Farhood Hoseinpur

TOWARDS SECURITY AND RESOURCE EFFICIENCY IN FOG COMPUTING NETWORKS

ACTA UNIVERSITATIS LAPPEENRANTAENSIS 1053

Farhood Hoseinpur

TOWARDS SECURITY AND RESOURCE EFFICIENCY IN FOG COMPUTING NETWORKS

Dissertation for the degree of Doctor of Science (Technology) to be presented with due permission for public examination and criticism in the Auditorium 1318 at Lappeenranta-Lahti University of Technology LUT, Lappeenranta, Finland on the 2nd of December, 2022, at noon.

Acta Universitatis
Lappeenrantaensis 1053

Supervisors	Professor Kari Smolander LUT School of Engineering Science Lappeenranta-Lahti University of Technology LUT Finland
	Associate Professor Pedro Juliano Nardelli LUT School of Energy Systems Lappeenranta-Lahti University of Technology LUT Finland
Reviewers	Professor Tommi Kärkkäinen Faculty of Information Technology University of Jyväskylä Finland
	Professor Antonio Marcos Alberti National Institute of Telecommunications Brazil
Opponent	Assistant Professor Ella Peltonen Faculty of Information Technology and Electrical Engineering University of Oulu Finland

ISBN 978-952-335-885-0
ISBN 978-952-335-886-7 (PDF)
ISSN 1456-4491 (Print)
ISSN 2814-5518 (Online)

Lappeenranta-Lahti University of Technology LUT
LUT University Press 2022

Abstract

Farhood Hoseinpur
Towards Security and Resource Efficiency in Fog Computing Networks
Lappeenranta 2022
63 pages
Acta Universitatis Lappeenrantaensis 1053
Diss. Lappeenranta-Lahti University of Technology LUT
ISBN 978-952-335-885-0, ISBN 978-952-335-886-7 (PDF), ISSN 1456-4491 (Print), ISSN 2814-5518 (Online)

The Internet of Things plays a crucial role in digitalising services in many domains directly affecting humans' day-to-day lives. However, with immense digitalisation, the Big Data phenomenon has become a critical issue. Big Data is, in general, characterized by five major features, including *Volume*, *Velocity*, *Value*, *Veracity*, and *Variety*, which are known as the 5 Vs of Big Data. In addition to its general characteristics, a new feature, i.e., geo-distribution, is also introduced by emerging IoT applications. These characteristics demand robust computing services for managing Big Data. Geo-distribution of Big Data and IoT services necessitates deploying computing services already at the network's edge. Hence, fog computing, a geographically distributed computing platform, was introduced to provide local computing services for IoT applications. However, because of its limited resources, i.e., computing, storage, energy, and bandwidth, fog computing faces challenges concerning implementing strong security measures.
This dissertation explores how to enhance security and privacy in fog computing by designing lightweight and efficient intrusion detection systems (IDS) to detect and prevent malicious attacks on IoT services. Furthermore, how to efficiently utilize computing resources in fog computing to enhance the quality of services and effectively facilitate the deployment of security mechanisms. The dissertation comprises five studies that employed a portfolio of research methods, including conceptualisation, proof of concept, and simulation modelling.
The contribution of this thesis is threefold. First, the dissertation proposes solutions to enhance and tailor anomaly-based IDS inspired by the biological immune system to detect network intrusions in the IoT efficiently. As a result, a new lightweight architecture, including an extra layer of protection called the innate immune system, was proposed, and a proof of concept was presented. Second, it proposes a security framework to manage the trust levels of the device nodes that join the computing pool in the fog network and control their access to processing data with different levels of criticality. Third, it proposes solutions for efficiently managing and utilizing the limited resources in fog computing. This includes the introduction of a concept called Smart Data that aims to reshape the existing perspective of Big Data computing from a passive to an active form. Further, the dissertation presents a resource management model and algorithms to efficiently map multi-task applications into fog computing networks to reduce communication delays and enhance the quality of service.

Keywords: internet of things, fog computing, resource management, intrusion detection system, security

Acknowledgements

This doctoral research began at the University of Turku in 2015 and continued from Lappeenranta Lahti University of Technology (LUT) until the autumn of 2022. I would like to express my appreciation to everyone who supported and encouraged me on this long journey.

First, I would like to express my deepest gratitude to my supervisors, Professor Kari Smolander and Associate Professor Pedro Juliano Nardelli, for their continued guidance and valuable support and encouragement.

I wish to thank Professor Tommi Kärkkäinen and Professor Antonio Marcos Alberti for their valuable reviews of this dissertation and for providing constructive comments that improved its quality. I also extend special thanks to Assistant Professor Ella Peltonen for agreeing to be the opponent at the public thesis defense.

I am extremely grateful to Professor Juha Plosila and Associate Professor Seppo Virtanen for their support and guidance on my research and publications. I want to express my sincere thanks to Professor Tapio Pahikkala for all his guidance, support, and outstanding feedback on my last publication.

I am deeply indebted to Professor Erkki Sutinen and Dr. Päivi Sutinen for their love and care and for welcoming me into their family. They opened their hearts and supported me on the hardest days of my life. I sincerely appreciate their kindness.

My warmest gratitude goes to my colleagues and collaborators for their valuable contributions to our joint publications. I would especially like to thank Dr. Payam Vahdani Amoli for his valuable input on our joint publications, and Mr. Babak Ziaee and Dr. Hesam Zolfaghari for their support and encouragement.

I would also like to take this opportunity to express my profound gratitude to Professor Jaber Karimpour and Mr. Mehdi Barazandetar for their unwavering support and belief in me since the beginning of my journey.

I want to extend my special thanks to Eficode Oy and all my wonderful colleagues for building such an excellent workplace and professional community. Furthermore, I want to express my deepest gratitude to Mr. Tuomas Leppilampi and Mr. Kalle Lindroos for their valuable encouragement and support, as well as for believing in me and giving me the opportunity to advance my professional career.

I am sincerely grateful to my parents, Mr. Reza Hoseinpur and Mrs. Roghiyeh Khorsandi, whose constant love and support keep me motivated and confident. My accomplishments and success undoubtedly owe to their love, care, and sacrifices. My deepest thanks go to my brother, Mr. Masood Hoseinpur, who keeps me grounded and always supports my adventures.

I would like to extend my sincere thanks to my parents-in-law, Mr. Heikki Helminen and Mrs. Kirsti Helminen, for their support, encouragement, trust, and solicitude.

Finally and most importantly, I would like to dedicate this thesis to my love, Hanna, who has made my life so wonderful and prosperous. These few words cannot express my deep appreciation for her love and endless support these past years.

Farhood Hoseinpur
December 2022
Lappeenranta, Finland

To my better half, Hanna
and our sweet daughter Rita

Contents

List of publications

This dissertation is based on the following original publications, referred to in the dissertation as Publications I, II, III, IV, and V. For all of the included publications, Farhood Hoseinpur was the principal contributor with regard to research design, planning, execution, and publication writing.

The corresponding publishers have granted the rights to include the following publications in the dissertation.

I. Hosseinpour, F., Vahdani Amoli, P., Farahnakian, F., Plosila, J., and Hämäläinen, T. (2014) Artificial Immune System Based Intrusion Detection: Innate Immunity using an Unsupervised Learning Approach. *International Journal of Digital Content Technology and its Applications.* Vol. 8(5). pp. 1-12.

II. Hosseinpour, F., Amoli, P. V., Plosila, J., and Hämäläinen, T. (2016). An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach. *International Journal of Digital Content Technology and its Applications.* vol. 10(5), pp. 34-46.

III. Hosseinpour, F., Plosila, J., and Tenhunen, H. (2016). An Approach for Smart Management of Big Data in the Fog Computing Context. *IEEE International Conference on Cloud Computing Technology and Science (CloudCom).* pp. 468-471. New York: IEEE.

IV. Hosseinpour, F., Siddiqui, A. S., Plosila, J., and Tenhunen, H. (2018) A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models. *Proceedings of Research and Practical Issues of Enterprise Information Systems. & Lecture Notes in Business Information Processing.* vol. 310, pp. 168-180. Cham:Springer.

V. Hosseinpour, F., Naebi, A., Virtanen, S., Pahikkala, T., Tenhunen, H., and Plosila, J. (2021). A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks. *IEEE Access.* vol. 9, pp. 152792-152802.

Related publications (not included in this dissertation)

This dissertation also builds on other related publications, which were not included in the dissertation portfolio of publications.

I. Hosseinpour, F., Bakar, K. A., Hardoroudi, A. H., and Kazazi , N. (2010). Survey on Artificial Immune System as a Bio-inspired Technique for Anomaly Based Intrusion Detection Systems. *International Conference on Intelligent Networking and Collaborative Systems.* pp. 323-324.

II. Hosseinpour, F., Meng, Y., Westerlund, T., Plosila, J., and Tenhunen, H. (2016). A Review on Fog Computing Systems. *International Journal of Advancements in Computing Technology*. vol. 8(5), pp. 1525-1530.

III. Hosseinpour, F., Plosila, J., and Tenhunen, H. (2016). ”Smart Data: A New Perspective of Tackling the Big Data Phenomena Leveraging a Fog Computing System. *International Journal of Digital Content Technology and its Applications.* vol. 10(5). pp. 119-130

IV. Vitabile, S., Marks, M., Stojanovic, D., Pllana, S., Molina, J. M., Krzyszton, M., Sikora, A., Jarynowski, A., Hosseinpour, F., Jakobik, A., Stojnev, A., Respicio, A., Moldovan, D., Pop, C., and Salomie, I. (2019). Medical Data Processing and Analysis for Remote Health and Activities Monitoring. *High-Performance Modelling and Simulation for Big Data Applications.* pp.186–220.

Nomenclature

Abbreviations

AIS	Artificial Immune System
ANN	Artificial Neural Networks
CDF	Cumulative Distribution Function
DDOS	Distributed Denial of Service
DOS	Denial-of-Service
GPS	Global Positioning System
GSM	Global System for Mobile
HIS	Human Immune System
IDS	Intrusion Detection System
IOT	Internet of Things
IP	Internet Protocol
LISP	Locator/Identifier Separation Protocol
MAC	Media Access Control
MIQP	Mixed-Integer-Quadratic-Programming
PKI	Public Key Infrastructure
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
RA	Remote Authentication
RBAC	Role-Based Access Control
RFID	Radio-Frequency Identification
SLA	Service Level Agreement
SOA	Service-Oriented Architecture
VM	Virtual Machine
VMM	Virtual Machine Monitor

1 Introduction

The rise of the Internet of Things (IoT) and cloud computing has pushed the evolution of traditional embedded devices into drastic changes. In this era, embedded devices are transforming into intelligent systems that are gradually being adopted and deployed in both general and special workplace environments and environments in the everyday lives of people (Nardelli, 2022). In this paradigm, information flows from sensors through gateways to a computing server, for example, the cloud, where a diverse set of applications in several domains can be provided for a professional or a layman. Emerging IoT applications generate massive amounts of data from various sensors, which are generally referred to as Big Data. Traditionally, the data is not processed in the proximity of a sensor but transferred as it is to a server that might be located in a cloud. However, transferring a constantly increasing amount of information from sensors to a cloud is not feasible. Hence, pre-processing the data already at the edge of the network will ease the computation load and complexity from the cloud servers, reduce the communication delay, and increase the energy efficiency in IoT systems (Narayanan et al., 2020).

Fog computing is a promising technology that extends cloud computing and its services to the edge of the network (Bonomi et al., 2014). Fog computing complements cloud computing services by providing a faster response for latency-sensitive applications. Data analysis at the fog computing layer is lightweight; therefore more advanced analyses and processing will be done at the cloud layer. Fog distinguishes from the cloud in its proximity to end users, geographical distribution, support for mobility, heterogeneity, dynamicity, and interoperability (Datta et al., 2015). Fog nodes are heterogeneous, ranging from switches, routers, base stations, edge clusters, or micro data centres. They host different operating systems and applications. To hide hardware and platform heterogeneity, virtualization technologies are used to develop a software abstraction layer on top of the hardware and host operating system (Bonomi et al., 2014; Osanaiye et al., 2017). The abstraction layer provides a uniform programmable interface for managing and controlling resources in the fog platform. Virtualization facilitates the co-existence and multi-tenancy of different applications in a single physical node for efficient utilization of the resources while ensuring the isolation of the multiple tenants in a single machine. Unlike cloud computing, fog computing is a distributed and highly dynamic computing network (Sahni et al., 2017). Fog computing nodes are geographically distributed at the edge of the network. They can be mobile and battery-enabled, contributing more to their geo-distributed and dynamic nature. Utilizing the available computing and storage capabilities of smart mobile devices in fog computing networks is one way to empower fog computing. This is referred to as cooperative computing (Sahni et al., 2017). Fog networks are mostly wireless supporting heterogeneous communication protocols ranging from 5G to ZigBee or 6LowPAN, facilitating interoperability between different IoT devices (Yousefpour et al., 2019; Mäki, 2021).

Fog computing nodes generally have considerably limited computing and storage capacities compared to cloud computing nodes. Hence, the efficient utilization and management of resources (i.e., time, energy, communication bandwidth, storage, and computing capacity) are crucial for the successful deployment of fog computing services.

From another perspective, the decentralised and open nature of fog computing networks makes them more vulnerable to security threats (Zhang et al., 2018). The predominance of wireless networks in fog networks poses considerable security concerns (Zhi et al., 2019). Moreover, due to limited resources in fog nodes, strong security measures are infeasible. Hence, fog computing is more vulnerable to classical network intrusions, such as man-in-the-middle, flooding, and port scanning (Khater et al., 2019; Hosseinpour et al., 2016). Furthermore, unlike cloud computing, fog nodes may belong to different service providers, including internet service providers, cloud service providers, end users, etc., resulting in more complicated trust, access control, and authentication issues (Zhi et al., 2019; Hosseinpour et al., 2018). Outsourcing the data into such diverse fog storage nodes also increases security threads with regard to integrity and unauthorised access.

1.1 Research objectives and questions

This thesis tackles research challenges in relation to *security* and *resource efficiency* in fog computing networks. In summary, the following research objectives and questions have been delineated:

Security: Detection of silent and zero-day attacks that are unknown to the system requires constant monitoring and behavioral analysis of the system's components and communication. Intrusion Detection Systems (IDS) with local decision making are the first line of defence against intrusive behaviour in fog computing systems. Due to the resource limitations of fog and IoT devices, lightweight IDS are highly desirable.
Further, in a highly dynamic fog computing system where computing nodes with different trust levels from different service providers or individuals are dynamically joining and leaving the system, a proper access control system is needed to protect the security and privacy of the data processed and stored in the fog computing nodes.
The goal of this thesis concerning the security challenges in fog computing is to enhance the security and privacy of IoT services in fog computing through lightweight intrusion detection, trust management, and access control. To address this objective, the following research questions are addressed in this thesis:

- *RQ1: How can an anomaly-based intrusion detection systems be enhanced to detect zero-day attacks?*
- *RQ2: How can distributed intrusion detection systems be adapted in a lightweight form for fog computing to protect IoT services against silent and zero-day attacks?*
- *RQ3: How can the security of IoT services be enhanced in cooperative fog computing networks?*

Resource efficiency: Since fog computing networks suffer from high heterogeneity, dynamicity, and limited resources, management, and efficient utilization of the resources is crucial to satisfy the Quality of Service (QoS) and Quality of Experience (QoE). In a fog

computing system, applications can enter and leave the system at run-time and have different requirements in terms of, for example, computing intensity, types of computations, memory usage, and power/energy consumption. Furthermore, at any time, some nodes might leave the fog network because of, for example, low battery levels or malfunctions, and, correspondingly, some new nodes might enter the fog network (Lee et al., 2017). In such a dynamic platform, complex resource management is needed to assign new or unfinished tasks on the fog layer while considering several constraints, for example, Service Level Agreement (SLA), power consumption, computing capacity, communication bandwidth, maximum tolerable latency, energy efficiency, and reliability.
Therefore, the goal of this thesis concerning the resource efficiency challenges in fog computing is to enhance resource utilization by introducing efficient approaches tailored to the specific characteristics and limitations of fog computing. To address this objective, the following research questions are addressed in this thesis:

- *RQ4: How can we decouple the dependency of application codes from the underlying hardware computing platform in a highly dynamic fog computing network to efficiently tackle the big data in IoT?*
- *RQ5: How can fog computing resources be optimally allocated to multitask IoT applications to reduce communication delays and optimise resource utilization in fog computing networks?*

1.2 Thesis Contribution

This thesis is an investigation to address the above questions by considering different existing aspects and constraints in the development of fog computing systems. Figure 1.1 shows a general overview of the works accomplished in this thesis and the cohesion of the publications. As shown in the figure, two main tracks have been investigated in fog computing, including *security* and *resource efficiency*, to tackle some of the main challenges in fog computing systems.
Security: Since fog computing is an open system, the risk of unseen network attacks is high. Anomaly-based intrusion detection systems are able to detect new types of attacks and abnormal behaviour in the network. However, the precision and effectiveness of anomaly detection systems rely heavily on the training datasets. With the rapid growth of the attackers' knowledge, relying on offline training data is a drawback for such anomaly-based IDS. The Artificial Immune System (AIS) is a biologically inspired technique that has proven to be an effective approach for anomaly detection. In Publication I, we proposed a multi-agent-based architecture for AIS-based IDS and verified its effectiveness. Furthermore, we introduced a concept called *innate immunity* by proposing a joint technique using unsupervised learning methods to feed into our AIS engine to empower the IDS to detect zero-day attacks. The innate immune system is the first response to the network attacks that provide online training data for anomaly-based IDS in addition to the offline training based on offline datasets.
Since the edge devices, including the sensors and fog computing nodes, are resource-constrained, utilizing the conventional IDS is not an efficient and feasible measure. Hence,

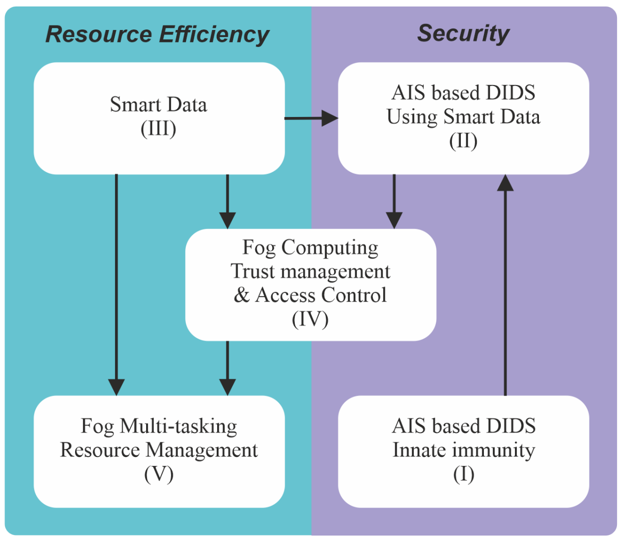

Resource Efficiency
Security
Smart Data (III)
AIS based DIDS Using Smart Data (II)
Fog Computing Trust management & Access Control (IV)
Fog Multi-tasking Resource Management (V)
AIS based DIDS Innate immunity (I)

Figure 1.1: Illustration of publications' cohesion. Each category is labeled with different colors.

the development of lightweight IDS is necessary to facilitate the adoption of such a protection mechanism in fog computing and edge devices. Therefore, in Publication II, utilizing the smart data technique that we presented in publication III, we presented a lightweight, multi-layered, and multi-agent architecture for distributed anomaly-based IDS tailored to a three-layered IoT system. We considered a smart IoT-based logistic system as our case study in this publication.

In cooperative fog computing systems, computing nodes come from different sources that may not be trustworthy. To limit the access level for such devices in publication IV, we proposed a framework based on role-based access control techniques and trust models. In this model, we considered the correctness of computing tasks to verify the trustworthiness and reliability of a node. As a result, nodes can gain trust and be promoted to process data at higher critical levels.

Resource Efficiency: Our target fog computing platform is a hierarchical and distributed computing system that receives data from sensors, pre-processes/processes/stores, or forwards them to the cloud computing layer. IoT applications that need computing services from such a system contain several distributed tasks with dependencies, each of which must be executed in one fog computing node. Thanks to virtualization technology, a fog node can host several application tasks, providing them an isolated run-time environment. The task modules of the applications are a single function/portion of the code that requires input data provided by precedent task modules or the sensors and produces specific output data sent to the subsequent tasks or present as the final output of the application. To run an application, the task modules must be *mapped* and deployed in a fog computing node. In a fog computing system, reducing communication and computation delays is one of the main goals to reduce service delay, network congestion, and bandwidth and energy

consumption, contributing to enhancing QoS and QoE. In this respect, the location of the data and the nodes that process them are essential. The closer the source of data and computing node the less communication delay will occur. Therefore, the code for each task module and, in general, the whole distributed application must be deployed as close as possible to the source of the data. However, it is possible that the other applications already occupy the closest nodes or the communication links are busy, and adding a new connection will degrade the connection in the existing applications within that link. Based on this, we propose an efficient resource management technique with joint optimization of the delay and deployment costs in Publication V.
As the number of IoT applications grows, the diversity of data also increases. This increase in the diversity of the data results in increase in the diversity of the applications that process them. However, such diverse applications may have several common task modules. Utilizing common task modules for multiple applications will reduce deployment cost and complexity in the fog computing layer. Furthermore, with a highly dynamic fog computing system, there is a possibility that some nodes that contain particular application codes will become unavailable or the source of the data will move to another geographical area. In this case, the set of computing nodes previously assigned for processing them becomes unreachable or costly to communicate with and delay-prone. To tackle this problem, we propose a concept called Smart Data in Publication III that aims to decouple the dependency of application codes from the hardware computing platform and instead embed the intelligence into the data itself by encapsulating sensory data with lightweight and modular applications codes.

1.3 Thesis Structure

This thesis consists of five chapters with the following contents: The current chapter describes the overall direction of the research by presenting an overview of the topic, the motivation behind the research, objectives and research questions, and the impact and contribution of the thesis. Chapter 2 presents the general landscape of the research relevant to the main tracks of the thesis, i.e., security and resource efficiency in fog computing networks. Furthermore, it addresses the research gap that this dissertation attempts to fill. Chapter 3 provides an overview of the publications in the dissertation portfolio and describes each publication's main findings and contributions, as well as the research methodology adopted in each publication. Chapter 4 summarizes the research results and theoretical and practical contributions of the thesis, as well as future research opportunities. Finally, Chapter 5 concludes the dissertation.

2 Background

This chapter presents the background and state of the art in the scientific literature related to the doctoral thesis topic. The following sections constitute the existing works published by other scholars, providing the theoretical background supporting the main topics covered in publications I - V.

2.1 Internet of Things (IoT)

Phenomenal augmentation in the number of smart devices and wireless and sensor technologies has led to the realization of a new era of computing called the Internet of Things (IoT). Kevin Ashton coined the term Internet of Things in 1999 to present an intelligent system for supply chains using Radio-Frequency Identification (RFID) tags that enabled the objects in the physical world to connect (Ashton et al., 2009). The IoT is composed of heterogeneous smart devices (referred to as *things*) ranging from mobile phones, intelligent network infrastructure, and in general, any device with computing, storage, and communication interconnected via an assortment of communication technologies, including Bluetooth, Wi-Fi, ZigBee, NB-IoT, LoRa, and GSM. The number of connected *things* is rapidly growing every day. According to a report from Cisco (Grossetete, n.d.), *"by 2023, there will be nearly 30 billion network-connected devices and connections, up from 18.4 billion in 2018. IoT devices will account for 50% (14.7B) of all networked devices by 2023, compared to 33% (6.1B) in 2018."* The IoT facilitates the digitalization of almost any old-fashioned service to improve citizens' living standards (Yaqoob et al., 2017). The extensive adoption of IoT technologies facilitates the transformation of many aspects of how we live. Many IoT-enabled smart appliances, home automation, smart energy management, electronic healthcare using wearable medical devices, etc., revolutionize consumers' lifestyles, promising a better quality of life, safety, and security. In the industry sector, intelligent IoT systems, such as mechanical and robotic automation systems, networked vehicles, intelligent traffic control, etc., are widely adopted to optimize production and improve services. This indicates the need to select the most suitable IoT platform (Ullah et al., 2020).

2.1.1 IoT Architecture

The IoT is a multi-layered system in which each layer carries out specific functionalities collaboratively with the other layers. Two widely considered IoT architectures include three-layered and five-layered architectures (Al-Qaseemi et al., 2016).

Three-layered Architecture: The three-layer architecture illustrated in Figure 2.1 is regarded as basic IoT architecture. As the name suggests, it consists of three layers: the perception, network, and application layers. The perception layer, also known as the physical layer, includes the physical devices equipped with sensors and actuators located at the edge of the network and interacting with the physical environment, sensing and collecting data, or carrying out actuation commands. The network layer provides connectivity to the devices in the perception layer to the application layer through an assortment

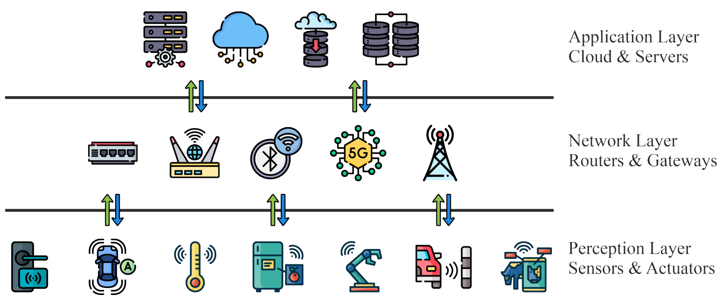


Figure 2.1: IoT three-layered architecture (Al-Qaseemi et al., 2016)

of wired and wireless communication technologies such as Bluetooth, Zigbee, and 5G. The network layer facilitates sending the sensory data from the perception layer to the application layer and other way around, sending the processing results or actuation commands from the application layer to the perception layer. Finally, the application layer consists of servers and storage, providing the required processing power to process collected data (Al-Qaseemi et al., 2016).

Five-layered Architecture: The five-layer architecture, as illustrated in Figure 2.2 consists of two additional layers of middleware and business in addition to the three layers described earlier. From another perspective, the application layer is further divided into three sub-layers in the five-layer architecture. The middleware layer consists of processing and storage hardware, such as distributed servers and databases, providing critical functionalities, such as aggregating and filtering, often called pre-processing, of the data and access control for the applications and devices (Bhawiyuga et al., 2019). The application layer in this model enables the management of IoT applications. Furthermore, the business layer is responsible for the global management of IoT systems, including services, business, and profit models. Based on the analytical results in this layer, the future directions and business strategies of IoT services are determined (Khan et al., 2012).

2.2 Big Data

In IoT systems, massive amounts of data, referred to as Big Data, are generated in the perception layer. Big Data is, in general, characterized by five main features as *Volume*, *Velocity*, *Value*, *Veracity*, and *Variety*, which are known as the 5 Vs of Big Data (Özköse et al., 2015). The volume of Big Data refers to the size, scale, or amount of the data required to be processed. The velocity of Big Data represents the speed of the generation of data. The variety of Big Data expresses the complexity of data as structured, semi-structured, unstructured, and mixed data. The value of the data reflects the added value of the data to the underlying process. Finally, the veracity of Big Data refers to the consistency and trustworthiness of the data being processed. Data veracity ensures the integrity, availability, and accountability of the data. In addition to its general characteristics, emerging IoT applications also introduce a new feature, namely *geo-distribution*. In IoT-based applications, the data is being, generated by sensors in different geographical locations through mobile or fixed sensors that need to be managed as a coherent whole.

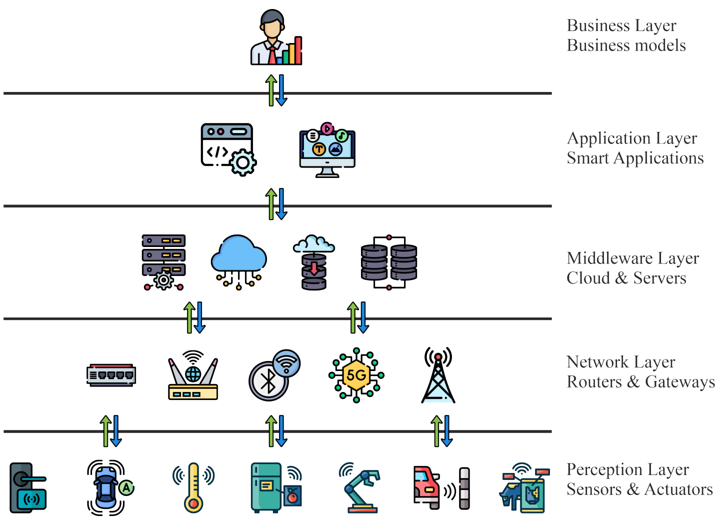


Figure 2.2: IoT five-layered architecture (Al-Qaseemi et al., 2016)

2.3 IoT Computing Solutions

Resource-constrained devices at the network's edge do not have sufficient processing and storage capabilities. Therefore, an external robust computing platform is required for processing such highly distributed Big Data in IoT systems. Robust computing platforms are crucial for processing Big Data in IoT to provide the desired service and guarantee the QoS and QoE while preserving data privacy and security.

Cloud Computing platforms were initially employed to accommodate IoT systems' storage and computing needs. In this approach, the data provided by the sensors is transferred to cloud servers in distant locations for further processing. However, transferring a constantly increasing amount of information from sensors to the cloud is not an optimum solution due to delays and latencies caused by communication to distant servers. To overcome this intrinsic limitation of centralized data processing in cloud computing, a new paradigm called *Fog Computing* has been introduced (Bonomi et al., 2014). Fog computing is characterized by heterogeneity, dynamicity, mobility, and geographical distribution that complement the characteristics of cloud computing services, providing local processing and faster responses for delay-sensitive or time-critical applications. It is considered a derivative of cloud computing, extending cloud services to the network's edge (Rafique et al., 2019).

Fog computing does not substitute cloud computing services but instead, with a well-defined interplay, complements the cloud computing services. Fog computing reduces latency and response time for frequent and delay-sensitive local user processes. By contrast, cloud computing offers powerful computing resources and more extensive data storage for the global data gathered from the more extensive geographical area.

