the Conference on Computer Design and Application (ICCDA 2010) (2010)
7. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In: Proceedings of 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing (2009)
8. Wong, H.J., Rendell, A.P. : The design of MPI based distributed shared memory systems to support OpenMP on clusters. In: Proceedings of IEEE International Conference on Cluster Computing (2007)
9. Wang, H., Potluri, S., Bureddy, D., Rosales, C., Panda, D.K.: GPU-aware MPI on RDMA-enabled clusters: design, implementation and evaluation. IEEE Trans. Parallel Distrib. Syst. **25**(10), 2595–2605 (2014)
10. Potluri, S., Wang, H., Bureddy, D., Singh, A.K., Rosales, C., Panda, D.K. : Optimizing MPI communication on Multi-GPU systems using CUDA inter-process communication. In: Proceedings of the IEEE International Conference on Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum (IPDPSW) (2012)
11. Balamurugan, B., Krishna, P.V., Rajya Lakshmi, G.V., Kumar, N.S.: Cloud cluster communication for critical applications accessing C-MPICH. In: Proceedings of the International Conference on Embedded Systems (ICES 2014) (2014)
12. Shivaramakrishnan, S., Babar, S.D.: Rolling curve ECC for centralized key management system used in ECC-MPICH2. In: Proceedings of the IEEE Global Conference on Wireless Computing and Networking (GCWCN 2014) (2014)
13. Matsuda, M., Kudoh, T., Kodama, Y., Takano, R., Ishikawa, Y.: Efficient MPI collective operations for clusters in long-and-fast networks. In: Proceedings of the IEEE International Conference on Cluster Computing (2006)
14. Rabenseifner, R.: Automatic MPI counter profiling of all users: first result on a CRAY T3E 900–512. In: Proceedings of the Message Passing Interface Developers and Users Conference 1999 (MPIDC 1999) (1999)

Wi-Fi Channels Saturation Using Standard Wi-Fi Gateway

Daniel Cortes Cañas[1], Brayan S. Reyes Daza[2], and Octavio J. Salcedo Parra[1,2(✉)]

[1] Universidad Nacional de Colombia, Bogotá D.C., Colombia
dacortesca@unal.edu.co, osalcedo@udistrital.edu.co
[2] Intelligent Internet Research Group, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia
bsreyesd@correo.udistrita.edu.co

Abstract. Considering the ever changing world of now, constantly growing, likewise changes the way that people connect with each other. Wireless networks are one of the main means of internet access today, thus the saturation of Spectra of 2.4 GHz has been the subject of countless discussions and studies on the market, such is the case that new communication protocols have been enabled and even enabled new Spectra as the 5 GHz for the continued expansion of wireless communications. In this experiment we will consider the technique of spatial multiplexing MIMO channels for 802.11n Protocol. Evaluating their features, by using a FTP Manager known as JPerf, including latency and ping, of each of the available channels.

Keywords: FTP · ADSL · 802.11n · MIMO

1 Introduction

In Colombia the number of connections to the internet has increased dramatically over the years 2012 and 2013 [3], just by looking at the number of connections going from 6′637.365 to 8′052.732 in this period, which represents an increase of 18 % (1′ 415.367). This vast increase in the number of connections to the internet in the city is caused by the internet boom which makes it almost a basic need for the people of all economical layers, also the different policies of the Ministry of Information technologies (Internet for everyone) which sought to obtain an agreement with the operators of broadband internet to provide affordable rates for the lower economical strata of the city, also the increasing use of ICT for education and other issues has changed today, passing from around 21594 connections per square kilometer up to about 26199, still a fairly significant number.

From that amount of connections, about 85 % are fixed ADSL connections, which leaves us a few 22 thousands fixed connections that have Wi-Fi, because of this, in neighborhoods with buildings of high population density, there is a greater condensation of connections thus leads to a saturation of the common spectrum in the country that is the 2.4 GHz, this spectrum is not only used by modems-routers and/or involved entities, also is used by wireless telephones (very common in the country), toy remote

© Springer International Publishing Switzerland 2015

control, short-wave radios, vehicle alarms, among many other appliances that we use every day, this, without taking in account other entities that may cause distortion or signal interference such as the architecture of the building, appliances emitting waves, etc.

In other countries like China, Korea, and Japan, who are strongly interested in the creation of the highest standards in telecommunications and information networks, they encourage the use of nets of 5 GHz for WiFi networks in homes and office, to alleviate the saturation of the known spectrum of 2.4 GHz, however, the low compatibility, the range of the signal and its low popularity has caused that the market has not been able to force the consumers migrate to this new spectrum. Nevertheless, research in the area motivates "internet of things" which tries to connect almost all to the internet, and for this, logically, it will require with time, more spectrum released for intercommunication of people and devices.

2 Review of Work

As a result of the saturation of the spectrum of 2.4 GHz it has begun a research to migrate to other spectrums, such as 5 GHz and also a research for optimal techniques for multiplexing channels such as the MIMO, however, as said in the introduction, it has to be added that we should also take into account interference from entities that communicate by Bluetooth [3]. Since these entities are communicating with the 802.15 protocol through the same spectrum, therefore, it favors the use of MIMO technologies (802.11n, 802.11ac) over old sequential technologies OFDM (802.11b, 802.11g, 802.11n). With the current, saturation it becomes impossible a transmission without interference or loss of data packets, in addition to this factors such as: the relative power of signal, the bandwidth, the displacement of mutual frequency, mutual overlap and the number of signals that collide are completely determining factors when assessing the interference of such entities on 802.11n Networks.

Regarding all facts and to reduce and mitigate the collision between signals, the use of MIMO technology Over 802.11n and 802.11ac protocols is required, and hence distribution Beamforming [4] of antennas (particularly those of LTE networks) to reduce SINR and likewise, in many cases as stated in this article, the omnidirectional antennas can cause major problems of interference than the unidirectional antennas, what is proposed is a multiple unidirectional antennas whose transmission is synchronized so that the interference by partially synchronous and asynchronous networks is reduced to avoid that multiplicity affects these transmissions directly.

On the other hand, another possible solution to prevent interference in the different channels of the WiFi network is to increase the coverage of the network by a technique of distribution through multiple jumps (multi-hop) [5], this technique consists in distributing points of access (Access points) to increase network coverage and also the stability of the network, in this article it was taken into account external factors as they are the walls and its composition, we also conducted several experiments with networks of 2, 3 and 4 points of access strategically allocated in the building and having outings of at least 100 Mbps, being completely stable speeds.

Another curious alternative is a multiplexing of OFDM channels with 802.15.4 protocol, which varies from 802.11 due to the fact that these possess alternative Spectra described in Table 1.

Table 1. Frequencies set for 802.15.4 Protocol around the world [6]

Location	Spectra (MHz)
Europe	868–868.6
Japan	950–958
China	314–316, 430–434, 470–510, 779–787
Korea	917–923.5
USA	902–928
Worldwide	2400

Despite using an algorithm that belongs to 802.11 protocol, legacy devices for 802.15.4 protocol, does not seem a problem since that entities that use it, ZigBee and MiFi [1], micro-controller cards, generate LR-WPAN networks with only investigative or experimental use, whose number of entities do not exceed eight, is also worth noting that there are few collisions when working on these networks except when working on the 2.4 GHz, if so is the case, we warn that the use of MIMO is completely required to avoid interference problems, also other WPAN networks are the Wi-Fi direct [2] which have algorithms that jump from channel to channel finding the more stable common between the two concerned entities, even running through channels that are permitted by law but which are no longer commercial, since they are already too saturated or simply its use is no longer optimal for networks.

3 Cognitive Radio

Cognitive radio is a proposal of dynamic channel assignment based on different techniques, the technique varies according to the requirement of the network, in the case of WiFi networks, the preferred technique is the detection by interference, however, just a few routers can carry out a pre-selection, in fact, those who have multiple antennas, MIMO and QoS [8] technology have multiple selectors which evaluate channels one by one, selecting quickly and without interruption as a destination one less saturated, for routers without QoS and low cost, the selector is sequential and the algorithm is FIFO, therefore sometimes interruptions with this type of routers arise, however, placing a fixed configuration of channel to one that is free, the performance of the router is similar to one that has QoS.

On the other hand, the use of techniques based on the transmitter and/or receiver, also is commonly seen in some routers, since fixed stability during certain time interval can be obtained by means of a negotiation algorithm, however, this technique has the defect of being little dynamic, causing service interruptions when the spectrum of the channel is quickly saturated and the router does not react at the same speed.

4 Technical Specifications

4.1 Arris TG862G DOCSIS® 3.0 (Router)

The Touchstone® DOCSIS® 3.0 Residential Gateway is an 8 × 4 advanced gateway product. The TG862 combines two analog voice lines, a 4-port Gigabit Router, and an 802.11n wireless access point.

The Wireless Transmit Power Output (EIRP) is between 19.5 dBm (802.11b), 17.5 (802.11n) with a standard error of ±1.5 dBm. The frequency range is between 2400–2483.5 MHz and has to antennas (one for transmit and one for receive).

4.2 Qualcomm Atheros QCA9565 (WiFi Adapter)

It's a WiFi mini card that includes protocols (802.11b/g/n) it almost has Bluetooth v4.0, 1 × 1 antennas and connection speed of 100 Mbps.

4.3 Qualcomm Atheros AR8145 (WiFi Adapter)

It's a WiFi mini card that includes protocols (802.11b/g/n) it almost has Bluetooth v4.0, 1 × 1 antennas and connection speed of 100 Mbps.

5 Test Conditions

Tests were performed on a tenth floor of a building whose population density is around 250.000/km^2. The router is located in a room surrounded by walls of concrete but free for transmission to the entities, which are 5 m from the same. Both cards Bluetooth will be activated on stand-by and it will be taken into account also the ping and jitter for each channel on the internet.

The experiment consists of the jPerf use to make transfers via FTP using only the 802.11n protocol and by placing a fixed configuration for the different channels (from 1 to 11), a test of ping's connection over LAN-LAN cable CAT5E type will be done to have a control test, also 30 points equivalent to the transfer speed for a second will be taken.

6 Results

The following charts represent the results of the experiment of bandwidth per channel. For the first analysis we have a via LAN CAT5E cable control experiment, to test the stability of the network which is pretty good.

In the first graph we see a very low variability of the bandwidth, in addition have a 60 ms ping and a jitter of 5 ms which are minimum values obtainable in this connection.

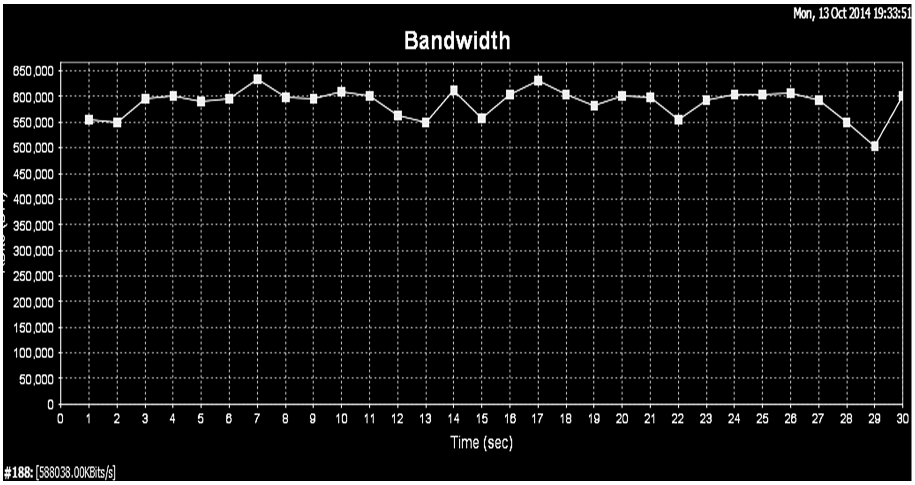


Fig. 1. Bandwidth LAN-LAN using a RJ-45 CAT5E cable.

In Fig. 2 we can see the bandwidth for the first 5 channels, we can see that in channels as the 2 and 3 have a of pretty high bandwidth variability regarding all other channels present, we can say that Channel 5 is the one that least varies, however the height of few peaks that it has is very notorious.

For the graph of Fig. 2 we see a clearly atypical behavior for the channel number 11, if we are based on the average bandwidth of Table 2 surprise us that it is of a few 1000 kbps ten times less than the channels with peaks of higher speed, is incredibly it is the most stable channel which was measured in the experiment. In general, we can say that these graphs (Fig. 2) are much more stable than those belonging to graph Fig. 1,

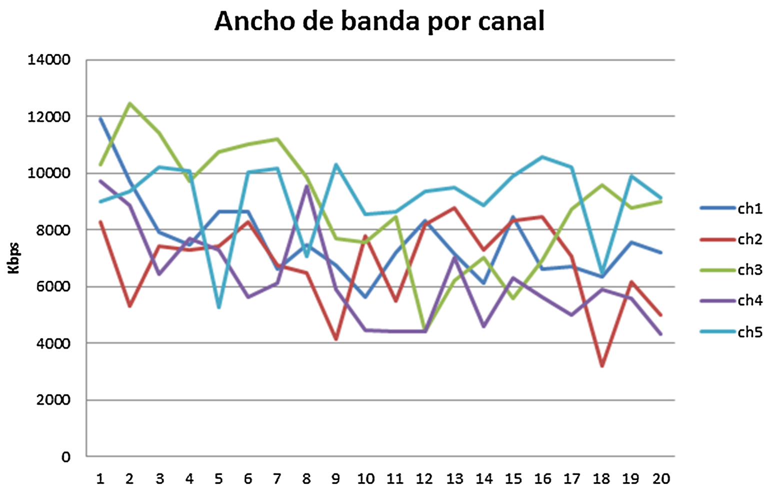


Fig. 2. Bandwidth for channels 1 to 5.

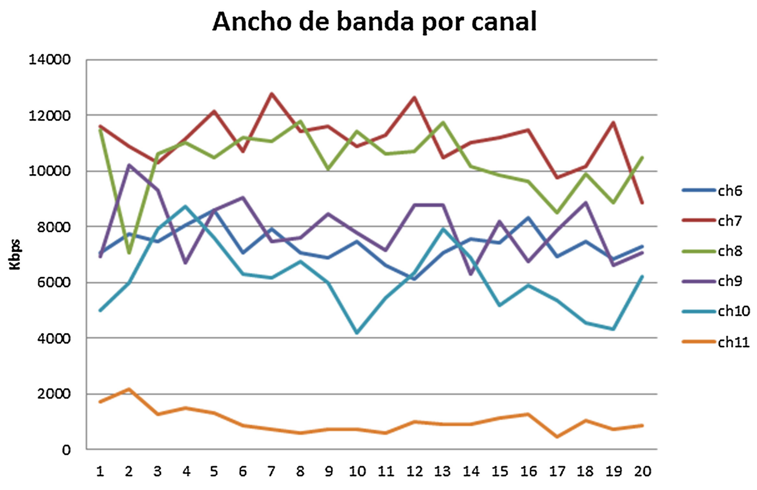


Fig. 3. Bandwidth for channels 6 to 11

also its amplitude is much lower than the addressed graph, and also presented highest speed averages. In the results of the experiment carried out the last time, a strong interference was notice on channels 10 and 11, which as seen in Fig. 2, they have enough instability and connection problems, also the test had to be repeated several times on these channels to obtain a reading consistent with experiment (Fig. 3).

Within the possible causes of the interference were evaluated entities of other nearby WiFi networks, electronic devices on the 2.4 GHz network such as wireless phones, radios, wireless controls among others.

On Table 2 we condense all the results of the experiment, adding additional calculations for the connection, such as the ping and jitter and statistical inference, such as average speed, median, standard deviation and variation coefficient in order to

Table 2. Average bandwidth, median, ping, jitter, standard deviation and variation per channel coefficient.

Channel	Prom.	Med.	Ping	Jitter	STDDev	CV %
1	7621.85	7340	81	5	1385.67	18.18
2	6855.10	7289	80	4	1503.58	21.93
3	8824.4	8880	79	3	2066.48	23.42
4	6231.7	5898	81	4	1626.6	26.1
5	9127.1	9437.5	87	30	1359.4	14.89
6	7346.6	7340	93	30	575.37	7.83
7	11097.65	11174	76	5	908.42	8.19
8	10331.8	10551.5	81	6	1134.58	10.98
9	7923.25	7831.5	70	4	1035.85	13.07
10	6134.05	6062	65	3	1211.36	19.75
11	1019.2	918	69	5	407.91	40.02

obtain more solid and clear results to get more accurate conclusions when assessing the findings.

7 Discussion of Results

After the experiment, we have noticed that the variability of the speed transfer via Wi-Fi through certain channels becomes bad. As we saw, there is the influence of external agents that use the same spectrum such as entities that communicate by Bluetooth or by radio frequency (wireless telephones, Walkie talkies, etc.), therefore, this factor is the first that should be considered in assessing the interference on a WiFi 802.11n. network. As a second element, we have the technique of multiplexing and transmission of information from the involved entities, in the case of use of OFDM, we observed that only in the case of requiring a robust infrastructure for the distribution of network, it would be optimal to use this method, on the other hand, the use of MIMO is ideal for wireless networks in homes and small distributions for companies.

If it is required to increase the coverage, the techniques of construction of networks that have multiple access points that allow to perform jumps would be ideal in this case, because as it increases WiFi network coverage, also increases the stability of the network and also reduces the interference between channels since it is transmitted through various channels and it should migrate in case that one of them has too much interference. Within the solutions that were found there could also be the use of the 2.4 GHz frequency channels which have been abandoned by disuse, despite the fact that in some cases it would need special equipment for transmitting and receiving, it can be a solution that clearly contributes to the stability of the network.

8 Conclusions

Based on the speed of transfer of different channels, it can be noticed clearly that the channels with less interference are located between 6 and 9 and, the largest are among the first and the last, there were several FTP falls on channel 10 and 11 which is reflected in a low level of transfer and a high CV with compared against the other channels.

Within the tested solutions, it is to be noticed that to avoid interference, techniques for MIMO multiplexing must be used for infrastructure of small and medium-sized networks and OFDM with unidirectional antennas placed in various points for large extension mobile networks or WiMAX networks which coverage should be quite broad. Additionally, keep in mind that if the network is located in enclosed spaces or with many walls, the network must necessarily have multiple wireless access points to make the repetition of the signal through multiple hops so the network remains stable and with reduced interference.

Finally, the solutions related to the use of abandoned channels (to the edges of the spectrum) are dismissed because its current popularity is so low that it makes that the costs of construction, improvement or adaptation of a network to rise too high, not to mention compatibility with client devices. Also is dismissed the use of exclusive protocols, such as for the ZigBee, 802.15.4, which while it may be perfectly free of interference, the complexity of construction and the low compatibility with 802.11 Networks, causes communication flaws in the devices. Finally, in experiments such as this one it also could and should be taken into account external interference factors such as concrete walls, aluminum and steel, the signal power and the Status of the network cards of the customer entities.

References

1. Xu, R., Shi, G., Luo, J., Zhao, Z., Shu, Y.: Muzi: multi-channel Zigbee networks for avoiding wifi interference. In: Proceedings of IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing 2011, pp. 1–7 (2011)
2. Feng, J., Liu, Z., Ji, Y.: Wireless channel loss analysis – a case study using wifi-direct. In: Wireless Communications and Mobile Computing Conference (Iwcmc) 2014, pp. 1–5 (2014)
3. Biran, A.: Wifi ofdm and bluetooth signal collision analysis in microwaves, communications, antennas and electronic systems (Comcas) 2011, pp. 1–4 (2011)
4. Ezri, D., Tsodik, G.: The impact of synchronization on receive beamforming with null steering in ofdm mimo systems. In: 2012 IEEE 27th Convention of Electrical & Electronics Engineers in Israel (IEEEI), pp. 1–4 (2012)
5. Hassan, S.: Optimal throughput analyisis for indoor multi-hop wireless networks in IEEE 802.11n. In: Wireless and Microwave Technology Conference (Wamicon) 2013, pp. 1–4 (2013)
6. Miaoxin, L., Zhuang, M.: An overview of physical layers on wireless body area network. In: 2012 International Conference on Anti-Counterfeiting, Security and Identification (ASID), pp. 1–5 (2012)
7. Salgado, L.: Algoritmo multivariable para la selección dinámicadel canal de backup en redes de radio cognitiva basado en el método fuzzy analitical hierarchical process. In: Tesis De Maestría En Ciencias De La Información, pp. 1–107 (2014)
8. Enchi, C.: Qos-aware power allocations for maximizing effective capacity over virtual-mimo wireless networks. In: 2013 International Conference on Selected Areas in Communications, vol. 31, pp. 1–15 (2012). IEEE Journal

Cloud Access Secure with Identity Based Cryptography

Houria Hamadi(✉), Abdelkrim Khireddine, and Abdelkamel Tari

University of A/Mira, Bejaia, Algeria
{hamadi.houria,abdelkrim.khired,tarikamal59}@gmail.com

Abstract. With the fast evolution of networks and Internet, appeared the concept of the Cloud, the requirements in terms of safety become more and more essential. This requires the introduction of advanced authentication methods, access control and identity management, while respecting the constraints of the services offered by this evolution, such as data exchange capacity and resources of terminals. The aim of our work is to propose cloud security architecture to be able to establish secure sessions between the nodes of a cluster in the cloud, and allow to different users a secure access to their data.

Keywords: Virtual networks · Cloud · Security · Authentication · Access management · Identity management · Identity-based cryptography

1 Introduction

In the classic computer systems, companies, operators, service providers put up computer parks to host their data and activate their services. But today, Cloud Computing represents an economic and practical alternative for these entities. Just like the electric power one century ago, the computing power and storage of information would be proposed for consumption by specialized companies. Therefore, companies would no need more appropriate servers, but would entrust this resources and services to Cloud providers using a set of physical resources (servers) organized within a cluster to provide, in the request, resources and services to users. The Cloud became today an effective solution to optimize the use of the physical resources and reduce their costs.