This thesis focuses on the fog computing platform, addressing two crucial challenges in adopting this technology for IoT systems: *resource efficiency* and *security*.

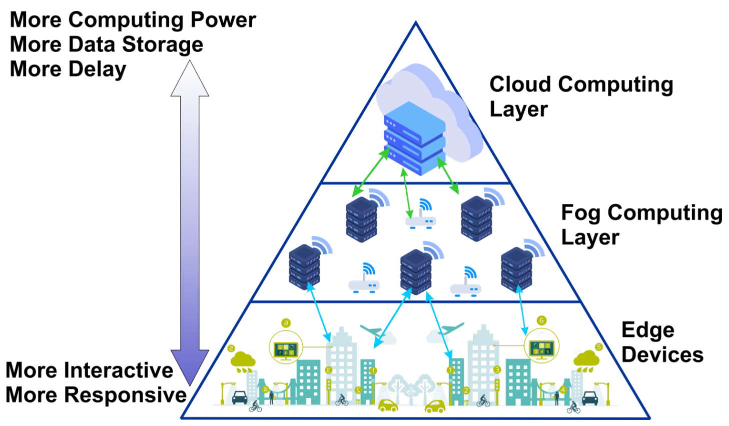


Figure 2.3: IoT computing architecture

2.4 Fog Computing

Fog computing was introduced in 2012 by Cisco as an additional computing layer near the edge of the network to complement cloud computing services. Bringing computational intelligence geographically near the end users provides new or better services for latency-sensitive, location-aware, and geographically distributed applications that, due to their characteristics, are not feasible merely through cloud computing. In this paradigm, smart devices and communication components with both computation and storage capabilities, such as, intelligent routers, bridges, gateways, and smart devices such as tablets and mobile phones, compose the fog computing platform. In this model, some frequent yet simple cloud tasks are delegated to fog, resulting in better performance for IoT applications.

In a systematic view, an IoT system that uses joint services from fog and cloud computing can be seen as a three-layered architecture (Figure 2.3). The cloud computing infrastructure is composed of homogeneous, high-performance resources deployed in a centralized fashion in the top layer. Accordingly, it provides a more comprehensive data analysis and storage over an extended period or geographical area. In the middle layer, fog computing extends the computation paradigm to the network's edge, providing local computing and storage for local services. Fog computing does not outsource cloud computing. Instead, it aims to provide physically closer computing and storage services to the users, provisioning new breed of applications and services with an efficient interplay with the cloud layer. The expected benefit is a better quality of service for applications that require low latency. Lower latency is obtained by performing a part of the data analysis already at the fog computing layer. Data analysis at the fog computing layer is lightweight; therefore more advanced analyses and processing will be done at the cloud layer. For example, the Fog layer pre-processes the raw data coming from the edge sensors before sending them to the cloud layer. Applications that do not require real-time computation or need high processing power are performed at the cloud layer.

2.4.1 Fog Computing characteristics

Fog computing has some distinct characteristics that make it a suitable platform for IoT:
Edge Proximity and Low latency: Since the fog computing nodes are distributed across the edge of the network, i.e., near the sensors and source of the data generation, the data does not need to travel a long distance to reach cloud computing. This is an essential feature of fog computing that reduces latencies and response times for IoT applications.
Geographical Distribution: Unlike cloud computing servers that are centrally located in a building, fog nodes are geographically distributed. Thus, they are the best option for computing data from highly distributed IoT systems. For example, fog computing can deliver high-quality streaming to moving vehicles through proxies and gateways along highways and tracks.
Support for mobility: Some IoT systems and their sensor sets may have a high mobility character. Therefore, frequent handovers among low-range fog servers are expected. This may lead to service disruptions, degrade the user experience, and decrease network resource utilization. Logically, anytime a user moves from one fog node's coverage area to another, a new connection must be established with a new network identifier. Thus, anytime that a user moves from one area to another, the IP address should also be changed with respect to the user's new location. Fog computing supports users' mobility by decoupling these two network functions using Locator/Identifier Separation Protocol or LISP. In LISP, both identifiers and locators can be IP addresses or arbitrary elements, such as a set of GPS coordinates or a MAC address. With distributed mobility management, fog computing can handle the change of the IP addresses as nodes move, thus ensuring seamless migration of the active session.
Heterogeneity: Fog computing comprises a wide variety of computing elements ranging from a macro or small cell base station, intelligent Wi-Fi access points, or any other mobile device with computing, storage, and network capabilities. Further, such elements may operate with various networking protocols ranging from low-range Zigbee, LoRa, SigFox, or 6LowPAN to high-range 5G networks.
Interoperability: Dealing with heterogeneous networks and computing elements, fog computing must be able to interoperate to give a wide range of services. Especially with the co-existence of cloud computing, fog computing should support interoperability to provide seamless services to edge users.
Dynamicity: Dynamicity is another distinct characteristic of fog computing networks. For example, fog nodes may dynamically join and leave the network due to mobility or power limitations. Or, the other way around, the edge sensors might be mobile and move from one local fog network to another.

2.4.2 Virtualization

One effective countermeasure to cope with the heterogeneity in fog computing networks is virtualization technology. Virtualization is a key enabler technology for resource management in fog computing that provides a seamless and integrated computing and storage platform for applications. Virtualization includes the process of abstracting the heterogeneous computing infrastructure and slicing it into Virtual Machines (VMs) to hide the

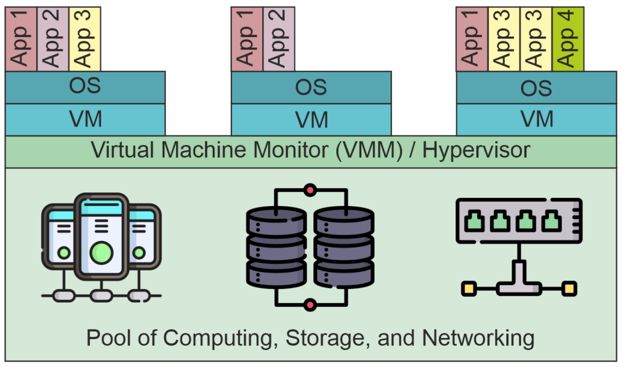

App 1
App 2
App 3
App 1
App 2
App 1
App 3
App 3
App 4
OS
OS
OS
VM
VM
VM
Virtual Machine Monitor (VMM) / Hypervisor
Pool of Computing, Storage, and Networking

Figure 2.4: Fog computing virtualization architecture.

details of the heterogeneity of hardware devices from the running applications. Virtualization facilitates the multi-tenancy and co-execution of multiple applications in a single physical node without interfering with each others' execution and violating the security of each application. Figure 2.4 shows the virtualization architecture of fog computing systems. A Hypervisor or Virtual Machine Monitor (VMM) sits on top of the heterogeneous hardware infrastructure and hosts operating systems and applications. The VMM presents guest operating systems with a virtual operating platform and manages the execution of them and their applications. From the VM perspective, the applications are run within VMs without considering the underlying hardware heterogeneity and distribution. From the perspective of physical machines, each VM needs to reside in one of the physical nodes within the fog computing system. A VMM should have built-in resource management and provisioning and service consolidation to optimally use the system resources, i.e, energy, bandwidth, processing capacity, and storage capacity.

Virtualization is the enabling technology for the distinctive functionalities of microservices and increasing their performance. Microservice architecture is an efficient approach for designing distributed applications (Dragoni et al., 2018; Oparin et al., 2019). In Microservice architecture, applications are developed as a collection of loosely coupled small-sized programs called *microservice*. A microservice is a cohesive, self-contained and independent piece of functionality with clear interfaces that deliver the service through a communication protocol over a network. Microservices are developed in the form of a black box package that provides intended computational service independent of their platform or programming language. Hence, the microservice architecture supports the heterogeneity of IoT applications. Applications developed as assemble of microservices feature better scalability, interoperability, and easier upgrade and large scale deployments compared to traditional monolithic, complex, and rigid applications (Banica et al., 2017).

Containerization, as lightweight virtualization, has brought significant advantages in implementation time, resource utilization, and low management cost, compared to its heavyweight counterparts (Pahl and Lee, 2015). Containers facilitate the cost-effective provision of microservices. Hence, they have been widely used to develop microservice applications (Singh and Peddoju, 2017; Sampaio et al., 2019; Ray et al., 2020).

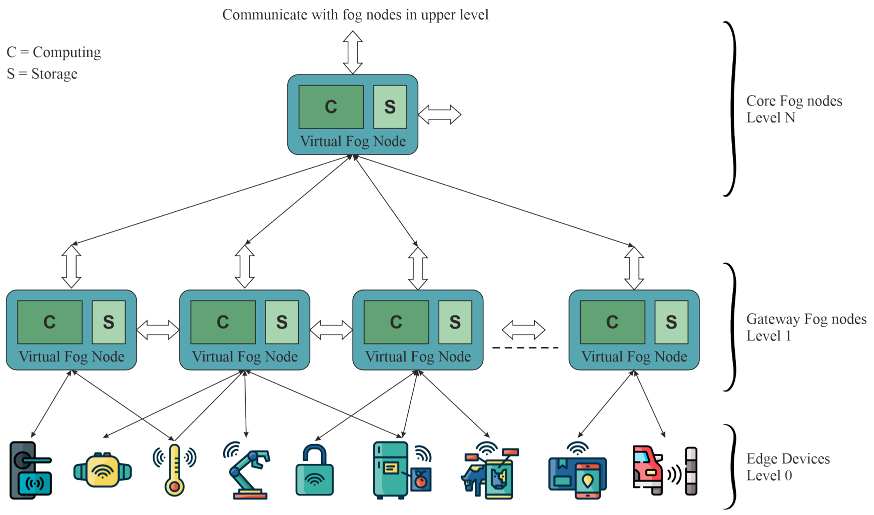


Figure 2.5: Hierarchical architecture of fog computing.

2.5 Fog Computing Architecture

Fog computing nodes deploy a virtualized and hierarchical topology and form a distributed computing platform. Nemirovsky (2012) introduced a structure for a physical fog node composed of multiple virtual fog nodes illustrated in Figure 2.5. Each physical node comprises computing and storage elements and has communication interfaces for communicating with adjacent fog nodes at the same, one step higher, or one step lower level of hierarchy. A virtual fog node also comprises computing, storage, and communication components and forms a multi-layered hierarchical structure. Consequently, a hierarchical architecture composed of several physical and virtual fog nodes forms the fog computing platform. In this paradigm, the virtual fog nodes are deployed as software agents that are composed of a virtual machine with the ability to run independently in different physical nodes. Figure 2.6 illustrates a hierarchical architecture of physical fog computing nodes on different levels (Nemirovsky, 2012).

2.6 Fog Computing Resource Management

Fog computing networks are dynamic and are formed based on resource-constrained devices. The dynamicity of the fog computing network is twofold. First, the service requests from heterogeneous IoT applications dynamically join or leave the fog computing networks as the edge devices in IoT could be mobile and travel from one geographical area to another. Second, fog computing nodes could be dynamically joining or leaving the network due to mobility of the nodes or limitations in energy sources for battery enable devices or sporadic wireless connectivity (Hosseinpour et al., 2018; Sahni et al., 2017). Such dynamicity and heterogeneity in both applications and platforms require smart and agile coordination through a resource management system to achieve optimal or near-optimal performance at run-time. Efficient utilization of the computing resources in this

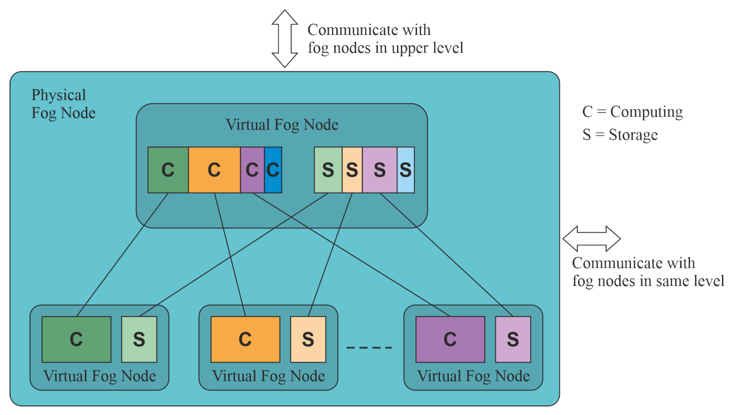


Figure 2.6: A physical fog node containing hierarchical virtual fog node.

platform will maximize the QoS and QoE. Whereas, an inefficient management of such resources-constrained platform will lead to even higher delays than sending the data for processing in the cloud (Rafique et al., 2019). Management of heterogeneous resources using virtualization technologies (e.g., VM, or container), faces different challenges compared to cloud computing as a result of the following limitations:

1. Communication deficiencies that are mainly because of heterogeneity of communication links, issues associated with low-bandwidth, intermittent, and less-reliable wireless networks, and limited communication resources, i.e., hardware interface (Wang et al., 2015b; Stojmenovic and Wen, 2014; Sanaei et al., 2014)

2. Limited available resources, (i.e,. computing, storage, memory, and power) in fog nodes (Sahni et al., 2017; Jiang et al., 2018; Dastjerdi et al., 2016)

3. Heterogeneity of fog computing nodes (Bonomi et al., 2012, 2014)

4. Dynamicity of the network caused by communication interruption, mobility of nodes, energy/power limitation and finally (Bonomi et al., 2012, 2014; Sanaei et al., 2014; Wang et al., 2015a)

5. Geo-distribution of the fog nodes (Bonomi et al., 2012, 2014)

Ghobaei-Arani et al. (2020) categorize the resource management approaches in fog computing into the following categories: application placement, resource scheduling, task offloading, load balancing, resource allocation, and resource provisioning. In the following section, we present a brief introduction to each of the techniques mentioned above and discuss the contribution of this thesis and the technique that we proposed for resource management in fog computing networks.

2.6.1 Application placement

IoT applications are typically composed of several sub-tasks (so-called a service) deployed in a virtualized environment that collaboratively provide the desired functionality following a Service-Oriented Architecture (SOA) (Ibrahim and Bench, 2017). In fog computing, optimal placement of services is crucial for maximizing the resource utilization of fog nodes and meeting the QoS requirements (Skarlat et al., 2017). The application placement problem in fog computing is a many-to-many mapping problem in which a set of services computing the IoT application needs to be mapped to a set of fog computing nodes considering a given set of constraints and a set of objective functions optimized (Selimi et al., 2019). In many studies, the targeted objectives in resource mapping are communication delays and energy efficiency (Rafique et al., 2019; Eyckerman et al., 2020; Goudarzi et al., 2021; Patro et al., 2021; Nashaat et al., 2020). In Publication V, we presented a mathematical model-based solution for resource allocation and application placement by formulating a Mixed-Integer-Quadratic-Programming (MIQP) problem minimising the communication delays in multi-task IoT applications.

2.6.2 Resource Scheduling

In resource scheduling, the main objective is to minimize a execution time of the given IoT service request. In this scenario, a set of sub-tasks or services in an SOA-based IoT application with given processing and QoS requirements need to be assigned for processing in a set of fog nodes with different computing and memory capabilities. The main objective of resource scheduling is to optimally schedule the execution of sub-tasks to minimize the application's total execution time. In general, the resource scheduling problem is an NP-hard optimization problem. Therefore, many studies have proposed metaheuristic algorithms to find a near-optimal solution to the problem (Potu et al., 2021; Ren et al., 2020; Wang and Li, 2019; Wu and Wang, 2019; Sun et al., 2018).

2.6.3 Task Offloading

In an IoT system in which the edge devices have limited computational power, battery energy, and storage space capabilities, the computation load could be offloaded to the fog computing nodes with richer computing capabilities, referred to as task offloading in fog computing. The main objectives of task offloading include avoiding service interruption, improving performance and throughput, and guaranteeing the QoS requirements. In this scenario, the computing-intensive tasks are often outsourced to fog or cloud computing. The application partitioning and generation of replicas of computation-intensive tasks and the decision to execute them locally or remotely in fog or cloud are done already in the edge device. According to Ghobaei-Arani et al. (2020), the task offloading can occur for different reasons such as load balancing, data management, latency management, security and, energy efficiency. Many studies have proposed model-based techniques such as queuing theory, game theory, and Lyapunov optimization to solve offloading problems (Bozorgchenani et al., 2017; Ahn et al., 2017; Deng et al., 2016; Liu et al., 2018; Ye et al., 2016; Urgaonkar et al., 2015).

2.6.4 Load Balancing

To enhance the responsiveness of the applications, one strategy is to evenly distribute the processing load to multiple distributed computing nodes, which is referred to as load balancing. However, However, due to the heterogeneity and dynamicity of the network, the load balancing in fog computing should be done based on the availability, processing capability, and the current load of the nodes to avoid processing bottlenecks by ensuring that no single node bears too much demand (Haghi Kashani and Mahdipour, 2022). Hence, load balancing faces challenges concerning network latency, system performance degradation due to continuous process migration, and lacking standards for describing possible scenarios (Ghobaei-Arani et al., 2020). The load balancing involves provisioning and de-provisioning instances of applications. Efficient load balancing in fog computing optimizes resource consumption, minimizes response time, and increases resource utilization (Chandak and Ray, 2019).

2.6.5 Resource allocation

The service request in an IoT system that incorporates both cloud and fog computing requires the allocation of resources in two separate distributing computing resource pools. Hence, the resource allocation in this scenario is a double-matching problem in that cloud/fog pairing for the given service request needs to take into account the constraints between the fog computing and the involved edge users in their coverage. Further, another way around, the fog/users paring needs to consider the constraints between the fog nodes and cloud servers. Hence, the resource allocation is NP-hard problem and is often formalized as approximation and heuristics to find near-optimal solution (Jiang et al., 2020; Lahmar and Boukadi, 2020; Lee and Lee, 2020; Mseddi et al., 2019; Gu et al., 2018; Jiao et al., 2019).

2.6.6 Resource Provisioning

Workload in IoT applications dynamically fluctuates over time. Such workload fluctuations result in over-provisioning, in which the allocated resources for a given IoT service are more than the actual load of that service, imposing unnecessary costs for the unused fog resource allocation. It may also result in under-provisioning, in which the allocated resources are less than the required processing power of the service in question to meet the quality of service and consequently leads to SLA violations. Thus, dynamic provisioning of the fog resources is necessary to minimise the service cost, optimize the resource utilisation and meet the QoS and SLA requirements (Shakarami et al., 2022; Ghobaei-Arani et al., 2020). In other words, the dynamic resource provisioning automatically adjusts when and how to allocate or release fog computing resources based on the real-time application requirements taking into account the QoS and SLA. Resource provisioning approaches are usually *reactive*, considering the run-time application requirement, or *proactive*, predicting the application requirement based on their computing trend (Santos et al., 2021a,b; Chandak and Ray, 2020; Etemadi et al., 2020; Santos et al., 2019).

2.7 Fog Computing Security and Privacy Issues

Mukherjee et al. (2017) argued that based on the specific characteristics of fog computing, i.e., mobility, heterogeneity, and large-scale geo-distribution, this platform faces different security and privacy challenges than its counterpart, i.e., cloud computing. Hence, the measures to cope with these issues differ from cloud computing. The authors classified fog computing security and privacy issues into following main categories:

2.7.1 Trust

Unlike cloud computing, fog computing is more flexible in leveraging different computing resources belonging to different parties. However, this flexibility adds more complexity in terms of trust management and security. Authentication and access control provide the foremost measure to establish an initial relationship between edge devices and fog computing nodes. However, solely relying on authentication and access control is not sufficient as the devices can malfunction or be compromised by malicious attacks. In such a scenario, trust fosters relations among the fog nodes and the edge devices based on previous interactions. Hence reputation-based models are widely used for trust management in IoT systems (Mukherjee et al., 2017). In Publication IV we presented a security framework for fog networks based on Role-Based Access Control (RBAC) and trust models. We defined different privileges and access levels for different types of participating fog computing nodes based on their trust level, which they gain based on their past reputation.

2.7.2 Authentication, Access Control and Secure Communication

To establish a secure connection, the edge devices should be authenticated to the fog network to access the resource pull in fog computing. However, the authentication and access control in fog computing faces challenges because of limited computing, storage, and power. Hence, the traditional authentication and access control methods based on Public Key Infrastructure (PKI) are not practical approaches for such resource-constrained devices. Further, performing authentication protocols in cloud servers, i.e., Remote Authentication (RA), is not a desirable solution due to the communication delays and single-point-of-failure problem for central cloud (Stojmenovic et al., 2016). To cope with these problems, hybrid and lightweight PKI are widely used, in which the initial connection is established with a shared key using public-key encryption, and then the subsequent communication is done by symmetric-key encryption with the established key (Stojmenovic et al., 2016).

2.7.3 End Users' Privacy

Preserving end users' privacy in fog computing is challenging because the fog nodes may collect privacy-sensitive data such as identities or usage patterns, locations, etc. For instance, as the edge nodes offload their processes to the nearest fog node, location, trajectory, and even mobility habits can be disclosed. Moreover, even with well-established security measures, critical privacy information can be exposed through side channels such as electromagnetic radiation, observation of certain activities timing, specific devices' power usage, and even light acoustic or heat emanations from equipment (Koo and Hur, 2018; Mukherjee et al., 2017).

2.7.4 Malicious Attacks (Intrusions)

Due to their specific characteristics, Fog computing faces different challenges than cloud computing in combating malicious attacks. On the one hand, the proximity to the end-users lowers the risks of routing attacks. On the other hand, with the limited computing resources in fog nodes, establishing strong security measures is not feasible, resulting in vulnerability to attacks by malicious entities on the network (Sadaf and Sultana, 2020). Furthermore, in a dynamic fog network in which new nodes join or leave the computing pool, most connected devices may not be mutually authenticated with the corresponding fog nodes making Denial-of-Service (DOS) attacks trivial. For example, a compromised device may request a frequent processing or storage request causing network congestion and delaying requests made by legitimate devices (Mukherjee et al., 2017). Intrusion Detection Systems (IDS) are potent mechanisms for detecting unforeseen intrusions in fog computing.

2.8 Intrusion Detection Systems

Detection of silent attacks in fog computing demands continuous monitoring and behavioural analysis of the system's internal processing components and communication within the network. Consequently, accurate and swift security monitoring and intrusion detection are essential to improve the security of fog networks. An intrusion detection system with local decision making will preclude failures caused by malicious attackers and, with a proper alert, prevent intrusion or mitigate its impact. Two common types of IDS include signature-based and anomaly-based IDS. Misuse-based or signature-based types of IDS are used to detect known attacks using predetermined signatures or attack patterns of similar attacks. By contrast, anomaly-based IDS are broadly used as defensive techniques in distributed network systems to address the detection of unknown or zero-day attacks that are unknown to the system. This process is done by monitoring and detecting the variations in the systems' behavior from a previously defined normal system profile.

2.8.1 Artificial Immune Systems

The artificial immune system (AIS) incorporates biologically inspired auspicious techniques to solve various problems in the information security field. The AIS is inspired by the Human Immune System (HIS), which is capable of differentiating internal cells and molecules of the body from foreign pathogens, so-called self and non-self, respectively, thus protecting the body against diseases (Aliyu et al., 2021; Leandro and Timmis, 2002). In the human body, the HIS mainly does this without any prior knowledge of attacking pathogens or their structure. Self and non-self discrimination is a significant attribute in

the AIS, which is utilized in developing efficient anomaly-based IDS (Brown and Anwar, 2021; Forrest et al., 1997). Similarly to HIS, which protects the human body against foreign pathogens, the AIS suggests a multi-layered protection structure for protecting computer networks against adversaries' attacks (Pamukov, 2017). This protection is accomplished through *Innate* or *Adaptive* mechanisms. Innate immunity is immediate; it is the first line of defence for the HIS and provides non-specific protection. Therefore, it has no prior knowledge of specific outsiders. The adaptive immune response, by contrast, is antigen-specific and is trained using a pre-defined profile of specific attacks (Ademokun and Dunn-Walters, 2010). Adaptive immunity also includes a "memory" that makes future responses against a specific antigen more efficient (Ademokun and Dunn-Walters, 2010). As with other anomaly-based detection techniques, the AIS also takes advantage of monitoring variations of the system's behavior according to a pre-defined normal activity profile as an adaptive immune mechanism. This is done through a learning phase in which a data set containing these profiles is utilized for this purpose. Hence, the efficiency of anomaly detection in the AIS is highly dependent on the learning data set.
Substantial research has been conducted so far on the enhancement and the utilization of AIS-based IDS. Most existing studies utilize a pre-defined and offline dataset as learning data for training the IDS. However, this will reduce the efficiency of the IDS by limiting its knowledge base to that particular learning data set. Moreover, it is extremely difficult to create a data set of self and non-self samples with all variations.

2.9 Chapter Summary

In this chapter, we presented the background of this thesis research work base on the state-of-the-art literature. Then, we discussed IoT architecture and its computing solutions. Next, we introduced fog computing as a promising technology to enhance the cloud computing shortcomings for delay-sensitive IoT applications. We then discussed the resource management, security and privacy issues in fog computing and the technological solutions in the literature to cope with those issues. In the next chapter, we will present the publication profile of the thesis and outline each publication's primary content, the research results and impact, the methodology used in the publication, and the research objective and questions addressed in each publication.

3 Publication Overview

This chapter presents an overview of the publications included in the dissertation portfolio, representing the research motivation, main context, relation to the research objectives and corresponding research questions, as well as findings and contributions.

3.1 Publication Outline

The dissertation portfolio consists of five publications covering two main topics, i.e., security and resource efficiency in fog computing. Table 3.1 presents an overview of the publication portfolio of the thesis, including the research methodology used in each publication and the research question addressed by the publication. Although a number of research publications were published during this thesis, only the most relevant publications focusing on the main research questions are included in the publication portfolio.

Table 3.1: Association of research questions with publications and research methods.

Publication	Title	Research Questions
Publication I	Artificial Immune System Based Intrusion Detection: Innate Immunity using an Unsupervised Learning Approach	How can an anomaly-based intrusion detection systems be enhanced to detect zero-day attacks?
Publication II	An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach	How can distributed intrusion detection systems be adapted in a lightweight form for fog computing to protect IoT services against silent and zero-day attacks?
Publication III	An Approach for Smart Management of Big Data in the Fog Computing Context	How can we decouple the dependency of application codes from the underlying hardware computing platform in a highly dynamic fog computing network to efficiently tackle the big data in IoT?
Publication IV	A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models	How can the security of IoT services be enhanced in cooperative fog computing networks?
Publication V	A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks	How can fog computing resources be optimally allocated to multitask IoT applications to reduce communication delays and optimise resource utilization in fog computing networks?

3.2 Publication I

Title: Artificial Immune System Based Intrusion Detection: Innate Immunity using an Unsupervised Learning Approach

3.2.1 Research Objective and Relation to the Whole Thesis

Due to geographical distribution, fog computing networks are more vulnerable to network attacks than central cloud computing. Therefore, protection against network intrusion is

a critical issue that could affect the availability, security and privacy of the computing services on this platform. To empower the fog computing networks, we started our research by investigating the most effective intrusion detection approach. However, the effectiveness of most of the existing IDS techniques suffered from dependency on the training dataset. Hence, the first research objective of this thesis was to provide a solution to improve the effectiveness of anomaly-based IDS by providing online training to detect zero-day attacks. Consequently, the following research question was formulated and answered in publication I:
How can an anomaly-based intrusion detection systems be enhanced to detect zero-day attacks?

3.2.2 Rationale and Context

Publication I presents an enhancement for the anomaly-based intrusion detection systems based on biologically inspired AIS techniques. AIS is a supervised machine learning approach that is capable of distinguishing abnormal behavior of the monitored system (called *non-self*) from the normal behavior (called *self*). The main drawback of many anomaly-based intrusion detection systems is relying on training labelled datasets sets to train the system against similar attacks to the labelled samples. Consequently, the efficiency of such IDS depends heavily on the comprehensiveness of the training data sets. Nevertheless, the IDS will not be able to distinguish zero-day attacks that they are not trained against.
This publication introduces a new and novel line of defence to complement the AIS, which we call *Innate Immunity*, by inspiring the HIS. HIS provides protection against foreign pathogens in the human body through *Innate* or *Adaptive* immune mechanisms. Innate immunity is the most immediate immune response and the first line of defence providing non-specific protection, meaning it has no prior knowledge of specific outsiders. By contrast, the adaptive immune response is antigen-specific and needs to be trained using a pre-defined profile of specific attacks. For example, the antibodies produced after exposure to a virus or a vaccine result from an adaptive immune response in the body. Hence, adaptive immunity also includes a “memory” that makes future responses against a specific antigen more efficient. AIS provides protection similar to that of the adaptive immune system in HIS by taking advantage of monitoring variations of the system’s behavior according to a pre-defined normal activity profile. In this publication, we introduced the innate immunity in AIS to address AIS-based IDS’s limitations.

3.2.3 Research Methodology

As a proof of concept, we used unsupervised learning methods to introduce innate immune responses to traditional AIS to reach this goal. Among several unsupervised machine learning approaches for detecting security attacks, clustering methods have proven effective. This publication examined the two most popular clustering algorithms, i.e., Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and K-means. We then fed the output of the unsupervised learning machine to our AIS engine to provide online training and empower the AIS to detect zero-day attacks. We also proposed a network measurement formula which dynamically calculates the threshold for clustering algorithms. To increase the accuracy of the clustering threshold and spot the changes in the normal network behavior, we calculated the standard deviation of the number of network flows in different windows from the last minute of the network traffic. We examined DBSCAN and K-means for forming the innate immune system. To evaluate the efficiency of the proposed method, we utilised KDD-Cup 99 dataset, which extracted from the DARPA-98 traffic network. In our experiments, the number of samples in the dataset was 22545 samples, sufficient for performing the evaluation.

3.2.4 Main Results and Contribution

In our evaluation, we measured the False Positive Rate (FPR), True Negative Rate (TNR), Accuracy (ACC), Recall (REC), Precision (PREC) and F1 score. According to our experiment, the DBSCAN algorithm outperformed K-mean by demonstrating a better detection rate. Table 3.2 presents the evaluation results and comparison of two clustering algorithms mentioned above for online learning in the innate immune mechanism.

Table 3.2: Comparison of two clustering methods for forming innate immune mechanism in AIS.

Algorithm	**FPR**	**TNR**	**ACC**	**REC**	**PREC**	**F1**
DBSCAN	0.008	0.991	0.771	0.589	0.987	0.738
K-Mean	0.156	0.843	0.607	0.431	0.788	0.557

To cluster the incoming traffic into self and non-self at run-time, our clustering engine constantly compares the number of network flows in different network resolutions (i.e., subnets of /0, /8, /16, /24) against a threshold that is dynamically calculated by our proposed network measurement formula. At run-time, we feed the output of the clustering engine, in the form of labelled real-time data, as the innate immune response, into the training engine of AIS to provide the AIS with online training against the zero-day attacks. However, the AIS system still uses offline training with some labeled datasets in a training phase to detect similar attacks to that of labelled samples. The online training through the innate immune system empowers the AIS for the new attack types that may not exist in the training samples.

3.3 Publication II

Title: An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach

3.3.1 Research Objective and Relation to the Whole Thesis

Intrusion detection is a computing and communication-intensive and delay-sensitive operation. Hence, most existing intrusion detection systems are not a practical solution for

resource-constrained edge and fog environments. To tackle this problem, lightweight intrusion detection systems tailored to the unique characteristics of IoT and fog networks must be developed to protect the edge devices and services in IoT systems against adversaries. However, due to the limited resources of the edge devices, the efficiency and performance of IDS are highly affected by its architectural design and implementation. Therefore, the second objective of this research was to design a lightweight IDS to enhance the security of IoT services. Consequently, the following research question was formulated and answered in publication II.
How can distributed intrusion detection systems be adapted in a lightweight form for fog computing to protect IoT services against silent and zero-day attacks?

3.3.2 Rationale and Context

Publication II presents an architecture for lightweight distributed IDS for IoT by utilizing the method we proposed in publication I. In this Publication, we considered IoT-based logistic systems and food delivery chains as our use case. Logistics is one of the main elements in a supply chain, and guaranteeing the security of logistic systems has become a crucial challenge at the global level. In particular, food safety and security have become the primary concerns for manufacturers and end-users. With the extensive digitalisation of supply chain management systems and adoption of IoT, new security threats increase cyber security concerns for supply chain and logistic systems. Therefore an IDS is required to tackle cyber threats in logistic systems.
We proposed a lightweight, multi-layered, multi-agent architecture for the distributed IDS in an IoT-based logistic system with a three-layered architecture. In this architecture, because the training phase in the AIS involves more computing-intensive processes than communicating-intensive processes, we deploy the AIS training engine in the cloud layer to train its detectors through the offline and online training approaches presented in publication I. Analyzing the intrusion alerts is delay-sensitive, requires more swift processing, and is a communication-intensive operation. Thus, we deployed the analyser agents of AIS at the fog layer. Furthermore, we deploy our detector agents as sensors for our IDS which monitor the behavior of the edge devices. Once any of these sensors detects an anomaly, they initiate a process for investigating the suspected abnormality by creating a Smart Data cell, a lightweight and active data bundle, containing the information about the suspected anomaly in its payload and the profile of the triggered detector in its metadata. We introduced the Smart Data concept in publication III. The Smart Data is sent to the fog layer for further analysis if a number of triggered detects for the same operation exceeds a predefined threshold to avoid false-positive errors. However, the IDS might fail to detect the silent attacks in this case. Because, unlike the short-term attacks, the silent attacks are launched over a more extended period and from distributed nodes while keeping the system's functionality as normal as possible to make it complex and more sophisticated to detect. Thus, the silent attacks may trigger fewer detectors, and the IDS may fail to detect threats. To tackle this problem, we store the suspected abnormal operation profile in the Smart Data cells over time, keeping them under monitoring. Therefore, if the number of triggered detectors over time exceeds a predefined threshold, the Smart Data cell is sent to

the fog layer for further analysis. The Smart Data cell is aggregated with other Smart Data cells coming from different edge devices in the fog layer if similar abnormal behaviors are reported in other devices. In case more comprehensive information is available for the analyser agent, which contributes to the enhancement of detection accuracy.

3.3.3 Research Methodology

Lightweight intrusion detection systems are crucial components for IoT systems to protect the services. In this publication, we presented a new lightweight architecture for AIS-based IDS for IoT systems by extending our proposed approach in Publication I. Our design leverages the multi-layer architecture of IoT systems by offering a three-layered, multi-agent and lightweight architecture for IDS based on AIS. We also utilized our proposed Smart Data approach for developing a lightweight and efficient intrusion analyzer for our proposed IDS. We also presented a new efficient clustering approach for training our IDS against silent and zero-day attacks. We improved the clustering engine using the DBSCAN algorithm presented in the publication I, to detect the silent attacks in the busy users' or server network flow by comparing the new network flow from these users with their previous profiles recorded earlier. If there is a deviation in their behavior of more than a threshold, the clustering engine will mark the new behavior as abnormal.

We utilized the KDD-Cup 99 data set extracted from the DARPA-98 traffic network and SSH Brute Force attacks from the ISCX dataset to evaluate our proposed architecture. As a proof of concept, we compared the efficiency of the new proposed clustering algorithm with the algorithm we presented in the Publication I.

3.3.4 Main Results and Contribution

According to the experimental results, our IDS was able to accurately cluster and label the normal and abnormal behaviors in the online training phase and detect similar attacks in the detection phase. We presented a comparison between the new proposed clustering algorithm and the algorithm presented in the Publication I. Our new proposed model outperforms our previous work in terms of false positive rate, true negative rate, accuracy, recall, and precision. Table 3.3 compares our new proposed method with the method we proposed in Publication I.

Table 3.3: Clustring methods proposed in Publication I and Publication II

	Proposed model in publication II	***Proposed model in publication I***
False positive rate	3.51%	4.53%
True negative rate	96.49%	95.47%
Accuracy	98.35%	96.23%
Recall	100%	95.3%
Precision	97.83%	91.21%

3.4 Publication III

Title: An Approach for Smart Management of Big Data in the Fog Computing Context

3.4.1 Research Objective and Relation to the Whole Thesis

In a highly dynamic and heterogeneous fog computing platform, deploying the required application code to process IoT Big Data is a challenging problem. Especially in the case of mobile sensors that move from one edge location to another, the local processing of data requires migration and deployment of the service to the new fog nodes within the proximity of the sensors' location to avoid communication delays in forwarding the data to the nodes that the service was initially deployed. In this case, deploying all the applications in all the fog nodes is inefficient in a resource-constrained fog computing network. Moreover, there are interoperability issues in deploying the application in heterogeneous fog nodes with different hardware architectures and operating systems. Hence, the IoT applications should be independent of their hardware platform so that they easily migrate to the new edge locations without service interruption due to their deployments. Consequently, the following research question was formulated and answered in publication III:
How can we decouple the dependency of application codes from the underlying hardware computing platform in a highly dynamic fog computing network to efficiently tackle the big data in IoT?

3.4.2 Rationale and Context

Publication III presents a new concept called Smart Data, which aims to tackle the Big Data in IoT systems by reshaping the raw and passive form of IoT data into intelligent and self-managed data cells. A Smart Data cell is a package of structured data generated by IoT sensors and a set of metadata encapsulated in a container. The metadata stored in the Smart Data cell incorporates a set of rules that define its behavior and govern its security, privacy and other functionalities that control and manage it. The container, in turn, executes the rules set in the metadata to achieve its intended evolution stage. As a result, Smart Data can evolve and become more meaningful information with reduced size. Figure 3.1 illustrates the architecture of a Smart Data cell.
Throughout the Smart Data lifecycle, the payload component undergoes a series of preprocessing steps, such as aggregation, filtering, compression, encryption thus converting it into more meaningful information. The metadata component retains key information and rules such as data source (sensors), the physical entity to which data belongs, timestamps, status and logs, rules for accessing and aggregating, and required steps for processing its payload. The metadata also stores the information extracted by processing the payload. Such information becomes more accurate and meaningful when the Smart Data cell is aggregated by other Smart Data cells, or receives data over a more extended period from its corresponding sensor. The container of Smart Data operates as a platform that fosters the management and execution of the rules specified in the metadata part. The container of the Smart Data has a modular structure in which different task modules of the IoT

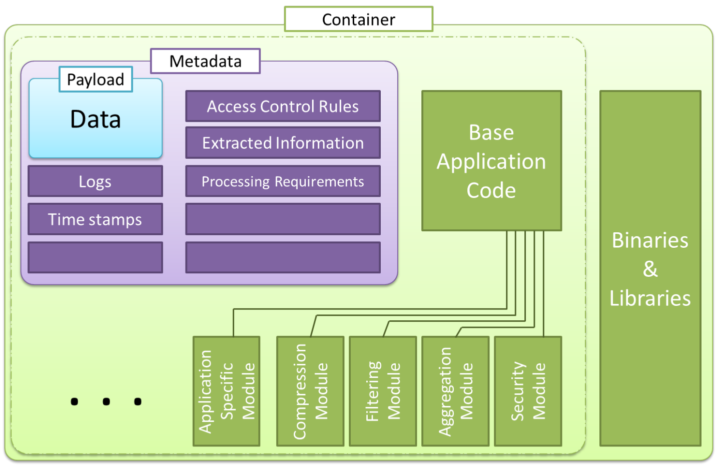


Figure 3.1: Smart Data architecture.

application to plugged in or removed on demand. This facilitates the Smart Data to be lightweight and scalable, allowing it to manage the overhead of carrying the code by removing unnecessary code modules and adding the required modules only when they are needed.

3.4.3 Research Methodology

In this publication, we designed and conceptualized the Smart Data concept by inspiring biological cell aggregation. We used containerization and active bundle technology (Othmane and Lilien, 2009) to design the structure of the Smart Data.

3.4.4 Main Results and Contribution

This publication was the first publication in the resource efficiency track of this thesis that focus on tackling the application deployment problem in dynamic and heterogeneous fog computing networks. The proposed method in this publication decouples the dependency of the application codes from the underlying hardware infrastructure, facilitating the execution of codes within a virtualized bundle capable of execution on different hardware or operating system settings. The proposed solution was a step toward revolutionizing the current perspective of data in IoT, opening many potential research opportunities to tackle emerging Big Data issues.

3.5 Publication IV

Title: A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models

3.5.1 Research Objective and Relation to the Whole Thesis

Cooperative computing in fog networks faces challenges with respect to managing trust and access permissions of arbitrary devices for the critical data of IoT services. Therefore a robust security framework must be in place to enhance the security of such cooperative fog computing. To address this goal, the following research question was formulated and answered in publication IV.

How can the security of IoT services be enhanced in cooperative fog computing networks?

3.5.2 Rationale and Context

Publication IV presents a security framework for managing the trust and access control for for computing nodes that newly join the fog computing network. Since fog computing suffers from limited resources, increasing the computing capabilities is a significant challenge in improving the QoS. To tackle this issue, in this publication, we proposed cooperative computing by leveraging the available and unused processing and storage capacities of surrounding smart devices. We argued that based on the literature, except for the peak time between 11:00 to 17:00, the average processor usage in personal smart devices is below 50%. This amount drops to less than 20% during the nighttime between 00:00 to 8:00.

With this motivation, we proposed utilizing the available resources of different smart devices to enhance the fog computing capability. However, utilizing such arbitrary untrusted devices will impose security risks. Hence, in this publication, we presented a combined method using reputation-based trust models and a role-based access control model (ABAC) to enhance the security of IoT services executed in these devices. We consider three types of fog nodes based on their processing capacities and trust levels. We also consider different levels of criticality for the data being processed. Once a new node joins the computing pool, it will be entitled to the lowest level of trust and the least access privileges. Nevertheless, over time, the node can gain higher levels of trust and access privileges based on its reputation for successfully executing the given assignment. Or in contrast, they may lose their trust or even be revoked from the computing network if their transaction reputation worsens.

3.5.3 Research Methodology

We calculated the trust score of each device based on important security parameters, including availability, reliability, data integrity, and turnaround efficiency. In addition to gaining higher trust scores to promote a node, we enforced the condition based on demands for nodes with higher trust levels to contribute to processing critical data. The trust level defines a role in this model, and the role defines the access privilege to data with different levels of criticality.

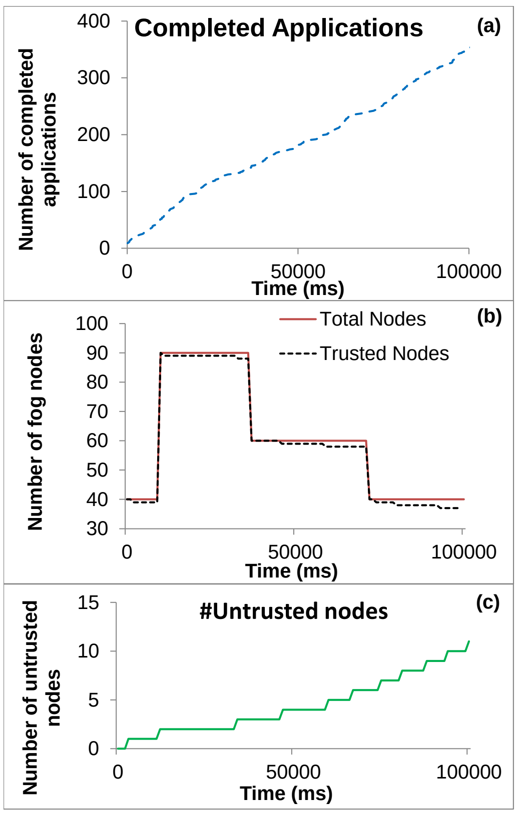


Figure 3.2: Experimental results of proposed framework in Publication IV.

3.5.4 Main Results and Contribution

We implemented the fog computing platform in the SystemC environment by modelling each processing device as a node with the capability of communicating with all other elements in its domain. We incorporated heterogeneous nodes with different processing capabilities into our platform. We introduced applications with distributed task graph models that dynamically enter and leave the fog system. Continuously, a dynamic node group joins and leaves the fog system, and the computing tasks are assigned to them. Based on the proposed model, we calculated the trust level of each dynamic node to assign more critical tasks. Based on our experiment, the system was able to promote some nodes based on their trust score and the demand for the new nodes at that level. Furthermore, some of the nodes that were unable to execute the tasks successfully and failed to improve their trust score were dropped from the computing pool. Figure 3.2 presents the experimental result of our proposed framework.

3.6 Publication V

Title: A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks

3.6.1 Research Objective and Relation to the Whole Thesis

Efficient utilization of the computing resources in fog computing is essential for enhancing the QoS and QoE. Inefficient scheduling and resource allocation of fog computing resources for IoT services can result in even higher delays than sending the data for processing in the cloud. High dynamicity and heterogeneity in both applications and network demand intelligent and agile coordination through a resource management system to achieve optimal or near-optimal performance at run-time. Furthermore, to deal with the ever-increasing complexity of IoT applications and ease their development, deployment, and management, multi-task applications and microservice architecture are gaining more popularity in the design and development of IoT applications. However, the allocation of fog computing resources to such distributed applications should be optimally done to avoid QoS and QoE degradation due to undesired communication delays between the components of multi-task applications. To address this goal, the following research question was formulated and answered in publication V.
How can fog computing resources be optimally allocated to multitask IoT applications to reduce communication delays and optimise resource utilization in fog computing networks?

3.6.2 Rationale and Context

Publication V presents a mathematical model for resource management in fog computing and service placement of distributed multi-task applications in that platform. As the number of connected *things* rapidly grows daily, the number of applications that provide services to the end users also grows. As a result, they also become more diverse and complex. Diversity is a desirable characteristic that is supported by the continuous digitalization of services. By contrast, the complexity is undesirable and results in several issues when scaling, upgrading, and inter-operating with other services. Traditional monolithic and centralized applications are not sufficiently feasible to cope with the application complexity (Butzin et al., 2016; Santana et al., 2021). Most IoT applications are developed as a collection of several functions that collaboratively provide the requested service by executing different operations on the incoming data stream from the sensors. Some of these operations, such as encryption, filtering, and visualization, are common are common in most IoT applications. Such common operations could be developed as standard task modules to reduce the complexity of applications. Butzin et al. (2016) argued that to handle the complexity in IoT, applications must be developed as small independent services. Developing value-added services necessitates combining best-of-breed services from different vendors to leverage the IoT's heterogeneity. They pointed out that service-oriented architecture (SOA) and microservice architecture follow this goal: *building one or multiple applications from a set of different services.*

Thus, conventional monolith applications can be developed as an ensemble of independent standard task modules with inter-dependencies forming multi-task applications. In a multi-task application, each task module performs a specific operation on the data, receives input data provided by precedent tasks, produces specific output data, and sends it to the subsequent tasks.
Offloading the multi-task application into distributed fog computing is a many-to-many mapping problem in which a set of fog computing resources should be allocated to a set of task modules of an application. Hence, in this publication, we presented a resource management model for allocating fog computing resources to multi-task IoT applications with the aim of reducing communication delays. In our model, we consider three different communications:

- The communication between the edge sensor and the gateway node that receives the data stream from that sensor
- The communication between the gateway node and the node in which the first task module of the application is deployed
- The communication between the set of fog nodes that host the task modules of the application.

Our goal in this publication was to reduce the communication costs and delay to enhance the QoS and QoE for IoT services running on fog computing. Figure 3.3 illustrates an example of mapping a multi-task application with a given task graph into a fog computing network. For simplicity and for the sake of illustration, we consider a network with a 4×4 grid-mesh network topology. The resource mapping algorithm needs then to select a group of fog computing nodes that can accommodate and process the containers of the task modules of the application, providing them with appropriate communication paths. In this example, the fog nodes $5, 10, 12, 13, 14$ are the selected nodes. Since fog nodes generally have limited resources in terms of storage capacity and processing power, some nodes cannot be chosen to host the containers of the task modules. These nodes, instead, can act as intermediate routing nodes. In the example, the nodes $6, 9, 15, 16$ are such routers.

3.6.3 Research Methodology

In this publication, we presented a mathematical model of multi-task applications and fog networks. We then formulated a mixed-integer-quadratic-programming (MIQP) optimization problem for mapping the task graph of applications into distributed fog computing network, considering different constraints with the objective of minimising the communications delays. We implemented our model in a commercial solver, i.e., Gurobi 9.1, to evaluate it and presented the key results.
However, since search space in the proposed model with MIQP optimization grows exponentially by adding more fog computing or sensor nodes to the system, this approach may not be feasible and efficient in the real-world scenario. Because in resource-constrained

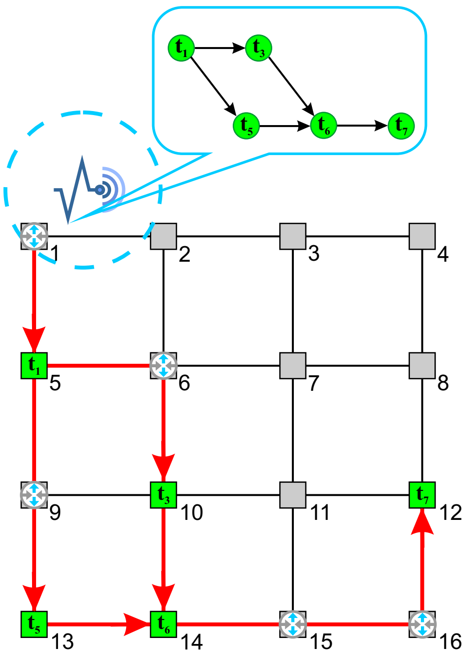


Figure 3.3: An example of distributed multi-task application mapping.

fog computing, dynamic decisions for resource mapping need to be done swiftly to avoid delays in service delivery time. Therefore, we also proposed a greedy algorithm as a heuristic method that guarantees a near-optimal solution that is considerably less computationally intensive and faster. Our proposed greedy algorithm demonstrated a highly scalable and near-optimal performance for resource mapping problems in fog computing networks for multi-tasking applications.

3.6.4 Main Results and Contribution

We validated the correctness of our proposed heuristic greedy-based algorithm by plotting a Cumulative Distribution Function (CDF) of both heuristics greedy and optimal algorithms by running 1000 simulation instances with different random seeds. Consequently, our greedy heuristic algorithm demonstrated near-optimal results, reaching on average, 93.2% of the optimal value.
To validate the performance of the proposed heuristic greedy-based algorithm, we compared the execution time for both optimal and greedy methods. Consequently, the heuristic greedy-based algorithm demonstrated a significantly lower and more predictable execution time than the optimal method, i.e., on average, 0.97% of the execution time required by Gurobi-based optimization.
To evaluate the scalability of our proposed heuristic greedy-based algorithm, we examined the CPU time and maximum used memory under different network settings. As a result, it was evident that our proposed heuristic greedy-based algorithm outperforms the Gurobi

solver by requiring significantly fewer resources, i.e., on average, 99.88% less CPU usage and 93.67% less memory usage. Furthermore, the resource usage for calculating the optimal solution in the Gurobi solver showed near-linear dependence on the number of sensors and a considerably steeper slope than the resource usage in the proposed greedy-based heuristic algorithm.

We also examined the effect of different network settings on the average total cost in both algorithms. As a result, the total cost generally increased when the number of sensor nodes increased. Further, the average total cost increases with the increased in arrival data rate.

4 Discussion

This chapter summarizes the dissertation findings, highlights research implications and proposes recommendations and further research directions on the topics covered in the dissertation.

4.1 Research Contributions

Motivated by the ever-increasing Big Data phenomenon and the demand for secure and efficient ways to manage and process such data in IoT domain, this dissertation posed the following research questions.

- *RQ1: How can an anomaly-based intrusion detection systems be enhanced to detect zero-day attacks?*
- *RQ2: How can distributed intrusion detection systems be adapted in a lightweight form for fog computing to protect IoT services against silent and zero-day attacks?*
- *RQ3: How can the security of IoT services be enhanced in cooperative fog computing networks?*
- *RQ4: How can we decouple the dependency of application codes from the underlying hardware computing platform in a highly dynamic fog computing network to efficiently tackle the big data in IoT?*
- *RQ5: How can fog computing resources be optimally allocated to multitask IoT applications to reduce communication delays and optimise resource utilization in fog computing networks?*

A set of research studies was conducted and documented in Publications I-V, employing various research methods to address security and resource efficiency challenges in fog computing from different perspectives to answer these research questions. In essence, the studies presented novel approaches to enhance and tailor the security mechanisms and optimize resource utilization in fog computing.
With the immense growth in the IoT application and digitization of the services, reliable and efficient computing platforms are crucial for successfully deploying and adopting these technologies. Based on the specific characteristic of IoT service architectures, fog computing was introduced to provide computing services already at the edge of the network through geographically distributed computing elements near the data source, thus reducing communication delays. However, fog computing faces challenges with regards to implementing strong security mechanisms because the computing elements in fog computing are resource constrained and therefore deploying strong security measures is often infeasible or inefficient. To fill this gap, this thesis, on the one hand, focused on designing and tailoring lightweight security solutions for detecting and preventing malicious intrusions to IoT services running in fog computing platforms and designing a lightweight yet efficient access control system, and on the other hand, designing resource management

systems to efficiently utilize the limited resource available in such platforms to facilitate the execution of services (including the security services).

In the first step toward our objectives, in the Publication I, we enhanced the efficiency of a biologically inspired artificial immune system-based intrusion detection system by introducing an extra layer of protection called innate immunity. We highlighted that the efficiency of any anomaly-based IDS heavily relies on the comprehensiveness of chosen training dataset. However, it is impossible to guarantee that the system will be trained against the new types of attacks that are not included in the training datasets. Therefore, the IDS fails to detect new emerging attacks that are known as zero-day attacks. To fill this gap, the innate immune system utilizes specially tailored unsupervised machine learning techniques to provide online training data which is able to cluster out the zero-day attacks.

However, intrusion detection includes processing and communication-intensive operations, which are not feasible to execute in the resource-constrained fog computing environment. To tackle this issue, in Publication II, we presented a new lightweight multi-agent-based architecture tailored to the specific characteristics of fog computing networks. In this architecture, the processing-intensive tasks are executed in cloud computing, and the communication-intensive tasks run in fog computing at the network's edge. We utilized the Smart Data technology that we presented in Publication III to enable lightweight detector agents at the edge devices without deploying the intrusion detection service. This will provide scalability and facilitate interoperability in different hardware and software platforms to execute intrusion detection operations without imposing too much processing load on the devices. We also enhanced the innate immune mechanism by introducing a new, improved clustering method using the DBSCAN algorithm.

As a countermeasure to tackle the limited resources in fog computing at the edge of the network, one possibility is to utilize the unused processing and storage capacity of smart consumer devices such as mobile phones, tablets. However, this imposes a security risk as the devices may not be trustable to process the critical information. To overcome this problem, in Publication IV, we proposed a framework based on trust management and role-based access control. With this framework, the fog platform can delegate some simple and non-critical tasks to new nodes joining the computing pool and evaluate their trust level based on the reputation that they earn for successfully delivering their processing assignments. Once a device gains a higher level of trust, it can be promoted to higher roles defined in the system and have access to processing more critical information.

As another step toward tackling resource limitations in fog computing, we proposed a resource management model in the Publication V, by mathematically modeling the fog computing system and developing mapping algorithms to map multi-task applications in fog computing networks efficiently. In addition to the optimal solution, we presented a greedy-based heuristic algorithm that demonstrated to be efficient, fast, and scalable while delivering a near-optimal solution for many-to-many mapping problems. Our proposed greedy algorithm requires significantly lower computing and memory resources compared to the optimal solution solved by a commercial Gurobi solver.

4.2 Implications and limitations for Research and Practice

In what follows, we discuss theoretical and practical implications and limitations of the dissertation.
To the best of our knowledge, the innate immunity in AIS was not studied before, and this study opened many research directions in the enhancement and optimization of AIS and its adoption in designing anomaly-based IDS. The proposed online unsupervised innate immune mechanism can be further used to complement and improve the training phase of any anomaly-based IDS, empowering them to detect zero-day attacks. However, distributed long-term attacks such as Distributed Denial of Service (DDOS) or botnet attacks can develop fake normal behaviour patterns and thus, they may not be detected by the clustering algorithms.
The proposed lightweight IDS architecture can be applied in various IoT systems monitoring anomalies, not only in the network and execution of the data in the IoT system but also in any abnormal behaviour detected from the physical entities that are monitored by the system. In this case, the proposed IDS can be applied to the digital twins of physical systems.
The proposed trust management and access-control framework can be applied to various domains such as wireless sensor networks, federated cloud computing and multi-access edge computing networks in which the existing computing pool needs to be extended on demand to address the service requirements.
The Smart Data concept has the potential to revolutionise the management of Big Data in IoT. With the ever-increasing complexity and diversity of IoT applications, the geographical expansion of services, increased heterogeneity of hardware, and operating systems, and, more importantly, mobility and dynamicity of applications and networks, the deployment of IoT applications will face challenges. The Smart Data aims to decouple the applications from the underlying hardware and bundle them into the data instead. This enables the scalability and interoperability of IoT applications while facilitating the management of data in a coherent way. However, from a practical perspective, moving the data in Smart Data bundles requires the migration of the container hosting the Smart Data. The migration of containers running stateful applications has some limitations that need further investigation.
The proposed resource management and application mapping model can be utilized in many IoT domains. Particularly, the heuristic greedy-based algorithm, which demonstrated significantly efficient and near-optimal resource mapping, could be used in critical applications that require real-time or near real-time resource mapping. However, in a real-world scenario, in addition to the communication delays, some other parameters, such as energy consumption, bandwidth utilization, deployment, and cost, should be considered in multi-objective optimisation for the resource mapping problems.

4.3 Proposals for Further Research

The current dissertation revealed some of the security and resource efficiency problems in fog computing and proposed solutions to tackle these problems. The dissertation opens and highlights many prospects for further research in the field. Below a summary of suggestions for further research is presented for each publication:

Findings in a Publication I and II provides valuable insight on approaches for developing lightweight and efficient IDS for IoT systems. However, there is a need for further research on enabling the clustering algorithms to detect and cluster long-term and silent attacks such as DDOS and botnet attacks. Additionally, to improve the efficiency of detection and required processing resources, utilizing other lightweight evolutionary algorithms such as Artificial Neural Networks (ANN), Particle Swarm Optimization (PSO), etc., could be studied for training the detectors of AIS.

The Smart Data concept is indeed in the early phases of conceptualization, and we believe that it has the potential to revolutionize the current perspective of Big Data management. However, running container migration faces some challenges when the running process is stateful. Therefore, we suggest studies be carried out on stateful container migration in the fog computing platform.

To further improve the proposed framework in Publication IV we suggest extending this framework by incorporating anomaly-based machine learning techniques to detect the malicious nodes joining the computation pool.

The communication cost inferred in the Publication V is an abstract parameter reflecting several other parameters such as delay, traffic, bandwidth, and energy. Therefore, there is a need to investigate further concrete aspects such as communication parameters when managing the resources in a real-world fog computing system. Furthermore, as part of future work, we intend to extend our proposed model by considering the communication bandwidth between network nodes to calculate realistic communication delays. The resource management model proposed in this publication can be extended and studied for microservice architecture applications in IoT. Deploying the microservice applications requires pre-deployment of the corresponding microservice in the fog node. Hence, the deployment of microservices in the fog node is one crucial parameter that needs to be considered in mapping problems. The deployment process, which includes cloning the container image from the nearest repository, building the container from the downloaded image and running the microservice, is often time-consuming, process intensive and energy consuming and imposes delays on the delay-sensitive applications. However, there is a trade-off between microservice deployment cost and microservice communication delays. Because deploying the microservices in the nearest nodes to reduce the communication delays may impose deployment costs. Or another way around, avoiding the deployment cost and running the services in the fog nodes that have the pre-deployed microservice may impose some communication delays as the nodes hosting the microservices of an application might be many hops away from each other. Thus, the resource mapping model should take into account the deployment cost in finding the optimum solution for mapping the set of microservices into fog nodes, i.e., deployment-aware resource management.