Cloud computing is a technology with vast impact on computer systems. The costs can be significantly reduced through the purchase at the request of CPU time, memory and storage, offering a great flexibility. However, the opening of systems and sharing of associated resources create many problems of security, which remains one of the major barriers for the adoption of these technologies [1]. In addition, cloud providers secure their systems only against external enemies using firewalls and secure connections. Ignoring the internal opponents, who represent a big threat.

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 109–118, 2015.
DOI: 10.1007/978-3-319-25744-0_10

Identities constitute a key element of safety. This information must be correct and available to all elements of cloud witch needs to validate access. Access control bases on the identity information to allow and constrain the access to cloud functioning and the underlying infrastructure. According to reports published by groups of protection of personal information supervisory in the United States, more than 148 incidents of identity theft, affecting about 94 millions identities, were held in 2005 in the United States alone (Mark, 2006). Identity theft is one of the most serious threats to online services security. Therefore, it is irresistible that SaaS providers have the means to authenticate the identity of each user trying to access to the system.

In order to improve the security of cloud infrastructure, security of identities and limiting access to authorized users, and by basing on the work already done and re-use of already existing solutions for other type of networks which we will adopt for virtual networks and reduce operating costs. Our purpose in this article is to define a system of access control to filter users wishing to connect, and preserve access to critical user's data while protecting their identities using a system of protection based on the identity-based cryptography and smart cards.

This paper is organized as follows: introduction, the second and third sections present an overview of existing works and definitions of various basic elements which will be used in this approche. Section 4 approaches the model of the solution and the proposed architecture. Section 5 details some security evaluations of our solution. Finally, Sect. 6 concludes and proposes perspectives of our work.

2 Basic Notions

2.1 Cloud Computing

Cloud computing is a model that allows quick access and request for a pool of shared computing resources (servers, storage, applications, bandwidth, etc.) without strong interaction with service provider. This definition of the NIST (National Institute of Standards and Technology) is widely taken to define the Cloud. Several major players like Microsoft and Google already propose different Cloud solutions. Companies hope to have more efficiency and reduce costs if they can access to an online service, and not manage their ICT (Information and Communication Technologies). Cloud brings, beyond the added value technology such as scalability, performance, etc., cost reduction and flexibility (to develop efficiency). In the following we are going to describe the cloud architecture that will be used in our work:

- **Cloud Service Provider:** the provider of cloud services, he provides to users and customers virtual infrastructure so that his customers can access to their storage spaces, and services requested and allocated.
- **Client:** is a user of the services offered by a cloud provider, he has access to the resources allocated, and has the rights and permissions on all of its data and services. A customer can be a single individual or a company. Customers can communicate between them and share data.

- **Other Users:** who are not necessarily customers in a cloud provider, but can access to the data stored in the cloud by a client, and having a number of authorizations defined by owners of the data (Fig. 1).

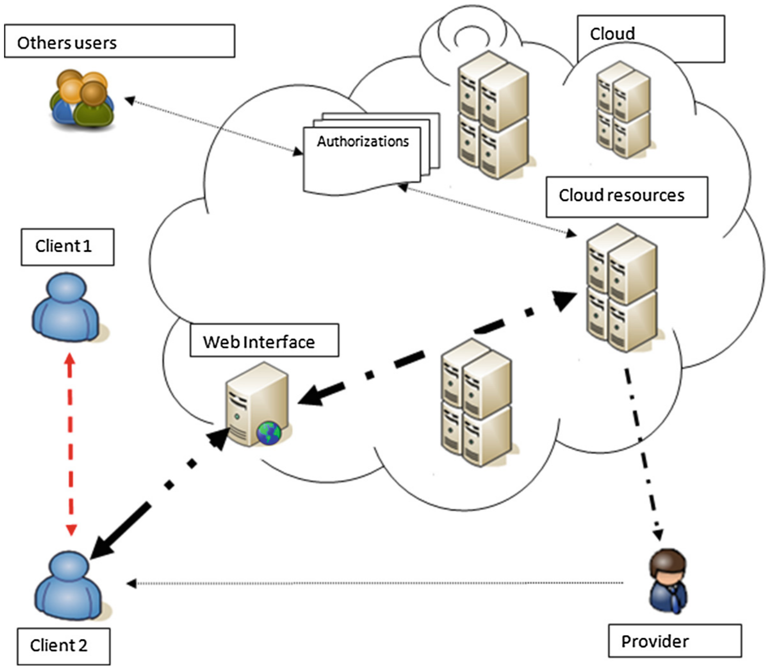


Fig. 1. Cloud architecture.

2.2 Identity-Based Cryptography

The concept of "identity-based cryptography" was introduced by Adi Shamir in 1984 [2]. Contrary to the conventional schemes, schemes based identity (IBC), offer the possibility of choosing freely the public key. Electronic certificates, known for their complexity, are replaced with easily remembered information, such as email address, IP address, such as public key. As far as the identity-based schemes do not require certification mechanism, they allow to simplify considerably the implementation of secure communication systems.

By their construction, plans based on identity (IBC) manage to solve the authentication problems, management and revocation public keys, however, they require the presence of an ultra-powerful authority PKG (Private key Generator), who provides to each user the corresponding private key to its public key. The confidence granted to this authority must be without defect, because it is inherently capable of regenerating the private key of any user, and therefore able to perform unauthorized signatures or decryption.

During the last decade, IBC has been improved by the use of Elliptic Curve Cryptography [3]. Consequently, the new identity based encryption and signature scheme appeared. To be able to take a client's private key, the PKG must first define a set of identifiers based on public characteristics. PKG product groups G_1, G_2 up to G_T and the coupling function ê of $G_1 \times G_2$ in G_T. G_1 and G_2 are additive subgroups of the group of points of an elliptic curve.

However, G_T is a multiplicative subgroup of a finite domain. G_1, G_2 and G_T have the same ordre q. Moreover, G_1, G_2 and G_T are generated by P, Q and the generator $g = \hat{e}\ (P, Q)$. The bilinear function ê is fired from the Weil [4].

After the specification of the groups, the PKG defines a set of hash functions in accordance with the IBC and the signature scheme used [5]. As such, the PKG defines a hash function $Hash_{pub}()$ to transform the client's identity (ID) into a public key as follows: $Pub_{ID} = Hash_{pub}(ID)$.

Generally, the public key of a customer is calculated as a hash of one of its identities which is a point of an elliptic curve [6] or a positive integer [7]. The PKG generates the private key of an entity using a local secret $S_{PKG} \in Z^*$ and a private key generation function $Priv_{Gen}()$. Note that the private key is calculated as follows: $Priv_{ID} = Priv_{Gen}(S_{PKG}, Pub_{ID})$.

3 Related Works

In the last years, cloud security and virtual environments became a topic attracting who research. Several studies have addressed the various problems of security and so proposing different solutions. To develop and improve the standard authentication with password [8] various works were realized to be in order to solve new security problems in a cloud environment, the use of identity-based cryptography (IBC) stays a very effective way to make sure of the identity of the user and be able to manage the identities of a number of users who increases more and more. With the development of internet technology to the cloud, access and identity management became crucial for the safety, and she allows limiting access to data and applications to the only authorized users.

Since their introduction, schemes based identity, have been the subject of intensive researches, yet it will have been necessary to wait 2001, and the works of Cocks [9], Boneh and Franklin [6], to have crypto systems that reply to conditions of safety and efficiency. The identity-based cryptography, is a new occurrence and an interesting domain for the security of virtual networks. IBC has was adapted for grids networks, this idea was explored by Lim and Robshaw in 2004 [10]. In their proposal, each virtual organization has its own PKG and all users share the same public characteristics certified by the network certification authority.

In 2005, Lim and Robshaw [11] explain a new concept of dynamic key infrastructure for the grid in order to simplify the management of keys already made in [10]. That is to say, every user takes care of the construction of its public characteristics and its distribution to other entities. It is in 2005, that Lim and

Paterson [12] suggest using IBC to secure grid environments. They describe several scenarios where IBC simplifies many cases, as the elimination of the use of the certificate, and the savings of bandwidth while using the approach proposed by Boneh and Franklin [10].

H. Li, Y. Dai, L. Tian, and H. Yang, [13] propose to use IBC as an alternative to SSL (Secure Sockets Layer)authentication protocol in the cloud, however, these solutions still suffer from necessary fiduciary hierarchy to assure a safe working system. [14] Presented a new security infrastructure by using the IBC for applications oriented Cloud services. In their proposal, the service URL is used to generate the public key.

4 Model and Architecture

The choice of IBC is motivated by several reasons. At first, IBC will allow us to take advantage of its simple keys management mechanism, which does not require the deployment of a public key infrastructure (PKI). Secondly, IBC allows generating public keys without needing for a prior calculation of corresponding private keys. That is, contrary to traditional patterns public key generation, IBC does not require to generate the private key before the public key. Indeed, customers need only first, generate public key-based characteristics that constitute their identities to define their public keys lists. Finally, for each new discussion, a different public key will be used to encrypt messages so the client generates and uses a new private key to decrypt, that will prevent us the use of the same key to encrypt or decrypt the discussions.

The objective of this work is to design and develop security architecture for the cloud. The proposed architecture will allow the management of identities, access and the security of communications between nodes in a cluster in Cloud. Our idea consist in naming every customer as a PKG that generates its own private keys, from a secret S_C, recovered after authentication with the PKG, and one of its public key Pub_C. These private keys used to authenticate, encrypt and decrypt messages exchanged between the client and the various nodes of the cluster. To define our architecture, we have to identify the different actors. In our context, the actors are the basic entities that our solution will count (provider, customer,...). We define the various entities as follows:

The provider, customer and other customers are already defined in the previous section: Basics notions.

- **PKG:** is an authority that authenticates and generates secret keys for all entities of the cluster. Each delivered key must have issued a validity date in order to manage and eliminate nodes which do not belong any more to the cluster. Every node that does not re-authenticate in a specific date its secret key will be invalid, and it cannot generate private keys. It also has a secret key and a public key list, that broadcast regularly in the network for new clients to have a way to communicate it.
- **Index Server:** Server in the cloud where are stored indexes that define the location of each data stored in a storage server.

- **Indexes Database:** contains all indexes of stored data. A batch of data stored or identifier may be designated by multiple indexes.
- **Storage Server:** it is a storage space where the data are stored. Each space memory where data is stored and saved under a code that we are going to name Index. Each lot of data saved by a client must have an identifier (keyword) that will be indicated when searching.

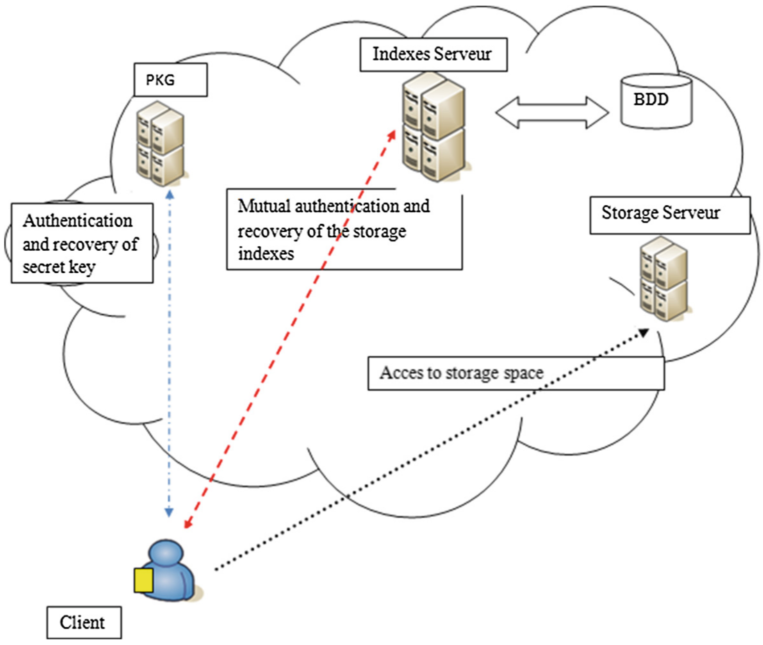

PKG
Indexes Serveur
BDD
Authentication and recovery of secret key
Mutual authentication and recovery of the storage indexes
Storage Serveur
Acces to storage space
Client

Fig. 2. Proposed architecture.

4.1 Prerequisites

In this section we will define the prerequisites used to design our model. First, each client generates its own list of public keys pub_C, that will be integrated in a certificate, signed by a certification authority, to validate their identity. This list will be used and distributed so this client is known within its cluster (Fig. 2).

We suppose as well, that the customer have already a registration and authentication with the provider, who granted him different access to its allocated services. Each customer holds a smart card which will allow him to protect its list of public keys as well as its identifiers of data stored in the storage servers. The choice of smart card came to avoid to the customer the memorization and loss of its important informations.

Note also, that the channel established between the client and the PKG is secured. This channel supports the mutual authentication, integrity and confidentiality of data exchanged. TLS (Transport Layer Security) [15], is chosen as it is among the best protocols which ensure secure transmission of messages.

Once the client is connected to a web space to reach his cluster, the first thing to do is to distribute its public key list on the network so that other nodes can connect with him. The client begins by mutual authentication with the PKG, always based IBC. Once authenticated, the customer request for secret key, PKG generates the secret key with a validity date. What will allow PKG to require the authentication of customer every time to remove inactive clients and release the cluster. PKG sends the secret key and the algorithm for generating private keys. The client's secret key will be stored in the smart card (Fig. 3).

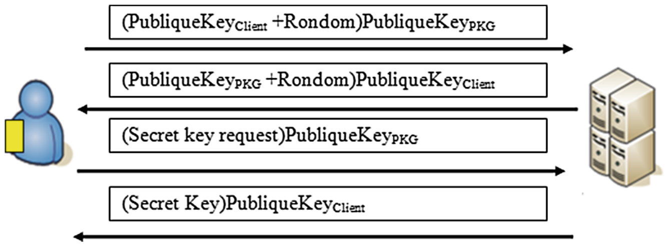

(PubliqueKeyClient +Rondom)PubliqueKeyPKG
(PubliqueKeyPKG +Rondom)PubliqueKeyClient
(Secret key request)PubliqueKeyPKG
(Secret Key)PubliqueKeyClient

Fig. 3. Recovery of the secret key.

Once he had his secret key, the customer can create himself its private keys with its public keys. Which will allow it to communicate with the server indexes to have access to its storage space. After a mutual authentication with the server indexes, the client sends a request with the list of data identifiers which it wants to access. The server indexes side verifies the existence of identifiers, sent by the client, in its database. If identifiers exist well, he gets back the list of appropriate indexes, that will allow the customer to have direct access to the storage server. Otherwise if the client has no access to the data it asked, the server indexes sends unauthorized access.

Our solution provides another advantage to the customer, which is the encryption of their data before saving. When the customer decides to take out of his storage space, the server gives him an opportunity to encrypt data with a key of his choice, thus, it will be the only person who holds the key for encryption and decryption. And as the client already uses identity-based cryptography then the choice of encryption its data will be based on IBC (Fig. 4).

5 Security Evaluation

In this section, we present an informal security analysis of our proposal. In addition, we express the possible refinements to limit other threats.

- **Data Privacy:** In our approach, the customer is responsible for encrypting data before storing them in the cloud. As the client acts as a PK authority, he

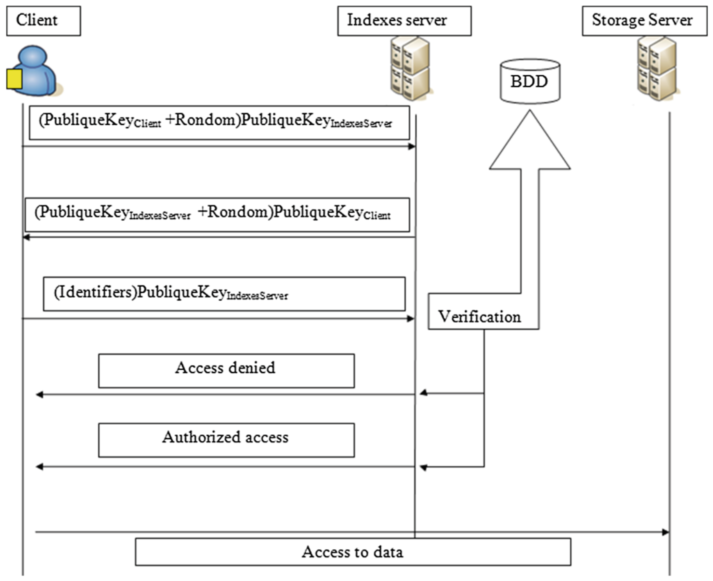

Client
Indexes server
Storage Server
BDD
(PubliqueKeyClient +Rondom)PubliqueKeyIndexesServer
(PubliqueKeyIndexesServer +Rondom)PubliqueKeyClient
(Identifiers)PubliqueKeyIndexesServer
Verification
Access denied
Authorized access
Access to data

Fig. 4. Schema of access to storage servers.

is able to manage its secrets and generate private keys. So, it is the only entity that knows the secret S_C, necessary to generate any decryption key. Thus, it is impossible for the provider or an attacker to recover the decryption key to encrypt data.

- **Control Data Access:** through our security model, data access is limited to authorized clients. IBC allows customers to protect their identity, preserve access to its data and even access to messages exchanged between the different nodes of the cluster to which it belongs. Even if an attacker has a list of client's public keys and tries to steal his identity, he cannot decrypt the messages which intended for him because he has not the secret key that will allow him to generate private keys. Secondly, although the supplier or a malicious user can obtain the access to data, we always guarantee the data privacy, as they can only have access to encrypted data. And they have no private key to decrypt the data.
- **Key Management:** cryptography based identity suffers attacks against the deposit of encryption keys, that is the PKG. However, our solution declines this problem, because every customers acts as a PKG for his own data and encryption and decryption keys. In addition, the secret key is saved in smart card, which makes it the mission impossible for an attacker to generate private keys, or even find them.

Finally, we are anxious to explain that our solution, limits some attacks and provides maximum protection of the private life and the information of the users of the Cloud.

6 Conclusion and Perspectives

The increasing need to secure access to sensitive data and to avoid the theft of identities in the cloud became a challenge for researchers in the field of security. In this article, we presented architecture of authentication for the cloud, with identity-based cryptography (IBC). Our solution is based on a particular method of IBC. First, each client is named as a PKG and takes care of the creation of his list of public keys, the generation of his private keys if necessary, and the secret key backup. The smart card comes to facilitate the job for the customer to properly protect its secret key and its various meta data. Our method also offers the user a chance to encrypt critical data as well as its meta data to avoid those who offers more privacy to its information.

We are anxious to make different tests security while simulating some attacks which aim authentication, client access safety and capacity of the authentication server to support different requests of authentications, and a comparative study between the different methods of encryption in performance time and CPU usage. Finally we presume that access security and data storage on the cloud is always full of challenges and of paramount importance and several research problems remain to be identified.

References

1. Cloud Security Alliance. Top Threats To Cloud Computing. Technical report, March 2010. http://www.Cloudsecurityalliance.org/topthreats.html
2. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
3. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., Secaucus (2003)
4. Blake, I., Seroussi, G., Smart, N., Cassels, J.W.S.: Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series). Cambridge University Press, New York (2005)
5. Ratna, D., Rana, B., Palash, S.: Pairing-based cryptographic protocols: A survey (2004). http://eprint.iacr.org/
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). http://dl.acm.org/citation.cfm?id=646766.704155
7. Sakai, R., Kasahara, M.: Id based cryptosystems with pairing on elliptic curve, Cryptology ePrint Archive, Report 2003/054 (2003). http://eprint.iacr.org/
8. Secure Password by Using Two Factor Authentication in Cloud Computing, Ali A. Yassin, Hai Jin, Ayad Ibrahim, Weizhong Qiang, and Deqing Zou, Services Computing Technology and System, Lab Cluster and Grid Computing, Lab School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China
9. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)

10. Lim, H.W., Robshaw, M.J.B.: On identity-based cryptography and grid computing. Lecture Notes in Computer Science, pp. 474–477 (2004). http://www.springerlink.com/content/ylj95gfgjxlb2131
11. Lim, H.W., Robshaw, M.: A dynamic key infrastructure for Grid. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 255–264. Springer, Heidelberg (2005)
12. Lim, H.W., Paterson, K.G.: Identity-based cryptography for grid security. Int. J. Inf. Secur. **10**(1), 15–32 (2011). http://dx.doi.org/10.1007/s10207-010-0116-z
13. Li, H., Dai, Y., Tian, L., Yang, H.: Identity-based authentication for cloud computing. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 157–166. Springer, Heidelberg (2009)
14. Schridde, C., Dörnemann, T., Juhnke, E., Smith, M., Freisleben, B.: An identity-based security infrastructure for cloud environments. In: Proceedings of IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS2010) (2010)
15. Dierks, T., Rescorla, E.: RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2, Technical report, August 2008. http://tools.ietf.org/html/rfc5246

SDN-Based QoS Aware Network Service Provisioning

Cosmin Caba(✉) and José Soler

DTU Fotonik – Networks Technology and Service Platforms Group, Ørsteds Plads, Building 343, 2800 Kongens Lyngby, Denmark
{cosm,joss}@fotonik.dtu.dk

Abstract. One of the applicability areas of SDN is for creating services for dynamic provisioning of network resources with strict QoS requirements. The research available in this field focuses mainly on the service logic implemented over the functionality of the SDN Controller (SDNC). However, there is much to be covered regarding the specific mechanisms used by the SDNC to enforce the QoS in the data plane devices. To this end, the current paper proposes a data plane QoS architecture, together with the invariants that have to be maintained by the SDNC in order to ensure predictable QoS for the network services. More specifically, the paper will look into on demand provisioning of Virtual Circuits (VCs) with specific QoS, based on the SDN paradigm. The aim is to analyze and compare the strategies for network resources management for two cases: a coarse granular and a fine granular VC provisioning service. Furthermore, the analysis is intended to serve as a basis for future implementations of SDN-based mechanisms for provisioning of bandwidth on demand.