Moreover, the resource management model proposed in this publication could be utilized to improve the efficiency and throughput of our proposed intrusion detection system in publications I and II. In this case, the detectors agents deployed in Smart Data bundles would be optimally mapped into a set of fog nodes that collaboratively process the intrusion alerts with minimal communication latency.

5 Summary and Conclusion

Continuous technology advancement in emerging IoT and IoE technologies faces challenges concerning the management, storage and processing of Big Data. Relying on current technologies such as cloud computing is insufficient for addressing Big Data management requirements. Hence, new technologies are required to reduce complexity, ease management, and boost Big Data processing. To this end, the fog computing platform at the edge of the network is introduced to reduce the processing load from the cloud by delegating some simple and frequent tasks from the cloud. However, unlike cloud computing, the fog computing platform is distributed and composed of devices with limited resources. As a result, implementing robust security mechanisms to protect the network is not feasible. To cope with this issue, lightweight security mechanisms must be tailored and designed according to fog computing characteristics. In addition, the resources in fog computing need to be managed efficiently to support the execution of different IoT applications and security services.
This dissertation has studied several approaches to tackle the above-mentioned problems, including:

- design and development of a lightweight intrusion detection system,
- development of a security framework based on trust models and a role-based access control system to facilitate the extension of computing resources in fog computing
- reshaping the passive form of data into active data bundles called Smart Data
- development of resource management models and algorithms to efficiently utilise fog computing resources.

Relying only on offline data sets limits the ability of IDS to detect the new emerging attacks known as zero-day attacks. Enriching the training data is one way to improve the efficiency of anomaly-based IDS. Unsupervised machine learning techniques are promising technology for clustering and labeling the network traffic in near-real time. Hence the output for unsupervised clustering could be used to enrich the anomaly-based IDS. In this dissertation, we tailored the DBSCAN clustering algorithm for clustering the network traffic to mark the abnormal connections. Furthermore, we utilized the unsupervised clustering algorithm to introduce a new protection layer to AIS-based IDS called innate immunity. Innate immunity is the first response to abnormal network behaviour, and a more thorough investigation is done by the AIS main engine that resembles the secondary and adaptive immune responses in the human immune system. We also introduced a new lightweight architecture for this IDS leveraging the IoT three-layer architecture. In our architecture, training, and analysis, which are more process intensive, occur in the cloud, and detection, which is collaborative and communication intensive, happens at the edge of the network in fog computing nodes to avoid communication delays. This enhances the efficiency of IDS, as the training agents are lightweight and do not consume too much of the nodes' resources. Moreover, to overcome the deployment cost of the detectors in the nodes, we utilize the Smart Data bundles deployed in containers able to detect intrusion and pre-process the intrusion alerts collaboratively.

To extend the processing capability of fog computing, one approach is to utilize the unused processing capacity of surrounding smart devices. In this case, the new nodes joining the computing pool must be monitored to ensure the security and privacy of data. Reputation-based trust models are proven to be effective in monitoring and evaluating the behavior of devices in a collaborative computing environment. However, these approaches must be complemented with other security measures, such as anomaly-based machine learning techniques, to detect malicious actions in the given assignments to the arbitrary fog nodes.

With the increasing complexity and diversity of IoT applications on the one hand and high mobility, heterogeneity and dynamicity of edge computing platforms, including fog computing, on the other hand, deployment of IoT services in geographically distributed edge environments faces challenges. We believe that to overcome this issue, the data and processing code should be decoupled from the underlying hardware infrastructure, and the data must be self-organizing and self-managed. To achieve this goal, we introduced a concept called Smart Data, which aims to revolutionize current passive forms of data into smart, self-managed, active data bundles. Although Smart Data is in very early phases of conceptualisation, it has the potential to solve many issues raised by the complexity of IoT realm.

Finally, another premise to consider in the complexity of IoT is to standardize the common operation on the data. Traditional monolithic and centralized applications are not sufficiently feasible to cope with the application complexity. In this case, the applications must be developed as a collection of small independent services following a SOA, and microservice architecture. However, efficient allocation of limited resources of fog computing to the set of tightly collaborative tasks forming an application is a crucial problem in ensuring successful deployment of services and for increasing QoS. Therefore, the problem could be further broken down into many-to-many mapping problems. In this dissertation, we modelled multi-task application mapping in a fog computing network and formulated an optimum MIQP problem. However, the search space for such an algorithm grows exponentially by adding more nodes and sensors. Therefore, even advanced mathematical solvers cannot process the problem reasonably fast. Hence, we proposed another heuristic algorithm based on a greedy principle, which proved to be lightweight, fast (cutting the execution time to less than 1% of the execution time of the commercial Gurobi optimiser executing the optimal MIQP model), highly scalable, and near-optimal (i.e., 93%).

References

Ademokun, A.A. and Dunn-Walters, D. (2010). Immune Responses: Primary and Secondary. In: *eLS*. John Wiley Sons, Ltd. ISBN 9780470015902.

Ahn, S., Gorlatova, M., and Chiang, M. (2017). Leveraging fog and cloud computing for efficient computational offloading. In: *2017 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pp. 1–4.

Al-Qaseemi, S.A., Almulhim, H.A., Almulhim, M.F., and Chaudhry, S.R. (2016). IoT architecture challenges and issues: Lack of standardization. In: *2016 Future Technologies Conference (FTC)*, pp. 731–738.

Aliyu, F., Sheltami, T., Deriche, M., and Nasser, N. (2021). Human Immune-Based Intrusion Detection and Prevention System for Fog Computing. *Journal of Network and Systems Management*, 30(1), p. 11. ISSN 1573-7705.

Ashton, K. et al. (2009). That â€˜internet of thingsâ€™ thing. *RFID journal*, 22(7), pp. 97–114.

Banica, L., Stefan, C., and Hagiu, A. (2017). Leveraging the Microservice Architecture for Next-Generation Iot Applications. *Scientific Bulletin : Economic Sciences*, 16(2), pp. 26–32. ISSN 1583-1809.

Bhawiyuga, A., et al. (2019). Architectural design of IoT-cloud computing integration platform. *Telkomnika*, 17(3), pp. 1399–1408.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. In: *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*, vol. 546, pp. 169–186. Studies in Computational Intelligence. Springer International Publishing. ISBN 978-3-319-05028-7.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog Computing and Its Role in the Internet of Things. In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pp. 13–16. New York, NY, USA: ACM. ISBN 978-1-4503-1519-7.

Bozorgchenani, A., Tarchi, D., and Corazza, G.E. (2017). An Energy and Delay-Efficient Partial Offloading Technique for Fog Computing Architectures. In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6.

Brown, J. and Anwar, M. (2021). Blacksite: human-in-the-loop artificial immune system for intrusion detection in internet of things. *Human-Intelligent Systems Integration*, 3(1), pp. 55–67. ISSN 2524-4884.

Butzin, B., Golatowski, F., and Timmermann, D. (2016). Microservices approach for the internet of things. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–6.

Chandak, A. and Ray, N.K. (2019). A Review of Load Balancing in Fog Computing. In: *2019 International Conference on Information Technology (ICIT)*, pp. 460–465.

Chandak, A.V. and Ray, N.K. (2020). Multi Agent Based Resource Provisioning in Fog Computing. In: *Trends in Computational Intelligence, Security and Internet of Things*, pp. 317–327. Springer International Publishing. ISBN 978-3-030-66763-4.

Dastjerdi, A., et al. (2016). Fog Computing: principles, architectures, and applications. *Internet of Things*, pp. 61–75. ISSN 0020-0255.

Datta, S.K., Bonnet, C., and Haerri, J. (2015). Fog Computing architecture to enable consumer centric Internet of Things services. *Proceedings of the International Symposium on Consumer Electronics, ISCE*, 2015-Augus, pp. 6–7.

Deng, R., et al. (2016). Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal*, 3(6), pp. 1171–1181.

Dragoni, N., et al. (2018). Microservices: How To Make Your ApplicationÂ Scale. In: Petrenko, A.K. and Voronkov, A., eds, *Perspectives of System Informatics*, pp. 95–104. Springer International Publishing.

Etemadi, M., Ghobaei-Arani, M., and Shahidinejad, A. (2020). Resource provisioning for IoT services in the fog computing environment: An autonomic approach. *Computer Communications*, 161, pp. 109–131. ISSN 0140-3664.

Eyckerman, R., Mercelis, S., Marquez-Barja, J., and Hellinckx, P. (2020). Requirements for distributed task placement in the fog. *Internet of Things*, 12, p. 100237. ISSN 2542-6605.

Forrest, S., Hofmeyr, S.A., and Somayaji, A. (1997). Computer Immunology. *Commun. ACM*, 40(10), p. 88â€“96. ISSN 0001-0782.

Ghobaei-Arani, M., Souri, A., and Rahmanian, A.A. (2020). Resource Management Approaches in Fog Computing: a Comprehensive Review. *Journal of Grid Computing*, 18(1), pp. 1–42. ISSN 1572-9184.

Goudarzi, M., Wu, H., Palaniswami, M., and Buyya, R. (2021). An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments. *IEEE Transactions on Mobile Computing*, 20(4), pp. 1298–1311.

Grossetete, P. (n.d.). IoT and the Network: What is the future? *Cisco*.

Gu, Y., et al. (2018). Joint Radio and Computational Resource Allocation in IoT Fog Computing. *IEEE Transactions on Vehicular Technology*, 67(8), pp. 7475–7484.

Haghi Kashani, M. and Mahdipour, E. (2022). Load Balancing Algorithms in Fog Computing: A Systematic Review. *IEEE Transactions on Services Computing*, 1374(c), pp. 1–18.

Hosseinpour, F., Amoli, P.V., Plosila, J., and Hämäläinen, T. (2016). An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach. *JDCTA nternational Journal of Digital Content Technology and its Applications*, 10, pp. 34–46.

Hosseinpour, F., Siddiqui, A.S., Plosila, J., and Tenhunen, H. (2018). A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models. In: *Research and Practical Issues of Enterprise Information Systems*, pp. 168–180. Springer International Publishing. ISBN 978-3-319-94845-4.

Ibrahim, N. and Bench, B. (2017). Service-Oriented Architecture for the Internet of Things. In: *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1004–1009.

Jiang, J., Tang, L., Gu, K., and Jia, W. (2020). Secure Computing Resource Allocation Framework For Open Fog Computing. *The Computer Journal*, 63(4), pp. 567–592. ISSN 0010-4620.

Jiang, Y., Huang, Z., and Tsang, D.H.K. (2018). Challenges and Solutions in Fog Computing Orchestration. *IEEE Network*, 32(3), pp. 122–129. ISSN 0890-8044.

Jiao, Y., Wang, P., Niyato, D., and Suankaewmanee, K. (2019). Auction Mechanisms in Cloud/Fog Computing Resource Allocation for Public Blockchain Networks. *IEEE Transactions on Parallel and Distributed Systems*, 30(9), pp. 1975–1989.

Khan, R., Khan, S.U., Zaheer, R., and Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In: *2012 10th International Conference on Frontiers of Information Technology*, pp. 257–260.

Khater, B.S., et al. (2019). A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing. *Applied Sciences*, 9(1), p. 178.

Koo, D. and Hur, J. (2018). Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing. *Future Generation Computer Systems*, 78, pp. 739–752. ISSN 0167-739X.

Lahmar, I.B. and Boukadi, K. (2020). Resource Allocation in Fog Computing: A Systematic Mapping Study. In: *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 86–93.

Leandro, C.N. and Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*. London. UK.: Springer-Verlag.

Lee, G., Saad, W., and Bennis, M. (2017). An online secretary framework for fog network formation with minimal latency. In: *IEEE International Conference on Communications*, pp. 1–6. ISBN 9781467389990, ISSN 15503607.

Lee, S.S. and Lee, S. (2020). Resource Allocation for Vehicular Fog Computing Using Reinforcement Learning Combined With Heuristic Information. *IEEE Internet of Things Journal*, 7(10), pp. 10450–10464.

Liu, L., Chang, Z., and Guo, X. (2018). Socially Aware Dynamic Computation Offloading Scheme for Fog Computing System With Energy Harvesting Devices. *IEEE Internet of Things Journal*, 5(3), pp. 1869–1879.

Mäki, V. (2021). *Feasibility Evaluation of LPWAN Technologies - Case Study for a Weather Station*. Ph.D. thesis. Lappeenranta-Lahti University of Technology LUT.

Mseddi, A., Jaafar, W., Elbiaze, H., and Ajib, W. (2019). Intelligent Resource Allocation in Dynamic Fog Computing Environments. In: *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pp. 1–7.

Mukherjee, M., et al. (2017). Security and Privacy in Fog Computing: Challenges. *IEEE Access*, 5, pp. 19293–19304.

Narayanan, A., et al. (2020). Key advances in pervasive edge computing for industrial internet of things in 5g and beyond. *IEEE Access*, 8, pp. 206734–206754.

Nardelli, P.H. (2022). *Cyber-physical Systems: Theory, Methodology, and Applications*. John Wiley & Sons.

Nashaat, H., Ahmed, E., and Rizk, R. (2020). IoT Application Placement Algorithm Based on Multi-Dimensional QoE Prioritization Model in Fog Computing Environment. *IEEE Access*, 8, pp. 111253–111264.

Nemirovsky, M. (2012). Fog Computing. In: *Cloud Assisted Services in Europe (CLASS) Conference, Bled 2012*.

Oparin, G.A., Bogdanova, V.G., Pashinin, A.A., and Gorsky, S.A. (2019). Microservice-oriented Approach to Automation of Distributed Scientific Computations. In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 236–241.

Osanaiye, O., et al. (2017). From Cloud to Fog Computing: A Review and a Conceptual Live VM Migration Framework. *IEEE Access*, 5, pp. 8284–8300. ISSN 21693536.

Othmane, L.B. and Lilien, L. (2009). Protecting Privacy of Sensitive Data Dissemination Using Active Bundles. In: *2009 World Congress on Privacy, Security, Trust and the Management of e-Business*, pp. 202–213.

Özköse, H., Ari, E.S., and Gencer, C. (2015). Yesterday, Today and Tomorrow of Big Data. *Procedia - Social and Behavioral Sciences*, 195, pp. 1042–1050. ISSN 1877-0428.

Pahl, C. and Lee, B. (2015). Containers and Clusters for Edge Cloud Architectures - a Technology Review. *In Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pp. 379–386.

Pamukov, M.E. (2017). Application of artificial immune systems for the creation of IoT intrusion detection systems. In: *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, pp. 564–568.

Patro, R., et al. (2021). Module Placement Scheme Using MPC4.5 with Markov Chain Process for Mobile Fog Computing Environment. In: *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 304–309.

Potu, N., Jatoth, C., and Parvataneni, P. (2021). Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments. *Concurrency and Computation: Practice and Experience*, 33(23), pp. 1–13. ISSN 15320634.

Rafique, H., et al. (2019). A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing. *IEEE Access*, 7, pp. 115760–115773.

Ray, K., Banerjee, A., and Narendra, N.C. (2020). Proactive Microservice Placement and Migration for Mobile Edge Computing. In: *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 28–41.

Ren, Z., et al. (2020). Resource scheduling for delay-sensitive application in three-layer fog-to-cloud architecture. *Peer-to-Peer Networking and Applications*, 13(5), pp. 1474–1485. ISSN 1936-6450.

Sadaf, K. and Sultana, J. (2020). Intrusion Detection Based on Autoencoder and Isolation Forest in Fog Computing. *IEEE Access*, 8, pp. 167059–167068.

Sahni, Y., Cao, J., Zhang, S., and Yang, L. (2017). Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things. *IEEE Access*, 5, pp. 16441–16458. ISSN 21693536.

Sampaio, A.R., Rubin, J., Beschastnikh, I., and Rosa, N.S. (2019). Improving microservice-based applications with runtime placement adaptation. *Journal of Internet Services and Applications*, 10(1), p. 4. ISSN 1869-0238.

Sanaei, Z., Abolfazli, S., Gani, A., and Buyya, R. (2014). Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges. *IEEE Communications Surveys Tutorials*, 16(1), pp. 369–392. ISSN 1553-877X.

Santana, C., Andrade, L., Delicato, F.C., and Prazeres, C. (2021). Increasing the availability of IoT applications with reactive microservices. *Service Oriented Computing and Applications*, 15(2), pp. 109–126. ISSN 18632394.

Santos, J., Wauters, T., Volckaert, B., and De Turck, F. (2019). Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications. In: *2019 IEEE Conference on Network Softwarization (NetSoft)*, pp. 351–359.

Santos, J., Wauters, T., Volckaert, B., and De Turck, F. (2021a). Towards end-to-end resource provisioning in Fog Computing over Low Power Wide Area Networks. *Journal of Network and Computer Applications*, 175, p. 102915. ISSN 1084-8045.

Santos, J., Wauters, T., Volckaert, B., and Turck, F.D. (2021b). Resource Provisioning in Fog Computing through Deep Reinforcement Learning. In: *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 431–437.

Selimi, M., et al. (2019). A Lightweight Service Placement Approach for Community Network Micro-Clouds. *Journal of Grid Computing*, 17(1), pp. 169–189. ISSN 1572-9184.

Shakarami, A., et al. (2022). Resource provisioning in edge/fog computing: A Comprehensive and Systematic Review. *Journal of Systems Architecture*, 122, p. 102362. ISSN 1383-7621.

Singh, V. and Peddoju, S.K. (2017). Container-based microservice architecture for cloud applications. In: *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 847–852.

Skarlat, O., et al. (2017). Optimized IoT service placement in the fog. *Service Oriented Computing and Applications*, 11(4), pp. 427–443. ISSN 1863-2394.

Stojmenovic, I. and Wen, S. (2014). The Fog Computing Paradigm: Scenarios and Security Issues. In: *Federated Conference on Computer Science and Information Systems*, vol. 2, pp. 1–8. ISBN 9788360810583.

Stojmenovic, I., Wen, S., Huang, X., and Hao, L. (2016). An overview of Fog computing and its security issues. *Concurrency and Computation: Practice and Experience*, 22(6), pp. 685–701. ISSN 15320634.

Sun, Y., Lin, F., and Xu, H. (2018). Multi-objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II. *Wireless Personal Communications*, 102(2), pp. 1369–1385. ISSN 1572-834X.

Ullah, M., Nardelli, P.H., Wolff, A., and Smolander, K. (2020). Twenty-one key factors to choose an iot platform: Theoretical framework and its applications. *IEEE Internet of Things Journal*, 7(10), pp. 10111–10119.

Urgaonkar, R., et al. (2015). Dynamic service migration and workload scheduling in edge-clouds. *Performance Evaluation*, 91, pp. 205–228. ISSN 0166-5316. Special Issue: Performance 2015.

Wang, J. and Li, D. (2019). Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing. *Sensors*, 19(5). ISSN 1424-8220.

Wang, S., et al. (2015a). Dynamic service migration in mobile edge-clouds.

Wang, Y., Uehara, T., and Sasaki, R. (2015b). Fog Computing: Issues and Challenges in Security and Forensics. *2015 IEEE 39th Annual Computer Software and Applications Conference*, 3, pp. 53–59.

Wu, C.g. and Wang, L. (2019). A Deadline-Aware Estimation of Distribution Algorithm for Resource Scheduling in Fog Computing Systems. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 660–666.

Yaqoob, I., et al. (2017). Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges. *IEEE Wireless Communications*, 24(3), pp. 10–16.

Ye, D., Wu, M., Tang, S., and Yu, R. (2016). Scalable Fog Computing with Service Offloading in Bus Networks. In: *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 247–251.

Yousefpour, A., et al. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, (February). ISSN 13837621.

Zhang, P., Zhou, M., and Fortino, G. (2018). Security and trust issues in Fog computing: A survey. *Future Generation Computer Systems*, 88, pp. 16–27. ISSN 0167739X.

Zhi, Y., Fu, Z., Sun, X., and Yu, J. (2019). Security and Privacy Issues of UAV: A Survey. *Mobile Networks and Applications*, pp. 1–10. ISSN 15728153.

Publication I

Hosseinpour, F., Vahdani Amoli, P., Farahnakian, F., Plosila, J., and Hämäläinen, T.
Artificial Immune System Based Intrusion Detection: Innate Immunity using an Unsupervised Learning Approach

Reprinted with permission from
International Journal of Digital Content Technology and its Applications
Vol. 8(5). pp. 1-12, 2014.
© 2014, AICIT

Artificial Immune System Based Intrusion Detection: Innate Immunity using an Unsupervised Learning Approach

[1]Farhoud Hosseinpour, [2]Payam Vahdani Amoli, [3]Fahimeh Farahnakian, [4]Juha Plosila and [5] Timo Hämäläinen
[1 Corresponding Author, 3 and 4] *Department of Information Technology, University of Turku, Finland.*
{farhos;fahfar;juplos}@utu.fi
[2 and 5]*Faculty of Information Technology, University of Jyväskylä, 40100, Jyväskylä, Finland.*
[2] *pavahdan@student.jyu.fi*
[5] *timo.t.hamalainen@jyu.fi*

Abstract

This paper presents an intrusion detection system architecture based on the artificial immune system concept. In this architecture, an innate immune mechanism through unsupervised machine learning methods is proposed to primarily categorize network traffic to "self" and "non-self" as normal and suspicious profiles respectively. Unsupervised machine learning techniques formulate the invisible structure of unlabeled data without any prior knowledge. The novelty of this work is utilization of these methods in order to provide online and real-time training for the adaptive immune system within the artificial immune system. Different methods for unsupervised machine learning are investigated and DBSCAN (density-based spatial clustering of applications with noise) is selected to be utilized in this architecture. The adaptive immune system in our proposed architecture also takes advantage of the distributed structure, which has shown better self-improvement rate compare to centralized mode and provides primary and secondary immune response for unknown anomalies and zero-day attacks. The experimental results of proposed architecture is presented and discussed.

Keywords: *Distributed intrusion detection system, Artificial immune system, Innate immune system, Unsupervised learning*

1. Introduction

Anomaly-based intrusion detection systems (IDS) have been broadly researched as defensive techniques to address the detection of unknown or zero-day attacks. Unlike misuse-based or signature-based types of IDS, which take advantage of the predetermined signature of known attacks, anomaly-based IDS deals with the detection of new types of attack that are unknown to the system. This process is done by detecting variation in the systems' behavior from a previously defined normal system profile. However, it is subject to false alarms as a result of the difficulty in defining the normal state during training. An increasing detection rate with fewer false alarms became an important challenge in the design of anomaly-based IDS.

The artificial immune system (AIS) comprises promising techniques in the form of biologically inspired computing that is applied to solving various problems in the information security field. The AIS is inspired by the human immune system (HIS), which has the ability to distinguish internal cells and molecules of the body from foreign pathogens, so called self and non-self respectively, and protects the body against diseases [1]. In the human body the HIS mainly does this without any prior knowledge of attacking pathogens and their structure. As self and non-self discrimination is a significant attribute in the AIS, it is proposed that it is utilized in designing efficient anomaly-based IDS [2]–[4]. The AIS suggests a multi-layered protection structure for protecting computer networks against attack, like HIS protection against foreign pathogens in the human body [5]. This protection is accomplished through *Innate* or *Adaptive* mechanisms. Innate immunity is immediate; it is the first line of defense for the HIS and provides non-specific protection. Therefore, it has no prior knowledge of specific outsiders. The adaptive immune response, on the other hand, is antigen-specific and is trained using a pre-defined profile of specific attacks [6]. Adaptive immunity also includes a "*memory*" that makes future responses against a specific antigen more efficient [7].

Like other anomaly-based detection techniques, the AIS also takes advantage of monitoring variations of the system's behavior according to a pre-defined normal activity profile as an adaptive immune mechanism. This is done through a learning phase in which a data set containing these profiles is utilized for this purpose. Hence, the efficiency of anomaly detection in the AIS is highly dependent on the learning data set. Substantial research has been conducted so far in the improvement and utilization of AIS-based IDS, the majority of which have utilized a pre-defined and offline data set as learning data for training the IDS. This will reduce the efficiency of the IDS by limiting its knowledge-base to that particular learning data set. Moreover, it is extremely difficult to create a data set of self samples with all variations. In order to cope with this problem, in this paper we have proposed an innate immune mechanism by using unsupervised learning methods as the first line of defense in AIS-based IDS. The innate immune system in our proposed architecture provides online and dynamic categorization of network flows to self and non-self, which is then used by the adaptive immune system to generate attack-specific detectors.

Machine learning methods can be organized based on the type of input available during training. There are three main categories of machine learning: supervised, semi-supervised and unsupervised algorithms. Supervised machine learning algorithms need to be trained by labeled data to distinguish the normal and abnormal behavior of the network. Semi-supervised machine learning algorithms can be trained by attack-free unlabeled data. The acquisition of labeled data from security experts, or finding attack-free data sets for both supervised and semi-supervised techniques, is costly. Recent studies showed the feasibility of unsupervised learning approaches in IDS in comparison with supervised or semi-supervised learning-based IDS. Unsupervised machine learning techniques formulate the invisible structure of an unlabeled data set without any prior knowledge. Clustering algorithms put objects based on their similarities into a cluster or clusters. Clustering algorithms have been used for unsupervised IDS to classify the behavior of the network [8]–[10].

The remainder of the paper is organized as follows. In section **2** we briefly review the AIS and unsupervised learning approaches. We discuss some of the most important related works in section **3**. Section **4** presents the proposed AIS-based IDS. Two main engines of the proposed DIS are explained in Sections **5** and **6**. In section 7, the experimental results of proposed architecture are presented and, finally, we discuss and conclude in Section **8**.

2. Background

In this section, we explain briefly the AIS and clustering method employed in our proposed IDS.

2.1. Artificial Immune System

The human immune system defends the human body against harmful and previously unseen foreign cells using lymphocyte cells. The foreign cells are called antigens, such as bacteria and viruses [1]. The artificial immune system is designed for the computational system and inspired by the HIS; it is applied to solving various problems in the field of information security, particularly intrusion detection systems [11], [12]. Moreover, it incorporates many attributes of the HIS, including diversity, error tolerance, dynamic learning, adaption and self-monitoring [3]. The AIS has the capability to differentiate between the “self” (cells that are owned by the system) and “non-self” (foreign entities to the system) as intrusions. Likewise, detectors similar to lymphocytes are deployed in computer system nodes to intercept and report any malicious activities.

The HIS employs a negative selection process to generate mature immune system cells called T-cells. Forrest et al. [2] proposed a negative selection algorithm to utilize this process of the HIS for a sophisticated anomaly-detection process. This process allows the detection of previously unseen harmful cells without any definition of specific harmful cells.

The algorithm includes three phases: defining self, generating detectors and monitoring the occurrence of anomalies. In the first phase, it establishes the normal behavior patterns of a monitored system to define "self". It regards the profiled normal patterns as "self" patterns. In the second phase, it generates a number of immature T-cells with random patterns that are compared to each self pattern defined in the first phase. If any generated pattern matches a self pattern, the pattern fails to become a detector and is thus removed. Otherwise, it becomes a mature T-cell detector and is utilized for monitoring subsequent profiled patterns of the monitored system. During the third phase, if a T-cell detector matches any newly profiled pattern, it is then considered that that new anomaly must have occurred in the monitored system.

2.2. Clustering Methods (Unsupervised Machine Learning)

There are several approaches to unsupervised machine learning. Clustering is one of the unsupervised machine learning techniques that have been used for IDS. Clustering techniques group (cluster) samples of data sets based on their similarities to find the outliers as the anomaly. Cluster association and centroid distance techniques are the two most important categories of clustering for anomaly detection. Two popular clustering approaches that have been applied in many proposed IDS are as follows:

1) Density-Based Spatial Clustering of Applications with Noise (DBSCAN) finds a number of clusters starting from the estimated density distribution of the corresponding samples. It requires two parameters: maximum radius of the neighborhood *(ε)* and minimum number of samples required to form a cluster (*minPts*). DBSCAN detects a density-connected cluster by discovering one of its core samples, *p*, and computes all samples that are density-reachable from *p*. It checks the *ε*-neighborhood of each sample *p*, *Nε(p)*, in the data set. If the *Nε(p)* of sample *p* consists of the least *minPts* samples, a new cluster containing all samples of *Nε(p)* is created.

2) K-means partitions the given data set into n clusters, in which each cluster has a cluster center (centroid). Any sample assigned to each cluster has a minimum distance to the centroid of the cluster. The Euclidean distance can be used to determine the distance between each sample and the centroid.

3. Related Works

Farmer et al. [13] put forward a new link between biological and computing sciences by proposing the artificial immunology model. Forrest et al. [14] proposed the most effective idea in the utilization of immunity in computer security for self and non-self discrimination. Following this work, they presented basic architecture [3] for the artificial immune system and took advantage of it in deploying the first AIS-based IDS, which was called LISYS. So far different frameworks have been presented in the utilization of the AIS for IDS. However, there are essentially two main approaches to applying AIS in DIS. One approach is classical self/non-self discrimination and another is the application of danger theory as a substitute for the former.

- In [14], a negative selection algorithm is proposed to discriminate between self and non-self entities. The algorithm first creates a set of detectors randomly and then compares it with a set of normal sets (self). If any detectors are matched with any self entity, the system eliminates them and the rest will be kept.
- As a substitute to self/non-self discrimination, the Danger Model was proposed by Matzinger [15], [16]. According to this hypothesis the main cause of an immune response is that a pathogen harms the system and thus is dangerous and not unknown to the system. Aickelin et al. [17] stated that in the IDS paradigm the danger is sensed and measured automatically after a number of intrusions because of the damage caused by the attack. Once a danger signal is detected, it will be transmitted to the nearest artificial antibody around the danger area.

In [5] a multi-layered structure consisting of detection, defense and user layers was proposed. Dal et al. [18] proposed a model in which the primary immune response is evolved through genetic algorithm to a secondary immune response with optimized detectors that are correspondent to memory cells in natural immune systems. This work was proposed as an enhancement to Forrest and Hofmeyr's work. However, their model still had the disadvantage of central processing with high processing overheads for large network traffic. In our previous work [19], we enhanced their model by proposing a distributed framework to reduce the processing overheads and to increase the efficiency of the IDS. Moreover, the distributed nature of this model resulted in a greater self-improvement feature for this IDS. This work, however, utilizes offline learning data for training the IDS. Since the network behavior is changed in a dynamic fashion, a new profile of normal and abnormal activity needs to be trained to the system dynamically. In order to solve this problem, in this paper we have proposed an innate immune mechanism by utilization of unsupervised learning methods to the primary detection of self and non-self flows, which features online and dynamic training of the adaptive immune system of AIS-based IDS.

Researchers have applied unsupervised machine learning algorithms in IDS to overcome the issues of training and detecting new attacks. For instance, in [20] they proposed a practical real-time solution for NIDS to detect known and unknown network attacks using unsupervised neural networks. They applied several neural networks to improve the detection rate of intrusions. In [8] they also proposed an unsupervised NIDS, which uses different clustering algorithms to detect attacks such as DOS/DDOS, Worm and Network scanning. In [9] they presented a tree-based subspace clustering technique for unsupervised NIDS in high-dimensional data sets. In the proposed model they have generated and analyzed cluster stability for each cluster by using an ensemble of multiple cluster indices. They have also introduced a multi-objective cluster labeling technique to label each stable cluster as normal or anomalous.

4. Proposed Intrusion Detection System

Figure 1 shows the proposed intrusion detection system, which consists of two main engines. The clustering engine performs network traffic clustering into the self or non-self clusters through unsupervised learning techniques. The AIS engine consists of agents that cooperate for intrusion detection. The term "agent" originally comes from Artificial Intelligence (AI) and refers to anything that can view its environment through sensors and act upon that environment using actuators . In this paper, the term agent refers to software agents. Compared to Dal et al.'s [18] work, in our previous work we proposed a distributed model in which we experimented increased performance and efficiency of these IDs as a result of a greater self-improvement rate compare to a centralized structure. This is due to generation of new memory cells and their dynamic synchronization and distribution to all hosts, and thus an enhanced secondary immune response.

The AIS engine trains the primary detectors generated by the negative selection algorithm based on received information from the clustering engine. Moreover, it improves the performance of primary detectors according to the intrusion report analysis from all hosts. In the architecture, the packet pre-processing module is responsible for extracting several attributes from the network traffic to create network flows. These attributes are selected based on the protocol types shown in Table 1.

Table 1. Network Flow specification for each type of packet

Packets Protocol	***Features***
IP	Source IP Address, Destination IP Address, Time of the First Packet, Time of the Last Packet, Duration
TCP	Source Port Number, Destination Port Number, Number of Packets, Number of SYN Packet, Number of SYN-ACK packet, Number of RST Packet, Number of RST-ACK Packet, Number of FIN-ACK Packet
UDP	Source Port Number, Destination Port Number, Number of Packets
ICMP	#Eco Request, #Eco Reply

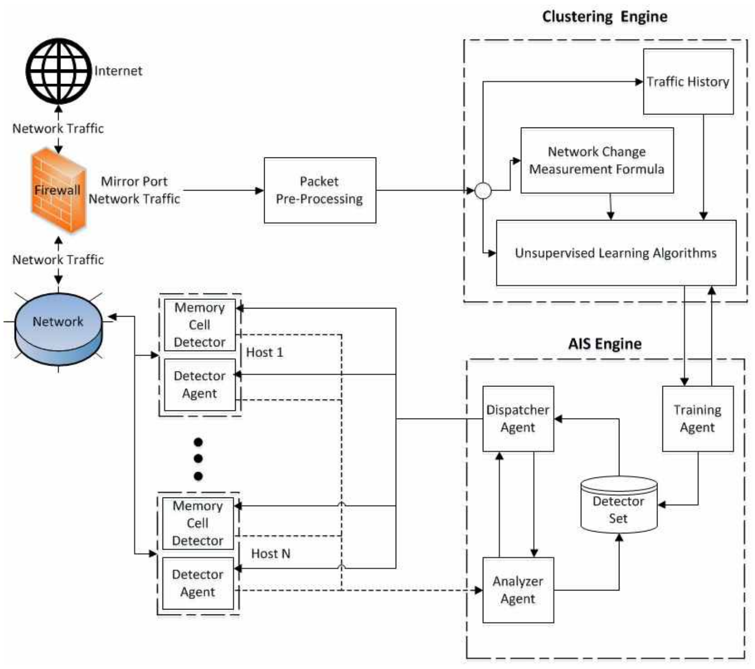


Figure 1. Proposed system architecture

5. Clustering Engine

In order to detect unseen intrusions without using any prior knowledge (training by labeled traffic or signature), we propose a clustering engine as innate immune response. The clustering engine utilizes the DBSCAN clustering method to group the real network traffic into clusters and consider them as self, while behaviors outside of the clusters will be considered as noise or non-self. For this purpose, the engine continuously compares the number of network flows, in different network resolutions (subnets of /0, /8, /16, /24), with a threshold which is dynamically calculated by our proposed network measurement formula in Table 2. Since high speed networks have larger amount of traffic, there is a significant possibility of losing the sign of network attacks. To overcome this issue the system will also monitor the behavior of the network in small resolutions to decrease the possibility of fading the attacks in the normal traffic.

To obtain an accurate threshold, the system needs to determine the previous behavior of the network. It is possible that small attacks to be fade with occurrence of heavy attacks, thus we have applied standardization on the number of network flows by using logarithm (Log) to increase the probability of detecting small attacks during the occurrence of heavy attacks. To determine changes in the network traffic, the system will calculate the “standard deviation” of the number of network flows in different windows from last minute of the traffic. As shown in Table 2 the previous 60 seconds of traffic is divided into four 15 seconds windows. For instance, δ_1 is the standard deviation of number of network flows in the first window which is from the last 65 seconds to the last 50 seconds of the previous network traffic. As it has been seen in so many datasets, it takes 2 to 3 seconds from starting time of the network attacks (such as DOS/DDOS attacks) till its own peak. To minimize the effect of early traffic of the attacks on the threshold, the network measurement formula considers a 5 seconds gap between every one minute of traffic to calculate the threshold for the current traffic. Sum of the highest standard deviations (from δ_1 to δ_4) and the heaviest traffic from the last minute of traffic can determine the highest traffic which could be accepted as normal. The following equation represents the network measurement formula which calculates the network traffic threshold "T_{nt}":

$$T_{nt} = \left(Max\{\delta_i | i = 1 \cdots 4\} + Max\{X_j \log X_j \,|j = 1 \cdots 60\}\right) \times \gamma$$

Where δ_i is standard deviation of number of network flows in i_{th} window and X_j is number of network fellow in j_{th} second of last minute's traffic. And γ is a coefficient value which can be set to determine the final threshold.

Table 2. Elements of network measurementformula

Last Minute Traffic				***Gap***	***Current Traffic***
Window 1	Window 2	Window 3	Window 4		T_{nt}
δ_1	δ_2	δ_3	δ_4		
65-50	50-35	35-20	20-5	5-0	
$X_j \log X_j$					

The cluster engine considers a network behavior as abnormal when the number of network flows is more than the calculated threshold. Whenever the number of network flows passes the threshold, the clustering engine will cluster the number of in-bounded and out-bounded network flows for each IP (from the past five minutes) to define the normal cluster and obtain an accurate "ε" as the maximum radius of the neighborhood in DBSCAN. Afterwards it will cluster the suspicious traffic with "ε" to find and flag the outliers (noise) as intrusion or non-self. Network intrusions such as DOS, DDOS, Spamming, Worm and UDP storm, generate a large number of network flows. We consider the number of inbound and outbound network flows as the other two essential features for clustering in order to detect these types of intrusion as outliers. Using the aforementioned features of network flow for the clustering engine increases the detection rate of network attacks. Moreover, it will reduce the time and space complexity of clustering algorithms compared to the previous proposed IDSs, which deals with data extracted from packets and bytes (payload). Detected intrusions will be marked as a non-self and set of marked flows will be forwarded to the training agent in the AIS engine to train the primary detectors.

6. AIS-based IDS

The proposed AIS-based IDS has two components: the AIS engine and the host side detectors. This section describes these components in the AIS-based IDS.

6.1. AIS Engine

In the AIS engine, we proposed three agents to employ the adaptive immune response of the AIS in IDS.

Training agent: first converts the network flow information into the binary string with a total 112-bit length as a flow profile (Table 3). Then, using the negative selection algorithm it generates and trains the primary detectors. The negative selection algorithm first generates a number of random detectors (immature detectors) and then trains them with samples of labeled flows from the cluster engine. If each immature detector matches each self sample of the data set, then the system will discard it and generate another in its place. After checking all immature detectors with all self samples, the remaining detector sets undergo the next step of the negative selection algorithm and become mature detectors. Each mature detector will be checked with all non-self samples of labeled flows. If a mature detector fails to match with some non-self samples, the system will discard this detector; otherwise, this detector will be added to a detector set. This process will continue until all non-self packets are matched with at least three mature detectors.

The negative section algorithm uses the r-Contiguous matching bit role [21] to check the matching between two strings. In this method, two strings are matched if they have at least *r* contiguous identical bits. Finally, the output of the negative selection algorithm is a set of primary detectors, which are archived and synchronized in detector set repository and then sent to the dispatcher agent for distribution to local hosts. These detectors are analogous to primary immune response in the HIS.

Table 3. Depiction of fields in flows profile strings

Name of the Field	Minimum and Maximum Value	Binary Strings Length (bits)
Destination IP Address	0.0.0.0 - 255.255.255.255	32
Source IP Address	0.0.0.0 - 255.255.255.255	32
Destination Port No	0 – 65535	16
Duration	0 – 65535	12
Protocol	0 – 65535	4
Source Port No	0 – 65535	16

Dispatcher agent: distributes detectors from the detector set to all hosts in the network. Moreover, it has the responsibility to communicate with all hosts and synchronize them according to changes in the detector set. It also forwards the reported flow from the memory cell detectors and detector agents to the analyzer agent.

Analyzer agent: employs the proposed genetic algorithm in [18] to evolve the highly fit detectors activated when an anomaly has been encountered. The suspected flow reported from the host's (discussed later) profile of activated detectors and their affinity with reported flow are analyzed in this agent, and an optimized detector, called the memory cell detector, is generated. A memory cell detector is a high-affinity and attack-specific detector with a higher detection ability and analogous to secondary immune response in the HIS [7].

A selection operation is undertaken on activated detectors to select the detectors with the highest affinity for cloning and formation of primary population for genetic algorithm. Those detectors having a fitness value greater than or equal to cloning threshold undergo cloning. The cloning threshold is set as follows.

$$\text{Cloning Threshold} = \frac{\sum_{i=0}^{n} \text{Fitness of detectors}}{n}$$

Where "n" is the total number of activated detectors.

Winner detectors that consist of cloned detectors and remain activated detectors are subjected to the genetic operators of Mutation, Crossover, and Reproduction, which facilitates the evolution of these detectors. This process is repeated and continued for a few generations until a detector with a fitness value higher than all winner detectors is generated. The optimized detector from the genetic algorithm is treated as a memory cell. Finally, the agent sends the memory cell to the dispatcher agent and adds it to the memory cell detector set.

6.2. Host Side Detectors

In order to improve the performance of IDS, detectors are distributed in all hosts in the network rather than a centralized structure. Moreover, the proposed distributed approach is robust and extendable. All inbound and outbound network flows are checked using these detectors. In each host, we consider two detectors as follows.

Detector agents: comprise a set of trained detectors that have the ability to discriminate between self and non-self flows. These detectors are non-specific and responsible for primary immune response for anomalies that occur for the first time. If a flow matches a detector with an effective affinity, that detector is considered an activated detector and the flow is suspected as an intrusion. To improve the accuracy of detection and reduce the false-positive errors in IDS, we have defined an intrusion threshold (T_i). If the number of activated detectors by a suspected flow is more than T_i, the flow is detected as an intrusion and its information is forwarded to the analyzer agent through a dispatcher agent.

Memory cell detectors: composed of a set of optimized detectors generated by the analyzer agent. As the secondary response of the AIS, memory cells have more accurate intrusion detection abilities. Hence, any flow that activates any of these detectors is treated as an intrusion and blocked by the hosts.

6.3. Detector Life Cycle and Non-self Updater

In order to maintain the efficiency of detectors, we propose to define a lifespan to eliminate unused or weak detectors. Due to machine learning errors, there is a possibility that some of the generated detectors have insufficient detection ability and remain inactivated during their lifespan. Such detectors have negative overheads to the system and reduce its performance. Therefore, in order to solve this problem we define a lifespan for all detectors, during which the number of times the detector is activated is counted. When the lifespan ends, if the counter is less than a threshold, the detector will be discarded and its profile forwarded to a clustering engine as feedback to improve its accuracy. Otherwise, the lifespan will be reset and the detector will remain in the main detector sets.

7. Experimental Result

To evaluate the efficiency of two popular clustering algorithms, we utilized KDD-Cup 99 data set, which is extracted from DARPA-98 traffic network. The number of samples of data set was 22545, which was sufficient for performing the evaluation and comparison between DBSCAN and K-means. The parameters for the different algorithms used are tabulated in Table 4. These parameter values were obtained in a series of preliminary experiments. Table 5 shows the achieved results of K-means and DBSCAN algorithms. We have measured the False Positive Rate (FPR), True Negative Rate (TNR), Accuracy (ACC) which estimated by dividing the total correctly classified positives and negatives by the total number of samples, Recall (REC) or True Positive Rate which estimated by dividing the correctly detected anomalies and the total number of anomalies, Precision of Anomalies (PREC) or positive predicted value which estimated by dividing the correctly classified positives by the total predicted positive count and finally the F1 score, which is the weighted average of the precision and recall. In this experiment, due to ability of DBSCAN for finding arbitrarily shaped cluster, this algorithm demonstrated a better rate of detection compared to K-mean.

Table 4. The parameters of the evaluated clustering approaches

n	*minPts*	*ε*
15	34.4%	1.09

Table 5. The comparison between clustering approaches

Algorithm	*FPR*	*TNR*	*ACC*	*REC*	*PREC*	*F1*
DBSCAN	0.008	0.991	0.771	0.589	0.987	0.738
K-Mean	0.156	0.843	0.607	0.431	0.788	0.557

In our experiment all of the parameters for the network traffic threshold (T_{nt}) is obtained automatically and we only set "γ" as 5. As shown in figure 3, during network attacks the behavior of network passes the threshold. For instance the network traffic during POD (Ping of Death) attack was 10 times more than the threshold or in DDOS (Distributed Denial of service) attack it was 6 times more than the threshold.

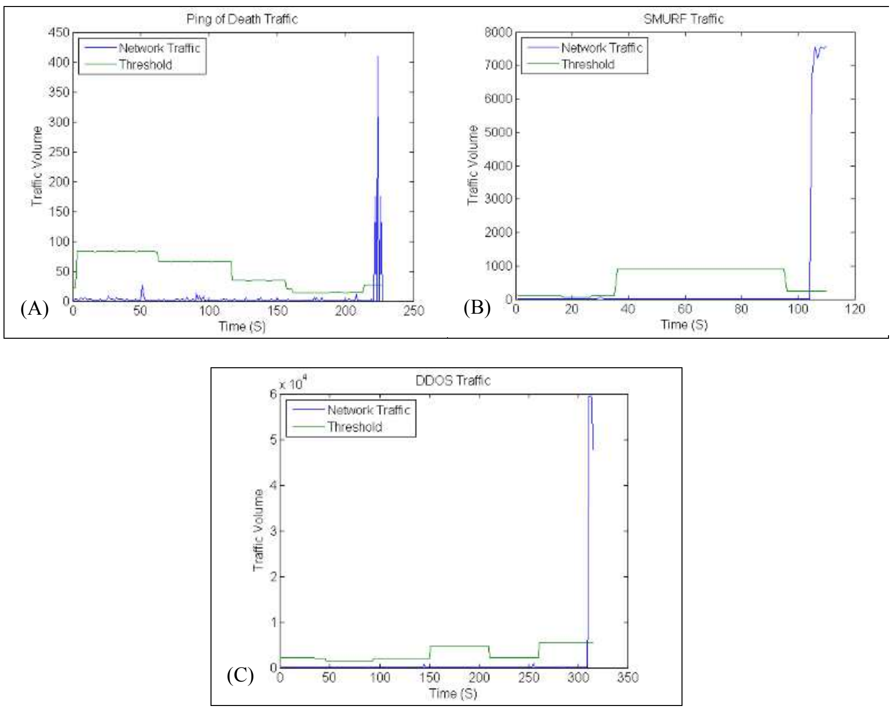
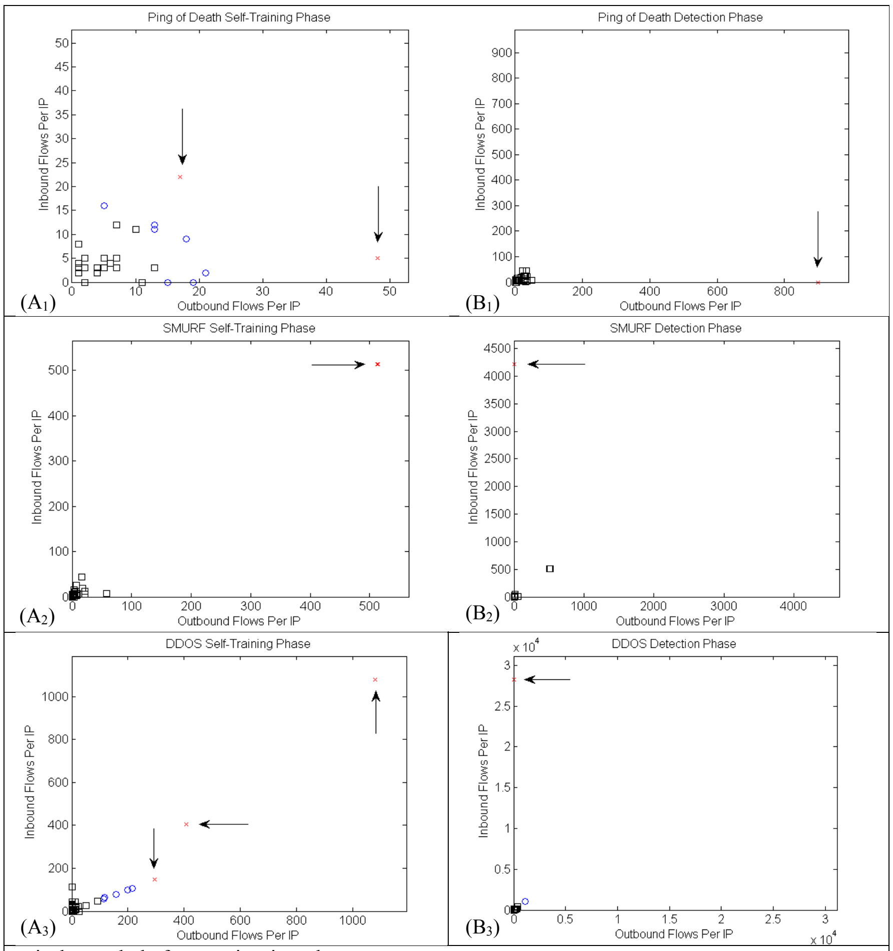


Figure 3. Behavior of network traffic during POD (A), SMURF (B) DDOS (C) attacks.

As mentioned before when the network traffic passes the threshold the system will flag that specific time slot as suspicious and the traffic in that time slot will undergo for clustering. Figure 4, (A_1, A_2, and A_3) shows the self-training phase of attacks such as POD, SMURF and DDOS. As DBSCAN needs two parameters "ε" and "*minPts*" (maximum radius of the neighborhood and minimum number of samples required to form a cluster), it sets "ε" as average Euclidian distance between all of the points in the dataset and "*minPts*" equal to 10% of the traffic sample. In Self-Training phase the clustering engine cluster the 5 min attack free time slot (the time slot before the suspicious behavior) to find out more accurate "ε". Then it will obtain the minimum distance between noises and clustered points as "φ". Based on our experimental results, sum of "φ" and "ε" will create more accurate and acceptable radius of the neighborhood for the DBSCAN during attack detection. Figure 4, (B_1, B_2, and B_3) shows the detection phase of POD, SMURF and DDOS attacks in which the noises are pointed by arrows in the diagram showing the attackers or the victim of the attacks. For instance during 1 to N network attacks such as POD as the intruder creates high amount of network flows the clustering engine consider this behavior as anomaly or non-self and in N to 1 network attacks such as SMURF or DDOS as the victims are flooded by huge number of requests the clustering engine considers this behavior as noise or non-self.



□ is the symbol of core points in a clusters
O is the symbol of density reachable point
x is the symbol of noise

Figure 4. Detection of network attacks by the clustering algorithm

In order to test the efficiency of the AIS engine and thus the proposed model, in our experiment the fitness value of "rc" for R-Contiguous matching bit algorithm is set to 13 and threshold of "T_i" is set to 3. Moreover, by testing the genetic algorithm for generation of memory cells, in different conditions, the probabilities of genetic operations of Crossover, Mutation, and Reproduction have been fixed to 30%, 40% and 30% respectively. The system is tested in both centralized and distributed mode. Figure 5 compares the self-improvement rate of AIS based IDS in central and distributed modes. According to this diagram, the self-improvement rate in distributed mode is better than centralized mode and it reaches to its stable maximum amount after only 6 rounds, while this happens after 10 rounds in centralized mode. This is because of dynamic distribution and synchronization of newly generated memory cells in each host to others.

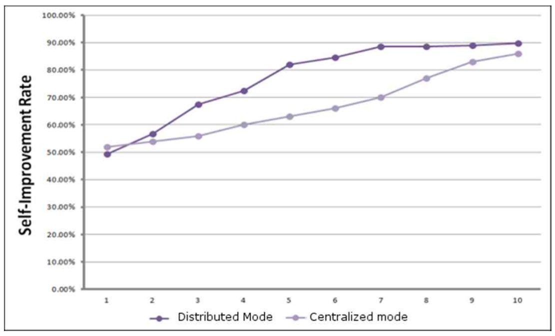


Figure 5. Comparison of self-improvement rate in distributed and centralized mode.

8. Conclusion and Future Work

In this paper, we presented a novel architecture for an intrusion detection system based on the artificial immune system. We proposed innate immunity using unsupervised machine learning methods. According to our primary experiments we conclude that among other methods, DBSCAN clustering is robust and has the greatest potential for this purpose. In this multi-layered framework, the clustering engine labels the network traffic as self and non-self without previous training or knowledge about network flow profiles, thus acting as the first line of defense in AIS-based IDS and providing innate immunity. We defined a network measurement formula as a dynamic threshold to facilitate the detection of abnormal network behaviors. The output of clustering is used to feed the training data for the adaptive immune system as online and real-time training data. Primary detectors after training are distributed to hosts in the network and provide primary immune response for our IDS. We presented the experimental results of proposed innate immune mechanism using our network measurement formula. We also demonstrated that the distributed structure for this IDS is more efficient than the centralized mode. Suspected intrusions reported from hosts are analyzed and an optimized memory cell detector is generated through a genetic algorithm process. Memory cells are attack specific detectors, which provide a secondary immune response. We defined detector life cycle to update and eliminate weak or inefficient detectors to enhance the performance of detection. Future work will mainly focus on detection of potential Bots and BotMater after DDOS attacks. As there is a specific behavioral structure in communications between the BotMaster and Bots in Botnet attacks, it is possible to cluster the communication between potential Bots (DDOS attackers) to detect the BotMaster.

9. References

[1] L. N. de Castro and J. Timmis, Artificial {I}mmune {S}ystems: A {N}ew {C}omputational {A}pproach. London. UK.: Springer-Verlag, 2002, p. 357.
[2] S. Forrest, S. A. Hofmeyr, and A. Somayaji, “Computer Immunology,” Commun. ACM, vol. 40, no. 10, pp. 88–96, Oct. 1997.
[3] S. A. Hofmeyr and S. A. Forrest, “Architecture for an Artificial Immune System,” Evol. Comput., vol. 8, no. 4, pp. 443–473, Dec. 2000.
[4] S. and G. G. Feixian, “Research of Immunity-based Anomaly Intrusion Detection and Its Application for Security Evaluation of E-government Affair Systems.,” JDCTA Int. J. Digit. Content Technol. its Appl., vol. 6, no. 20, pp. 429 – 437, 2012.
[5] M. Tan, H. Yu, Z. Zhao, Z. Liu, and F. Liu, “An artificial immunity-based proactive defense system,” in Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on, 2007, pp. 2239–2243.
[6] et al. Xishuang, D., “Multi-word-Agent Autonomy Learning Based on Adaptive Immune Theories,” JDCTA Int. J. Digit. Content Technol. its Appl., vol. 7, no. 3, pp. 723–745, 2013.

[7] A. A. Ademokun and D. Dunn-Walters, "Immune Responses: Primary and Secondary," in eLS, John Wiley & Sons, Ltd, 2001.
[8] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised Network Intrusion Detection Systems: Detecting the Unknown Without Knowledge," Comput. Commun., vol. 35, no. 7, pp. 772–783, Apr. 2012.
[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "An Effective Unsupervised Network Anomaly Detection Method," in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, 2012, pp. 533–539.
[10] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," Commun. Surv. Tutorials, IEEE, vol. 10, no. 4, pp. 56–76, 2008.
[11] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "An artificial immune system architecture for computer security applications," Evol. Comput. IEEE Trans., vol. 6, no. 3, pp. 252–280, Jun. 2002.
[12] F. Hosseinpour, K. A. Bakar, A. H. Hardoroudi, and N. Kazazi, "Survey on Artificial Immune System As a Bio-inspired Technique for Anomaly Based Intrusion Detection Systems," in Proceedings of the 2010 International Conference on Intelligent Networking and Collaborative Systems, 2010, pp. 323–324.
[13] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," Phys. D Nonlinear Phenom., vol. 22, no. 1–3, pp. 187–204, 1986.
[14] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on, 1994, pp. 202–212.
[15] P. Matzinger, "Essay 1: The Danger Model in Its Historical Context," Scand. J. Immunol., vol. 54, no. 1–2, pp. 4–9, 2001.
[16] P. Matzinger, "The Danger Model: A Renewed Sense of Self," Science (80-.)., vol. 296, no. 5566, pp. 301–305, 2002.
[17] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger Theory: The Link between AIS and IDS?," in Artificial Immune Systems, vol. 2787, J. Timmis, P. Bentley, and E. Hart, Eds. Springer Berlin Heidelberg, 2003, pp. 147–155.
[18] D. Dal, S. Abraham, A. Abraham, S. Sanyal, and M. Sanglikar, "Evolution Induced Secondary Immunity: An Artificial Immune System Based Intrusion Detection System," in Computer Information Systems and Industrial Management Applications, 2008. CISIM '08. 7th, 2008, pp. 65–70.
[19] Hosseinpour F., Meulenberg A., Ramadass S., Vahdani Amoli P., and Z. Moghaddasi, "Distributed Agent Based Model for Intrusion Detection System Based on Artificial Immune System," Int. J. Digit. Content Technol. its Appl., vol. 7, pp. 206–214, 2013.
[20] M. Amini, R. Jalili, and H. R. Shahriari, "RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks," Comput. Secur., vol. 25, no. 6, pp. 459–468, 2006.
[21] T. Stibor, "Foundations of r-contiguous Matching in Negative Selection for Anomaly Detection," Nat. Comput., vol. 8, no. 3, pp. 613–641, Sep. 2009.

Publication II

Hosseinpour, F. , Amoli, P. V. , Plosila, J. , and Hämäläinen, T.
An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach

Reprinted with permission from
International Journal of Digital Content Technology and its Applications
vol. 10(5), pp. 34-46, 2016.

An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach

[1]Farhoud Hosseinpour, [2]Payam Vahdani Amoli, [3]Juha Plosila, [4] Timo Hämäläinen, and [5]Hannu Tenhunen
*[*1, 3 and 5] Department of Information Technology, University of Turku, Finland.*
{farhos;juplos;hatenhu}@utu.fi
[2 and 4]Faculty of Information Technology, University of Jyväskylä, 40100, Jyväskylä, Finland.
[2] pavahdan@student.jyu.fi
[5] timo.t.hamalainen@jyu.fi

Abstract

The Internet of Things (IoT) is widely used in advanced logistic systems. Safety and security of such systems are utmost important to guarantee the quality of their services. However, such systems are vulnerable to cyber-attacks. Development of lightweight anomaly based intrusion detection systems (IDS) is one of the key measures to tackle this problem. In this paper, we present a new distributed and lightweight IDS based on an Artificial Immune System (AIS). The IDS is distributed in a three-layered IoT structure including the cloud, fog and edge layers. In the cloud layer, the IDS clusters primary network traffic and trains its detectors. In the fog layer, we take advantage of a smart data concept to analyze the intrusion alerts. In the edge layer, we deploy our detectors in edge devices. Smart data is a very promising approach for enabling lightweight and efficient intrusion detection, providing a path for detection of silent attacks such as botnet attacks in IoT-based systems.

Keywords*: Intrusion Detection Systems, Smart Data, Fog Computing, Internet of Things*

1. Introduction

International commerce is developing all the time, which puts pressure on the whole supply chain. Logistics is one of the main factors in a supply chain, and assuring the safety of logistic systems has become one of the critical challenges at the national, European and global levels. Especially, food safety and security has become one of the major concerns for manufacturers and end users. China is utilizing a system for its national food industry that enables a citizen to track the whole supply chain of food and drinks. The European Commission also envisions an integrated approach to food safety aiming to ensure a high level of food safety within the European Union through coherent farm-to-table measures and adequate monitoring [1]. Moreover, food and beverages industries are keen to prove the quality and authenticity of their products from farmland to a dining table. In Figure 1 a holistic view of the supply chain is shown. The safety of the transportation needs to be guaranteed throughout the entire supply chain. Thus, having a system that is capable of protecting the supply chain is of utmost concern - particularly in international markets.