Keywords: Software defined networking · Openflow · OVSDB · Information model · Services · QoS · Virtual circuit · Virtual switch

1 Introduction

Because of its inherent characteristics (i.e. programmability, network wide optimization), SDN is considered a suitable solution for dynamic provisioning of network resources. As a result, there is a growing interest in the industry and academia in leveraging the SDN paradigm to make the service provisioning process more automated and efficient with respect to the network resource utilization [1]. The current paper focuses on the management of network resources, in the context of SDN-based service provisioning with strict QoS requirements.

A possible technology for deploying QoS in IP networks is DiffServ-aware MPLS-TE [2], which is based on the integration of a DiffServ data plane architecture with an MPLS-TE control plane. The current paper describes a similar method for offering QoS guarantees for network services, but using the SDN paradigm. More precisely, the data plane QoS mechanism presented here is similar to the DiffServ architecture: applying per-flow admission control at the edge and prioritization among the traffic classes in the core of the network [3]. With regard to the control plane, the current work proposes a Constraint-Based Routing (CBR) algorithm, which computes the paths in the network,

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (eds.):MSPN 2015, LNCS 9395, pp. 119–133, 2015.
DOI: 10.1007/978-3-319-25744-0_11

given a set of constraints for the paths. Unlike [2], the method described here uses a centralized control plane logic based on SDN instead of the distributed MPLS-TE control plane.

The type of service discussed in this paper is provisioning of Virtual Circuits (VCs) with strict QoS requirements. A VC is considered to be a stream of packets (consisting of one or more traffic flows) identified and treated uniformly across the packet-switched network, and it is used to carry customers' traffic across a provider's domain. A request for a VC, initiated by a customer towards an Internet Service Provider (ISP), is assumed to contain a set of parameters associated with the VC such as *start time*, *duration*, *QoS profile*, and other path constraints. If the provider accepts the request for the service, it has to reserve resources in its network such that the offered service complies with the QoS profile. The architecture proposed here for the provider's network is based on the SDN paradigm, thus the enforcement of the QoS profile is managed by the SDN Controller (SDNC). Two QoS parameters are taken into account: *bandwidth* and *delay*. Other QoS parameters, such as jitter, packet loss, availability, are considered for inclusion in future work.

Following the first introductory section, the second section describes the generic control plane architecture used throughput the paper. In Sect. 3 the related work is presented and the main differences from the current work are outlined. Section 4 discusses the proposed data plane QoS architecture and Sect. 5 continues with a description of a prototype for the SDNC platform. The algorithm for VC provisioning is detailed in Sect. 6. Section 7 discusses the validation tests for the data plane QoS mechanism, and Sect. 8 concludes the paper.

2 Architecture Overview

Figure 1 illustrates the high-level SDN architecture used throughout the paper, consisting of a set of data plane forwarding devices (e.g. D1, D2, and D3) and the SDN Controller (SDNC). Two interfaces are defined for the SDNC [4]: (1) the Application-Controller Plane Interface (A-CPI), used for receiving service requests, and (2) the Data-Controller Plane Interface (D-CPI) which enables the following functions:

- *Management Functions*: are fulfilled through a management protocol and are associated with provisioning and configuration of data plane resources (e.g. ports, queues).
- *Control Functions*: are associated with real-time operations such as orchestration of traffic flows, statistics gathering, etc. The control functions presented here are based on the Openflow (OF) protocol [5], hence the forwarding devices contain a set of flow tables that store the OF entries installed by the SDNC.

As illustrated in Fig. 1, the SNDC can be divided in two parts containing the management and control functions respectively, and an Information Model (IM) providing a representation of the network resources (virtual and physical). A more detailed view of the platform‘s architecture will be presented in Sect. 5. Reservation of network resources for the case of VC provisioning requires that two mechanisms are in place:

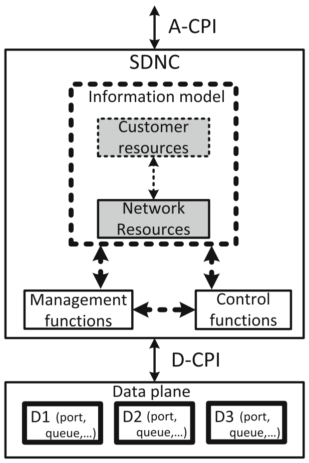


Fig. 1. SDN architecture used throughput the paper

- A control plane IM together the strategy for managing the customer and network resources (dotted line in Fig. 1). The IM consists of two sub models: the *network resources* and the *customer resources*. The *network resources* sub model provides a representation of the underlying network, while the *customer resources* sub model provides a representation of the resources allocated to customers from the underlying network.
- A data plane QoS architecture, which consists of the data plane resources (i.e. ports, queues, etc.), to support the QoS enabled SDN services.

Two types of traffic flows may exist in the data plane: (1) best effort flows, for which reservation of network resources is not needed, and (2) QoS flows, which require a certain level of QoS (resource reservation). The current paper focuses only on the latter category of traffic flows because best-effort flows do not require QoS guarantees. Depending on the operational model for VC provisioning, the strategies for resource management to be applied inside the SDNC may be different. The paper covers two scenarios for network resources management, which are applicable to a wide range of operational models for VC provisioning:

- *Scenario 1* describes a fine granular model for VC provisioning where each VC has its own QoS profile. The QoS profile is defined by the path selected for the VC in the network, and the level of QoS assigned at each output port along the path, by choosing a certain queue (i.e. traffic class).
- *Scenario 2* assumes a coarser granular model where several VCs share a subset of the QoS characteristics. More specifically, several VCs may share the same path in the network, being differentiated among themselves only by the traffic class (i.e. output queue).

3 Related Work

Some of the work done in the area of SDN-based, on demand provisioning of network resources is targeted towards automated, policy-based network provisioning [6, 7], while other is targeted towards traffic engineering across Wide Area Networks (WANs) [8–12]. Dynamic allocation of network resources is also required inside the data centers and many studies address this challenge. As an example, an Openflow-based algorithm for allocation of bandwidth resources between virtual machines in data centers is presented in [13], while in [14] the authors describe a platform for integrated provisioning of compute, storage and network resources in data centers. However, most of the related work focuses on the service logic for QoS-aware resource provisioning, leaving out the details of how the network resources are managed and provisioned in the data plane. In some of the papers, such as [11], the authors mention that the proposed QoS architecture is based on traffic classification and rate limiting at the edge of the network, although no description is provided on how the reservation of logical resources inside the SDNC is enforced in the forwarding devices. The current paper complements the aforementioned works in that it provides a description of how the high-level processes for resource reservation are materialized into the actual reservation of resources in the forwarding devices.

The work that is closest to the one presented here is [15], which defines a data plane QoS architecture for SDN based on similar principles (i.e. a combination of queues and rate limiters), but with different constraints for the resources. Unlike [15], the current paper contains a definition of how the resources are managed inside the SDNC in order to provide deterministic QoS. In addition, the two models for resource reservation described in the next sections are aimed to serve as a basis for building any SDN framework for bandwidth on demand provisioning with specific QoS parameters.

In [16] the authors present the implementation and testing of an SDN framework for network resource allocation with guaranteed QoS, for cloud services provisioning. Their solution, implemented and tested using Open vSwitch, relies on traffic classification using priority queues at the output ports. Although the work described herein follows the same implementation as [16], the two algorithms for VC provisioning proposed in this paper are suitable for various other applications and not only provisioning of cloud services.

A concept related to the work presented here, which is discussed in various research papers [9, 17], is Network-as-a-Service (NaaS). NaaS enables abstraction of the network infrastructure for end-to-end provisioning of network services, and QoS is often included as one of the criteria in the provisioning process. The authors in [17] present and SDN-based framework for NaaS. They describe an analytical model to provide end-to-end connectivity with bandwidth and delay constraints, and show that by allocating a sufficient amount of bandwidth for a traffic flow, the delay can be bounded. The data plane QoS control mechanism presented in the next section is based on the same argument.

In particular, the current paper proposes a detailed solution for network resource provisioning with QoS guarantees, covering both the data plane and the control plane of the SDN architecture: it precisely describes the data plane QoS mechanisms and it presents the control plane algorithm for resource management. Moreover, the implemented VC provisioning SDNC platform exposes a set of REST APIs for QoS configuration which opens the data plane QoS mechanism to other QoS-aware services. Finally, the data plane QoS mechanism is validated through a series of tests performed on a virtualized testbed based on Mininet [18].

4 Data Plane QoS Control Mechanism

As shown in Fig. 2, the proposed data plane QoS control mechanism uses rate limiters and priority queues to perform per-flow admission control, and prioritization among different traffic classes (i.e. one best effort class and several QoS classes). The admission control process occurs at the ingress forwarding devices and it is dynamically configured by the SDNC according to the measured network parameters and the demands for resources. Throughout the configuration process, the SDNC must maintain the data plane invariants (i.e. relations between the resources) described later in this section. These invariants ensure that there is no congestion in the output queues for the QoS flows. Based on the observations in [19], by controlling the congestion in the output queues, the queueing delay experienced by the packets can be bounded. In the current work, the configuration of the data plane devices (i.e. priority queues) is administrated by the SDNC through the Open vSwitch Database Management (OVSDB) protocol [20], on the D-CPI. OVSDB works for Open vSwitch software switches [21], which represent the base of the virtualized environment used for testing the proposed services. For other types of network elements, other protocols such as Network Configuration protocol (NETCONF) [22] can be used.

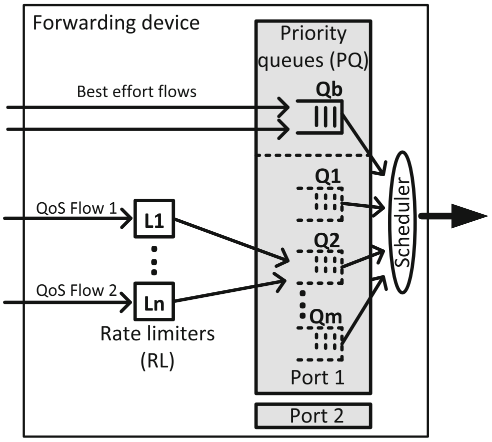


Fig. 2. Resource model for forwarding device

The bandwidth of each output port inside a forwarding device is shared among several priority queues: (1) a queue for best effort flows (Qb), and (2) one queue for each QoS class (Q1…Qm) (Fig. 2). Four parameters are considered for the priority queues: the minimum serving rate (R_{min}), the maximum serving rate (R_{max}), the queue size (S), and the queue priority. The scheduler at the output port (Fig. 2) is responsible for serving the priority queues according to the configured rates.

For the best effort queue, Qb, there is no per flow resource management (i.e. no per flow admission control using rate limiters) hence, all best effort flows share Qb. However, the bandwidth associated with Qb may not be enough to serve all the best effort flows, causing congestion thus leading to a poor level of service for this traffic class (e.g. increased delay, packet drops). The following relation exists between the serving rates of each priority queue, where C_p is the capacity of the channel:

$$R_{min} \leq R_Q \leq R_{max} \leq C_p \tag{1}$$

The rate guaranteed for a queue is R_{min} (Eq. 1). However, the rate of a queue (R_Q) can increase up to R_{max}, if there is bandwidth available at the output port (i.e. the port's bandwidth is shared).

The QoS flows are processed in two stages: the first stage consists of rate limiters (i.e. L1, Ln). The rate limiters perform admission control for the incoming flows according to a set of token bucket parameters: average rate R_L, which is guaranteed for a flow (Committed Information Rate), and bucket size B which denotes the maximum burst size (Excess Information Rate). The second stage consists of priority queues (i.e. Q1, Q2…Qm), one for each QoS traffic class. The set of priority queues used for the QoS flows is denoted as the QoS slice for the remaining in the paper Considering that $R_{L,i}$ is the guaranteed rate for the QoS flow arriving at rate limiter Li, and $R_{min,Qj}$ is the guaranteed serving rate of a priority queue, Qj (which belongs to the QoS slice), Eq. 2 describes the resource allocation for Qj. N represents the set of rate limiters that are applied to the QoS flows belonging to the traffic class associated with Qj (one rate limiter per QoS flow). M denotes the set of priority queues in the QoS slice.

$$\sum R_{L,i} \leq R_{min,Qj}, i \in N, j \in M \tag{2}$$

By applying the constraint specified in Eq. 2 to each priority queue in the QoS slice, the SDNC ensures that there is always enough bandwidth in the QoS slice to serve QoS flows with at least the guaranteed rate (i.e. R_L). A QoS flow may get additional bandwidth, as specified by the burst size (B), if there is bandwidth available in the priority queue of the corresponding traffic class. The integration of rate limiting and prioritization allows the SDNC to control congestion on the output queues, hence offering bandwidth and delay bounds for the QoS flows. The equation governing the slicing of the entire output port bandwidth is:

$$\sum R_{min,Qi} + R_{min,Qb} \leq C_p, i \in M \tag{3}$$

Equation 3 translates into that the sum of the guaranteed bandwidth for the priority queues and the best effort queue cannot be higher than the capacity of the port. Furthermore, the available bandwidth (i.e. not reserved or not used at one moment) at the output port is shared among the queues, including the best effort queue (i.e. Qb), according to their priorities, up to the R_{max} level of each queue. The sharing of the bandwidth available at the output port is managed by the SDNC, which also manages the queuing delay by controlling the congestion at the output port. Besides, the end-to-end propagation delay for the traffic flows is reflected in the length of the path computed by the CBR inside the SDNC. Hence the SDNC has control over the two major delay components: queuing delay and end-to-end propagation delay.

5 Prototype Implementation

Figure 3 depicts the architecture of the SDNC platform which is currently under implementation. The data plane is based on Open vSwitch (OVS) software switches, and it is emulated by using Mininet [18]. OVS offers the possibility to configure queues for traffic control at the output ports with the following parameters: R_{min},R_{max}, C_q and priority, with the same semantics as described in Sect. 4 [18]. In the current implementation, the OVSDB protocol is used on the D-CPI to remotely configure the data plane QoS resources [20] from the SDNC, based on the concepts described in the previous section. Correspondingly, the OF v1.0 protocol is used by the SDNC to install flow entries in the flow tables inside the forwarding devices (Fig. 3). The SDNC described herein is implemented as an extension to the open source SDNC platform Floodlight [23].

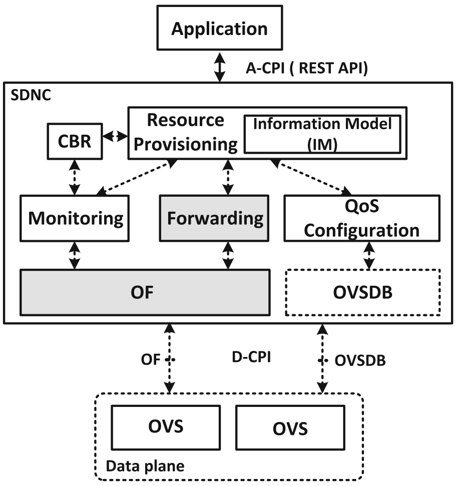


Fig. 3. Architecture of the VC provisioning SDNC platform

Some of the modules illustrated in the architecture are re-used from Floodlight (depicted in grey in the figure) while the rest of the modules are newly added or under implementation. The role of each module can be summarized as follows:

- *OF* and *OVSDB* Modules: they realize the communication with the forwarding devices at the D-CPI. OF is used to install entries in the flow tables of the forwarding devices, and the OVSDB management protocol is used to configure the rate limiting and priority queues in the data plane forwarding devices.

- *Monitoring* Module: it gathers statistics from the network and applies various processing on these statistics such as averaging over several measurements to cope with spikes in the amplitude of the measured parameter, which otherwise negatively affects the routing algorithm (causing unnecessary oscillations).
- *CBR* Module: implements the Shortest Widest Path (SWP) [24] algorithm, which consists of two steps: (1) choses a path which satisfies the bandwidth constraint and (2) if multiple paths exist after step 1, it chooses the one with the lowest end-to-end delay. Because the monitoring and routing operations are computationally intensive, they may hinder the performance of the SDNC when the arriving rate of VC requests is high. In order to alleviate this issue, the current implementation offers the possibility to adjust several performance related parameters (e.g. rate of monitoring probes, rate of route computations) of the Monitoring and CBR modules through a REST API.
- *Forwarding* Module: it provides capabilities for pushing OF entries into the flow tables of the forwarding devices.
- *QoS Configuration* Module: it exposes a set of primitives for configuring the QoS resources in the data plane, through the OVSDB protocol. These primitives are used by the modules residing inside the controller through a Java API, and by external applications through a REST API exposed at the A-CPI.
- *Resource Provisioning* (RP) Module (Fig. 3): it contains the logic for the VC provisioning service. To realize its purpose, the RP module uses the capabilities offered by other internal modules to get information about the state of the network (Monitoring module), to get routes that satisfy the QoS constraints (CBR module), and to configure the QoS mechanism in the data plane (QoS Configuration module). Furthermore, it implements the Information Model (IM) depicted in Fig. 4. One of the challenges for the RP module is to maintain a consistent view of the resources in the IM in order to avoid erroneous allocation of resources to different VCs. This may lead to unwanted congestion in the output queues.

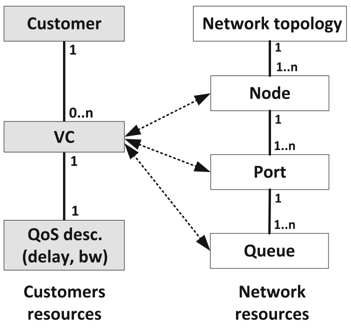


Fig. 4. Information Model (IM)

As the IM in Fig. 4 suggests, the resources used by a customer (left side) are a subset of the network resources (right side). With respect to the customer resources, a VC has a QoS descriptor which can be modified over time if the customer demands a change in the VC. The following two components define the QoS characteristics of the VC:

- A path in the network comprising a set of nodes and ports. The path reflects a certain level of QoS as computed by the CBR algorithm.
- The traffic class, which is defined by the output queue used to forward the traffic belonging to the VC at each output port.

In the light of the above, the strengths of the prototype SDNC platform are:

- The RP module provides a REST API for external applications to demand VCs with specific QoS, thus facilitating the convergence between the applications and the network
- The QoS Configuration module exposes a REST API for configuring the data plane QoS resources such as creation, modification and deletion of priority queues. This facilitates the creation of external network services that require QoS support (i.e. queue management) from the SDNC.
- It provides a REST API for configuration of the performance-related parameters of the Monitoring and CBR modules.

6 The VC Provisioning Algorithm

Considering that the requests for services arrive from external entities (e.g. customers), the initial processing and validation of these requests belongs to the provider's management plane and is out of the scope of this paper. Moreover, it is assumed that the provider's network architecture is SDN-based, hence the entity receiving the requests for service provisioning from the management plane is the SDNC. Depending on the VC scenario considered, the VC provisioning algorithm implemented in the RP module varies slightly. This is explained in the following, and illustrated in Fig. 5.

6.1 Scenario 1

Initially, the request for a new VC is received by the SDNC. The CBR algorithm is executed to compute a path that satisfies the QoS demands for the new VC. If no feasible path for the VC is found (satisfying the QoS requirements), the request is rejected and a "Failure" reply is sent to the requesting entity. Conversely, if there is a feasible path for the new VC, the IM is updated with new objects reflecting the addition of a new VC with the corresponding QoS descriptor to the requesting customer (Fig. 4). Moreover, the customer resources (i.e. bandwidth) have to be subtracted from the network resources, within the IM. In the next step the RP module sends the QoS configuration commands to the forwarding devices along the computed path. The QoS configuration comprises rate limiters on the ingress device, priority queues on the provider core devices and the egress device, matching the QoS for the requested VC. The OF entries are also installed into the forwarding devices allowing the creation of the path for the VC. If the configuration of the data plane forwarding devices is successful the RP module replies with a "Success" status to the initial VC request. On the other hand, if the configuration is not successful, the objects created in the IM are destroyed, the resources are de-allocated, and a "Failure" reply is sent to the customer.

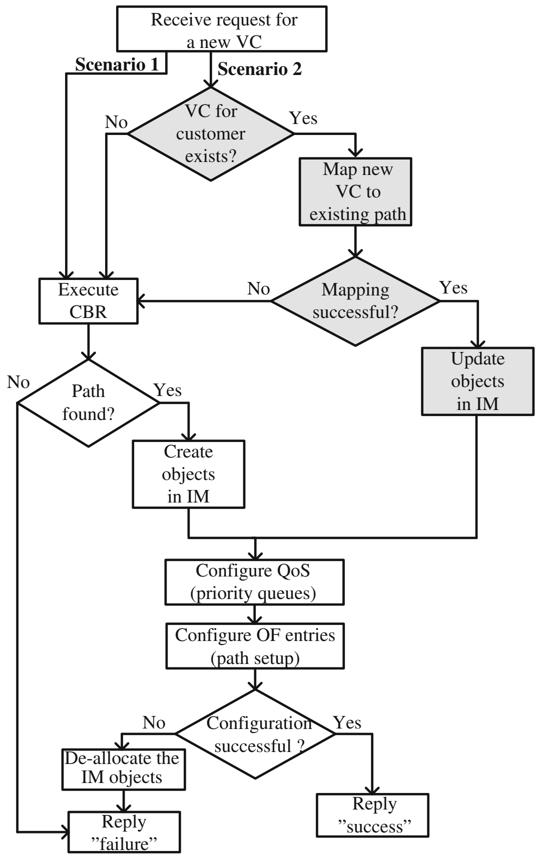


Fig. 5. Flow diagram for the VC provisioning algorithm

6.2 Scenario 2

The grey blocks in Fig. 5 denote the portion of the logic which is particular to this scenario. After receiving the VC request, the RP module verifies if there is already a VC established between the same end-points, for the customer who has initiated the request. If there is no existing VC, then the algorithm proceeds in a similar manner as described for scenario 1. On the other hand, if a VC already exists, the RP module attempts to map the new VC to the path of the existing VC, considering the QoS characteristics and the availability of data plane resources (i.e. bandwidth) along the existing path. While it is possible that the new VC is mapped on the same path as the existing VC, different queues are used at each output port in order to reflect the difference in QoS characteristics between the two VCs. If the mapping is possible, the RP module updates the IM: the existing VC is allocated more resources, and potentially a second set of queues to denote that there are two different QoS flows multiplexed within a VC. After the IM is updated, the logic continues as explained in the previous scenario.