Along with the ever-increasing demand for technological solutions for supply chain management systems, Internet of Things (IoT) has been identified as a promising technology to ease the management and monitoring of the system. On the other hand, such technologies increase cyber security concerns for supply chain and logistic systems. Furthermore, the usage of commercially available off-the-shelf (COTS) components or just-in-time (JIT) manufacturing processes increases the security threats as most of them originate from unsecured foreign facilities. A single failure in a logistics system may result in significant consequences for the shipping materials or human being. A failure can be caused by adversaries who try to compromise the functionality of a system by disrupting software, hardware, physical environment or its connectivity. The attack model for cyber-physical systems comprises short and long term attacks. In short term attacks, adversary immediately tries to disrupt the system and cause a failure. The second type is more sophisticated and difficult to detect as the adversary tries not to leave any footprint by disturbing the system's functionality, before fully

[1] Corresponding Author: Email: farhos@utu.fi

intruding to several components to launch a distributed attack. An intrusion detection system (IDS) is required to tackle cyber threats in logistic systems.

In IoT applications such as cyber-physical systems, safety and security of monitoring a physical environment is a critical issue that is underestimated in recent and current studies. First, unlike in typical computer systems, in IoT systems the physical environment can be affected through IoT actuators. Second, attackers can affect a cyber-physical system by manipulating the physical environment. Moreover, as we are dealing with resource constrained devices in IoT, lightweight approaches need to be undertaken to ensure the quality of service and feasibility of such security measures.

The remainder of the paper is organized as follows. In Section **2** we briefly review the IoT and fog computing technologies. In Section 3 we discuss IDS systems. In Section 4 we present a new promising approach for lightweight IDS called Smart Data. Section **5** presents the proposed Artificial Immune System (AIS) based IDS. In Section 6, the experimental results of the proposed architecture are presented and, finally, we end with some concluding remarks in Section **7**.

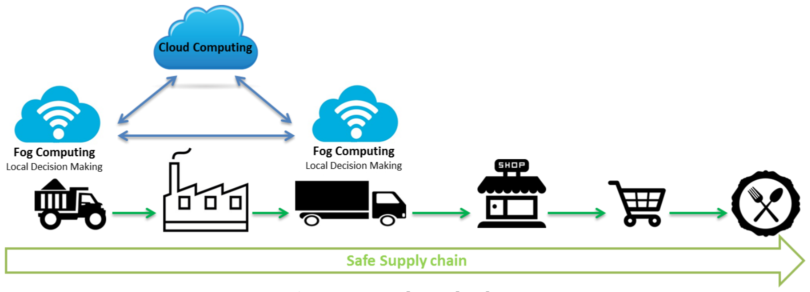


Figure 1. Food supply chain

2. Internet of Things and Fog Computing

The last decade has witnessed the wide deployment of IoT technology in various application domains, and its pervasive role will continue to strengthen in the future. IoT is a concept that realizes communication and control among a very large set of different devices [2]. By connecting the devices, such as sensors, communication devices and data processing units, IoT allows distributed, autonomous decision making and intelligent data processing and analysis [3]. The cloud computing is being recognized as a success factor for IoT, providing ubiquity, reliability, high-performance and scalability [4]. However, because of its geographically centralized nature and communication implications, cloud computing based IoT fails in applications that require very low and predictable latency and which are geographically distributed, fast mobile, or large-scale distributed control systems [5]. Fog computing provides a promising technology to tackle the low-latency and geographical distribution required by IoT devices.

Fog computing is a paradigm extending cloud computing and its services to the edge of the network as shown in Figure 2. Fog is distinguished from the cloud in its proximity to end-users/nodes, dense geographical distribution, support for mobility, heterogeneity, interoperability and pre-processing. Fog computing does not replace cloud computing. On the contrary, it complements cloud computing and aims to provide a computing and storage platform physically closer to the end nodes, provisioning a new breed of applications and services with an efficient interplay with the cloud layer [5]. The expected benefit is a better quality of service for applications that require low latency. Lower latency is obtained by performing data analysis already at the fog layer. This data analysis is lightweight, and therefore more advanced analyses and processing will be done at the cloud layer. Naturally, some applications do not require real-time computation, or they need high processing power, and therefore they are performed at the cloud layer. Fog devices, located at the fog layer, are heterogeneous in nature, ranging from end-user devices and access points to edge routers and switches allowing their use in a wide variety of environments.

The fog layer embodies software modules in the form of fog services and embedded operating systems. At the fog layer, it is also possible to analyze gathered data obtained from the sensor layer and thus make decisions locally. Local decision making is a key to reduce latency, and thus to provide quick responses to unusual behaviors, for example, in the case of security and safety breaches. Such local processing requires lightweight, energy efficient algorithms and applications that can be performed nearby the source of data.

In a systematic view, fog computing is composed of distributed and heterogeneous resources that are deployed based on a hierarchal model. In this model, the fog nodes constitute a virtualized and hierarchical topology and provide a distributed computing platform. Each physical node is composed of computing and storage components and has interfaces for communication with neighboring fog nodes at the same, one step higher, or one step lower level of the hierarchy. Figure 3 illustrates a hierarchical architecture of physical fog computing nodes at different levels [4].

The fog computing platform shown in Figure 2 gathers data from different sensors via a wireless communication module using wireless communication protocols such as Wi-Fi, 6LoWPAN and Bluetooth Low Power depending on the used application. The gateway uses the data in the analysis to anticipate and spot any abnormal behavior in the system. In the case of food transport, the intelligent gateway gathers the food quality data from several sensor nodes. The raw food quality information can be transformed into information (structured data) by developing fog computing services. Naturally, the priority of the safety services is higher that of the other services to ensure prompt action for all threats.

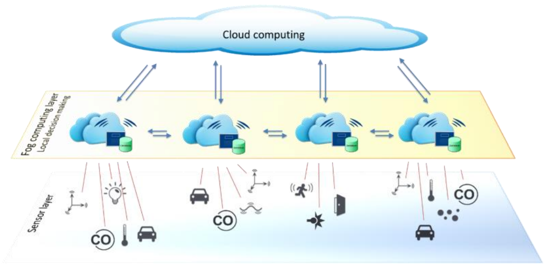


Figure 2: Fog computing platform

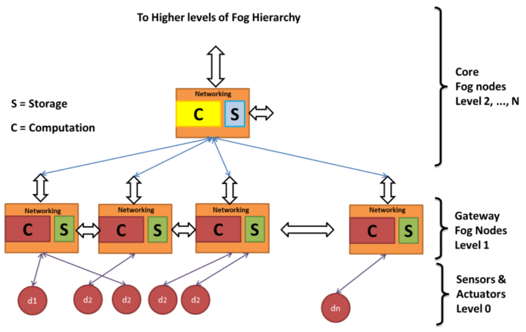


Figure 3. Hierarchical architecture of fog computing.

IoT is not only applied to consumer electronics such as wearables and sport gadgets but also, in the industry in various applications including building/home automation, smart cities, smart manufacturing and smart logistics. In logistics, safety is essential given the mission-critical deployments of logistic systems. ISO 60601 defines safety as “the avoidance of hazards to the physical environment due to the operation of a device under normal or single fault condition.”

3. Intrusion Detection Systems

Detection of silent attacks requires constant monitoring and behavioral analysis of the system’s components and communication. Therefore, precise and swift safety monitoring and intrusion detection are of utmost importance in IoT-based logistic systems. An Intrusion Detection System (IDS) with local decision making will prevent failures caused by adversaries and decide proper alert to prevent intrusion or to mitigate the impact of an intrusion. Anomaly-based IDS have been broadly studied as defensive techniques to address the detection of unknown or zero-day attacks. Unlike misuse-based or signature-based types of IDS, which take advantage of the predetermined signature of known attacks, an anomaly-based IDS deals with the detection of new types of attacks that are unknown to the system [6]. This process is done by detecting variation in the systems' behavior from a previously defined normal system profile.

The artificial immune system (AIS) comprises promising techniques in the form of biologically inspired computing that are applied for solving various problems in the field of information security including IDS. The AIS is inspired by the human immune system (HIS), which has the ability to distinguish internal cells and molecules of the body from foreign pathogens, so called self and non-self respectively, and to protect the body against diseases [7]. Like HIS protection against foreign pathogens in the human body, the AIS suggests a multi-layered protection structure for protecting computer networks against attacks [8]. This protection is accomplished through the Innate or Adaptive mechanisms. The innate immune response is immediate and the first line of defense for the HIS and provides a non-specific protection. Therefore, it has no prior knowledge of specific outsiders. The adaptive immune response, on the other hand, is antigen-specific and trained using a pre-defined profile of specific attacks [9]. This is done through specific autoimmune cells called antibodies which act as agents of HIS in the body. The adaptive immune system also includes a “memory” that makes future responses against a specific antigen more efficient [10]. Likewise, the AIS is also proposed as a distributed agent-based system [11]–[13] that is composed of a set of detectors generated as a result of a self-training phase in its main engine. We presented an architectural design of a distributed AIS system in [14]. The AIS comprises three main parts:

1) A training engine that has the responsibility of learning from an initial learning data set and training its detectors. Such processes are complex and require powerful processing units to enable a real-time monitoring of the system. Hence, a cloud computing at the center of the network is the best option to set up the training engine. Training the detectors is an intensive task that is done at the initialization phase of the AIS at the center of the network. Hence, it does not need much communication with the nodes at the edge of the network at this phase. The training process is done through a negative selection algorithm which we presented in our previous work [15]. In the analyzing phase, in turn, the AIS requires more communication between the infected nodes and the main engine.
2) An analyzer engine has the responsibility of analyzing anomalies detected by the detectors to come up with an intrusion alert or to reject the false positive signals. To increase the detection precision, an optimized detector that is called a memory cell detector is generated. Based on our previous works [15], [16], we have utilized genetic algorithms to generate the most optimized memory cell detectors based on the profile of the reported attack and also the triggered detectors. Since the analyzer engine requires more communication with the edge devices, we deploy the analyzer engine on the fog layer
3) A set of detector sensors that are accommodated in each node executing the monitoring task. The detectors are distributed in the network providing an intelligent and collaborative monitoring of the network and the computing nodes. Because each type of attack could be carried out in various forms, to increase the precision of the detection, at the learning phase, a number of different detectors are generated for each attack, and the best detectors, which have more affinity with the targeted attack profile, are selected. So, each type of attack could be detected by a number of detectors. Once an anomaly occurred in a node, a number of detectors will be triggered by the anomaly. If the number of the triggered detectors is more than a threshold, then the anomaly will be reported to the analyzer engine for further analyses. In this case, based on the result provided by the analyzer engine, a more accurate intrusion alert will be given.

Like other anomaly-based detection techniques, the AIS also takes advantage of monitoring variations of the system's behavior as an adaptive immune response, according to a pre-defined normal activity profile. This is done through a learning phase in which a data set containing these profiles is utilized for this purpose. Hence, the efficiency of anomaly detection in the AIS depends heavily on the learning data set. Substantial studies have been conducted so far for improvement and utilization of AIS-based IDS, the majority of which have utilized a pre-defined and offline data set as the learning data for training the IDS. This will reduce the efficiency of the IDS and limit the knowledge base to that particular learning data set. Moreover, it is extremely difficult to create a data set of self-samples with all variations. To cope with this problem, we proposed an online self-training method for our AIS based IDS in [16] using unsupervised machine learning methods, which act as an innate immune response. The innate immune system provides an online and dynamic categorization of network flows into self and non-self flows, which is then used by the adaptive immune system to generate attack-specific detectors.

In integrating the AIS technology in IoT and fog computing systems, a challenging task is building a lightweight smart agent system that is computing and energy efficient and also requires less communication to save the network bandwidth. To this end, we take advantage of the smart data approach, which we presented in [17].

4. Smart Data

Smart data is an active and intelligent data structure using a fog computing system, which facilitates the management of Big Data in IoT-based applications. Such a data cell is initially very simple and light-weight, but it evolves (grows) when traveling through the hierarchical fog computing system towards the cloud, merging with other cells or vice-versa, if the data moves from the cloud towards the actuators.

Figure 4 illustrates the general structure of a smart data cell. The smart data is composed of three main parts: payload data, metadata, and virtual machine. The payload contains the main data collected from the sensors. It undergoes a series of processing or pre-processing steps and is thereby converted into more meaningful information. The metadata part of smart data contains key information such as the source of data (sensors), the destination of data, the physical entity which data belongs to, timestamps, current status and logs as well as rules for accessing, fusing or diffusing, and processing data, for example. The virtual machine part, in turn, acts as a platform which enables and manages the execution of the rules specified in the metadata part. The VM at the very beginning stage contains only basic application codes. Then, it evolves by adding other code modules of the application when they are needed. Each code module provides specific functionalities and services to the smart data. The modular structure of the VM component makes smart data extendable, allowing it to manage the overhead of carrying the code by removing unnecessary code modules and adding the required modules only when they are needed. To enable this, we consider a remote code repository node which contains all necessary code modules as plugins. Whenever a smart data cell requires a specific code module, it communicates with the code repository node and requests for the required code module. To minimize the communication involved in downloading the plugins, the most recently downloaded plugins are also cached in the physical fog nodes for some period of time. So, if the requested code module does not exist in the local fog node, it will be downloaded from the remote code repository node. We have presented a detailed design and specification of our smart data in [17].

Indeed, the smart data acts as a software agent that is able to travel through a fog computing and IoT network, monitor, gather data, and process/pre-process them. The main objective of encapsulating a set of data already at the sensor level, instead of constantly sending discrete data, is to reduce the communication overheads in a very resource constrained environment as well as to reduce the data velocity in the Big Data context. To enable lightweight intrusion detection through the AIS system in a fog computing based IoT environment, we utilized the smart data as a package of suspected anomaly which is needed to be processed and determine if a real intrusion happens.

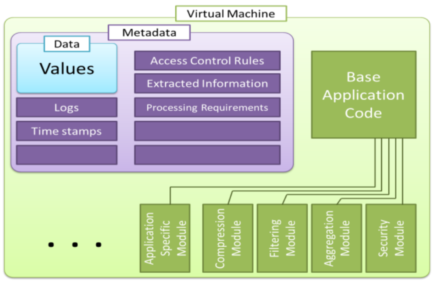


Figure 4. Structure of Smart Data

5. Proposed Intrusion Detection System

As discussed earlier, the training phase in the AIS main engine involves more intensive processes without a significant need for communicating with the distributed nodes at the edge of the network. Hence, the cloud computing at the center of the network is the proper computing platform to run such computations. In contrast, detection by the trained detectors requires less computing and a higher amount of communication compared to processes in the main engine. According to this strategy, we have developed our IDS architecture based on the three-layered structure of fog-based IoT systems (Figure 5). Based on this architecture, the cloud computing accommodates the IDS main engine which is composed of two sub-engines called a clustering engine and a training engine. The clustering engine, using unsupervised clustering methods, divides the primary network traffic into self (normal) and non-self (intrusion) packets which are used as the online training data set for our AIS based IDS. The training engine, in turn, trains a set of detectors based on the learning data obtained from the clustering engine by using a negative selection algorithm. These detectors are called primary detectors. The primary detectors, after the training phase, are stored in a detector repository database at the cloud and also distributed to the devices at the edge of the network.

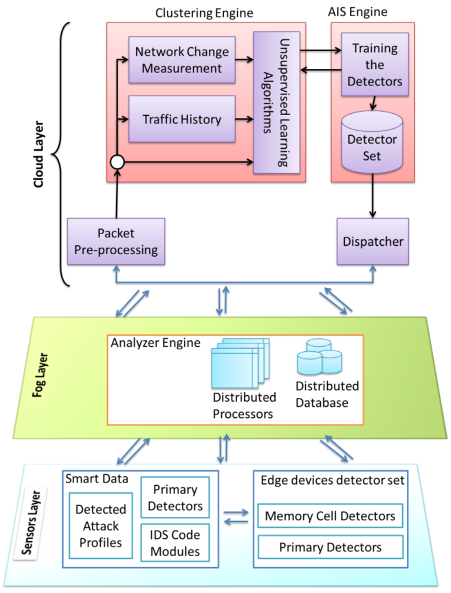


Figure 5. Proposed IDS architecture.

The primary detectors at the edge of the network act as sensors for our IDS which monitor the behaviour of the edge devices. If an anomaly is detected by any of these sensors, they initiate a process for investigating the anomaly by producing a smart data cell. Such smart data contains the information about the suspected connection in its payload and the triggered primary detector in its metadata. In order to increase the precision and avoid false alarms, the smart data is sent to the fog layer for investigation only if the number of triggered detectors exceeds a threshold. Based on our previous work [16], we set this threshold to three. The smart data in the fog layer will fetch the required code modules to build an optimized detector called a memory cell detector for detecting that particular type of attacks. The memory cell detector will be sent to the detector repository in the cloud and from there it will be distributed to all the other devices at the edge.

In this case, the connections that do not trigger a sufficient number of detector sensors will be omitted. So, the IDS will not be able to monitor the trend of the system and detect the long-term or silent attacks. The silent attacks, unlike the short-term attacks, are launched over a longer period of time and from distributed nodes, while keeping the system's functionality as normal as possible to make it difficult and more sophisticated to detect. To cope with this problem, we take advantage of the smart data concept. Smart data has the ability to store and encapsulate the sensory data (profile of a suspected connection detected by any of detector sensors) over a time. We introduce the time dimension to enable the IDS to detect the long-term attacks. If the number of triggered detector sensors is more than a predefined threshold, the smart data cell will be sent to the fog computing platform for further analysis. Otherwise, if the number of triggered detector sensors is less than the threshold, the information of a suspected attack will be stored in a smart data cell. In this case, after a particular time interval, the smart data cell will be sent to the fog computing. In the upper level, the smart data cell will be aggregated with other smart data cells coming from other devices. So, if the similar anomaly had occurred in another device, the aggregated smart data will include the profile of suspected connections collected from a larger amount of devices over time. In a similar way, smart data will be aggregated at the higher levels of the fog hierarchy (Figure 3) with other smart data collected from a larger geographical area. In this case, the smart data become mature which means it contains the information of suspected connections from a larger amount of distributed devices over a longer period. Hence, the silent attacks will become more visible. Once the number of similar attacks that are aggregated and collected by smart data becomes larger than a threshold, the smart data will fetch the code module for analyzing the attack. In this phase, if the attack pattern is similar in all suspected anomalies then the system will alert an intrusion.

In the following subsections the detailed functionalities of each component are discussed:

5.1. Clustering Engine

To detect unseen intrusions without using any previous knowledge (training by labeled traffic or signature), we introduce a clustering engine as an innate immune response. The clustering engine employs the DBSCAN clustering technique to classify the real network traffic into clusters and count them as self, while behaviors outside of the clusters will be deemed as noise or non-self. For this purpose, the engine continuously compares the number of network flows in different network resolutions (subnets of /0, /8, /16, /24), with a threshold which is dynamically computed by our suggested network measurement formula in Table 2. Since high-speed networks have a higher amount of traffic, there is a notable probability of missing the sign of network attacks. To overcome this issue, the system will also control the behavior of the network in small resolutions to minimize the possibility of fading the attacks in the regular traffic.

To obtain a precise threshold, the system requires determining the past behavior of the network. It is probable that small attacks to be fade with the existence of large attacks, thus we have applied standardization on the number of network flows by using logarithm (Log) to increase the probability of detecting small attacks during the existence of large attacks. To determine changes in the network traffic, the system will calculate the "standard deviation" of the number of network flows in different windows from last minute of the traffic. As shown in Table 1 the previous 60 seconds of traffic is broken into four 15 seconds windows. For instance, δ_1 is the standard deviation of the number of network flows in the first window which is from the last 65 seconds to the last 50 seconds of the previous network traffic. As it has been seen in so many datasets, it takes 2 to 3 seconds from starting time of the network attacks (such as DOS/DDOS attacks) till its own peak. To reduce the impact of initial traffic of the attacks on the threshold, the network measurement formula considers a 5 seconds gap between every one minute of traffic to calculate the threshold for the current traffic. The sum of the highest standard deviations (from δ_1 to δ_4) and the heaviest traffic from the last minute of traffic can determine the highest traffic which could be accepted as normal. The following equation represents the network measurement formula which calculates the network traffic threshold "T_{nt}":

$$T_{nt} = \left(Max\,\{\delta_i | i = 1 \cdots 4\} + Max\,\{X_j \log X_j \,| j = 1 \cdots 60\}\right) \times \gamma$$

Where δ_i is standard deviation of number of network flows in i_{th} window and X_j is number of network fellow in jth second of last minute's traffic. And γ is a coefficient value which can be set to determine the final threshold.

Table1. Elements of network measurement formula

Last Minute Traffic				*Gap*	*Current Traffic*
Window 1	Window 2	Window 3	Window 4		T_{nt}
δ_1	δ_2	δ_3	δ_4		
65-50	50-35	35-20	20-5	5-0	
$X_j \log X_j$					

The DBSCAN algorithm requires two parameters: the maximum radius of the neighborhood (β) and the minimum number of samples required to form a cluster (α). The real network contains traffic from different classes of users such as normal users, busy users, and servers. In general, the number of busy users and servers is smaller than α thus they may not form a cluster in DBSCAN. Since the proposed model in our previous work [16] considers all of the network behavior (in the clean traffic windows) as normal, this will increase the acceptable distance β for DBSCAN by a high value of Δ to include all of the points inside the nearest cluster. Clustering the data with a high value of acceptable distance increases the false negative rate (FNR) in certain cases. To overcome this issue, we will propose a new method which compares the previous behavior of outliers to distinguish normal high traffic users from intrusions.

Similar to the proposed model in our previous work [16], whenever the volume of network flows passes the threshold, the cluster engine uses the DBSCAN to cluster the in-bounded and out-bounded network flows for each machine to find the attacker/s.

During the training phase, DBSCAN will obtain the most accurate α and β from the most recent clean network traffic. In our proposed model, the network traffic can be considered "clean" if it occurs before the threshold (T_{nt}) raise the alarm. Technically, the normal users will form into clusters while the density of busy users or servers may not reach the required level. Nevertheless, since training phase uses the clean network traffic the proposed model will consider outliers as busy machines with normal profiles.

Afterward, to find the anomalous outliers which caused the high volume of network traffic, the clustering engine clusters the suspicious network traffic window. The outliers' IP addresses from detection phase will be compared to their previous profile. If the distance of current behaviors and the previously seen behavior does not exceed the acceptable distance β, the clustering engine will mark it as normal high traffic machine. Otherwise, if the new behaviors of outliers IP exceed the distance it will consider the behavior of that machine as abnormal. It is important to note that if the outlier IP addresses do not have any profile from the training phase, the clustering engine will mark it as abnormal.

5.2. Training Engine

The training engine has the responsibility of training the primary detectors of the IDS. As discussed earlier, because training of the detectors does not require much communication with the edge of the network, this component is located in cloud computing at the center of the network. The training engine first transforms the network flow information into binary strings with a total 112-bit length as a flow profile (Table 2). Then, utilizing a negative selection algorithm, it trains and creates the primary detectors. The negative selection algorithm first creates some random detectors (immature detectors) and then trains them with samples of marked flows from the cluster engine. If any immature detector matches with any self-sample of the data set, then the system will drop it and create another in its place. After checking all of the immature detectors with all self-samples, the remaining immature detectors undergo the next step of the negative selection algorithm and become mature detectors. Each mature detector will be checked with all non-self samples of labeled flows. If a mature detector fails to match with some non-self samples, the system will discard this detector; otherwise, this detector will be added to the final detector set. This process will continue until all non-self packets are matched with at least three mature detectors.

The negative section algorithm utilizes the r-Contiguous matching bit role proposed in [18] to check the matching between two strings. In this method, two strings are matched if they have at least r contiguous identical bits. Finally, the output of the negative selection algorithm is a set of primary detectors, which are archived and synchronized in a detector set repository in the cloud and then distributed to the edge devices. These detectors are analogous to primary immune response in the HIS.

Table 2. Depiction of fields in flows profile strings

Name of the Field	Minimum and Maximum Value	Binary Strings Length (bits)
Destination IP Address	0.0.0.0 - 255.255.255.255	32
Source IP Address	0.0.0.0 - 255.255.255.255	32
Destination Port No	0 – 65535	16
Duration	0 – 65535	12
Protocol	0 – 65535	4
Source Port No	0 – 65535	16

5.3. Analyzer Engine

The analyzer engine has the responsibility of analyzing detected anomalies and giving intrusion alert. It employs the proposed genetic algorithm in [19] to evolve the highly fit detectors activated when an anomaly has been encountered. The analyzer engine requires more communication with the edge devices so we deploy the analyzer engine in the distributed fog computing at the edge of the network. In order to save the deployment cost of the analyzer engine in the fog computing, we take the advantage of the modular structure of the smart data cells. Indeed, we deploy the analyzer engine in code repositories in fog computing. If a smart data which is sent to the fog computing needs the processes of this engine, it fetches the required code module from the nearest code repository. The smart data contains 1) the suspected flow reported from the hosts, 2) profile of the activated detectors, and 3) their affinity with reported flow. It utilizes the fetched code modules to analyze the anomaly and generates an optimized detector, called memory cell detector. A memory cell detector is a high-affinity and attack-specific detector with a higher detection ability and analogous to secondary immune response in the HIS [10]. The following operations are carried out in this case:

A selection operation is undertaken on activated detectors to select the detectors with the highest affinity for cloning and formation of primary population for genetic algorithm. Those detectors having a fitness value greater than or equal to cloning threshold undergo cloning. The cloning threshold is set as follows.

$$\text{Cloning Threshold} = \frac{\sum_{i=0}^{n} \text{Fitness of detectors}}{n}$$

Where "n" is the total number of activated detectors.

Winner detectors that consist of the cloned detectors and remaining activated detectors are subjected to the genetic operators of Mutation, Crossover, and Reproduction, which facilitates the evolution of these detectors. This process is repeated and continued for a few generations until a detector with a fitness value higher than all the winner detectors is generated. The optimized detector from the genetic algorithm is treated as a memory cell.

5.4. Host Side Detectors

The detectors are distributed to the edge devices. Figure 6 illustrates the architecture of edge devices that comprises of sensors, actuators, connection platforms, and processing units. The IDS detectors in the edge devices act as sensors for our IDS. All inbound and outbound network flows are checked using these sensors. In each device, we consider two detectors as follows.

Primary Detectors: comprise a set of trained detectors that have the ability to discriminate between self and non-self flows. These detectors are non-specific and responsible for the primary immune response for anomalies that occur for the first time. If a flow matches a detector with an effective affinity, that detector is considered an activated detector and the flow is suspected as an intrusion. To improve the accuracy of detection and reduce the false-positive errors in IDS, we have defined an intrusion threshold (T_i). If the number of activated detectors by a suspected flow is more than T_i, the flow is detected as an intrusion.

Memory cell detectors: composed of a set of optimized detectors generated by the analyzer engine. As the secondary response of the AIS, memory cells have more accurate intrusion detection abilities. Hence, any flow that activates any of these detectors is treated as an intrusion.

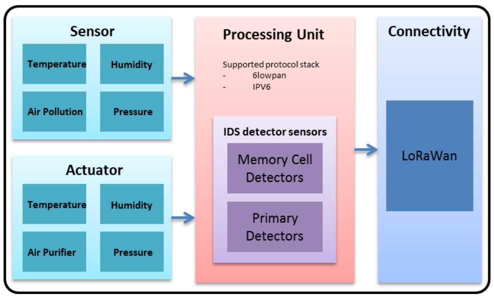


Figure 6. Architecture of Edge Devices

6. Experimental Result

To evaluate the efficiency of two popular clustering algorithms, we utilized KDD-Cup 99 data set, which is extracted from DARPA-98 traffic network. In addition, we have tested our model on SSH Brute Force from ISCX dataset [20]. Since today most of the servers with SSH protocol limit the number of user attempt, we have changed the SSH Brute Force attack in ISCX to a distributed model, which a various number of bots have participated in it. Figure 7 show the network's behavior during the attack. As shown in Figure 7 (A) the ratio of outbound flows to the threshold is below one because the number of attackers is high. However, in Figure 7 (B) the threshold for inbounded traffic raise alarm since all of the traffic goes to the limited number of machines.

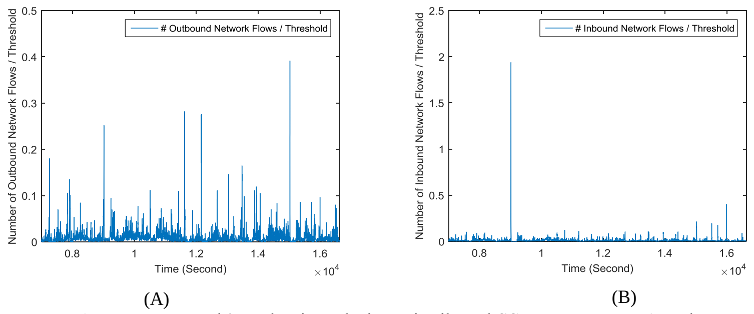


Figure 7. Network's Behaviour during Distributed SSH Brute Force Attack

Figure 8 shows the self-training phase during distributed SSH Brute Force attack. As mentioned before the clustering engine marks the IP addresses of the machines which were located inside the clusters as normal. However, the IP addresses of outliers will be profiled as busy users or servers. As shown in Figure 9 during the comparison phase all of the outliers will be compared to their previous history. If the distance does not exceed the threshold, the cluster engine will mark them as normal devices (with high traffic). Otherwise, if the device exceeds its traffic abnormally, the clustering engine will mark it as the abnormal device. Figure 10 shows the final decision of clustering engine.

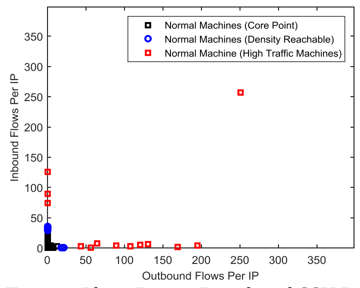


Figure 8. Self-Training Phase During Distributed SSH Brute Force Attack.

Table 3 shows the comparison of average performances of the new proposed model and our previous work. To evaluate the performance of “different behavioral classes” feature in the new proposed model we have added traffic from busy users and servers during the occurrence of an intrusion. Since the proposed model compares the behavior of outliers with their previous history, the overall performance was higher than our previous proposed model [16].

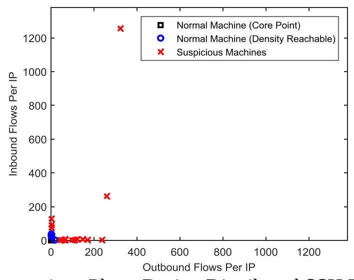


Figure 9. Comparison Phase During Distributed SSH Brute Force Attack.