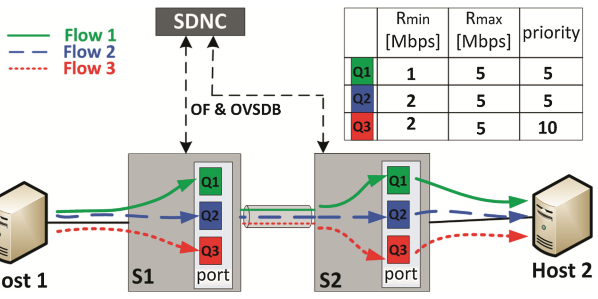
Considering both scenarios, the difference between them lies in the way the VCs belonging to the same customer are managed. For the first scenario each VC is characterized by a path (i.e. a set of devices and output ports) and the queue to be used at each output port to forward the traffic for that VC. In other words, each VC is handled individually thus its QoS profile can be accurately controlled, ensuring better QoS isolation between the VCs. In contrast, for the second scenario several VCs belonging to one customer may share the same path. Therefore, even if the VCs are assigned different queues along the path, they are still handled as a single unit. This significantly reduces the flexibility in cases when the VCs have to be rerouted to a new path for traffic engineering purposes: the new path has to satisfy more various QoS constraints since several QoS flows are inter-dependent due to the shared path.

As a result, the provisioning model described in Scenario 1 is better suited for cases where good QoS isolation between the VCs is mandatory, such as provisioning of VCs for transport over carrier packet networks using IP/MPLS. On the other hand, scenario 2 is suitable for business models where customers purchase VCs that carry traffic flows with different QoS requirements between the same two end-points, and strict QoS isolation between the traffic flows multiplexed in the same VC is not mandatory.

At the same time, the main advantage of the provisioning model for Scenario 2 is that the computational intensive CBR algorithm can be avoided in some of the cases (Fig. 5), thus reducing the initial delay for the flow setup.

7 Validation of QoS Mechanism

The testbed illustrated in Fig. 6 is used to validate the proposed data plane QoS mechanism. More precisely, the purpose of the tests is to verify if the behavior of the queues in OVS instances is according to the QoS mechanism described in Sect. 4, enabling further development of the proposed QoS aware services. Mininet has been used to emulate the data plane. A simple network topology consisting of two OVS instances has been chosen for the test, just enough to investigate the queue behavior of the data plane forwarding devices.



	Rmin [Mbps]	Rmax [Mbps]	priority
Q1	1	5	5
Q2	2	5	5
Q3	2	5	10

Fig. 6. Mininet testbed

Both devices (S1 and S2) have three queues at the output ports (i.e. Q1, Q2, and Q3), each queue being configured as described in the figure (with R_{min},R_{max}, and priority).

During the tests, the queue management (e.g. creation, modification, deletion) was done through the REST API exposed by the QoS Configuration module (Fig. 3). Upon receiving a REST request for queue configuration, the QoS Configuration module converts the request into a set of OVSDB commands, which are further sent to the devices by the OVSDB module. The capacity of the output ports on S1 and S2 is 5 Mbps.

To test the behavior of the queues, three TCP traffic flows (i.e. flow 1, flow 2, and flow 3) are sent from host 1 to host 2 in various combinations, each of them having a rate of 10 Mbps. Each traffic flow is mapped onto one of the queues. The traffic flows used for the current tests do not emulate realistic network traffic volumes. Instead they are just used to drive the rate limiting queues in order to illustrate their behavior. With this setup, by looking at the flow throughput it is possible to validate if the behavior of the queues corresponds to their configuration and to the expected behavior (as presented in Sect. 4). In particular, the most interesting characteristic to investigate is how the available bandwidth is shared among the queues.

For the first test, the traffic flows are transmitted one at a time. The average throughput (over approx. 30 instantaneous samples) measured for each flow is shown in Fig. 7, together with the standard deviation. As the results illustrate, the traffic flows have an average rate of 4 Mbps thus indicating that each flow is given all the capacity of the output port as long as there are no other flows transmitting at the same time. However, according to the queue configuration (R_{max} is 5 Mbps) the average flow throughput should be 5 Mbps. In spite of this, the impact of lost packets on TCP's congestion avoidance mechanism reduces the average throughput to 80 % of the port capacity (thus only 4 Mbps). Besides, the small difference in average throughput for the three flows can be attributed to several causes such as traffic generation tools, measurement operation, packet scheduler but also to the fact that the testbed uses software switches, which have less than ideal performance. Nevertheless, the results confirm that the capacity of the output port can be used entirely by one flow if the other flows are not transmitting.

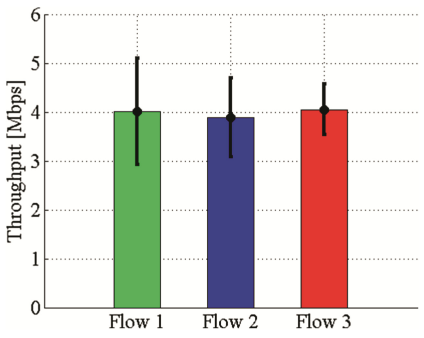


Fig. 7. Average flow throughput (with standard deviation) when each flow is transmitted individually

For the second test, flows 1 and 2 are transmitted simultaneously hence they share the capacity of the output port. Capacity sharing is done according to the configuration of the corresponding queues (Q1 and Q2): the guaranteed rates (R_{min}) for flow 1 and flow 2 are 1 and 2 Mbps respectively, summing up to 3 Mbps. Since the capacity of the port is 5 Mbps, the remaining bandwidth (2 Mbps) is shared between the two queues and reflected in the measured average flow throughput as illustrated in Fig. 8 (left side). As shown in the figure, flow 1 is given an extra bandwidth of 0.48 Mbps and flow 2 an extra bandwidth of 0.84 Mbps, beyond the guaranteed minimum rates for each of the flows (1 and 2 Mbps respectively). While both flows have equal priority (5), the extra bandwidth is not evenly distributed, but proportionally to the guaranteed minimum rate for each flow, resulting in almost twice extra bandwidth for flow 2 (0.84 Mbps) than for flow 1 (0.48 Mbps). This behavior is as described in Sect. 4 (Eq. 3) with the addition that the bandwidth sharing depends also on the guaranteed rate (R_{min}) and not only on the priority configuration of the queues.

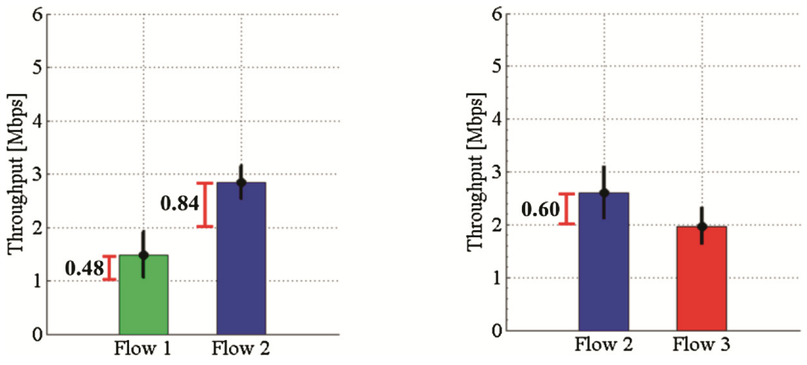


Fig. 8. Average flow throughput (with standard deviation) for test 2 (left side) and test 3 (right side)

The third test addresses the bandwidth sharing among queues with different priorities (i.e. Q2 and Q3) and the results are depicted in Fig. 8 (right side). In this case, the higher priority queue (Q2) gets all the extra bandwidth (0.6 Mbps), while the lower priority queue only gets the guaranteed minimum rate (approx. 2Mbps). In this case, due to queue prioritization, Q3 (thus flow 3) would be given extra bandwidth only after Q2 would have received its maximum rate (5 Mbps). Since for this particular configuration the maximum rate for Q2 is equal to the port capacity, Q3 receives extra bandwidth only if flow 2 (transmitting from Q2) lowers its rate under 2.6 Mbps.

Concluding, the test results confirm that the proposed VC provisioning service, which is supported by the data plane QoS mechanism described in Sect. 4, can be implemented and tested in a virtualized environment which is based on OVS. To ensure QoS guarantees in such a setup, the SDNC must carefully manage the queue configuration and allocation, and the flow admission in the network.

8 Conclusion

In this paper, data plane QoS mechanisms and the control plane logic for managing the data plane resources have been proposed. The combination of a prioritization-based QoS mechanisms and a CBR algorithm resembles the DiffServ-enabled MPLS-TE, as described in [2]. The benefit of applying SDN is that the provisioning of VCs becomes more agile and automated.

As also mentioned in Sect. 3, the current work represents a contribution in the field because discusses the aspect of network resource provisioning with QoS guarantees at all the layers of the SDN architecture: it proposes a detailed data plane QoS mechanism, together with the control plane algorithms for managing the data plane resources. The main goals of the proposed models (both data plane and control plane) are to be realizable with the available technology and offer predictable QoS for a wide range of network resource provisioning scenarios. An architecture for a platform incorporating all these mechanisms has been proposed, and the details for a prototype implementation of the platform have been presented. The platform opens the possibility for integration of the applications and the network through the REST APIs exposed at the A-CPI. In addition, the capabilities for managing the data plane resources (i.e. creation, modification and deletion of priority queues) are exposed as REST APIs, enabling the creation of other SDN-based network services that need QoS support in the data plane.

The data plane QoS architecture has been tested in a virtualized environment based on Open vSwitch and the qualitative and quantitative results confirm that the proposed mechanism behaves as described in this paper. Future work will focus on a thorough quantitative comparison between the two scenarios with respect to the level of QoS isolation between individual flows, the maximum network throughput achieved in each case, and the request blocking ratio. The final implementation of the VC provisioning SDNC platform will be available as open source to the research community for experimenting and creating services on top of it. At the moment, it is possible to get the experimental version of the source code by contacting the authors.

References

1. Boucadair, M., Jacquenet, C.: Software-Defined Networking: A Perspective from within a Service Provider Environment. RFC 7149, March 2014
2. Filsfils, C., Evans, J.: Engineering a multiservice IP backbone to support tight SLAs. Comput. Netw. Int. J. Comput. Telecommun. Network. **40**(1), 131–148 (2002)
3. Blake, S., Black, D., Davies, E., Wang, Z., Weiss, W.: An architecture for differentiated services. RFC 2475, December 1998
4. Open Networking Foundation: SDN Architecture. Issue 1, June 2014
5. Open Networking Foundation: OpenFlow Switch Specification, version 1.0.0. (2009)
6. Bari, M.F., Chowdhury, S.R., Ahmed R., Boutaba, R.: PolicyCop: an autonomic QoS policy enforcement framework for software defined networks. In: IEEE SDN for Future Networks and Services, Trento, Italy, pp. 1–7, November 2013
7. Ferguson, A.D., Guha, A., Liang, C., Fonseca, R., Krishnamurthi, S.: Participatory networking: an API for application control of SDNs. ACM SIGCOMM Comput. Commun. Rev. **43**(4), 327–338 (2013)
8. Hong, C.Y., et al.: Achieving high utilization with software-driven WAN. In: Proceedings of the ACM SIGCOMM, Hong Kong, China, pp. 15–26 (2013)

9. Bueno, I., Aznar, J.I., Escalona, E., Ferrer, J., Garcia-Espin, J.: An OpenNaaS based SDN framework for dynamic QoS control. In: IEEE SDN for Future Networks and Services, Trento, Italy, pp. 1–7, November 2013
10. Mechtri, M., Houidi, I., Louati, W., Zeghlache, D.: SDN for inter cloud networking. In: IEEE SDN for Future Networks and Services, Trento, Italy, pp. 1–7, November 2013
11. Jain, S., et al.: B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Comput. Commun. Rev. **43**(4), 3–14 (2013)
12. Egilmez, H.E., Dane, S.T., Bagci, K.T., Tekalp, A. M.: OpenQoS: an openflow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks. In: Proceedings of the Signal and Information Processing Association Annual Summit and Conference, Hollywood, California, US, pp. 1–8, December 2012
13. Guo, J., Fangming, L., Haowen, T., Yingnan, L., Hai, J., John, L.: Falloc: fair network bandwidth allocation in IaaS datacenters via a bargaining game approach. In: Proceedings of the ICNP, Gotingen, Germany, pp. 1–10, October 2013
14. Benson, T., Akella, A., Shaikh, A., Sahu, S.: CloudNaaS: a cloud networking platform for enterprise applications. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, Cascais, Portugal (2011)
15. Kim, W., et al.: Automated and scalable QoS control for network convergence. In: Proceedings of the INM/WREN, San Jose, California, US (2010)
16. Anand, V. A., Kaiqi, X.: Quality of Service (QoS) guaranteed network resource allocation via software defined networking (SDN). In: 12th International Conference on Dependable, Autonomic and Secure Computing, Dalian, China, pp 7–13, August 2014
17. Qiang, D.: Network-as-a-service in software-defined networks for end-to-end QoS provisioning. In: 23rd Wireless and Optical Communication Conference (WOCC), Newark, NJ, pp. 1–5, May 2014
18. Bob, L., Brandon, H., Nick, M.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks
19. Davie, B., et al.: An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246, March 2002
20. Pfaff, B., Davie, B.: The Open vSwitch Database Management Protocol. IETF RFC 7047, December 2013
21. http://openvswitch.org/. Accessed, March 2015
22. Enns, R., et al.: Network Configuration Protocol (NETCONF). IETF RFC 6241, June 2011
23. http://www.projectfloodlight.org/floodlight/. Accessed, March 2015
24. Wang, Z., Crowcroft, J.: Quality-of-service routing for supporting multimedia applications. IEEE J. Sel. Areas Commun. **14**(7), 1228–1234 (1996)

Poisson-Based Anomaly Detection for Identifying Malicious User Behaviour

Andrey Sapegin(✉), Aragats Amirkhanyan, Marian Gawron, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam, P.O.Box 900460, 14440 Potsdam, Germany
{andrey.sapegin,aragats.amirkhanyan,marian.gawron, feng.cheng,christoph.meinel}@hpi.de

Abstract. Nowadays, malicious user behaviour that does not trigger access violation or alert of data leak is difficult to be detected. Using the stolen login credentials the intruder doing espionage will first try to stay undetected: silently collect data from the company network and use only resources he is authorised to access. To deal with such cases, a Poisson-based anomaly detection algorithm is proposed in this paper. Two extra measures make it possible to achieve high detection rates and meanwhile reduce number of false positive alerts: (1) checking probability first for the group, and then for single users and (2) selecting threshold automatically. To prove the proposed approach, we developed a special simulation testbed that emulates user behaviour in the virtual network environment. The proof-of-concept implementation has been integrated into our prototype of a SIEM system — Real-time Event Analysis and Monitoring System, where the emulated Active Directory logs from Microsoft Windows domain are extracted and normalised into Object Log Format for further processing and anomaly detection. The experimental results show that our algorithm was able to detect all events related to malicious activity and produced zero false positive results. Forethought as the module for our self-developed SIEM system based on the SAP HANA in-memory database, our solution is capable of processing high volumes of data and shows high efficiency on experimental dataset.

Keywords: Anomaly detection · Intrusion detection · User behaviour · Authentication

1 Introduction

To withstand the huge flow of attacks on the corporate network, a modern enterprise security department possesses plenty of security systems, such as firewalls, Intrusion Detection Systems (IDS) or Security Information and Event Management Systems (SIEM), and Data Leak Prevention Systems (DLP). If correctly integrated into enterprise environment in accordance with thoroughly developed information security controls, these systems could effectively identify an attack

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 134–150, 2015.
DOI: 10.1007/978-3-319-25744-0_12

on the network, or data leakage by company's employees. Even if the attack was not detected immediately, or if the attacker cleans log messages from the exploited server, the copy of the original log messages should be available on the central logging server or SIEM system. Later historical analysis of these logs could reveal the most of attacks or access violation events.

But how could these systems identify an intrusion, that does not induce enormous data flow and/or any access violation at all? For example, if the user identity (e.g., credentials) were stolen using social engineering methods and an attacker uses it to gather information about the enterprise network via VPN. Or if the company employee doing espionage collects information using only the systems and data he is authorised to access[1].

Such cases are usually not in the focus of intrusion detection or data leak prevention systems. On the one hand, signature-based intrusion detection systems are focused on access violation or hacking scenarios [1–3] and often cannot effectively detect such issues. On the other hand, most anomaly-based intrusion detection systems usually operate on the large set of features, mainly focusing on protocol anomalies, network traffic analysis and policy violations [4–7], while others analyse too specific user activities, e.g. VPN access or host-wise user actions [8–10]. Further, data leak prevention systems mainly concentrate on data itself and are yet unable to control data movements between departments within the organisation [11,12].

Nonetheless, we believe that such malicious user activity should be processed and can be easily detected. Indeed, the malicious user activity pattern should differ from the usual one and this difference could be modelled. In particular, we focus on patterns of user logon events and believe that in case of malicious behaviour such pattern changes are trackable.

During the past two decades there was a significant progress in the machine learning methods used for anomaly detection, including user behaviour models. The most popular approaches utilise Principal Component Analysis [13,14], Support Vector Machines and neural networks [15–17], Naive Bayes classifiers [18,19], Markov chains and Hidden Markov Models [20,21], as well as hybrid techniques [22,23]. Using these techniques authors often reach 90–100% precision and very low false positive rates. However, presented solutions are rather heavy for smooth processing of high data volumes, which is relevant for intrusion detection systems. As already mentioned above, existing systems based on these techniques either do not focus on the described particular type of user anomaly or are too specific for the analysis of user behaviour. In contrast, we offer a lightweight approach to effectively detect user's misbehaviour by tracking logon events in the whole network.

In this paper we propose a Poisson-based anomaly detection algorithm to identify malicious user behaviour. Different to the techniques mentioned above, the Poisson model is fairly simple, but still could be used for modelling user arrivals [24,25]. In particular, we use it to analyse the frequency of user logon

[1] An insider could also intentionally avoid moving large portions of data or copying it to untrusted storage to stay undetected by Data Leak Prevention system.

events on different workstations and servers, since the simple Poisson model fits perfectly for high-speed event analysis. Our anomaly detection approach considers not only logon patterns of single users, but also behaviour of user groups. This measure helps us to avoid producing false positive results on unusual user login actions, which are however common for other users in the same user group. Further, we developed a self-adaptive threshold selection mechanism for anomaly detection, which also allows us to significantly reduce the number of false positive events. To check the accuracy of the resulted proof-of-concept solution, we utilise our experimental dataset, generated with the newly developed simulation platform.

The rest of this paper is organised as follows. Section 2 describes the simulation of user behaviour and the generated dataset, Sect. 3 provides details on the anomaly detection algorithm and Sect. 4 demonstrates the detected results. Finally, we mention future work in Sect. 5 and conclude in Sect. 6.

2 Simulated Dataset

Since we are interested in special cases of malicious user behaviour, we decided to simulate the dataset, that will contain such relevant cases. Beside that, the simulated dataset also has benefits for development of supervised machine learning algorithms like Poisson-based anomaly detection. Indeed, there is no need to clean-up training data since we could simply choose not to include attacks in the scenario for algorithm training. It is also easier to analyse efficiency of the algorithm, as we know exactly which malicious behaviour should be captured by anomaly detection from the simulated testing dataset.

2.1 Target Scenarios

We decided to analyse the case when a user is authorised to access a resource, but the resource is however never used neither by this user nor by any other users (from the same group) under normal working conditions. This case emulates an example of malicious user behaviour, when the user credentials were stolen and are now used to collect information from the enterprise network avoiding any access violations to stay undetected by an intrusion detection system. Under this case we describe 2 scenarios: (1) normal, which will be used to generate a training dataset, and (2) abnormal, used to generate a testing dataset, where all malicious events should be captured automatically. The details are illustrated in Figs. 1 and 2.

Figure 1 describes normal behaviour of users in a small network. Alice, Bob and Carol are usual network users. All users have access to both Server_1 and Server_2, however, only Bob and Carol regularly log on to the Server_1, and nobody accesses Server_2. Like in the real network, a user first needs to login to their PCs, and only then he could access a server remotely. The users could also login on each other's PCs, e.g. Carol logs on to the PC1 of Alice. Finally, the

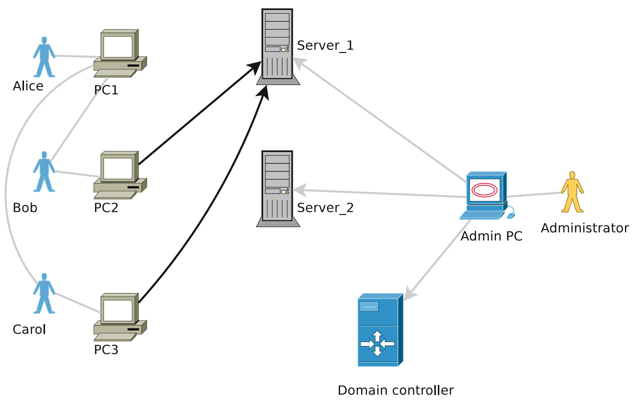


Fig. 1. Normal behaviour of users in the network

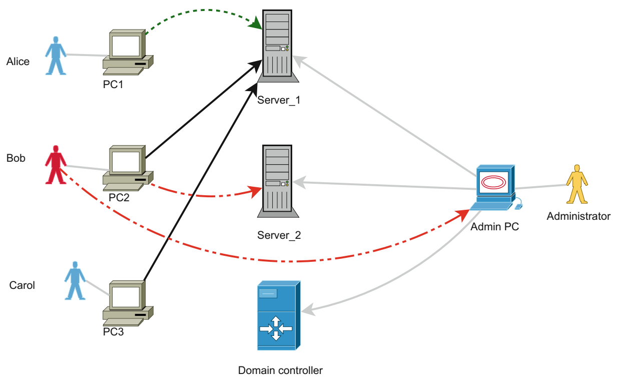


Fig. 2. Malicious user behaviour of user Bob

network administrator has an access to all virtual machines in the network and regularly uses all Servers (including Domain Controller) as well as his own PC.