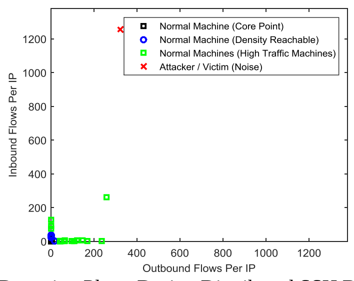


Figure 10. Detection Phase During Distributed SSH Brute Force Attack.

Table 3. Performance Evaluation.

	New proposed model	**Previous Proposed model** [16]
False positive rate	3.51%	4.53%
True negative rate	96.49	95.47%
Accuracy	98.35%	96.23%
Recall	100%	95.37%
Precision	97.83%	91.21%

To examine the effectiveness of the AIS engine and thus the proposed model, in our test, we set the fitness value of "rc" for R-Contiguous matching bit algorithm to 13 and the threshold "Ti" to 3. Furthermore, with experimenting the genetic algorithm for the formation of memory cells, in different circumstances, the values for the probability of genetic operations of Crossover, Mutation, and Reproduction have been set to 30%, 40% and 30% respectively. The system is examined in both centralized and distributed mode. Figure 11 corresponds the self-improvement rate of AIS based IDS in the central and distributed forms. According to this chart, the self-improvement rate in distributed mode is better than centralized mode and it reaches to its steady maximum amount after only 6 cycles, while this happens after 10 cycles in centralized mode. This is due to the dynamic distribution and synchronization of recently created memory cells to each device.

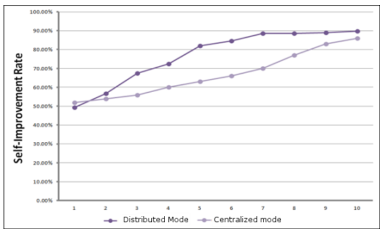


Figure 11. Comparison of self-improvement rate in distributed and centralized mode.

7. Conclusion

Development of lightweight intrusion detection systems is critical for the safety and security of advanced IoT-based logistics systems. In this paper, we presented a new lightweight architecture for an AIS based IDS for IoT systems. This paper extends our previous work [16], into a three-layered structure of IoT systems including the cloud, fog and edge layers. We utilized our proposed smart data approach to develop a lightweight and efficient analyzing engine in fog computing platform for our IDS. Smart data is a very promising framework for enabling lightweight and efficient intrusion detection providing also a path for detection of silent attacks such as botnet attacks in IoT-based systems. We also presented a new approach for clustering the primary network connections which is more efficient method than the one used in our previous work. Our future work will mainly focus on detection of potential botnet attacks using the smart data technology.

Acknowledgement

This work was supported by Turku University Foundation and EIT Digital.

8. References

[1] "Food Safety: overview," *European Commission*, 2016. [Online]. Available: http://ec.europa.eu/food/index_en.htm.

[2] Y. J. Guo-Zhen TAN, Hao Wang, "IoT-based Distributed Situation Awareness for Traffic Emergent Events," *Int. J. Adv. Comput. Technol.*, vol. 5, no. 7, pp. 1050–1059, 2013.

[3] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet : The Internet of Things Architecture , Possible Applications and Key Challenges," in *10th International Conference on*

Frontiers of Information Technology Future, 2012.

[4] A. R. Biswas and R. Giaffreda, "IoT and cloud convergence: Opportunities and challenges," *2014 IEEE World Forum Internet Things*, pp. 375–376, Mar. 2014.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*, vol. 546, N. Bessis and C. Dobre, Eds. Cham: Springer International Publishing, 2014, pp. 169–186.

[6] S. and G. G. Feixian, "Research of Immunity-based Anomaly Intrusion Detection and Its Application for Security Evaluation of E-government Affair Systems.," *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 20, pp. 429 – 437, 2012.

[7] L. N. de Castro and J. Timmis, *Artificial {I}mmune {S}ystems: A {N}ew {C}omputational {A}pproach*. London. UK.: Springer-Verlag, 2002.

[8] M. Tan, H. Yu, Z. Zhao, Z. Liu, and F. Liu, "An artificial immunity-based proactive defense system," in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, 2007, pp. 2239–2243.

[9] et al. Xishuang, D., "Multi-word-Agent Autonomy Learning Based on Adaptive Immune Theories," *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 7, no. 3, pp. 723–745, 2013.

[10] A. A. Ademokun and D. Dunn-Walters, "Immune Responses: Primary and Secondary," in *eLS*, John Wiley & Sons, Ltd, 2001.

[11] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer Immunology," *Commun. ACM*, vol. 40, no. 10, pp. 88–96, Oct. 1997.

[12] F. Hosseinpour, K. A. Bakar, A. H. Hardoroudi, and N. Kazazi, "Survey on Artificial Immune System As a Bio-inspired Technique for Anomaly Based Intrusion Detection Systems," in *Proceedings of the 2010 International Conference on Intelligent Networking and Collaborative Systems*, 2010, pp. 323–324.

[13] S. A. Hofmeyr and S. A. Forrest, "Architecture for an Artificial Immune System," *Evol. Comput.*, vol. 8, no. 4, pp. 443–473, Dec. 2000.

[14] F. Hosseinpour, K. A. Bakar, A. Hatami Hardoroudi, and A. Farhang Dareshur, "Design of a new distributed model for Intrusion Detection System based on Artificial Immune System," in *Advanced Information Management and Service (IMS), 2010 6th International Conference on*, 2010, pp. 378–383.

[15] F. Hosseinpour, A. Meulenberg, S. Ramadass, P. Vahdani Amoli, and Z. Moghaddasi, "Distributed Agent Based Model for Intrusion Detection System Based on Artificial Immune System," *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 7, pp. 206–214, 2013.

[16] F. Hosseinpour, P. V. Amoli, F. Farahnakian, and J. Plosila, "Artificial Immune System Based Intrusion Detection : Innate Immunity using an Unsupervised Learning Approach," *JDCTA Int. J. Digit. Content Technol. its Appl.*, vol. 8, no. 5, pp. 1–12, 2014.

[17] F. Hosseinpour, P. Juha, and H. Tenhunen, "Smart Data: Reshaping Data Structure in IoT for Tackling the Five Vs of Big Data using Fog Computing." TUCS Technical Reports 1159, 2016.

[18] T. Stibor, "Foundations of r-contiguous Matching in Negative Selection for Anomaly Detection," *Nat. Comput.*, vol. 8, no. 3, pp. 613–641, Sep. 2009.

[19] D. Dal, S. Abraham, A. Abraham, S. Sanyal, and M. Sanglikar, "Evolution Induced Secondary Immunity: An Artificial Immune System Based Intrusion Detection System," in *Computer Information Systems and Industrial Management Applications, 2008. CISIM '08. 7th*, 2008, pp. 65–70.

[20] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

Publication III

Hosseinpour, F. , Plosila, J., and Tenhunen, H.
An Approach for Smart Management of Big Data in the Fog Computing Context

An Approach for Smart Management of Big Data in the Fog Computing Context

Farhoud Hosseinpour
Department of Information Technology
University of Turku, Turku, Finland
Email: farhos@utu.fi

Juha Plosila
Department of Information Technology
University of Turku, Turku, Finland
Email: juplos@utu.fi

Hannu Tenhunen
Department of Information Technology
University of Turku, Turku, Finland
Email: hatenhu@utu.fi

***Abstract*—In this paper, a new approach for tackling Big Data in Internet of Things (IoT) systems is presented. We approach the problem from the data perspective rather than only focusing on the computing platform. We design and develop a concept that we call Smart Data. Taking advantage of a hierarchical fog computing system, we reshape the raw and passive form of the data generated by IoT sensors to intelligent and self-managed data cells that are able to evolve and become more meaningful information with reduced size. We believe that smart data will revolutionize the current perspective of data and will open many potential research directions to tackle emerging big data issues.**

***Index Terms*—Smart Data, Big Data, Fog Computing, Internet of Things**

I. INTRODUCTION

Big data is currently becoming a critical research focus along with the growth of Internet of Things (IoT) and Internet of Everything (IoE) technologies. Relying on current technologies such as cloud computing is not efficient for addressing the requirements of big data management [1]. Hence, new technologies are needed to reduce the complexity, ease management and boost the processing of big data. Preprocessing of raw sensory data is one of the most efficient ways to reduce the load of big data in cloud computing. To this end, a fog computing platform at the edge of the network is introduced to reduce the processing load from the cloud by delegating some simple and frequent tasks to the fog (pre-processing) [2]. A virtualized and hierarchical architecture of fog computing provides a distributed computing and storage platform near to the edge sensors for local and latency sensitive applications. In this model, the raw data generated in edge sensors is pre-processed in the local fog, and more meaningful and efficient data with reduced volume and velocity are sent to the cloud for further and global processing and storage. Expected benefits of utilizing the fog computing technology in the IoT architecture include: local and hence faster processing and storage for geo-distributed and latency sensitive applications, reduced communication overhead, and reduced volume and velocity of big data before sending it to the cloud.

Relying merely on the advancement of computing technologies for big data processing and management will not completely address the involved issues. In this doctoral project, approaching the problem from a different perspective, we aim to reshape the raw, passive and unstructured form of data in IoT to an intelligent and active form, while preserving and enhancing many other important parameters such as energy efficiency, scalability, throughput, quality of service as well as privacy and security. For this, we will introduce a new intelligent, self-managed and lightweight data structure that we call Smart Data. We believe that smart data has potential to revolutionize the current perspective of data and will open many potential research directions to tackle emerging big data issues. The smart data is a package of encapsulated structured data generated by IoT sensors, a set of metadata, and a virtual machine. It is controlled and managed through the metadata that accommodates a set of rules that define its behavior and govern its security, privacy and other functionalities. The virtual machine, in turn, executes the rules set in the metadata.

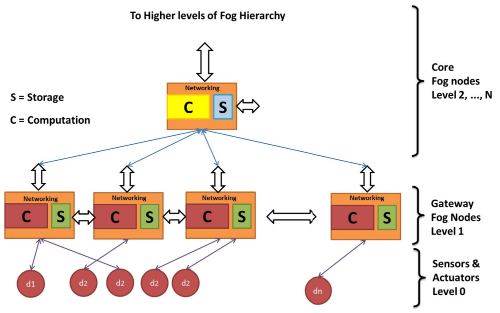
The remainder of the paper is organized as follows. We discuss the fog computing technology and its integration with IoT systems in Section 2. In Section 3 we introduce our smart data model and present its structure and, finally, we discuss and conclude in Section 4.

II. FOG COMPUTING

Cloud computing is being recognized as a success factor for IoT, providing ubiquity, reliability, high-performance and scalability. However, due to its geographically centralized nature as well as communication implications, cloud computing-based IoT fails in applications that require a very low and predictable latency, are geographically distributed, are fast mobile, or are large-scale distributed control systems [2]. Fog computing is a promising technology proposed and developed by Cisco [3], complementing the cloud computing services by extending the computing paradigm to the edge of the network. Fog computing introduces an intermediate layer between the edge network or the end nodes and the traditional cloud computing layer. Bringing the computational intelligence geographically near to the end users will provide new or better services for latency sensitive, location-aware and geo-distributed applications which due to their characteristics are not feasible merely through cloud computing. The expected benefit is faster computation times for requests that require low latency. In addition to the requirement of low latency, fog computing is a promising technology for dealing with big data

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

2330-2186/16 $31.00 © 2016 IEEE
DOI 10.1109/CloudCom.2016.77



To Higher levels of Fog Hierarchy
S = Storage
C = Computation
Networking
C
S
Core Fog nodes Level 2, ..., N
Gateway Fog Nodes Level 1
Sensors & Actuators Level 0
d1
d2
dn

Fig. 1. Hierarchical architecture of fog computing.

generated from IoT-based systems with a vast number of nodes spread across a large area. IoT-based systems introduce a new dimension for characterizing big data, namely geo-distribution, along with its generally known characteristics. Cloud based IoT systems dealing with Big Data [2] have to process a large amount of data at any time. Fog computing, as a middleware, can pre-process raw data coming from the edge nodes before sending them to the cloud layer. As a result, the fog layer not only reduces the amount of work needed in the cloud layer by generating meaningful results from raw data, but also reduces the monetary cost of computing in the cloud layer.

In a systematic view, the fog computing is composed of distributed and heterogeneous resources that are deployed based on a hierarchal model. In this model, fog nodes deploy a virtualized and hierarchical topology and provide a distributed computing platform. Each physical node is composed of computing and storage components and has interfaces for communication with neighboring fog nodes at the same, one step higher, or one step lower level of hierarchy. Figure 1 illustrates a hierarchical architecture of physical fog computing nodes in different levels [4].

III. Smart Data

To tackle the big data in IoT systems we approach the problem from a different perspective to reduce the management efforts, computing load and communication overhead imposed by big data. We design and develop a new data structure by embedding the intelligence required to process the payload into an encapsulated data cell that we call Smart Data. Deployment of application code in each processing fog node is costly and not efficient in fog computing-based IoT systems [5]. Hence, the smart data eliminates this need resulting in saving cost, energy and network traffic as well as better Quality of Service (QoS). Smart data are first generated by sensors in a very basic form and then they evolve through their lifecycle by being processed and converted to meaningful information as well as complemented and amended with new features such as security and privacy. A fog computing platform is the main enabler for implementing our smart data concept.

A. Structure of Smart Data

In general, a smart data element is a cell of encapsulated data consisting of a payload, metadata, and a virtual machine. Smart data is inspired by the Active Bundle (AB) technology introduced by Ben Othmane in 2009 [6] for protection of sensitive data: An active bundle or AB is a software construct, which bundles together the following three components: (1) sensitive data (2) metadata, which contain information describing sensitive data and prescribing its use; they can include a privacy policy for the sensitive data (which control the access to sensitive data or their portions), as well as the rules for AB dissemination; and (3) virtual machine (VM), which controls and manages how its AB behaves, thus making the AB active; the essential task of the VM is enforcement of the privacy policy specified by metadata.

The smart data has a similar structure to the active bundles. However, our smart data, rather than only protecting the data within the bundle, is an intelligent unit which is able to evolve and participate in the operation of an IoT application. Generally, the smart data is a standalone unit that through the resources provided by the underlying hierarchical fog computing platform undergoes a series of pre-processing steps, evolving by getting more attributes, such as security and privacy aspects, and involved rules. A basic and lightweight version of smart data is generated by IoT sensors. It evolves (grows) when it travels through the hierarchical fog computing system towards the cloud, merging with other cells. The process is the opposite when data moves from the cloud towards the actuators, i.e., data are transformed stepwise into a distributed set of elementary cells.

Figure 2 illustrates the general structure of a smart data cell. The smart data is composed of three main parts: payload data, metadata and virtual machine. In IoT based scenarios, where data is generated and transferred continuously in a resource constrained environment, communication activities can consume a considerable amount of energy. In our smart data, data generated by each sensor are encapsulated into smart data bundles and are communicated to their gateways in specific intervals. The main objective of encapsulating a set of data already at the sensor level, instead of constantly sending discrete data, is to reduce the communication overheads in a very resource constrained environment as well as to reduce the data velocity in the big data context. The payload component of the smart data undergoes a series of pre-processing steps and is thereby converted into more meaningful information. Pre-processing of data includes different operations such as aggregation, filtering, compression, and encryption.

The metadata part of smart data contains key information such as the source of data (sensors), destination of data, the physical entity which data belongs to, timestamps, current status and logs as well as rules for accessing, fusing or diffusing, and processing data, for example. In addition, the metadata part stores information extracted by processing the payload data. Such information obtains more accurate values when the data is processed and aggregated with other data

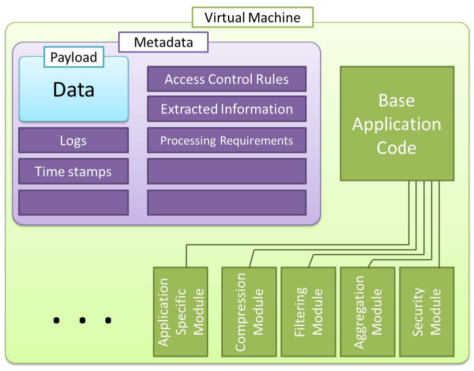


Fig. 2. Structure of Smart Data.

from different sensors or the same sensor over a longer period of time.

The virtual machine part, in turn, acts as a platform which enables and manages the execution of the rules specified in the metadata part. The VM at the very beginning stage contains only basic application codes. Then, it evolves by adding other code modules of the application when they are needed. Each code module provides specific functionalities and services to the smart data. The modular structure of the VM component makes smart data extendable, allowing it to manage the overhead of carrying the code by removing unnecessary code modules and adding the required modules only when they are needed.

The code modules are installed into a smart data cell as a plugin. To enable this, we consider a remote code repository node which contains all necessary code modules as plugins. Whenever a smart data cell requires a specific code module, it communicates with the code repository node and requests for the required code module. To minimize the communication involved in downloading the plugins, the most recently downloaded plugins are also cached in the physical fog nodes for some period of time. So, if the requested code module does not exist in the local fog node, it will be downloaded from the remote code repository node. In order to avoid the overhead of integrating the whole application code to each smart data, at early stage smart data includes only basic application code which provides it some basic functionalities such as communication and lightweight encryption. A new modules of the application code integrates to the base code whenever there is a need for a new functionality. Each module contains a set of program codes which provides a certain service or accomplishes a certain operation on the data. For example, there could be an aggregation module, an encryption module, and compression modules.

B. Smart Data Lifecycle

A lifecycle of smart data is a time span in which a data cell is generated, processed, stored, used and destroyed. A basic form of smart data is first generated in sensors. At this point, smart data is considered immature. Immature smart data contains a semi-structured and encapsulated set of raw data in its payload. The data is stored in the payload of the smart data in a very basic format already in the sensors. This semi-structured data needs to be re-formatted and structured according to the applications context to make complex analysis and access possible for the application. Furthermore, some basic information, such as the source of data, destination of data, and time stamps are specified in the metadata part of each smart data cell already in the sensors. The payload and metadata of the smart data are encrypted using lightweight symmetric ciphers in the sensors to ensure their integrity and confidentiality during inter-node transfers.

Once the smart data has been transmitted from the sensors to their corresponding gateways, they are decrypted and then aggregated with other smart data collected from the neighbour sensors. Then, the aggregated smart data is encrypted and access control policies for its payload are set in its metadata at this point. For this purpose, the smart data cell communicates with the code repository unit to fetch the required code modules and rules for the access control and aggregation.

The gateways in IoT systems are usually resource unconstrained devices and responsible for managing the sensors and their data. In the gateways, after the aggregation stronger encryption mechanisms could be applied to smart data cells. At this point, smart data becomes semi-mature, having aggregated data and developed its metadata to a higher level of complexity. The semi-mature smart data undergoes then a series of further pre-processing tasks, such as filtering and compression through the next higher levels of the fog computing hierarchy and becomes mature.

The aggregation process also involves transforming data from different sources to a single summative data. Such summative data will have a smaller size compared to the sum of the original data before the aggregation which results in lower network traffic and improved performance in processing the data. The aggregation of sensory data will also reduce the variety of big data in smart data. Depending on the application, filtering and aggregation might be applied on the smart data in the fog computing network hierarchy until the smart data reaches a desired level of quality. So, defining an optimal fog computing cluster in which the smart data is processed with the least energy consumption and time, and with the best values for the volume, velocity, value, veracity and variety of the data, is a challenging research problem in designing the smart data. The mature smart data will have significantly less volume and velocity and higher value and veracity. During its lifecycle, semi-mature or mature smart data would be compressed and stored in the local memory systems provided by the fog nodes, so that it would be swiftly available for local applications and processes. On the other hand, mature data from each local fog node in multiple geographical areas is sent to the cloud for global processing and stored for long-term purposes. Also, specific rules for destroying smart data can be set in the metadata part to destroy the payload part in the case of detected security breaches, or if a given validity period expires.

C. System Architecture

In our model, utilizing the virtualized architecture of fog computing, we integrate smart data as virtual fog computing nodes in an existing fog computing platform (Figure 1). The sensors are the lowest layer of our fog computing hierarchy . Once smart data is generated by the sensor nodes, a set of encapsulated data is sent to the gateways within a particular time frame. The nearest available gateway which receives smart data directly from a set of sensors is considered the current managing fog node, or the gateway fog node, for these underlying sensors. Moreover, as it is the nearest point to the edge sensors, it is considered the level one (L_1) fog node for this set of the sensor nodes (Figure 1). A gateway fog node has the primary responsibility for managing smart data received from its underlying reporting sensors, initiating and managing fog computing-based processing and storage for this data and also forwarding actuation commands to specific actuators within its area. The gateway fog nodes, once having received smart data from multiple sensors, supervise aggregation of this data and assign required computing and memory resources for the involved processing. Depending on the underlying IoT service, the gateway nodes also establish distributed local storage within the local fog computing platform or (pre-)process the contents of the smart data in the fog computing platform. Evolution of smart data involves procedures such as aggregation, filtering, compression, encryption, and access control.

The fog computing platform in our model is hierarchical, which enables stepwise evolution of smart data at each level of hierarchy. At the level one (L_1) of this hierarchy, the gateway nodes are connected directly to the sensor nodes (L_0) from one side and to their upper fog level (L_2) from the other side. The nodes at each level are able to communicate with each other (typically with their neighbours at the same level) and accomplish distributed tasks. Once the tasks have been completed the nodes pass data to their parent node at the next higher level of hierarchy. This stepwise process is continued until data reaches a desired degree of maturity at the highest hierarchy level of the fog computing platform. Consequently, the architectural model is scalable and extendable both vertically and horizontally.

Generally, the tasks at the lower levels of hierarchy are more detailed and concern relatively small amounts of data, while at the higher levels of hierarchy more general tasks on larger amounts of data are carried out. The data collected from the lower levels is aggregated at the higher levels. Moreover, a higher hierarchy level deals with data from a larger geographical region than a lower hierarchy level. From the real-time performance point of view, task allocation is done in such a way that processes with low/tight latency requirements are executed at lower hierarchy levels, closer to users. This enables real-time tasks to execute and deliver results very fast, improving user experience. Such processes have some specific characteristics. First, they typically use data related to fewer sensors (corresponding to smaller geographical areas), and therefore a relatively small amount of data is involved. Second, generally at lower levels, a larger number of individual computing devices are involved in processing compared to higher levels. Tasks with no or loose latency requirements can be executed at higher levels of hierarchy. Smart data processed in the fog computing platform becomes eventually mature and refined and is stored either in the distributed fog storage or sent to the cloud utilizing high performance communication technologies such as 3G and broadband connections. In case a local decision is made within the fog computing platform, the resulting commands will be sent to corresponding actuators through the involved gateways, without disturbing the cloud. Therefore, by pre-processing data locally in the fog computing platform, the workload of cloud computing can be significantly reduced.

IV. Conclusion

With wide application of IoT/IoE technologies, new techniques will be required to deal with big data. In this paper, by reshaping the current raw and passive structure of data into an intelligent and active form, we introduced the smart data concept. This concept combined with the fog computing technology will have potential to revolutionize the current perspective of data in IoT and will open many potential research opportunities to tackle emerging big data issues. Smart data takes advantage of hierarchical and virtualized model of fog computing and provides better means for pre-processing big data originated from IoT sensors. As the next phase of this project, our purpose is to implement the smart data model in a realistic scenario and to develop efficient solutions for relevant operations such as filtering, aggregation, and compression based on the proposed smart data concept.

References

[1] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," in *International Conference on Collaboration Technologies and Systems*, 2013, pp. 48–55.

[2] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*. Springer International Publishing, 2014, ch. Fog Computing: A Platform for Internet of Things and Analytics, pp. 169–186.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things characterization of fog computing," in *MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–15.

[4] M. Nemirovsky, "Fog computing," Cloud Assisted Services in Europe (CLASS) Conference, Bled 2012, Tech. Rep., 2012.

[5] L. Gu, D. Zeng, and S. Guo, "Cost efficient resource management in fog computing supported medical cyber-physical system," *EEE Transactions on Emerging Topics in Computing*, vol. 6750, no. c, 2015.

[6] L. B. Othmane and L. Lilien, "Protecting privacy of sensitive data dissemination using active bundles," in *World Congress on Privacy, Security, Trust and the Management of e-Business*. IEEE, 2009, pp. 202–213.

Publication IV

Hosseinpour, F., Siddiqui, A. S., Plosila, J., and Tenhunen, H.
A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models

Reprinted with permission from
Research and Practical Issues of Enterprise Information Systems.
pp. 168-180, 2018.

A Security Framework for Fog Networks Based on Role-Based Access Control and Trust Models

Farhoud Hosseinpour[1](✉), Ali Shuja Siddiqui[2], Juha Plosila[1], and Hannu Tenhunen[1]

[1] Department of Future Technologies, University of Turku, Turku, Finland
{farhos,juplos,hatenhu}@utu.fi

[2] Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, USA
asiddiq6@uncc.edu

Abstract. Fog networks have been introduced as a new intermediate computational layer between the cloud layer and the consumer layer in a typical cloud computing model. The fog layer takes advantage of distributed computing through tiny smart devices and access points. To enhance the performance of the fog layer we propose utilization of unused computational resources of surrounding smart devices in the fog layer. However, this will raise security concerns. To tackle this problem, we propose in this paper a novel method using a trust model and Role Based Access Control System to manage dynamically joining mobile fog nodes in a fog computing system. In our approach, the new dynamic nodes are assigned non-critical computing tasks. Their trust level is then evaluated based on the satisfaction rate of assigned tasks which is obtained through different computing parameters. As the result of this evaluation, untrusted nodes are dropped by the fog system and nodes with a higher trust level are given a new role and privileges to access and process categorized data.

Keywords: Fog computing · Cloud · Access control · Trust model

1 Introduction

The benefits achievable by deploying scalable applications serving a large number of users simultaneously are rapidly generating novel innovations and expanding the reach of cloud computing. The cloud computing has replaced the need for owning large private data centers for service providers who want to deploy their projects with minimum infrastructure cost [1]. Cloud computing provides scalability for applications in manyfold by enabling addition and removal of processing nodes at runtime as needed. Although cloud computing has deemed itself useful in many scenarios [2], it is not viable for applications that require low latency and predictable feedback such as Smart Grids, industrial automation systems or intelligent transport systems [3]. This is due to the fact that systems in a cloud service are geographically distributed. For the alleviation of this issue, "*fog computing*" [4] has been introduced as a complementary concept to cloud computing. Cloud computing can be defined using a layered computation model. Typically there are two layers: a cloud layer and a consumer layer. Recently a new computational layer, called a fog layer, has been introduced to the model. The fog layer resides between the cloud and consumer layers in the network's edge nodes like sensors and Internet-of-Things devices. Fog computing introduces the concept of location to cloud computing where traditionally non-locational computing has been dominant. Additionally, the fog layer also provides extra computational resources to the cloud layer.

Fog computing is currently an evolving new technology which aims to supplement already established cloud computing platforms to expand their application domain. Fog computing provides a location based expansion of the cloud by using heterogeneous computing devices and access points to which end nodes connect to communicate with the cloud. Bringing the computational intelligence geographically near to the end users provide new or better services for latency sensitive, location-aware and geo-distributed applications that due to their characteristics are not feasible merely through cloud computing. Delegating some simple yet frequent tasks of the cloud to the fog results in better performance for IoT-based applications [5]. In this paradigm, intelligent networking devices with both computation and storage capabilities, i.e., intelligent routers, bridges, and gateways, compose the fog computing platform near to the edge of the network. However, such devices are resource constrained and have computing and storage limitations.

Increasing computing capabilities of fog computing is a major challenge to improve the Quality of Service (QoS). To this end, one possible way is to leverage processing and storage capabilities of surrounding smart devices [6]. Smart devices have become an ubiquitous part of modern life. According to the Global Internet Phenomena Report Spotlight 2016 from Sandvine, the Waterloo-based broadband network equipment company in North America [7,8]: "*The average household was found to have at least seven active, connected devices in use every day, while at the top end of the spectrum, 6% of households tuned in with more than 15 active devices, a marked increase over previous years. Whereas home roaming via mobile devices such as tablets and smartphones accounted for only nine percent of traffic five years ago, it now represents almost 30% of home internet traffic across North America.*" Falaki et al. [9] developed a tool called SystemSens and investigated resource usage such as CPU, memory, and battery in smartphones. According to this study, except for the pick time between 11:00 to 17:00 the average CPU usage in all tested users is below 50%. This amount drops to less than 20% during the night time between 00:00 to 8:00.

Having this motivation, leveraging the available computing power of numerous different smart devices will enhance fog computing. However, utilizing resources of such devices for fog computing will impose some security challenges.

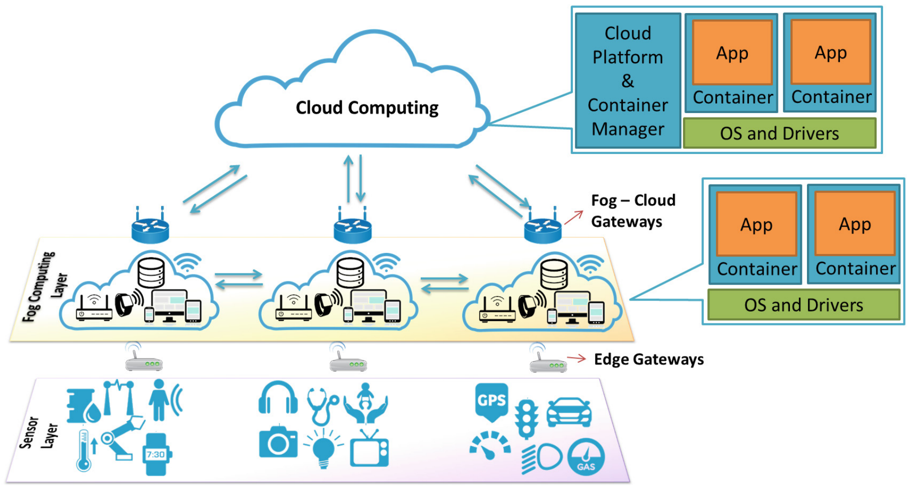


Fig. 1. Fog computing platform.