Different to the normal scenario, Fig. 2 shows an example of malicious user behaviour.

In Fig. 2, Bob connects to the Server_2, which was never used before neither by him, nor by other users, despite the fact, that all users are authorised to access this server. We classify such cases as malicious user behaviour. Bob's local connection to Admin PC should be then classified malicious as well. However, the fact, that Alice connects to the Server_1 should be classified as benign, even though it was also not expected for her. We do not mark such cases as suspicious, since the Server_1 has been regularly accessed by other users from the same group (Bob and Carol). We therefore classify only Bob's behaviour as malicious and will try to capture it during the analysis phase.

To implement the described scenario, we set up a testbed with Windows domain and use a self-developed simulation tool to generate Active Directory logs for further analysis.

2.2 Virtual Testbed

Our simulation scenario bases on the testbed with virtual machines running on the VMware ESXi virtualisation server. For all PCs — PC1–PC3, as well as Admin PC — we use virtual machines with Microsoft Windows 7. Server_1 and Server_2 have Microsoft Windows Server 2003 installed, while for the Domain controller we setup Microsoft Windows Server 2012. The Domain controller just provides Active Directory services, DHCP and DNS, meanwhile all virtual machines have Remote Desktop [26] enabled. Using this testbed, we simulate logins on the user PCs as well as connections to the servers and domain controller.

To simulate user logon and logoff events, we decided to utilise the VNC [27] access functionality of the VMware ESXi hypervisor. We have created a program in Python, that is able to simultaneously connect to different virtual machine consoles and execute user actions, such as logon and logoff, based on the screen capture and recognition functions provided by Python Imaging Library [28]. To simulate connections from user PCs to servers, we use Remote Desktop Protocol [26]. The simulation software is able to unlock the computer (by sending the "Ctrl-Alt-Del" combination and typing in a password) and then start the PowerShell script located on the Desktop of each virtual PC. This script opens the connection via Remote Desktop to the predefined server. Since every logon event (both local and via RDP) is saved by as the Windows Event and reported to the domain controller, we are able to realistically reproduce Active Directory logs for our scenario of user behaviour, including both local logon/logoff events and connections to server.

To control the simulation process, we specify an actions order in the scenario files, that have the format as presented in Fig. 3.

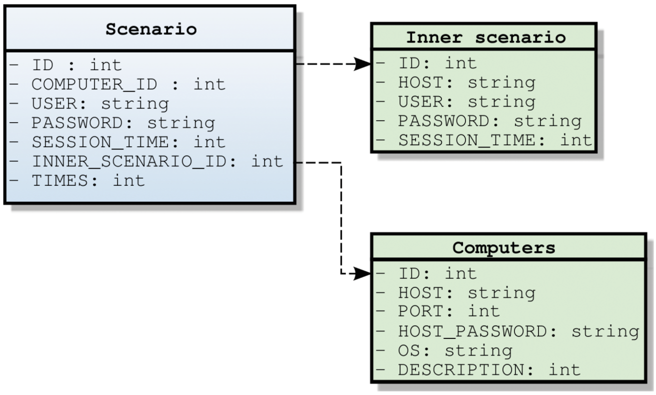


Fig. 3. Scenario schema used in simulation software

Table 1. Computers

ID	Host	Port	Host password	OS	Description
1	192.168.42.20	5908	<PC1pass>	win7	PC1
2	192.168.42.20	5905	<PC2pass>	win7	PC2
3	192.168.42.20	5906	<PC3pass>	win7	PC3
4	192.168.42.20	5901	<PC4pass>	win7	Admin PC

Table 2. Scenario for simulation of normal user behaviour

ID	Comp. ID	User	Password	Session time	Inner scenario ID	Times
1	4	Administrator	<adminpass>	300	1	15
2	4	Administrator	<adminpass>	200	4	10
3	4	Administrator	<adminpass>	300	5	20
4	4	Administrator	<adminpass>	215	0	30
5	1	Alice	<pass1>	205	0	40
6	2	Bob	<pass2>	195	2	66
7	1	Bob	<pass2>	199	0	6
8	3	Carol	<pass3>	300	3	45
9	1	Carol	<pass3>	280	0	25
10	2	Bob	<pass2>	220	0	30
11	3	Carol	<pass3>	200	0	44
12	4	Administrator	<adminpass>	300	0	11
13	1	Alice	<pass1>	250	0	22

Table 3. Inner scenario for simulation of normal user behaviour

ID	Host	User	Password	Session time
1	10.10.21.1 (Domain Controller)	Administrator	<adminpass>	235
2	10.10.21.110 (Server_1)	Bob	<pass2>	240
3	10.10.21.110 (Server_1)	Carol	<pass3>	250
4	10.10.21.110 (Server_1)	Administrator	<adminpass>	250
5	10.10.21.109 (Server_2)	Administrator	<adminpass>	230

We store the scenario schema as defined in Fig. 3 in .csv[2] files.

The 'Scenario' file describes user logins onto different PCs, one user and one PC per line. The entries in the 'Scenario' file are executed in parallel, but always wait for computer lock: like in the real world, only one user could use one PC at the same time. The specific PC is selected from 'Computers' file by ID. Each entry in the scenario file is executed several times, depending on the value in 'TIMES' column. Every such iteration ends when SESSION_TIME is expired.

[2] Comma-separated values.

Table 4. Simulation scenario containing malicious user behaviour

ID	Comp. ID	User	Password	Session time	Inner scenario ID	Times
1	4	Administrator	<adminpass>	300	1	15
2	4	Administrator	<adminpass>	200	4	10
3	4	Administrator	<adminpass>	300	5	20
4	4	Administrator	<adminpass>	215	0	30
5	1	Alice	<pass1>	205	0	40
6	1	Alice	<pass1>	205	6	40
7	2	Bob	<pass2>	195	2	66
8	3	Carol	<pass3>	300	3	45
9	2	Bob	<pass2>	220	0	30
10	3	Carol	<pass3>	200	0	44
11	1	Alice	<pass1>	205	6	40
12	4	Administrator	<adminpass>	300	0	11
13	2	Bob	<pass2>	245	7	31
14	1	Alice	<pass1>	250	0	22
15	4	Bob	<pass2>	200	0	25

Table 5. Inner simulation scenario containing malicious user behaviour

ID	Host	User	Password	Session time
1	10.10.21.1	Administrator	<adminpass>	235
2	10.10.21.110	Bob	<pass2>	240
3	10.10.21.110	Carol	<pass3>	250
4	10.10.21.110	Administrator	<adminpass>	250
5	10.10.21.109	Administrator	<adminpass>	230
6	10.10.21.110	Alice	<pass1>	229
7	10.10.21.109	Bob	<pass2>	230

The ‘Computers’ file stores information for VNC connection to the virtual machine, including hostname and password of the VNC service running on the VMware ESXi server. The different virtual machines are accessed through different ports.

If the scenario contains INNER_SCENARIO_ID, that is not 0, the inner scenario is executed. In the ‘inner scenario’ we describe remote connections from user’s PC to one of the servers. The options from the inner scenario will be used as parameters for PowerShell script to initiate Remote Desktop connection to the specified server for SESSION_TIME, defined in the inner scenario.

We now provide the content of our scenario files in Tables 1, 2, 3, 4 and 5.

All together, Tables 1, 2, 3, 4 and 5 reflect the normal and abnormal scenarios, described earlier in Figs. 1 and 2. While running these scenarios, we have captured 38568 Windows Events on the Domain Controller[3]. These events should also cover cases of malicious user behaviour (of user Bob), that we tried to detect using machine learning algorithm, as described in the next section.

3 Poisson-Based Anomaly Detection

Since the Poisson's distribution is the one commonly used for modelling user arrival and network traffic [25,29], we selected it to model user logon events on the workstations in the simulated network.

To identify anomalies in the user behaviour, we first divide the data into time intervals. We choose the appropriate time interval heuristically, since it depends on the time range of a dataset, the number of users and on how often they perform logon events[4]. For our simulated data we consider 15 min as optimal time interval.

After selection of the time interval, we calculate number of logon events per time interval for each user group on each workstation to check the probability of this number of logon events according to the Poisson's formula:

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{1}$$

where x is number of logon events per time interval. Please see Algorithm 1 for details.

The Algorithm 1 first calculates the value of λ_{group} for each {user_group, workstation} pair. For a Poisson's distribution, this value could be calculated as average number of logon events per time interval (line 2). The λ_{group} value represents a model of normal user group behaviour, based on the training data. On the lines 4–11 we apply our model on testing data to identify events, that differ from modelled normal behaviour of user groups. Using the same time interval as for training data, we count number of logon events for each {user_group,workstation} pair in the testing data (line 6) and calculate its probability (according to the Formula 1). On the line 7 we compare the probability of particular number of logon events for a user group on some workstation within a particular time interval with the threshold value. If the probability is less than threshold, we mark a triplet user_group,workstation,time_interval as *suspicious* on the line 8.

For all revealed *suspicious* events we utilise Algorithm 2 to identify which users from the highlighted groups caused it.

[3] Before execution of the analysis, the captured Windows Events need to be parsed and filtered. During this step, we have extracted 1958 filtered events available for the analysis.

[4] E.g., for a small enterprise with up to 10 users and few logon events per day on the sole company's internal server it hardly makes sense to set a time interval to less than 1 day. While for a big company with thousands of employees it could be reasonable to calculate number of logon events per minute.

Algorithm 1. Anomaly_detection_groups($threshold_{group}$)

```
 1: for all {user_group,workstation} from training_data do
 2:     $\lambda_{group}$ = average number of logon events per time_interval
 3: end for
 4: for all time_interval in testing_data do
 5:     for all {user_group,workstation} from testing_data do
 6:         logons = number of logon events
 7:         if PoissonsProbability({user_group,workstation}, logons, $\lambda_{group}$) < $threshold_{group}$ then
 8:             mark {user_group,workstation,time_interval} as suspicious
 9:         end if
10:     end for
11: end for
```

Algorithm 2. Anomaly_detection_users($threshold_{user}$)

```
 1: for all {user,workstation} from training_data do
 2:     $\lambda_{user}$ = average number of logon events per time_interval
 3: end for
 4: for all time_interval in testing_data do
 5:     for all {user,workstation} from testing_data where {user_group,workstation,time_interval} is suspicious do
 6:         if PoissonsProbability({user,workstation},$\lambda_{user}$) < $threshold_{user}$ then
 7:             mark {user,workstation,time_interval} as anomaly
 8:         end if
 9:     end for
10: end for
```

Similar to Algorithm 1, the Algorithm 2 calculates λ_{user} based on training data. First, we find an average number of logon events per time interval for each {user, workstation} pair (line 2). Next, from testing data, we select all users related to *suspicious* events identified by Algorithm 1 and check the probability of number of logon events per time interval for each user and each workstation (lines 4–10) using Formula 1. If the probability is less than threshold, we mark all events related to the triplet {user,workstation,time_interval} as **anomalies**.

Using a two-step probability check – first for user groups, and then for particular users from suspicious groups – we avoid a false positive anomaly alerts for situations when the user performs some action (login on the workstation), which is not usual for him personally, but usual for his user group.

The main limitation of the model based on the training data subset is that the behaviour of all users in the training data will be considered and modelled as normal. In the situations, where the unidentified intrusion happened first within time range of training data, the user behaviour of the intruder also will not be classified as anomalous in the testing data. However, if all anomalies in the training data could be easily identified and cleaned without anomaly detection approach, they could be also easily identified without it in the testing data, making anomaly detection approach useless.

Nonetheless, all new issues of anomalous user behaviour, that appear first in the testing data, should be revealed with this algorithm. Moreover, if the normal user behaviour could be simulated (see Sect. 2) instead of creating testing dataset from captured real data, it will guarantee that training dataset has no traces of any malicious actions and therefore improve the quality of anomaly detection on testing dataset.

To further improve our algorithm, we also identify the optimal value of both $threshold_{group}$ and $threshold_{user}$. Please see Subsect. 3.1 for details.

3.1 Identifying Optimal Threshold Value

To find an optimal threshold for probability of logon events, we use so-called "elbow method". First of all, we calculate number of detected suspicious groups for different $threshold_{group}$ values[5] using Algorithm 3.

To check different $threshold_{group}$ values, Algorithm 3 first calculates maximal Poisson's probability of number of logon events for each user group and workstation within a time interval (lines 1–8). The maximum probability value is used as a reference to set an upper bound for the $threshold_{group}$ (line 10). We then select 20 different threshold values from $max\_probability/2$ down to 0 and calculate number of detected suspicious groups using Algorithm 1 (lines 10–15) for each of 20 threshold values[6]. Finally, using these data, we find an

Algorithm 3. OptimalGroupThreshold

```
1: for all time_interval in testing_data do
2:   for all {user_group,workstation} from testing_data do
3:     interval_probability = PoissonsProbability({user_group,workstation},$\lambda_{group}$)
4:     if interval_probability > max_probability then
5:       max_probability = interval_probability
6:     end if
7:   end for
8: end for
9: i = 0
10: for $threshold_{group} = max\_probability/2$ to 0 step $max\_probability/20$ do
11:   Anomaly_detection_groups($threshold_{groups}$)
12:   suspicious_groups[i] = count number of suspicious groups
13:   thresholds[i] = $threshold_{group}$
14:   i++
15: end for
16: OptimalGroupThreshold = ElbowPoint(suspicious_groups,thresholds)
```

[5] Number of anomalies for different values of $threshold_{user}$ could be calculated in the similar way.

[6] Similar to this approach, we use Algorithm 2 to find optimal value of $threshold_{user}$. However, this value should be precomputed before we execute Algorithm 2. So there will be no suspicious user groups, that are checked on the line 5 of the Algorithm 2, since they are not found yet. Therefore, we disable this criteria for determining optimal threshold value and calculate Poisson's probability on lines 6–7 of Algorithm 2 for all {user,workstation} pairs.

optimal threshold value with so-called "elbow method" (line 16). According to this method, the optimal threshold value could be an "elbow point" — point where the curvature of the interpolated function reaches it's extremum. To find this extremum, we calculate a second order central difference for our set of 20 {threshold,number_of_suspicious_groups} points. Similar to second order derivative for functions, second order central difference reflects curvature for discrete data[7]. The maximum absolute value of second order central difference indicates therefore the maximal curvature in the corresponding point.

The second order central difference could be calculated as follows:

$$\delta^2_{y_k} = y_{k+2} - 2 * y_{k+1} + y_k \tag{2}$$

We therefore use Formula 2 to find the "elbow point", that should correspond to the optimal threshold value. Please see Algorithm 4 for details.

Algorithm 4. ElbowPoint(array,thresholds)

```
1: for i = 0 to length(array)-2 do
2:     $\delta^2_{array_i} = |array[i+2] - 2 * array[i+1] + array[i]|$
3:     if max_difference < $\delta^2_{array_i}$ then
4:         max_difference = $\delta^2_{array_i}$
5:         OptimalThreshold = thresholds[i]
6:     end if
7: end for
8: return OptimalThreshold
```

Algorithm 4 calculates the value of the second order central difference for first $n-2$ points from *array* and finds its absolute maximum — point with maximum curvature (lines 1–4). The corresponding threshold value (line 5) would be optimal according to the described method. The calculated $threshold_{group}$ and $threshold_{user}$ (found in a similar way) values are later used in Algorithms 1 and 2.

Therefore, we have developed a set of algorithms that are able to automatically determine optimal threshold values, depending on the dataset, and then identify cases of anomalous user behaviour (logon events), taking into account both user and group information. Please refer to the next section for sample results, received on the simulated dataset.

4 Anomaly Detection Results

Let's review the generated dataset and check if our algorithm is able to detect cases of simulated malicious user behaviour. We present distribution of logon-related events from the training dataset in Fig. 4, while Fig. 5 shows the same distribution for the testing dataset.

[7] Curvature of interpolated function based on discrete data points, e.g. number of suspicious groups or anomalies for different threshold values.

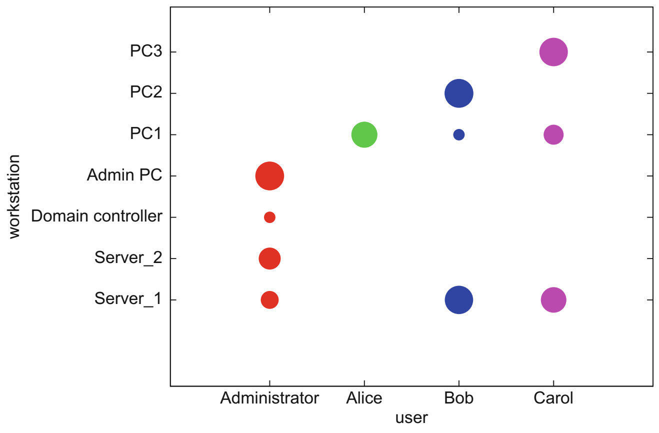


Fig. 4. Distribution of user logon events in the training dataset

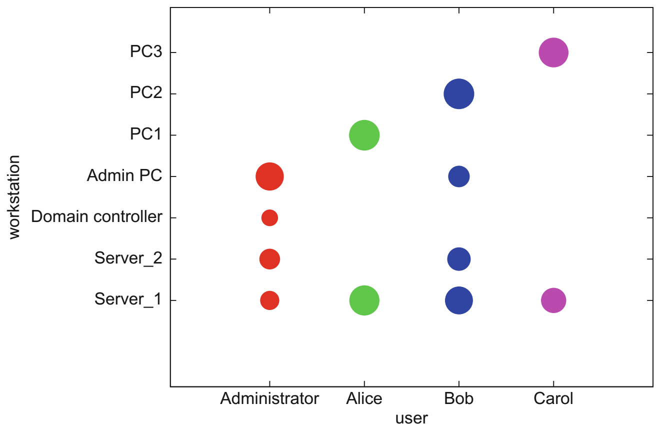


Fig. 5. Distribution of user logon events in the testing dataset

Figures 4 and 5 describe the distribution of simulated logon events in the testbed network. Each circle points to the logon of the user (on the x-axis) on the workstation or server (on the y-axis). The logarithmically scaled size of the circles reflects number of events for each particular user and workstation/server. Both Figures correspond to original simulation scenarios (see Sect. 2), which provides an extra proof, that no logon activities were lost during data filtering.

After our Poisson-based anomaly detection algorithm learns user behaviour model from training dataset, the model is applied on testing dataset to find anomalies, that do not fit into the learned model. We present the identified anomalies on Fig. 6.

Figure 6 shows results of anomaly detection using the same representation as Figs. 4 and 5. Compared to Fig. 5, it presents the subset of events (including

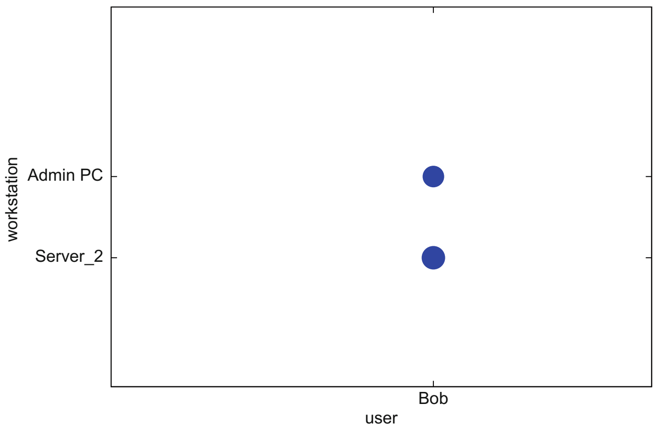


Fig. 6. Detected user behaviour anomalies

users and workstation), that were marked as anomalous by our algorithm. The results prove that our algorithm has successfully captured all logon-related events for user Bob on both Server_2 and Admin PC. All these events are malicious according to the scenario described in Sect. 2 and Fig. 2. Moreover, no other event was classified as anomaly, which implies that our approach to perform two-step probability check — first for user groups and then for each user individually — helped to avoid producing any false positive results.

The presented results demonstrate high efficiency of our approach for modelling of user behaviour. So, in our future work we plan to proceed from current proof-of-concept implementation to fully-featured module for our Security Information and Event Management System. Please see next section for further details.

5 Discussion and Future Work

Since the offered Poisson-based anomaly detection algorithm works well on the simulated dataset, we decided to integrate it into our Real-time Event Analysis and Monitoring System [30,31].

The usage of SAP HANA [32] and Object Log Format [33,34] in REAMS allows us to process high amounts of heterogeneous events from different original log formats. Even though in this paper we concentrate on our Active Directory testbed and Windows Events, deep log normalisation provided by REAMS enables processing of log messages in any format (currently supported formats include Windows Events, SAP Netveawer, SAP Moonsoon and various syslog message types such as Apache, Cisco IOS, iptables and many others). The Windows Event format is the most problematic for parsing, as it stores different data in the same schema elements as well as uses different schema elements for same data. This feature results in the situation, when the same data (e.g. username) should be also stored in different fields of normalised Object Log Format. Different to Windows Event, other log formats do not have such problems and their normalisation almost always guarantees that data from different log message types is always stored in the same fields of Object Log Format.

Therefore, only minor or no changes of our current algorithm implementation in SQL SCRIPT language are needed to take into account normalisation specifics of different formats and select normalised users, groups, workstations and timestamps for processing.

However, there is still a room for improvement of offered anomaly detection algorithm. First of all, our current implementation ignores time-wise fluctuations of user behaviour during the day, week, month or year periods. Indeed, user activity in different datasets could be bounded to working days or weekend time, as well as local timezones, vacation plans and so on. Identifying and taking these fluctuations into account will allow us to significantly improve precision of anomaly detection on such datasets.

Second, the current list of features for anomaly detection only includes user name, group and workstation. We plan to extend it to be able to model access to specific data or other workstation resources, which in turn will improve precision of anomaly detection.

Finally, we need to provide a speed optimisations for real-time anomaly detection. The current implementation works in SAP HANA, which allows to rapidly process high amount of events. However, the offered method for automatic threshold detection calculates number of anomalies in whole dataset for each of 20 threshold values, which significantly slows down the detection speed. In our future work we will research this question and try to implement threshold detection based on data samples rather than whole dataset.

All these measures – together with other REAMS features – will allow us to achieve significant progress in modelling of user behaviour as well as provide a complete and efficient prototype of SIEM system.

6 Conclusion

We have developed an effective approach to detect malicious user activity even when such activity does not violate any access or data protection policies. Our results show that our algorithm is able to detect all existing malicious events without producing any false positive alarms. Such precision was possible due to the performed two-step probability check. By checking the Poisson probability first for each {user_group,workstation} pair, and later – for {user,workstation} pairs, we are able to correctly classify legitimate cases when the resource accessed by the group of users was not accessed by one of the users from that group for a long time period.

Further, we offer an automatic selection of an optimal threshold value. This feature is especially relevant in context of SIEM system [35]. Due to this option, the future integration of our anomaly detection algorithm into our own prototype of SIEM system (REAMS) will not require a user of such system to input the parameters for anomaly detection. This ability helps to improve the user experience for system operators, which becomes extremely important for sophisticated and often overloaded user interfaces of security platforms.

Our concepts were checked using the proof-of-concept implementation, that utilizes simulated dataset. The simulation tool emulates user activity to produce real events on the domain controller within our testbed. The use of a simulated dataset does not only allow to precisely estimate efficiency of our algorithm, but also to emulate normal user behaviour in cases, when cleaning up the training dataset could be considered too complex. Even though for cases, when real training dataset contains undetected malicious activity, so that our algorithm is unable to detect them later in the data, all new issues of malicious user behaviour, that were not captured into training dataset, will be detected.

We hope that the future integration of our Poisson-based anomaly detection approach into a SIEM system prototype like REAMS will significantly improve its ability to detect cases of malicious user behaviour and achieve higher level of security in the monitored network.

References

1. Nanda, S., Cker Chiueh, T.: Execution trace-driven automated attack signature generation. In: Proceedings - Annual Computer Security Applications Conference, ACSAC, pp. 195–204 (2008)
2. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M.: A survey of intrusion detection techniques in cloud. J. Netw. Comput. Appl. **36**(1), 42–57 (2013)
3. Patel, A., Qassim, Q., Wills, C.: A survey of intrusion detection and prevention systems. Inf. Manag. Comput. Secur. **18**(4), 277–290 (2010)
4. Maciá-Fernández, G., Vázquez, E., Garcia-Teodoro, P.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(12), 18–28 (2009)
5. Scarfone, K., Mell, P.: Guide to intrusion detection and prevention systems (IDPS) (2007)
6. Ihler, A., Hutchins, J., Smyth, P.: Adaptive event detection with time-varying poisson processes. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2006, p. 207. ACM Press, New York (2006)
7. Wu, S.X., Banzhaf, W.: The use of computational intelligence in intrusion detection systems: a review. Appl. Soft Comput. **10**(1), 1–35 (2010)
8. Berthier, R., Rhee, W., Bailey, M., Pal, P., Jahanian, F., Sanders, WH: Safeguarding academic accounts and resources with the University credential abuse auditing system. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), pp. 1–8, IEEE, June 2012
9. Chapple, M.J., Chawla, N., Striegel, A.: Authentication anomaly detection: a case study on a virtual private network. In: Proceedings of the 3rd Annual ACM Workshop on Mining Network Data, pp. 0–5 (2007)
10. Oh, S.H., Lee, W.S.: An anomaly intrusion detection method by clustering normal user behavior. Comput. Secur. **22**(7), 596–612 (2003)
11. Liu, S., Kuhn, R.: Data loss prevention. IT Prof. **12**(2), 10–13 (2010)

12. Shabtai, A., Elovici, Y., Rokach, L.: A survey of data leakage detection and prevention solutions (2012)
13. Viswanath, B., Ahmad Bashir, M., Crovella, M., Guha, S., Gummadi, K.P., Krishnamurthy, B., Mislove, A.: Towards detecting anomalous user behavior in online social networks. In: Proceedings of the 23rd USENIX Security Symposium (USENIX Security)
14. Ringberg, H., Soule, A., Rexford, J., Diot, C.: Sensitivity of PCA for traffic anomaly detection. In: Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems - SIGMETRICS 2007, p. 109 (2007)
15. Salem, M.B., Stolfo, S.J.: Modeling user search behavior for masquerade detection. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 181–200. Springer, Heidelberg (2011)
16. Mukkamala, S., Janoski, G., Sung, A.: Intrusion detection: support vector machines and neural networks. In: Proceedings of the IEEE International Joint Conference on Neural Networks (ANNIE), pp. 1702–1707 (2002)
17. Chen, W.-H., Hsu, S.-H., Shen, H.-P.: Application of SVM and ANN for intrusion detection. Comput. Oper. Res. **32**(10), 2617–2634 (2005)
18. Koc, L., Mazzuchi, T.A., Sarkani, S.: A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. Expert Syst. Appl. **39**(18), 13492–13500 (2012)
19. Muda, Z., Yassin, W., Sulaiman, M.N., Udzir, N.I.: A K-Means and naive bayes learning approach for better intrusion detection. Inf. Tech. J. **10**(3), 648–655 (2011)
20. Ye, N., Zhang, Y., Borror, C.M.: Robustness of the markov-chain model for cyber-attack detection. IEEE Trans. Reliab. **53**(1), 116–123 (2004)
21. Khanna, R., Liu, H.: System approach to intrusion detection using hidden markov model. In: Proceeding of the 2006 International Conference on Communications and Mobile Computing - IWCMC 2006, p. 349. ACM Press, New York (2006)
22. Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J.: Modeling intrusion detection system using hybrid intelligent systems. J. Netw. Comput. Appl. **30**(1), 114–132 (2007)
23. Chen, Y., Li, Y., Cheng, X., Guo, L.: Survey and taxonomy of feature selection algorithms in intrusion detection system. In: Lipmaa, H., Yung, M., Lin, D. (eds.) Inscrypt 2006. LNCS, vol. 4318, pp. 153–167. Springer, Heidelberg (2006)
24. Klein, R.W., Roberts, S.D.: A time-varying poisson arrival process generator. Simulation **43**(4), 193–195 (1984)
25. Yu, H., Zheng, D., Zhao, B.Y., Zheng, W.: Understanding user behavior in large-scale video-on-demand systems (2006)
26. Remote desktop protocol. http://msdn.microsoft.com/en-us/library/aa383015.aspx
27. Virtual network computing. http://www.hep.phy.cam.ac.uk/vnc_docs/index.html
28. Python imaging library. http://www.pythonware.com/products/pil/
29. Chandrasekaran, B.: Survey of network traffic models. Waschington University in St. Louis CSE, pp. 1–8 (2009)
30. Roschke, S., Cheng, F., Meinel, C.: An advanced IDS management architecture. J. Inf. Assur. Secur. **5**, 246–255 (2010)
31. Real-time event analysis and monitoring system. https://hpi.de/en/meinel/security-tech/network-security/security-analytics/reams.html
32. SAP HANA. http://www.saphana.com

33. Sapegin, A., Jaeger, D., Azodi, A., Gawron, M., Cheng, F., Meinel, C.: Hierarchical object log format for normalisation of security events. In: 2013 9th International Conference on Information Assurance and Security (IAS), IAS 2013, pp. 25–30, IEEE, December 2013
34. Sapegin, A., Jaeger, D., Azodi, A., Gawron, M., Cheng, F., Meinel, C.: Normalisation of log messages for intrusion detection. J. Inf. Assur. Secur. **9**(3), 167–176 (2014)
35. Ali, M.Q., Al-Shaer, E., Khan, H., Khayam, S.A.: Automated anomaly detector adaptation using adaptive threshold tuning. ACM Trans. Inf. Syst. Secur. (TISSEC) **15**(4), 1–30 (2013)

Analysis of Privacy and Security Exposure in Mobile Dating Applications

Constantinos Patsakis[1](✉), Athanasios Zigomitros[1], and Agusti Solanas[2]

[1] Department of Informatics, University of Piraeus, Piraeus, Greece
kpatsak@gmail.com,azigomit@unipi.gr

[2] Smart Health Research Group, Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia, Spain
agusti.solanas@urv.cat

Abstract. Millions of people around the globe try to find their other half using Information and Communication Technologies. Although this goal could be partially sought in social networks, specialized applications have been developed for this very purpose. Dating applications and more precisely *mobile* dating applications are experiencing a continuous growth in the number of registered users worldwide. Thanks to the GPS and other sensors embedded in off-the-shelves mobile devices, dating mobile apps can provide location aware content, not only about the surroundings, but also about nearby users. Even if these applications have millions of registered users, it can hardly be said that they are using the best standards of security and privacy protection.

In this work we study some of the major dating applications and we report some of the risks to which their users are exposed to. Our findings indicate that a malicious user could easily obtain significant amounts of fine-grained personal information about users.

Keywords: User profiling · Location privacy · Online social networks · Security and privacy exposure

1 Introduction

The wide adoption of ICT has drastically modified the way we exchange information and interact with other people. Nowadays, our capacity to communicate with others is no longer bounded to our immediate surroundings. Instead, we can easily interact with people from distant places in almost real time.

Mobile technology has become commonplace and the possibility to easily reach a huge market has fostered the development of numerous and diverse applications. In addition, smartphones and tablets are steadily improving their computing capabilities and come equipped with accelerometers, GPS and a series of other sensors that enable developers to deploy more sophisticated solutions to exploit spatial information. As a result of these new means of interaction, users are more engaged and developers can make them actively participate and provide added-value content and information.

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 151–162, 2015.
DOI: 10.1007/978-3-319-25744-0_13

A good example of these communication and behavioural changes can be found in mobile dating applications. The way in which people look for partners has drastically changed as a result of the use of mobile technology. However, using technology does not come without risks. Due to their very nature, dating applications contain sensitive information. Note that in addition to the sexual orientation and preferences of users, modern dating apps are context-aware and allow the finding of nearby partners. Moreover, user profiles contain other sensitive data such as political views or beliefs.

Certainly, users would like to share some personal information with potential partners as these could create criteria to filter candidates. Notwithstanding, users might not want this shared information to be publicly available. Additionally, the information exchanged between two users might be private and very sensitive. Thus, information leaks could have a huge impact on the reputation of individuals and should be avoided.

It might be thought that these privacy and security issues are well-known and defined. Hence, one would expect that developers put in place the right measures to secure their applications and deter any information leak, especially since these applications are used by millions of people.

In this article we study 18 mobile dating and chatting apps and highlight doubtful software development practices, which we have found to be commonplace. The detected vulnerabilities are, in many cases, quite obvious so are their solutions. The vulnerabilities that we discuss might have a very big impact on the users in a variety of ways and might affect millions. More importantly, we have observed that those vulnerabilities can be exploited very easily and require little, if any, computer skills. Actually, there is no need for reverse engineering of the applications. In our experiments, the most *sophisticated* scenario implies to intercept network packages *e.g.* by using a proxy, while in the simplest scenario, the attacker just needs to eavesdrop exchanged messages.

2 Related Work on S&P in Apps

The use of ICT has helped us to communicate and exchange information more easily, specially thanks to the Internet. However, it has opened the door to remote attacks that might affect millions. With the aim to fight against these attacks many organizations strive to raise awareness about secure web development. For instance, OWASP[1] compiles the well-known OWASP Top Ten survey, which highlights the most critical web application security flaws. This survey provides a very good insight on what developers should be aware of when developing applications and it illustrates how attackers would try to penetrate into a web service. Other surveys such as [1] provide similar results.

Mobile applications could be developed as stand-alone services running in a mobile device. However, in most cases, they do not rely on local resources only but use web content and infrastructure. Very frequently, most of the mobile apps content is retrieved, uploaded, and updated through the Internet, and the data

[1] https://www.owasp.org.

gathered by means of the embedded sensors are used to provide value-added services and functionalities.

Currently, the most used operating systems for smart phones are Android and iOS. Android provides a permission-based security model, however, it has been shown that it has vulnerabilities [2,3]. Beresford et al. [4] created *MockDroid*, a modified version of Android which allows users to intervene and revoke the access rights of applications to particular resources at run-time. Definitely, this approach has a very big impact on the usability of the application because it limits some functions. Notwithstanding, it allows users to define their own privacy policies and reduce their possible information leakage. A similar approach was followed with *TISSA* [5], which allows users to fine-tune their privacy policies and the access level of specific applications at run-time. Enck et al. approached the problem from a different perspective and introduced *TaintDroid* [6]. Instead of constantly intervening, their Android mod tracks down which sensitive data is requested by an app and when it is used, hence, providing real-time analytics. Their analysis can be very useful in the automated characterization of benign, grayware and malware [7]. A more user friendly approach has currently been embedded in *CyanogenMod*[2], one of the most well-known customized distributions of Android, allowing users to block access to apps from specific resources.

In the iOS arena, where publishing an app in the official app store passes some stricter filters, Egele et al. [8] made a comprehensive study over a sample of 1,400 iOS apps (3/5 from the iTunes App Store and 2/5 from the Cydia repository). While most apps were found to be benign, their analysis revealed that more than half of them was leaking the unique ID of the device. To provide users with a clearer overview about how their information flows in their mobile devices, Wetherall et al. [9] created two tools: a browser plugin and a mobile app. These tools alert users of dangerous logins (a.k.a. logins performed without encryption) and inform them about which private information is collected.

Burattin et al. [10] illustrated that while users think that some of their information is hidden, e.g. list of friends, it can be recovered from the OSNs with various methods. Focusing on dating apps, Qin et al. [11] show how an attacker could obtain the real location of users in several well-known dating apps. It was showed that the main reason for this exposure is the poor randomization and obfuscation techniques used.

Recently, the Electronic Frontier Foundation published a review[3] to determine what security is offered by secure messaging products. The review clearly indicates that there are many problems in the architecture of most applications and the user cannot be considered secure as for instance she cannot verify the contact's identity, or the service provider has access to the users' traffic.

3 Experimental Setup

In our experiments we have approached the problem of obtaining information from a non-invasive perspective. Given the legal constrains to apply reverse

[2] www.cyanogenmod.org.

[3] https://www.eff.org/secure-messaging-scorecard.

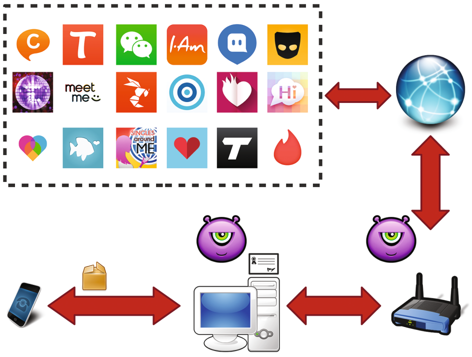


Fig. 1. Experimental setup

engineering on an application,s and considering the capacities of an average user or a network administrator, we have installed a proxy that intercepted the messages targeted towards our apps. To ensure that the content of the intercepted packages can be analysed by the proxy, we generated a root certificate for it and installed it in the smartphone. Therefore, even if the packages were encrypted, their content could be read by the proxy. In this sense, the setup is analogous to a *man-in-the-middle attack*, with the only, but important, difference that we are controlling all sides, both the benign and the malicious.

The above setup mimics two real-world attack scenarios: In the first scenario, a network administrator might try to gather as much information as possible about the users in his network. Note that, as it is shown in our findings, there is no need for the network administrator to request his users to install a certificate in order to attack them, as many applications are using unencrypted traffic or leave important information in the header so the administrator only needs to watch regular network traffic. In principle, beyond the malicious network administrator, anyone who can perform traffic sniffing can execute these attacks.

The second scenario involves a malicious user with cyber-stalking intentions. The attacker wants to find probable victims and their whereabouts. To this end, he/she intercepts the packages that are sent and received from the applications and he uses them to extract further fine-grained information. Figure 1 illustrates our setup and where we expect our adversary to be logically located.

To capture the data from the studied apps we have used Fiddler 2[4] as the debugging proxy and we have installed its certificate in a smartphone. All the experiments were conducted using an iPhone 4 with iOS 7.0.4. However, as it is going to be discussed, the operating system does not have any relevance to the vulnerabilities that we have found. The vulnerabilities that we discuss in this work are mainly due to the exchanged traffic between the apps and web servers that host the services and, in fact, the apps *per se* have not been analysed.

[4] http://www.telerik.com/fiddler.

4 The Findings

We have analysed 18 mobile dating and chatting applications to study whether they send sensitive HTTP traffic, include the current location of users, send the actual distance to other users, use static links, etc. An overview of our findings is depicted in Fig. 2 and in what follows we briefly describe the specific details/vulnerabilities of each studied application.

ChatOn: ChatOn uses HTTPS for all its traffic. However, many details could be leaked through the URL of the API. For instance, an eavesdropper can easily find many information about the user's phone, namely, model, operating system version, IMEI, IMSI, telephone number, user ID and app version. The app sends the telephone numbers of all user's contacts to Samsung, and the received packets contain additional information like contacts' birthday. The RESTful API that is used exposes users actions and the profiles that they visit.

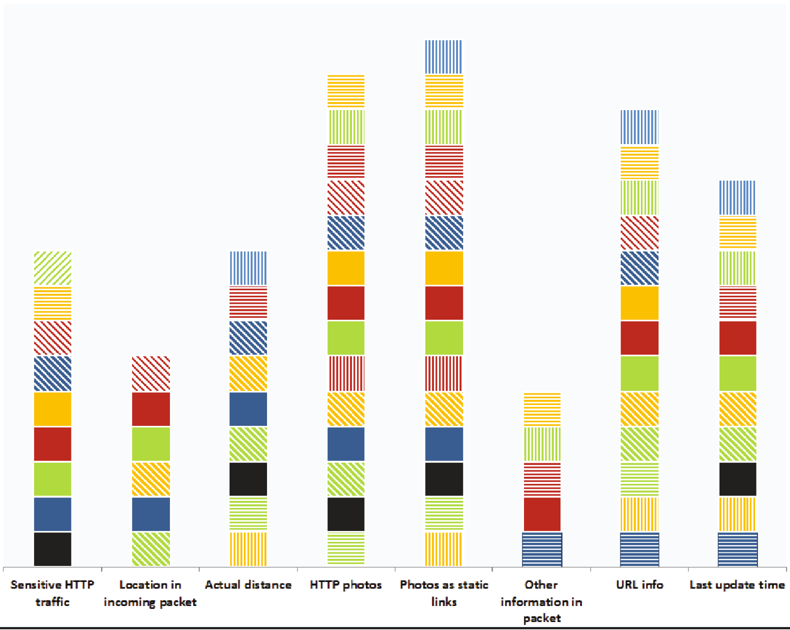
Grindr: Grindr uses HTTPS for most of its communications, but photographs are sent by using static links over HTTP. The API that is called from the mobile app might allow eavesdroppers to extract the actual user location and his/her application ID from the sniffed URL. Additionally, the URL discloses the user's activity and his/her device OS. Moreover, exchanged packets contain the distance only for users that consented and the application might display the relative user distance. However, the messages contain the actual users' location.

Hornet: Hornet encrypts its traffic using HTTPS, but sends the distance with 10 m accuracy. Photos are static links sent over HTTP. The API calls allow an adversary to deduce user activity, e.g. chatting, browsing profiles, etc. simply by capturing the URLs that users request.

I-Am: Apart from the authentication which is executed over HTTPS, the rest of the traffic of I-Am goes over HTTP, hence, allowing an adversary to have full access to user private data. Images are sent over HTTP and as static links. Regarding spatial data, the actual user location is sent in the URL without any encryption or obfuscation, and the exact distance to other users is sent in the packet along with their birthday.

LOVOO: All traffic of LOVOO is sent over HTTPS, with the exception of photos, which are sent over HTTP. It is interesting to notice that links are dynamic and expire. The actual distance between users is sent with a rounding of 100 m, along with their relative (x, y) coordinates. Thus, an adversary could recover the actual locations. Also, the API calls expose in the URLs that are requested the user location, his/her preferences and his/her overall activity.

MeetMe: MeetMe uses mixed HTTP/HTTPS traffic. The user location and his/her preferences are visible in the URL. The actual location of the user is included in the packet if other users are nearby, otherwise their distance is given in Km. Photos are shared over HTTP, and user commands can be seen in the URL.



Application	Version	Installations	Code	Application	Version	Installations	Code
ChatOn	3.0.2	100m-500m	▤	Singles around me	3.3.1	500K-1m	🟥
Grindr	2.0.24	5m-10m	▥	SKOUT	4.4.2	10m-50m	▧
Hornet	2.0.14	1m-5m	▤	Tagged	7.3.0	10m-50m	🟨
I-Am	3.2	500K-1m	■	Tango	5.8	100m-500m	▧
LOVOO	2.5.6	10m-50m	▧	Tinder	4.0.9	10m-50m	▤
MeetMe	9.2.0	10m-50m	🟦	Tingle	1.12	-	▥
MoMo	5.4	1m-5m	▧	Waplog	3.0.3	5m-10m	▤
POF	2.40	10m-50m	▥	WeChat	6.0.1	100m-500m	▨
SayHi	3.6	10m-50m	🟩	Zoosk	8.6.21	10m-50m	▥

Fig. 2. Discovered vulnerabilities per application. Since the App Store is not reporting the downloads, the numbers are from Google Play.

MoMo: While MoMo uses HTTPS to exchange messages with the server, it does not hide users' location. More precisely, the packets that are received from the app contain fine-grained distance information from other users. In addition, URLs contain the visited profiles as well as the current user ID and photos are sent over HTTP by using static links.

Plenty of Fish: It uses HTTP but all messages are encrypted, which most likely means that the app contains a local key to decrypt the contents. On the bad side, photos are sent over HTTP as static links.

SayHi: It uses Facebook for its authentication and then sends everything in clear text. Packets include the fine-grained location of other users and their activity can be seen in the requested URLs. An eavesdropper could also intercept user conversations. Photos are sent over HTTP using static links.

Singles Around me: It sends the exact location of other users in the packet. Photos are sent over HTTP with some of them being dynamic links and others being static. However, the received packet contains an additional field: users' emails. In principle, a user needs to ask for the permission of other users to access their email, but this is always sent in the packet. This app significantly exposes users because URLs contain the IDs that a user has been watching. Hence, revealing his/her preferences and activity.

SKOUT: SKOUT uses HTTPS only for its authentication but the rest of the traffic is sent over HTTP. It sends the exact distance to other users in the packets and then obfuscates it in the frontend of the app. The API of SKOUT exposes the user activity because it shows whether the user is typing a message, visiting a profile, etc. Since traffic is sent over HTTP, chat messages are unencrypted.

Tagged: All traffic is sent over HTTP, so all messages can be intercepted and the API exposes user's activity and preferences. Photos are sent over HTTP as static links.

Tango: Tango transmits over HTTP and all messages can be intercepted. The API exposes user's activity as well as his/her phone number and preferences. Photos are sent over HTTP as static links and the messages contain the real location of other users.

Tinder: Tinder uses HTTPS traffic to deter eavesdroppers but the messages contain the Facebook ID of the users. Hence, an adversary can access even more data. Packets do not contain the actual location but the distance to other users, which can be further tuned to recover their actual locations. Photos are sent over HTTP as static links.

Tingle: Tingle does not use location data of other users in the received packets. However, messages contain other important information. Like Singles Around Me, it includes other users' emails and, additionally, it has a device tag indicating, for example, that the user has switched the device. Moreover, Tingle displays the actual location of the user in the URL, allowing an eavesdropper to identify him/her. The URL used by the API contains users' queries, hence, exposing their preferences. Finally, photos are sent over HTTP as static links.

Waplog: Waplog transmits over HTTP. While it does not send the location, it exposes emails of other users. Photos are sent over HTTP as static links. In addition, the API exposes information about the user's device in the URL, the session key and the user's hashed password.

WeChat: WeChat uses HTTP for all its traffic and sends all information in an encrypted file. In our experiments, we didn't notice any handshake between the application and the API to generate a cryptographic key. Therefore, we may safely deduce that the application is installed with a static hard-coded key, same for each user that can be derived by reverse engineering the application. Therefore, one could use the key to fully manipulate the data of all users.

Zoosk: Zoosk uses HTTPS for its traffic. The requested URLs expose the phone model and its OS as well as the user activity. Finally, photos are sent as static links over HTTPS.

5 Discussion

After analysing the above mobile dating and chatting applications we have observed several trends that are next discussed.

Users' Locations: Many of the studied apps hand over the locations of other users so as to display them and show whether users are nearby. A typical example is illustrated in Table 1 where "Singles Around Me" sends a JSON file containing the exact GPS location of a user. This approach is susceptible to many attacks. It seems that a better solution would be to use distances instead of real locations. However, as shown by Qin et al. in [11], even distances can be used, via trilateration, to disclose the location of users. A better approach towards protecting users' locations, would be to use *Private Proximity Testing* protocols such as [12]. These protocols disclose a single bit of information, *i.e.* whether two users are nearby or not. By using this approach the desired functionality is provided whilst, at the same time, users' exposure is drastically reduced.

Unencrypted Transmission Channels: In [13] it is shown that many mobile apps use SSL/TLS code, which is potentially vulnerable to man-in-the-middle attacks. Notwithstanding, these apps were, at least, trying to protect sensitive user data. On the contrary, our analysis shows that major apps transmit private data over HTTP instead of HTTPS. It could be said that using HTTPS might imply an overhead in terms of computation and bandwidth. However, in this kind of apps, where very sensitive information is managed, it is difficult to support the use of HTTP.

Multimedia: In general, our study shows that the handling of multimedia content is very poor. In [14], Patsakis et al. highlight the privacy and security risks related to the sharing of multimedia content in Online Social Networks. Note that the case of mobile dating apps could be considered even more sensitive and might imply further dangers. First, it is apparent that using HTTP (instead of HTTPS) allows an attacker to intercept and change the content of the received messages. Second, the use of static links can be considered a very important security vulnerability because an eavesdropper could easily find the images of the

Table 1. Example of a JSON packet from “Singles Around Me”.

```
{
 "username":"s---------eam",
 "email":"d----------96@yahoo.com",
 "gender":2,
 "interestedIn":1,
 "country":"United Kingdom",
 "region":"London, City of",
 "city":"",
 "gps":[38.------------2,23.8-------------5],
 "age":39,
 "photo":"http://www.singlesaroundme.com/images/cache/1/
    photoface_11-----_--5_--5.jpg",
 "photos":[],
 "birthYear":1974,
 "birthMonth":--,
 "birthDay":-,
 "lastOnline":"2014-10-06 03:28:07 PM",
 "profileAnswers":{"1":"5’ 7" - 170cm",
 "3":"prefer not to say",
 "21":"Married","30":"straight","25":"brown","31":"blonde","2
    6":"white","28":"none",
 "29":"Sagittarius","38":["dating","serious relationship","
    friendship"],
 "37":"Greek","36":["English"],"32":"socially / occasionally"
    ,
 "34":"socially/occasionally","35":"quite fit","40":"
    Christian - Other",
 "41":"University graduate","42":"yes living with me","43":"
    yes"},
 "privacySettings":{"gps":0,"profile":0}
}
```

profiles that a specific user has visited. These images would allow the adversary to identify the sexual orientation and preferences of users. Depending on the openness of the society, giving away the precise sexual orientation and preferences of a user can lead to social discrimination or even legal actions.

Hidden Information and URL Parameters: Thanks to our analysis, we have noticed that in several apps, the data packets sent to users contain hidden information about other users. For instance, they contain the email of others, even when they have not consented. Table 1 illustrates this for “Singles Around Me”. In the case of Tingle, the analysed data packets contained a device identification field that would allow an attacker to know when users change or switch their devices. Also, we have observed a doubtful development practice, this is, the addition of sensitive parameters in the URLs of the API. These parameters allow an attacker to obtain lots of information. For instance, users’ activities (*e.g.*

Table 2. User exposure from the URL. For obvious reasons, the sensitive information has been suppressed.

Application	**URL**
ChatOn	`https://gld1.samsungchaton.com/prov4?imei=----- \&imsi=-----\&model=iPhone4\&clientversion=3.0.2\ &platform=iPhone\%20OS\&osversion=7.0.4`
	`https://prov4?imei=------\&countrycallingcode= 30\&phonenumber=-----\&imsi=----\&model=iPhone4\ &clientversion=3.0.2\&platform=iPhone\%20OS\ &osversion=7.0.4`
Grindr	`https://primus.grindr.com/2.0/broadcastMessages? applicationVersion=2.0.24\&hasXtra=0\&lat=53.----- \&lon=6.2----\&platformName=iOS\&platformVersion=7.0. 4\&profileId=36850131`
MoMo	`https://api.immomo.com/api/profile/1121----?fr=98----`
SKOUT	`http://i22.skout.com/services/ServerService/ GCUserTyping`

browsing of profiles, chatting, etc.), the telephone number or even the location of the user could be monitored simply by eavesdropping the communications. Some examples of the information that can be leaked through the URL are illustrated in Table 2. While in all cases the traffic is encrypted, one can see that e.g. ChatOn broadcasts the IMEI, IMSI and user's phone number in the URL along with some details about the phone, Grindr broadcasts the user's location his ID and some data about the device. Similarly, MoMo leaks which profiles a user is visiting and SKOUT shows what the user is doing, in this case typing.

6 Conclusions

The steady growth in the number of users of dating services might get the attention of cybercriminals, who try to obtain personal information that could be used for user profiling, blackmailing, defamation, and even identity theft.

Web-based dating services are pretty well-established and well-known programming practices are put in place (in most cases). However, we have realised that in dating services provided by mobile platforms the situation turns out to be quite different. Mobile applications have the ability to collect very sensitive information (*e.g.* users current location, sexual orientation and preferences) and one might expect that their security measures are even more robust than those implemented on their web-based counterparts. However, our analyses have shown that there are significant security holes that could be easily used even by inexperienced attackers to obtain very sensitive information.

After analysing a significant number of diverse mobile dating applications, we have concluded that most of them are vulnerable to simple sniffing attacks, which could reveal very sensitive personal information such as sexual orientation, preferences, e-mails, degree of interaction between users, etc.

We interpret the results of our study in two main ways: First, we believe that disclosing these vulnerabilities might foster the interest of the society in protecting its privacy, and will raise an alarm for those companies that provide insecure services. Second, we believe that most of the detected vulnerabilities have very simple solutions that do not require much effort. Thus, by disclosing these problems we open the door to a reframing of the programming practices in mobile dating applications.

Acknowledgments. This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the *OPERANDO* project (Grant Agreement no. 653704) and is based upon work from COST Action *CRYPTACUS*, supported by COST (European Cooperation in Science and Technology).

Dr. Solanas is partly funded by La Caixa Foundation through project "SIMPATIC" RECERCAIXA'12, by the Government of Catalonia under grant 2014 SGR 537, and by the Spanish Ministry of Economy and Competitiveness under project "Co-Privacy", TIN2011-27076-C03-01.

References

1. Cenzic, Application vulnerability trends report, Technical report (2014). http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf
2. Grace, M.C., Zhou, Y., Wang, Z., Jiang, X.: Systematic detection of capability leaks in stock android smartphones. In: 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, 5–8 February 2012
3. Au, K.W.Y., Zhou, Y.F., Huang, Z., Lie, D.: Pscout: analyzing the android permission specification. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 217–228. ACM (2012)
4. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: Mockdroid: trading privacy for application functionality on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications, pp. 49–54. ACM (2011)
5. Zhou, Y., Zhang, X., Jiang, X., Freeh, V.W.: Taming information-stealing smartphone applications (on Android). In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 93–107. Springer, Heidelberg (2011)
6. Enck, W., Gilbert, P., Chun, B.-G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: Taintdroid: an information flow tracking system for real-time privacy monitoring on smartphones. Commun. ACM **57**(3), 99–106 (2014)
7. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A study of android application security. In: Proceedings of the 20th USENIX Conference on Security, SEC 2011, p. 21. USENIX Association, Berkeley (2011). http://dl.acm.org/citation.cfm?id=2028067.2028088
8. Egele, M., Kruegel, C., Kirda, E., Vigna, G.: Pios: detecting privacy leaks in iOS applications. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011. The Internet Society, San Diego, 6th–9th February 2011
9. Wetherall, D., Choffnes, D., Greenstein, B., Han, S., Hornyack, P., Jung, J., Schechter, S., Wang, X.: Privacy revelations for web and mobile apps, p. 21 (2011)
10. Burattin, A., Cascavilla, G., Conti, M.: Socialspy: browsing (supposedly) hidden information in online social networks, CoRR abs/1406.3216. http://arxiv.org/abs/1406.3216

11. Qin, G., Patsakis, C., Bouroche, M.: Playing hide and seek with mobile dating applications. In: Cuppens-Boulahia, N., Cuppens, F., Jajodia, S., Abou El Kalam, A., Sans, T. (eds.) SEC 2014. IFIP AICT, vol. 428, pp. 185–196. Springer, Heidelberg (2014)
12. Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., Boneh, D.: Location privacy via private proximity testing. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2011. The Internet Society, San Diego, 6th–9th February 2011
13. Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., Smith, M.: Why eve and mallory love android: an analysis of android SSL (in)security. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 50–61. ACM, New York (2012)
14. Patsakis, C., Zigomitros, A., Papageorgiou, A., Solanas, A.: Privacy and security for multimedia content shared on OSNs: issues and countermeasures. Comput. J. **58**, 518–535 (2015). doi:10.1093/comjnl/bxu066

CBUPRE to Secure MANETs from DoS Attacks

Radha Krishna Reddy Pallavali[1], Samia Bouzefrane[2](✉), and Selma Boumerdassi[3]

[1] Computer Science and Engineering, Lisbon, Portugal
Pallavali@gmail.com
[2] CEDRIC Labs, CNAM, Paris, France
Samia.Bouzefrane@cnam.fr
[3] INRIA, Paris, France
Selma.Boumerdassi@inria.fr

Abstract. Nowadays, mobile devices are an increasingly useful part of our daily life. Networks through which mobile devices communicate are named Mobile Ad-hoc Networks (MANETs). Without using a fixed infrastructure, mobile nodes dynamically build a wireless network for communication. In order to ensure reliable communications, a main security issue in MANETs is protection from Denial of Service (DoS) Attacks. From past decade, only a few proposals have been made to secure MANETs from DoS Attacks. Generally, the normal encryption schemes protect data confidentiality from unauthorized users, but these techniques are limited to encrypted data sharing between mobile nodes. To overcome this issue, in this paper we propose to use a new proxy re-encryption technique called Compression-Based Unidirectional Proxy Re-Encryption (CBUPRE) for secure data exchange and efficient bandwidth use even during DoS attacks. This technique uses a new node called Dynamic TCP Proxy Node to check the authorization of nodes, to encrypt at a proxy level, and to secure the network against unintended nodes. Based on this, secure data exchange can be achieved with encrypted data. We present a correctness proof of how, through the CUPRE technique, the data can be transferred to the destination node in the group within the low bandwidth network securely even during DoS attacks.

1 Introduction

As stated in [1], MANETs are assemblies of mobile nodes with the capability of configuring any momentary network without the assistance of any established infrastructure or centralized supervision. MANETs are not permanent in nature and are self-organizing because they use multi-hop communication mechanisms. In these mechanisms, each mobile node itself acts as a router to help other nodes. The features of MANETs, such as dynamic topology, mobility, and resource constraints make MANETs suitable to operate under disaster conditions, in the battlefield, news feeds, video-audio transmission, multiparty video games, etc. As described in [2], the nodes in MANETs create a small group network and stay connected to perform particular tasks. In this network, any node may join and leave at any moment in time, resulting in

© Springer International Publishing Switzerland 2015
S. Boumerdassi et al. (Eds.): MSPN 2015, LNCS 9395, pp. 163–175, 2015.
DOI: 10.1007/978-3-319-25744-0_14

dynamic changes in the network. Due to being dynamic in nature, these networks are vulnerable to denial of service (DoS) attacks, eavesdropping of communication channels and e-commerce transactions modifications [3]. In MANET, two nodes can communicate directly with each other as long as they are in radio communication range. In sensitive environments (e.g. military), data exchange must be secured between the communicating parties through intermediate nodes.

In general, DoS attacks are the most critical attacks in computing, relatively easy to implement and extremely difficult to detect and to prevent [4]. The DoS attack has several forms: HTML DoS attack, XML DoS attack or distributed DoS attack (DDoS attack). It consumes an entire network's resources or machine services, making resources unavailable to its respective users [5]. This kind of attack can affect any layer of an open systems interconnection model. In MANETs, the DoS attacks can be performed on MAC layer by several methods [6]. Because MANETs are resource-constraint based networks, the secure group communication must be efficient and inexpensive.

Radio signals can be jammed by wireless signal jamming, disruption of existing communication paths, and saturation of a normal node with a large number of fake communication requests by an illegitimate user. It is even easier for malicious insiders to successfully achieve DoS attacks that consume the constrained energy of the mobile nodes by flooding the network with garbage packets [7]. To achieve a MANETs secure group communication, a group key needs to be established among multiple nodes to ensure a secure exchange of secret information within the group [4]. Due to the distributed nature of the users and the systems, there is a growth of data exchange on existing network connections [8]. Due to slow networks, this results in poor performance. To overcome this problem, organizations moved to data compression techniques [9–12]. To convert original data into alternative formats with lower size, data compression techniques are used.

The major compression technique of fixed length code with n-grams uses 128 most frequently occurring n-grams. This algorithm generates 266 symbols based on frequent key sets used. Where 1 to 87 is for blank, upper and lower spaces, digits and special characters, 88 to 255 for n-grams compression = (Master + Combining). Here Master with 8 bits for A, E, I, O, N, T, U, blank and Combining with 21 bits for blank, plus everything except J, K, Q, X, Y, Z. It takes the original plain text as input and gives the compressed text as an output. A fixed length window 'n' is used to scan each word of the data; all possible n-grams are extracted, stored and analyzed. In n-grams we assumed that each word started with a letter and ended with space. With the below formula we can calculate how many n-grams are possible in a given data dictionary:

$$\mathrm{N}\ (\mathrm{n}) = \sum\nolimits_{w-n}^{m} (w - n + c) f(w)$$

Where *N(n)* be the number of n-grams in the text, m be the maximum word length, *f(w)* be the frequency of words with length *w*, *c* = *2* with spaces.

Finally, the unidirectional proxy re-encryption (UPRE) technique is used to authenticate user nodes in the group and to exchange data and secret keys as well [13]. The UPRE scheme that allows a secret key holder node to create re-encryption key, and the proxy can use this key to translate a message that already encrypted by the sender node. This can be done without providing the authority on encrypted data to the proxy node. In detail, the proxy node can neither recover the sender node's secret key nor decrypt the sender's ciphertext. However, UPRE schemes are under digital rights management (DRM) [14], distributed file storage systems [23], law enforcement [16], etc., In this case, there is no question of compromise at the proxy level re-encryption. Note, we can observe this type of compromise in Apple iTunes DRM [23]. The following are the few properties of UPRE, makes data transformation is more secure [17]:

Unidirectional: the encryption process is allowed from Alice to Bob only, but not from Bob to Alice.

Interaction prevention: for re-encryption key generation, the secret key of Alice and public key of Bob are only required, means that Bob's secret key is not required.

Authentic access: the encrypted ciphertext can decrypt by Alice because she is the authentic user.

Proxy invisibility: Actually the proxy is transparent, but we allowed the senders encrypted message, that can only be opened by the intended recipient (receiver).

Collusion-resistance: Alice and proxy both are working together, but they doesn't know anything about Bob's secret key.

In this paper we propose to use a combination algorithm for MANETs to secure data sharing and, for efficient network bandwidth usage, to combine a compression technique with UPRE. Even during an attack to a network, data already transmitted to that network must reach its destination node securely. We expect this to be achieved through data compression techniques and effective use of available bandwidth even during network attacks.

2 Related Work

Due to their inherent mobility patterns and dynamic topology, MANETs are susceptible to DoS attacks [18]. DoS attacks on MANETs have received significantly less attention than DoS attacks on wired networks. Many solutions have been proposed to protect networks from DoS attacks, some of which have also revealed solutions for ad-hoc networks problems.

In [7], the authors describe the design of a capability-based security mechanism (CapMan) to defend DoS attacks on MANETs, in particular for multi-path communications, where participating nodes are distributed along multi-path communications. The nodes maintain a global view based on summary of capability message exchange, and then dynamically adjust its local constraints to prevent DoS attack on a specific node. The exchange of summary messages between nodes can alter and consume all the network bandwidth.

In [3], the authors used IEEE 802.11 distributed coordination function (DCF) protocols to detect the DoS attacker in MANETs with different PHY layer techniques, by employing several MAC layer protocols. In Bianchi's model, the baseline is used for theoretical throughputs of different technologies. Even though the attacker doesn't have valid information about a transmitted packet, he/she manipulates the IEEE 802.11DCF standards to achieve successful packet transmission on his/her machines, consuming legitimate users' nodes bandwidth. A two-dimensional Markov Chain is used to determine the network capacity. The following are a few issues that that paper didn't explain: DoS attacks related with other layers, no Quality of Service guaranteed through IEEE 802.11 DCF, no priority-based traffic and if all the nodes want to communicate at a single point of time, many collisions occur, leading to congestion collapse.

In [19], the authors implemented six different types of DoS attacks (flood, sinkhole, black hole, selective forward, selfish node attacks and re-request flood) in an ad-hoc on demand distance vector (AODV) and presented an ad-hoc on-demand multi-path distance vector-intrusion detection system (AOMDV-IDS) protocol, to detect and avoid malicious nodes in MANETs. It's a three-phase method: initial trust establishment, detection and elimination of intruder, and alternate path establishment. AODV is combined with the simple model to discover trustworthy paths and only single time route discovery is initiated. Packet forwarding ratio is used to evaluate node's trustworthiness and a continuous product of node trusts to estimate a path trust. There are a number of major problems with that paper, however: it does not include in the model a security for key exchanges to find initial trust among the nodes, there is no explanation on how the verification of intruder and elimination will take place, intermediate nodes can lead to inconsistent routes if the source sequence number is very low, and there is unnecessary bandwidth consumption due to periodic beaconing.

Especially problematic type of DoS attack is called the gray-hole attack. This attack consumes all the system's resources, and isolates legitimate users from the network by dropping the received data packet during the route discovery phase, [20] deals with this type of attack. To detect the attack, the authors propose a novel statistical approach. Based on two Bayesian classification models (Bernoulli and Multinomial with AODV), a routing protocol is developed. Bernoulli's mathematical model is used to identify the behavior of a node using vectors, and based on three exclusive types of packets: route request, route reply, and data packets. From these, the probability of dishonesty of a node over a defined period of time is calculated. The detection scheme is described as follows: the node listens to its neighborhood periodically and collects information about the forwarded packets for its neighbors. The information is then filtered and mathematically modeled as vectors of behaviors. Using the Bayesian filter, the probability of dishonesty for a given vector is calculated using a decentralized process, in order to fit with the mobility and with the dynamic topology of MANETs.

In [21], the authors improved the avoiding mistaken transmission table (AMTT) to prevent flooding attack, which targets DoS attacks on network layer of MANETs through the following three phases: route discovery, route maintenance and data transfer. This improvement is based on the neighbor suppression technique, which identifies malicious nodes during route construction phase. If a malicious node found, then it is isolated for some time and given an adequate penalty. This system monitors the behavior of the network to avoid DoS attacks at network layer. Each node checks the blacklist field in IAMTT before transferring the data packet. The problems are: once a DoS attacker node is found, this approach considers that node as a normal node temporarily. However, if it happens a second time, then the node is put into a blacklist. Sometimes a normal node may request a route again and again, which may be considered as RREQ flooding attack and the node, even though legitimate, can be considered as misbehaving and blacklisted from the network forever.

These existing solutions are based on a variety of security protocols aiming to provide secure communication (from DoS attacks) between nodes without reducing the size of the data exchanged. These recent schemes are not; however, secure enough, because all the techniques are not related with cryptographic techniques. Also, each of these solutions deals with only one specific type of DoS attack. To overcome this limitation, we propose a solution, described in the next section, which applies the combination of data compression technique with UPRE while securing the communication in MANET's environment and maintaining an efficient bandwidth usage as well. The major contribution is that, the fixed length codes compression technique is combined with unidirectional proxy re-encryption technique. The encryption technique provides security for the data that the sender node wants to send via the network to the destination node, whereas the compression technique reduces the data size before it sending to the network for transmission. The main theme of compression technique is that, when the data was transmitted through the network during the DoS attack submitted to the destination node, by using the low bandwidth. In case of UPRE, the data transmitted securely even a key was compromised with the attacker, because the data was encrypted twice.

3 CBUPRE Methodology

CBUPRE: The proposed CBUPRE technique is to secure the communication between nodes in the group and efficient bandwidth usage even during DoS attacks in MANETs. With this technique, we improved upon the UPRE technique based on fixed length codes compression. In sender nodes side, encryption technique follows the compression technique to send the data securely in the MANETs group. In the same way, at the destination node side, the above scheme is performed in reverse so that the original plain text is achieved.

CBUPRE takes place in the following two phases:

1. Fixed Length Codes (FLC) compression.
2. Unidirectional Proxy Re-Encryption technique.

3.1 Phase 1-FLC

The main functionality of the FLC, for instance, two nodes are connected in remote LANs by a pair of routers, bridges and modems. At each end-point of the network connection, compression is done by network devices or by computers. Before putting data into the packet, the data is compressed first, because the compression of on packets generates the same number of packets. So there is no use to compress the packets. If data encryption is required, then data should be compressed first and then encrypted, before placing it in the packet [22].

3.2 Phase 2-UPRE

The second phase of our CBUPRE is UPRE [23]; the proxy is fully trusted because data sharing is done through the node identity and data compression. The Identity

Proxy node generates proxy level key in the group. The proxy signers in the proxy group are responsible for proxy key generation behalf of the original user identity.

Encryption E = (Setup, User level Key Generation, User level Encryption, Proxy level Key Generation, Proxy level Encryption).

As mentioned above, the encryption of compressed data takes place twice, once in a sender node level with senders' private key, and then at the proxy level with a re-encryption key. Then the achieved cipher-text is transferred to destination node, before receiving at destination side the proxy-level cipher-text decrypted with public key and sent to the destination node. Then with the sender nodes' public key decrypts user level cipher-text and received original compressed text.

3.3 Assumptions

Based on the below assumptions and related work, we can say that our proposal does not fall into single point failure problem, secure communication achieved, and efficient bandwidth usage, and key management is also achieved.

- This group communication is under distributed control.
- All the nodes in the group acts as end-systems as well as routers to forward the packets [21].
- Dynamic TCP proxies [24].
- Clustering based group key management [25].
- The newly joining or exiting nodes must follow the protocol standard.

3.4 Methodology of CBUPRE Technique

Our scheme is compression-based unidirectional proxy re-encryption and it is an improved version of UPRE [13].

3.4.1 Sender Side Encryption

The message space M, W is the word space in the message space = $w_1, w_2, \ldots, w_n$, C is the compressed text, C' = cipher text (already compressed), usk_i, upk_i are user level secret and public key pair, node identities Id_i, Id_j and etc.

1. **Setup:** uses system's parameters to setup our approach. We assume that k implicitly included in the following algorithms. The system parameters include a description of a compressed data and a description of a cipher text.
   a. **FLC:** The conditional probability $\Pr(w_i|w_i^{n-1})$ of the word space for an arbitrary n-gram $W = (w_1, w_2, \ldots, w_n)$ as follows:
      i. $\alpha(w_1^n)$ if w_1^n exists
      ii. $\beta(w_1^{n-1})Pr(w_n|w_2^{n-1})$ if w_1^{n-1} exists
      iii. $Pr(w_n|w_2^{n-1})$ Otherwise. Where $\alpha(w_1^n)$, $\beta(w_1^n)$ are smoothed probabilities
   b. **UPRE:** The two primes p and q such that $q|p-1$ and the bit-length of q is the security parameter k, g be the generator of group G, which is the subgroup of Z_q^*

with order q. Choose four hash functions $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow Z_q^*$, $H_2 : G \rightarrow \{0,1\}^{l_0+l_1}$, $H_3 : \{0,1\}^* \rightarrow Z_q^*$ and $H_4 : G \rightarrow Z_q^*$. Here l_0 and l_1 are security parameters determined by k, and the message space M is $\{0,1\}^{l_0}$. The system parameters are $param = (q, G, g, H_1, H_2, H_3, H_4, l_0, l_1)$.

2. **Compression:** It takes the original plain text as input and gives the compressed text as an output.
   a. For example the trigram of a word space W is follows: $\Pr(w) \approx \prod_k \Pr(w_k|w_{k-2}.w_{k-1})$, then the probability of $\Pr(w_k|w_{k-2}.w_{k-1}) \approx \frac{frequency(w_{k-2}w_{k-1}w_k k)}{frequency(w_{k-2}w_{k-1})}$
3. **User level key pair generation:** user's identity is the input and the public, private key as a pair is the output.

$$UKeyGen() : Pick usk_i = (x_{i,1} \overset{\$}{\leftarrow} Z_q^*, x_{i,2} \overset{\$}{\leftarrow} Z_q^*) and\, set\, upk_i = (upk_{i,1}, upk_{i,2}) \\ = (g^{x_{i,1}},\, g^{x_{i,2}}).$$

4. **User level Encryption:** Compressed text and sender node's private key are inputs and the output is first/user level cipher-text.
   $UEncrypt(upk_i = (upk_{i,1}, upk_{i,2}), C)$: To encrypt a compressed text $C \in W \in M$:
   a. Pick $\overset{\$}{\leftarrow} Z_q^*$ *and computeD* $= (upk_{i,1}^{H_4(upk_{i,2})} upk_{i,2})^u$.
   b. Pick $\overset{\$}{\leftarrow}\{0,1\}^{l_1}$, compute $r = H_1(C, w)$.
   c. Compute $E = (upk_{i,1}^{H_4(upk_{i,2})} upk_{i,2})^r$ and $F = H_2(g^r) \oplus (C||w)$.
   d. Compute $= v + r \cdot H_3(D, E, F) mod q$.
   e. Output the ciphertext $C' = (D, E, F, s)$.
5. **Proxy level key pair generation:** based on the identity of participant nodes, the key pair is generated for proxy level encryption process.
   $PKeyGen(usk_i, upk_j)$: On input user *i's* secret key $usk_i = (x_{i,1}, x_{i,2})$ and user *j's* public key $upk_j = (upk_{j,1}, upk_{j,2})$, this algorithm generates the proxy level encryption (re-encryption) key $rk_{i\rightarrow j}$ as below:
   a. Pick $h \overset{\$}{\leftarrow} \{0,1\}^{l_0}$ *and* \$ $\{0,1\}^{l_1}$, *compute* $v = H_1\ (h, \pi)$.
   b. Compute $V = upk_{j,2}^u$ and W = $H_2(g^v) \oplus (h||\pi)$.
   c. Define $rk_{i\rightarrow j}^{<1>} = \frac{h}{x_{i,1}H_4(upk_{i,2}) + x_{i,2}}$. Return $rk_{i\rightarrow j} = (rk_{i\rightarrow j}^{<1>}, V, W)$.
6. **Proxy level Encryption:** First/user level cipher-text re-encrypted at proxy level by user's private key, so proxy level cipher-text (re-encrypted) resulted, and transferred to destination node.
   $PEncrypt(rk_{i\rightarrow j}, C_i', upk_i, upk_j)$: On input a proxy level encryption (user i to user j) key $rk_{i\rightarrow j} = (rk_{i\rightarrow j}^{<1>}, V, W)$, an original ciphertext $C_i' = (D, E, F, s)$ under public key $upk_i = (upk_{i,1}, upk_{i,2})$, this algorithm re-encrypts C_i' into another C_j^* under destination node public key $upk_j = (upk_{j,1}, upk_{j,2})$ as follows:
   a. If $(upk_{i,1}^{H_4(pk_{i,2})} upk_{i,2})^s = D \cdot E^{H_3(D,E,F)}$ does not hold, return $\perp$.
   b. Otherwise, compute $' = E^{rk_{i\rightarrow j}^{<1>}}$, and output (E', F, V, W).

Let $= H1(C,w), v = H1(h,\pi)$, the re-encrypted ciphertext is of the following forms: $C_j^* = (E', F, V, W) = (g^{r.h}, H_2(g^r) \oplus (C||w), upk_{j,2}^v, H_2(g^v) \oplus (h||\pi))$.

3.4.2 Receiver Side Decryption

Decryption can be done in two levels, proxy level, user level and finally decompression to get the original text message from sender node:

7. **Proxy level Decryption:** proxy level encrypted data C^* and destination node's public keys are inputs and user level cipher-text is the output.
   C^* is a re-encrypted ciphertext in the form $C' = (E', F, V, W)$:
   a. Compute $(h||\pi) = W \oplus H_2(V^{1/uski,2})$ and $(C||w) = F \oplus H_2(E'^{1/h})$.
   b. Return C' *if* $V = upk_{i,2}^{H_1(h,\pi)}$ and $E' = g^{H_1(C,w)\cdot h}$ hold; else $\perp$.

8. **User level Decryption:** User level cipher-text *C'* decrypted by using user's public key, so it results compressed text decrypted by using user's public key, so it results compressed text.
   C is an original ciphertext (compressed) in the form $C = (D, E, F, s)$:
   a. If $(upk_{i,1}^{H_4(pk_{i,2})} upk_{i,2})^s = D \cdot E^{H_3(D,E,F)}$ does not hold, return $\perp$.
   b. Otherwise, compute $(C||w) = F \oplus H_2(\frac{1}{E^{xi,1H4(upki,2)+xi,2}})$.
   c. Return C if $E = (upk_{i,1}^{H_4(upk_{i,2})} upk_{i,2})^{H_1(C,w)}$ holds; else return $\perp$.
9. **FLC Decompression:** through the compressed text and the key set, the destination node extracts original plain text finally.

N-grams decompression of word space $W \in M$ message space = $(\frac{frequency(wk)}{N})$, where N-number of words. For example to unigrams: $r(w) \approx \prod_k Pr(w_k)$.

4 Experimental Discussions

The main theme of this experiment is to determine the effectiveness of our compression based unidirectional proxy re-encryption technique approach for secure group communication in MANETs and efficient bandwidth utilization even at the time of DoS attack occurred. In this proposed framework, there are nine steps as follows: compressed data with fixed length codes compression technique by using keyset of the source node, setup, key generation, User level encryption, proxy key generation, proxy level encryption, Proxy Decryption and User decryption on destination node, and then finally decompression.

Step 1: The setup phase, which sets-up the system, is ready to deal with compression technique and unidirectional proxy re-encryption technique. The inputs are the word space within the message space, and security parameters. Based on this, the output is system parameters, and includes a description of a finite compressed message space and a description of a cipher text.

Step 2: In the second step, the original data which we would like to transfer to the destination node in MANET communication is compressed by using the keyset from source node with fixed length codes n-grams compression technique before packetting. This is because of the compression technique on packets gives the same number of packets, which is not useful.

Step 3: This user-level key generation step generates key pair based on the node identity on either nodes.

Step 4: In the fourth step, we will get user level cipher-text from compressed text and private key of sender node identity.

Step 5: This step is proxy-level key generation; it services the proxy to encrypt at proxy level whenever the destination node requires the data from source. This is the one way key generation from source to destination based on their identities.

Step 6: This is the proxy-level encryption step (proxy re-encryption) of user level cipher text by proxy node. The proxy will encrypt the first level cipher-text and transfer to destination node.

Step 7: This is the proxy-level decryption step in destination node side. In this step when destination node is ready to download data, the proxy re-encrypted message is decrypts at proxy level and gives to destination node.

Step 8: This is the user level decryption step, in this process the user gets compressed data from user level encrypted data by using his public key.

Step 9: This is the final step in destination node side to get original data from compressed text, based on decompression scheme with keyset.

Based on the working nature of our algorithm, we compared it with existing DoS detection and prevention methodologies. The Table 1, which shows the major advantages, such as bandwidth usage during the DoS attack, the layers in which the attack could be detected, and the cryptographic technology. The next section correctness proof, that supports our proposed scheme based on our CBUPRE.

Table 1. Comparisons between existing approaches with CBUPRE

	Detection	Prevention	No. of nodes	Bandwidth	Layer	Cryptography
CapMan		√	Dynamic		Transport	
IEEE802.11 DCF	√		Fixed		Data link	
AOMDV	√		–		–	
Statistical Method	√		Fixed		–	
IAMTT		√	–		Network	
CBUPRE	√		Dynamic	√	Presentation & Network	√

5 Proof of Correctness

This section proves our proposed scheme, which ensures a secure MANET group communication. Also, we assume that every incoming node is an authenticated node. This makes an incoming node is possible to study secure data transmission scheme in MANET group communication.

Lemma 1. All members of MANET's groups must know that they are using n-grams fixed length codes compression technique to reduce the burden on network bandwidth.

Proof. The n-grams FLC technique uses same number of bits for each symbol. For example, K-bit code compression technique supports 2 k different symbols. The below illustration shows the usage of FLC compression technique with n-grams. Where n = 1, 2, 3….. We take the following text for our example illustration.

```
"Was receiving a percentage from the farmer till such time as the
advance should be cleared off oak found + that the value of stock,
plant, and implements which were really his own would be about suffi-
cient to pay his debts, leaving himself a free man with the clothes he
stood up in, and nothing more.

<C vi > < P 88>
THE FAIR "THE JOURNEY" THE FIRE TWO months passed away. We are brought
on to a day in February, on which was held the yearly statue or hiring
fair in the country-town of Casterbridge. At one end of the street
stood from two to three hundred blithe and hearty laborers waiting
upon change "all men of the stamp to whom labor suggests nothing worse
than a wrestle with gravitation, and pleasure nothing better than a
renunciation of the same among these, carters and wagoner's were
distinguished by having a piece of whip-cord twisted round their
hats;"
```

Based on FLC n-grams compression for the above text, we have to calculate the different frequencies. First the unigram frequencies, then most frequently occurred bigrams, and then trigrams based on the bigram frequencies and etc. from the above text the n-grams where n = 1, 2, 3 calculated and displayed in Table 2.

Table 2. Trigrams fixed length compression for the above text

Count n = 1	Count n = 2	Count n = 3
125551	20452 e	11229 th
72431 e	17470 he	9585 the
50027 t	17173 t	8989 he
…..	…..	…..
…..	…..	…..

Based on this n-grams FLC technique the size of the data is reduced. So we can say that the data compression is guaranteed and the size of data is reduced by using n-grams fixed length compression technique.

Lemma 2. In the MANET group, every node is identified by their node identity. And all group members must know it.

Proof. We would like to prove this one by contradiction, assuming that there are two nodes with identities I d i and I d j that want to exchange data in a MANET group. In this MANET group nodes I d i and I d j are already authenticated based on their node identity. Thus, two different cases can occur:

Case 1: the two nodes I d i and I d j are in direct communication in the group, which means no intermediate nodes are available between them in the link. So there is no need to think about attacker nodes. Why? Because even if the network bandwidth is very low our method works effectively due to data compression.

Case 2: the nodes I d i and I d j are not neighbours, but communicated in the same MANET group. Between these two nodes there other nodes that are also authenticated by their node identity. So the transmission occurs on whichever link is near with few hops and authenticated nodes between them. This way, the data is shared securely.

Thus, the MANET group is formed securely with node identifications. This proves that there is a contradiction with the assumption that each group maintains authenticated group members only.

Lemma 3. When an unauthorized node wants to join in the group with fault identity in proxy level, it cannot decrypt the encrypted data.

Proof. By Lemma 2 case 2, if any unauthorized node between authorized nodes I d i and I d j wants to join in the network with fault identity N a at proxy level. Even though this N a node accepted to participate in group communication, the proxy level encrypted data will not be decrypted. Why? Because it doesn't contain the proxy level key of particular attacked node. The proof could drive to a conclusion that every node must have a proxy level encryption key to decrypt proxy level cipher-text.

Lemma 4. When an unauthorized node wants to join in the group with faulty identity in user level, it cannot decrypt the encrypted data and cannot decompress it.

Proof. By Lemma 2 case 2, if any unauthorized node N a wants to join between authorized nodes I d i and I d j in network with fault identity at user level. Even though this N a node accepted to participate in group communication, the user level encrypted data will not be decrypted. Why? Because it doesn't consist the user level secret key of particular attacked node. At peak moment, it got decrypted user level cipher-text, but it doesn't know the meaning of the compressed text message.

The proof could drive to a conclusion that every node must have a secret key to decrypt proxy level cipher-text and must know the data compression technique pattern which is used at source node.

Theorem. In MANETs, group communication, secure data transformation is guaranteed, even in lower data bandwidth.

Proof. This can be done through Lemmas 1, 2, 3 and 4.

6 Conclusions

In this paper, we proposed an efficient scheme for secure data exchange and bandwidth usage in MANETs, based on a Unidirectional Proxy Re-Encryption technique (CBUPRE) to make them secure from DoS attacks. With this scheme, the direct and secure communications between any identified nodes in a group can be implemented. Due to fixed length codes compression technique, the data transformation ratio is increased and the delay of communication is lowered. Hence the scheme is more flexible for MANETs. Any member node in a group can transform the data securely due to algorithms unidirectionality. Our future work is to simulate CBUPRE based on the hypothesis that we made, and to construct more secure and efficient group communication for MANETs based on CBUPRE.

References

1. Kaur, T., Toor, A., Saluja, K.: Defending manets against flooding attacks for military applications under group mobility. In: 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1–6 (2014)
2. Konate, K., Abdourahime, G.: Attacks analysis in mobile ad hoc networks: modeling and simulation. In: 2011 Second International Conference on Intelligent Systems, Modeling and Simulation (ISMS), pp. 367–372 (2011)
3. Soryal, J., Saadawi, T.: IEEE 802.11 denial of service attack detection in manet. In: Wireless Telecommunications Symposium (WTS), pp. 1–8 (2012)
4. Pushpalatha, K., Chitra, M.: Gamanet: A ganatic algorithm approach for hierarchical group key management in mobile adhoc network. In: 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), pp. 368–373 (2013)
5. Reddy, P.R.K., Bouzefrane, S.: Analysis and detection of dos attacks in cloud computing by using qse algorithm. In: 2014 IEEE International Conference on High Performance Computing and Communications, 2014 IEEE 6th International Symposium on Cyberspace Safety and Security, 2014 IEEE 11th International Conference on Embedded Software and Systems (HPCC,CSS,ICESS), pp. 1089–1096 (2014)
6. Lolla, V., Law, L., Krishnamurthy, S., Ravishankar, C., Manjunath, D.: Detecting mac layer back-o timer violations in mobile ad hoc networks. In: 26th IEEE International Conference on Distributed Computing Systems, ICDCS 2006, p. 63 (2006)
7. Jia, Q., Sun, K., Stavrou, A.: Capman: capability-based defense against multi-path denial of service (dos) attacks in manet. In: 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), pp. 1–6 (2011)
8. MacVittie, L.: Understanding advanced data compression (2013)
9. Base, O.D.: Oracle advanced compression with oracle database 12c (2015)
10. Oberhumer, M.F.: lzop (2010)
11. Seward, J.: bzip2 and libbzip2, version 1.0.5, a program and library for data compression (2007)
12. loup Gailly, J., Adler, M.: A massively spiffy yet delicately unobtrusive compression library (2013)
13. Wang, H., Cao, Z., Wang, L.: Multi-use and unidirectional identity-based proxy re-encryption schemes. Inf. Sci. **180**, 4042–4059 (2010)
14. Smith, T.: Dvd jon: buy drm-less tracks from apple itunes, 18 March 2005
15. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**, 1–30 (2006)
16. Ivan, A.A., Dodis, Y.: Proxy cryptography revisited. In: NDSS (2003)
17. Reddy, P.R.K., Sivaramaiah, S., Sesadri, U.: Secure data forwarding in cloud storage system by using umib proxy. Int. J. Comput. Technol. **10**, 1905–1912 (2013)

18. Wu, B., Chen, J., Wu, J., Cardei, M.: A survey of attacks and countermeasures in mobile ad hoc networks. In: Xiao, Y., Shen, X.S., Du, D.-Z. (eds.) Wireless Network Security, pp. 103–135. Springer, USA (2007)
19. Gupta, H., Shrivastav, S., Sharma, S.: Detecting the dos attacks in aomdv using aomdv- ids routing. In: 2013 5th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 380–384 (2013)
20. Rmayti, M., Begriche, Y., Khatoun, R., Khoukhi, L., Gaiti, D.: Denial of service (dos) attacks detection in manets through statistical models. In: Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–3 (2014)
21. Ahir, S., Marathe, N., Padiya, P.: Iamtt - new method for resisting network layer denial of service attack on manet. In: 2014 Fourth International Conference on Communication Systems and Network Technologies (CSNT), pp. 762–766 (2014)
22. Allen Kent, J.G.W.: Artificial Intelligence and ADA to Systems Integration: Concepts: Methods, and Tools (1992)
23. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**, 1–30 (2006)
24. Ouyang, T., Jin, S., Rabinovich, M.: Dynamic tcp proxies: coping with disadvantaged hosts in manets. In: 29th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops 2009, pp. 530–536 (2009)
25. El-Sayed, A.: Clustering Based Group Key Management for MANET. In: Awad, A.I., Hassanien, A.E., Baba, K. (eds.) SecNet 2013. CCIS, vol. 381, pp. 11–26. Springer, Heidelberg (2013)

Author Index