In this paper, we present a novel approach to tackle this problem by leveraging containerization technology to provide isolation for fog computing tasks in external smart devices. This is further supported by role-based access control and trust models.

The rest of the paper is organized as follows. In Sect. 2, we give an overview of the newly emerging technology of fog computing. In Sect. 3, related works to access control mechanisms for smart devices are presented and discussed. Then, in Sect. 4, we present our proposed framework. The results and discussion on the implemented model are presented in Sect. 5, and, finally, concluding remarks are given in Sect. 6.

2 Application of Fog Computing

Fog computing introduces an intermediate layer between the edge network or the end nodes and the cloud layer (Fig. 1). The fog layer can be implemented using the same components as the cloud layer. The fog layer provides computation in a geographical location. It aims to provide a computing layer physically closer to the end node so that the computing capabilities can be brought near to consumers. The expected benefit is obtaining faster computation times for requests that require low latency. This can play an advantageous role in promotion of the Internet of Things (IoT) [10]. Utilizing fog computing reduces the overhead of communication with cloud through internet and provides a faster response for applications which require lower latency. This is made possible by locally executing such processes in the fog layer and forwarding only those which do not require real-time computation or require higher processing power to the cloud layer. Schulz et al. [3] have investigated different latency critical IoT applications.

According to their study, factory automation applications have the highest critical latency requirements in the range of 0.25 to 10 ms. Process automation, Smart Grids and intelligent transport systems are in the next place in their ranking.

In addition to the requirement of low latency, fog computing as middleware can pre-process raw data coming from the edge nodes before sending them to the cloud. Cloud computing, dealing with Big Data [11], has to process large amounts of data at any time. As a result, the fog layer not only reduces the amount of work needed in the cloud to generate meaningful results, but it can also reduce the monetary cost of computing in the cloud layer.

2.1 Fog Layer Structure

The most important and beneficial aspect of fog computing is the location proximity to the end nodes. The fog layer can be deployed on intelligent access points and gateways that not only connect the edge nodes to the cloud layer but also provide additional computing resources near the edge of the network. In addition to that, independent computing nodes such as smart devices can be added to the fog layer for the sole purpose of computation. Fog nodes can connect to each other to form a mesh. This can also be envisioned as a peer-to-peer (P2P) network with either centralized master controllers or a decentralized implementation without any controllers. Fog nodes cooperate and pool their resources to complete a task. The fog layer can be a dynamic network because some nodes might dynamically join and leave the network due to mobility or power limitations. Or, the other way around, the edge sensors might be mobile and move from one local fog network to another. A robust orchestration system is required to manage the execution of applications in such a dynamic environment without violating QoS and security.

Virtualisation: To support multi-tenancy of different applications and to achieve elasticity in large-scale shared resources, fog computing takes advantages of visualization technologies. A physical fog node can accommodate several virtual fog nodes. A fog computing platform is composed of several physical and virtual fog nodes that are deployed based on a hierarchical architecture [12] (Fig. 2). Virtualisation technology based on Virtual Machines (VM) is not efficient or even feasible approach for resource constrained fog computing nodes. Containers are a new lightweight alternative for traditional VMs that are ideal for a fog computing platform. Containers provide OS level virtualisation without a need for deployment of a virtual OS. Hence, they are lightweight and significantly smaller in size than VMs. Containers provide a self-contained and isolated computing environment for applications and facilitate lightweight portability and interoperability for IoT applications [13]. Moreover, data and resource isolation in containers offers improved security for the applications running in fog nodes.

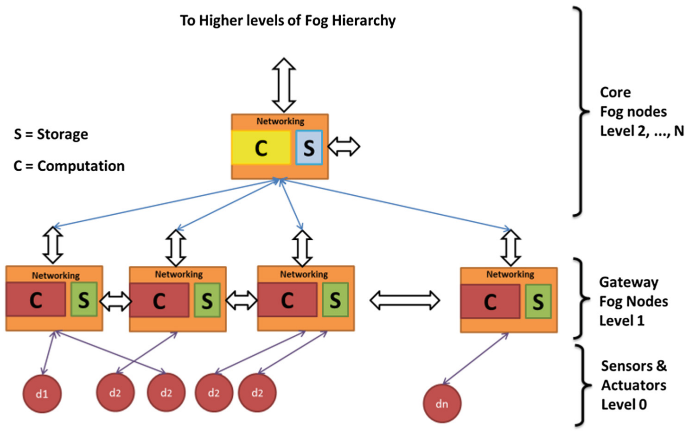

To Higher levels of Fog Hierarchy
S = Storage
C = Computation
Networking
C
S
Core Fog nodes Level 2, ..., N
Gateway Fog Nodes Level 1
Sensors & Actuators Level 0
d1
d2
dn

Fig. 2. Architecture of fog computing.

3 Related Works

Fog computing was introduced in 2012 by Cisco [4] as an additional computing layer near to the edge of the network, to complement cloud computing services. We discussed challenges for adoption of this technology in IoT applications as well as its security issues in our review paper [12]. Due to location proximity to the edge of the network, mobility of edge sensors, and also resource limitations, enabling scalable, flexible, and real-time strategies for resource allocation is very challenging. Oueis et al. [14] proposed a cluster-based resource allocation scheme for fog computing in which a cluster of fog computing resources is logically built depending on the profile of computation offloading request from an IoT device or a fog node. Yi et al. [15] investigated security and privacy issues of fog computing and pointed out that unlike the cloud computing that the cloud service provider owns all computing devices, fog computing is more flexible to leverage different computing resources belong to different parties. This flexibility adds more complexity in the terms of trust management and security. Misra and Vaish [16] proposed a cluster-based and multilevel hierarchical architecture for Wireless Sensor Networks (WSN) to establish an authentication mechanism. They deployed a multi-level access control system for each logical cluster using Role-Based Access Control (RBAC) model. They proposed a reputation-based trust model to assign a role for a node and form the logical cluster in WSN. They calculated the reputation value based on the behaviour of a node for successful transmission of data. Salonikias et al. [17] addressed access control issues in fog computing for an intelligent transport system as a case study. They pointed out that fog computing has dispersed nature and sensors can enter and leave the network arbitrarily or the other way around, fog nodes could also be mobile. Hence, traditional identity-based authentication is not a feasible approach in this case. To cope with this problem, they proposed utilizing Attribute-Based Access

Control (ABAC) model in which the authentication is based on the attributes of the subject (in this case, fog node) trying to access a data rather than their identity. In [18] the author discussed the importance of granting access based on the level of trust to individuals. They innovated a mobile device called MS-Ro-BAC for implementation of role-based access control. The MS-Ro-BAC manages access and network authorizations through of role-based access control with no dedicated hubs, servers, special hard-drives or local administrators.

4 Proposed Framework

In this paper, we propose a framework for secure utilization of surrounding smart devices' processing capabilities in a fog computing platform. We use containers as virtualization technology in our fog platform. The reason for this is that they are: (1) lightweight and require less computing and storage, (2) easily portable, (3) platform independent and provide interoperability in a heterogeneous network and, (4) provide isolation of the application that utilize shared resources, which results in better security. We also design and develop an access control system based on the RBAC model to provide authentication for dynamic fog nodes joining the fog computing network. We consider three different kinds of fog nodes according to their capabilities and trust levels. As discussed earlier, a typical fog network is composed of smart communication nodes with the capability of acting as access points. This way they can communicate with edge sensors and also forward preprocessed data to an upper level in the cloud. Also, we propose utilization of dynamic nodes, each of which provides either processing resource only or combined processing and access point resource. Such nodes, after having been identified within the fog network, can join fog computing and share their resources.

Our framework employs trust models in transactions pertaining to data transfer and administration. This adds an extra layer of security and guarantees that untrusted nodes are not able to access sensitive data over the network. Dividing trust into levels will allow segregation of operations and data based on their criticality.

Whenever a node is made part of the fog for the first time, it is assigned the lowest level of trust and the least access privileges to the data to be processed as no knowledge of its previous transactions exists. However, after some transactions, the dynamic nodes can improve their reputation and gain a higher level of trust. They might also be disjoined from the network if any malicious actions are detected, or if they will no longer be in the vicinity of the computing environment. In cases like this, the nodes' access privileges need to be revoked. To make this possible, a manager node is required for managing the task allocation and participating nodes. Figure 3 illustrates the proposed framework in which the fog layer consists of four types of nodes: Fog Manager Node (FMN), Static Node (SN), Dynamic Node (DN) and Processing Node (PN). Any of these nodes have different roles and hence different privileges are assigned to them. A role for a node is defined based on its capability (Processing only or Processing and

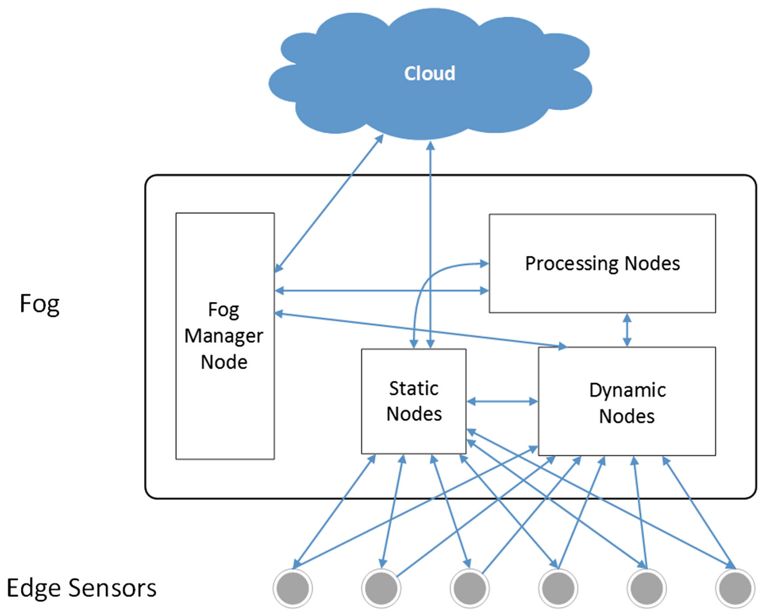


Fig. 3. Proposed framework.

communication) and its current level of trust. The following section describes the definition of roles and trust in more detail.

4.1 Roles

A role for a node defines its privileges for accessing different kinds of data and for participating in processing tasks. In this framework we assume three categories of information based on criticality: *non-critical*, *moderate* and *critical* data. The FMN assigns the roles to nodes based on their reputation and trust levels as well as their capabilities. We define four roles according to the node types in this framework. With each role within fog layer, a set of permissions is assigned. This set limits the nodes' access to certain types of data and defines their privileges and responsibilities in processing certain tasks. Table 1 summarize the security privileges of each node according to its role.

A detailed description of each node is presented in the following.

Fog Network Manager is the central overseer of the fog network. Whenever a node wants to join the network, it must contact the FNM. If the connecting node is an edge node, the FNM will send this node the address of the active fog nodes based on location proximity. The edge node will then connect to the nearest fog node and start sending its data. In case a fog node goes offline, the edge node will be provided with the address of the next suitable nodes to connect to. If a smart device attempts to join the fog network, the FNM will assign the connecting node with the lowest trust level. On the other hand, if a

Table 1. Assignment of privileges according to roles.

Role	Privileges						
	Processing	Edge communication	Cloud communication	Verifying the tasks	Assigning a task	Adding new nodes	Revoking access
FNM		×	×	×	×	×	×
SN	×	×	×	×			
DN	×	×					
PN	×						

node needs to be deleted from network (due to malfunctioning or permanently disconnecting), the FNM will revoke access rights from that node and update the list of active fog nodes to the edge sensors as well as the cloud layer. The FNM is also responsible for promoting or demoting the roles of participating fog nodes according to their trust levels.

Static Nodes: These nodes are used for connecting edge devices to the fog layer. They are static in nature and are expected to be available at all times. By default, they are assigned the trust level *High*. They can either process data themselves or they can request the FNM to initiate a task on some other processing node or an access point. Upon completion of a task, they can forward the data themselves to the cloud layer.
Dynamic Nodes are dynamic and are intended to be used as access points as well as processing nodes. They start with trust level *Low* and gain more levels as trusted more by the FNM. They do not themselves send data to the cloud layer but, instead, they use the static access points for the purpose.
Processing Nodes are dynamic and are used exclusively for processing data. Since they are dynamic, their initial trust level is *Low* and it is increased as the node gains more trust. They cannot themselves connect to the cloud layer nor act as access points but, instead, they use static access points for sending their data. These nodes only share their processing resources. Therefore, their tasks are assigned by the network manager or static nodes. After processing the task the results needed to be sent to a static nodes to be forwarded to the cloud layer.

4.2 Trust Management

A trust level is a measure of the reliability of a participating node. Trust management is applied to dynamic fog nodes of the network. Static fog nodes at any time are considered to have the highest level of trust. Trust in our system is defined in terms of a nodes' privilege to process a certain type of data. It can be divided into multiple levels but in this paper, we will consider the division into three levels:

– *Low*: This is the lowest level. Dynamic fog nodes are initially assigned this level upon joining the network. The FMN assigns tasks of the lowest priority

and criticality to these nodes. Data computed by nodes with this trust level is sent to one of the static nodes to be verified before sending to the cloud layer.
- *Moderate*: This is the second level of trust. On this level, the data is considered to be of moderate criticality. The fog node handling this data is assumed to be reliable, and the result generated by the node will be sent directly to the cloud layer. Dissatisfaction in the service of nodes in this level will demote the node to the low level. However, dissatisfaction up to a pre-determined level can still be tolerated.
- *High*: This is the highest level of trust. The data which is considered to be most critical by the application is handled by the nodes at this level. The requirement of processing is not only that the data be processed correctly, but also that the nodes maintain the highest level of service. The data processed by nodes in this level is sent directly to the cloud layer.

The trust level of each dynamic fog node evolves over time and can change on interaction with other nodes. We utilize already established trust algorithms for our implementation. There can be several ways to calculate the trust level for a node. Manuel [19] investigated different factors to evaluate trust value of a recourse in cloud computing. They claim that combination of multiple trust factors such as availability, reliability, data integrity, and turnaround efficiency should contribute to the trust model of a resource. According to this study, in our proposed framework we calculate the trust value based on all attributes mentioned above.

In the following we discuss each of these attributes and present a formula to compute the trust value of a resource based on those attributes:
Availability is a measure to ensure that a resource is operational and accessible to authorized parties whenever needed. A resource is deemed unavailable if (1) it is too busy to process and responds a task request, (2) it denies a task request, or (3) it is just shut down. Availability of a resource Av_R is calculated based on the following equation over a period of time:

$$Av_R = \frac{Ac}{Sb} \tag{1}$$

where Ac denote the number of computing tasks accepted by a resource and Sb denote the total number of tasks submitted to that resource.
Reliability or success rate of a resource is a measure and quality of a resource in consistently performing according to its specifications in specified time. Reliability of a resource Re_R defines its success rate in the completion of the tasks that it has accepted and is calculated based on the following equation over a period of time:

$$Re_R = \frac{Cs}{Ac} \tag{2}$$

where Cs denote the number of accepted tasks completed successfully by a resource, and Ac is the total number of accepted tasks by that resource.

Data Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire lifecycle. Integrity ensures that information is not modified by unauthorized entities. Data Integrity of a resource Di_R is calculated based on the following equation over a period of time:

$$Di_R = \frac{Cm}{Ac} \tag{3}$$

where Cm denote the number of tasks that a resource successfully preserves data integrity, and T is the total number of accepted tasks completed successfully by a resource.
Turnaround Efficiency is a quality that a resource accomplishes a task within the time that it promises. Turnaround is a time frame that starts from when a broker sends a processing request to a resource till the time that the resource completes the task successfully. Turnaround Efficiency of a resource Te_R is calculated based on the following equation over a period of time:

$$Te_R = \frac{Pt}{At} \tag{4}$$

where Pt denote the Promised Turnaround time by a resource for completion of a task and At is the Actual Turnaround time by a resource for the completion of a task.

Trust Value of a Resource: The overall trust value for a resource is calculated based on composition of all attributes of a resource with following equation:

$$TrustValue_R = (a * Av) + (b * Re) + (c * Di) + (d * Te) \tag{5}$$

where $a+b+c+d = 1$ are coefficient positive numbers that define the weight of each attribute and Av, Re, Di, and Te are respectively average value for Availability, Reliability, Data Integrity, and Turnaround Efficiency over determined time T.

Task assignment done by the FNM is also dependent on the trust levels. To ensure that each trust level would have the required number of nodes to perform all tasks defined for that trust level, we will use the weight function to evaluate the need to increase the trust level of a dynamic node. It would be preferred for a node to be promoted to a higher trust level if there is a shortage of nodes at a higher level. For each dynamic fog node, the weight is calculated as:

$$Weight = \begin{cases} 1 - \frac{N_{req}}{N_{avail}} \ if & N_{req} < N_{avail} \\ 0 & otherwise \end{cases} \tag{6}$$

where N_{req} is the number of nodes required at the next higher trust level, and N_{avail} is the number of nodes available at the next higher trust level.

5 Results and Discussion

The fog computing platform is implemented in SystemC environment. Each processing unit is modeled by a SystemC module which can communicate with all

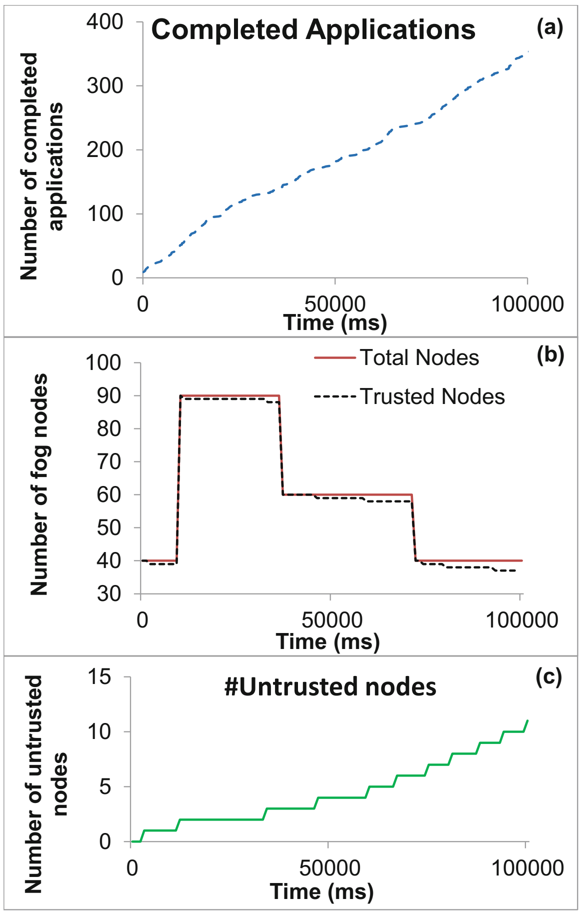


Fig. 4. Experimental results.

the other processing elements in its domain through a SystemC channel. We have considered heterogeneous nodes with different processing capabilities. The execution frequency for each processing elements varies between 500 MHz up to 4 GHz. Applications enter and leave the system based on a randomized amount of workload during the time. Each application is modeled as a task graph where each task should be assigned to a processing element exclusively. Execution of the tasks are independent of each other, and only the data transfer between tasks connects two tasks to each other. Therefore each task can be run at a different frequency. The fog system comprises of a number of fog nodes which include static nodes, processing nodes, and a fog manager node. Along the time, a group of dynamic nodes joins and leave the fog system. The fog manager assigns the tasks to the newly joined dynamic node and calculates their trust level based on the Trust formula. So, once any of the new dynamic nodes reaches to the desired trust level, then the fog system upgrades their role to become trusted fog node.

Figure 4(a) shows the number of completed applications in the fog system during the time. Figure 4(b) illustrates the total number of the nodes once the dynamic nodes join and leave the fog system. The dashed line shows that the system is able to detect and eliminate the untrusted nodes in each interval. As it can be seen, while the number of nodes in the fog system increases, the rate of the completed applications also increases. And finally Figure 4(c) shows the total number of identified untrusted nodes during the execution time.

6 Conclusion

The fog computing paradigm extends cloud computing and services to the edge of the network to support geographical distribution and mobility of end users. In this paper, we presented a security framework for fog computing infrastructure. After discussing the potential application of fog computing, we argued that to increase the performance of the fog layer we can take advantages of vacant resources of surrounding smart devices such as smart phones and tablets. To tackle the security issues that are imposed by this technique we proposed an implementation of role-based access control in conjunction with trust models. We presented how our proposed framework can contribute to solving security issues of a fog network. In our framework, we defined a method to calculate trust levels based on computing tasks assigned to the nodes. Moreover, we presented algorithms for implementation of our framework. According to our implementation results, the fog system was able to distinguish the trusted and untrusted dynamic nodes. However, in addition to secure access control and authentication methods, secure computation schemes need to be undertaken to guarantee the security and integrity of data in a fog network.

Acknowledgment. This work was supported by University of Turku Foundation, EIT Digital and the Department of Information Technology - University of Turku.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. In: 2009 Fifth International Joint Conference on INC, IMS and IDC, pp. 44–51. IEEE (2009)
3. Schulz, P., Matthe, M., Klessig, H., Simsek, M., Fettweis, G., Ansari, J., Ashraf, S.A., Almeroth, B., Voigt, J., Riedel, I., Puschmann, A., Mitschele-Thiel, A., Muller, M., Elste, T., Windisch, M.: Latency critical IoT applications in 5G: perspective on the design of radio interface and network architecture. IEEE Commun. Mag. **55**(2), 70–78 (2017)

4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ser. MCC 2012, pp. 13–16. ACM, New York (2012)
5. Sehgal, V.K., Patrick, A., Soni, A., Rajput, L.: Smart human security framework using Internet of Things, cloud and fog computing. In: Buyya, R., Thampi, S.M. (eds.) Intelligent Distributed Computing. AISC, vol. 321, pp. 251–263. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-11227-5_22
6. Shi, W., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
7. Sandvine Intelligent Broadband Networks: 2016 global internet phenomena report, Latin America & North America. Sandvine - Intelligent Broadband Networks, Technical report (2016)
8. MacLean, J.: Households now use an average of seven connected devices every day. Cantech Letter, Technical report (2016)
9. Falaki, H., Mahajan, R., Estrin, D.: SystemSens: a tool for monitoring usage in smartphone research deployments. In: Proceedings of the Sixth International Workshop on MobiArch, ser. MobiArch 2011, pp. 25–30. ACM, New York (2011)
10. Ashton, K.: That 'Internet of Things' thing. RFiD J. **22**, 97–114 (2009)
11. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.: Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute (2011)
12. Hosseinpour, F., Meng, Y., Westerlund, T., Plosila, J., Liu, R., Tenhunen, H.: A review on fog computing systems. Int. J. Adv. Comput. Technol. (IJACT) **8**(5), 48–61 (2016)
13. Bellavista, P., Zanni, A.: Feasibility of fog computing deployment based on Docker containerization over RaspberryPi. In: Proceedings of the 18th International Conference on Distributed Computing and Networking, ser. ICDCN 2017, pp. 16:1–16:10. ACM, New York (2017)
14. Oueis, J., Strinati, E.C., Barbarossa, S.: The fog balancing: load distribution for small cell cloud computing. In: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), pp. 1–6 (2015)
15. Yi, S., Qin, Z., Li, Q.: Security and privacy issues of fog computing: a survey. In: Xu, K., Zhu, H. (eds.) WASA 2015. LNCS, vol. 9204, pp. 685–695. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21837-3_67
16. Misra, S., Vaish, A.: Reputation-based role assignment for role-based access control in wireless sensor networks. Comput. Commun. **34**(3), 281–294 (2011)
17. Salonikias, S., Mavridis, I., Gritzalis, D.: Access control issues in utilizing fog computing for transport infrastructure. In: Rome, E., Theocharidou, M., Wolthusen, S. (eds.) CRITIS 2015. LNCS, vol. 9578, pp. 15–26. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33331-1_2
18. House, T.: Mobile secure role based access control (MS-Ro-BAC) device. In: Proceedings of the SoutheastCon, pp. 542–546. IEEE, April 2005
19. Manuel, P.: A trust model of cloud computing based on quality of service. Ann. Oper. Res. **233**(1), 281–292 (2015)

Publication V

Hosseinpour, F. , Naebi, A. , Virtanen, S. , Pahikkala, T. , Tenhunen, H., and Plosila, J.
A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks

Reprinted with permission from
IEEE Access
vol. 9, pp. 152792-152802, 2021.

Received October 28, 2021, accepted November 8, 2021, date of publication November 10, 2021, date of current version November 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127355

A Resource Management Model for Distributed Multi-Task Applications in Fog Computing Networks

FARHOUD HOSSEINPOUR[1], (Member, IEEE), AHMAD NAEBI[2], SEPPO VIRTANEN[1], (Senior Member, IEEE), TAPIO PAHIKKALA[1], HANNU TENHUNEN[1,3], AND JUHA PLOSILA[1], (Member, IEEE)

[1]Department of Computing, University of Turku (UTU), 20500 Turku, Finland
[2]System Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China
[3]School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden

Corresponding author: Farhoud Hosseinpour (farhos@utu.fi)

This work was supported by the European Institute of Innovation Technology (EIT Digital) and the Academy of Finland under Project 311273 and Project 335512. The work of Farhoud Hosseinpour was supported by the Research Awards of the Nokia and Tekniikan Edistämissäätiö (TES) Foundations.

ABSTRACT While the effectiveness of fog computing in Internet of Things (IoT) applications has been widely investigated in various studies, there is still a lack of techniques to efficiently utilize the computing resources in a fog platform to maximize Quality of Service (QoS) and Quality of Experience (QoE). This paper presents a resource management model for service placement of distributed multitasking applications in fog computing through mathematical modeling of such a platform. Our main design goal is to reduce communication between the candidate nodes hosting different task modules of an application by selecting a group of nodes near each other and as close to the source of the data as possible. We propose a method based on a greedy principle that demonstrates a highly scalable and near-optimal performance for resource mapping problems for multitasking applications in fog computing networks. Compared with the commercial Gurobi optimizer, our proposed algorithm provides a mapping solution that obtains 93% of the performance, attributed to a higher communication cost, while outperforming the reference method in terms of the computing speed, cutting the mapping execution time to less than 1% of that of the Gurobi optimizer.

INDEX TERMS Greedy, fog computing, Internet of Things, modelling, optimization, resource management.

I. INTRODUCTION AND BACKGROUND

The past decade has witnessed a wide deployment of the Internet of Things (IoT) technology in various application domains, and its pervasive role will continue to strengthen in the future [1]. The emerging IoT applications, through various sensors, generate massive amounts of data that are generally referred to as big data. Traditionally, data is not processed in the proximity of a sensor but transferred as it is to a server that might be located in the cloud. However, transferring the constantly increasing amount of information from sensors to the cloud is not feasible. To overcome the intrinsic limitation of centralized data processing in cloud computing, a new paradigm called fog computing was introduced. Fog computing is characterized by heterogeneity, dynamicity, mobility, and geographical distribution that complement cloud computing services, providing local processing and faster response for delay-sensitive applications. It is considered a derivative of cloud computing that extends its services to the network's edge [2].

Fog computing does not replace cloud computing services but rather, by a well-organized interplay, complements the cloud computing services. Fog computing reduces the delay and response time for frequent and delay-sensitive local user requests. In contrast, cloud computing provides powerful computing resources and more extensive data storage for the global data collected from a larger geographical area. Figure 1 illustrates the basic architecture of a three-layered IoT system based on cloud and fog computing.

While the effectiveness of fog computing in IoT applications has been widely investigated in various studies [3]–[7], there is still a lack of techniques to efficiently utilize

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

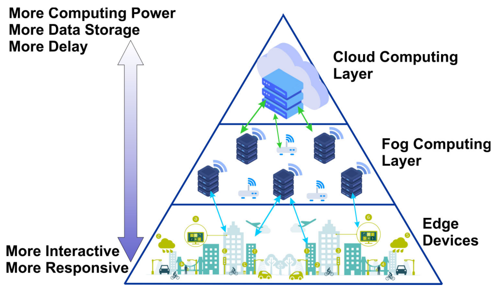


FIGURE 1. IoT architecture based on cloud and fog computing.

computing resources in this platform to maximize Quality of Service (QoS) and Quality of Experience (QoE). Inefficient scheduling and resource allocation for user services can actually result in even higher delays than sending the data for processing in the cloud [2]. Due to the aforementioned unique characteristics of fog computing, the resource allocation problem becomes more challenging and sophisticated. Understanding the nature of fog computing and the analogy between fog and cloud computing helps in systematic modeling of a fog computing framework and efficient resource management in the fog.

The fog layer is a dynamic and resource-constrained computing system, where both to-be-executed applications and executing mobile nodes can enter and leave the system at run-time, which results in high dynamicity of the workload and platform. Moreover, such applications and computing nodes are heterogeneous. Applications can be related to various services, and nodes range from switches and routers to base stations and even edge clusters or micro data centres [8], [9]. Such dynamicity and heterogeneity in both application and platform demand intelligent and agile coordination through a resource management system to achieve optimal or near-optimal performance at run-time.

A. RESOURCE MANAGEMENT

The resource management problem in fog computing has been widely investigated from different perspectives. The authors in [10] investigate the trade-off between the power consumption and delay in a cloud-fog computing system. They formulate the workload allocation problem in a cloud-fog scenario by modeling the power consumption and delay functions in the cloud and fog as well as communication delay function for dispatch. They conclude that cloud computing is more energy-efficient than fog computing while fog computing, due to the proximity to the users, can improve the performance of cloud computing by reducing communication latencies. In [11], the authors formulate a joint optimization problem for allocating the computation resources to maximize utility or the satisfaction rate and minimize carbon footprint for video streaming services in fog computing. As a solution, they propose a proximal distributed algorithm for large scale fog systems.

Communication costs between fog computing nodes are very important parameters to be optimized. Finding an optimal path to communicate the processing requests is a key task in building a robust resource allocation scheme in fog computing. The authors in [12] propose a Steiner tree based caching scheme to produce an optimal Steiner tree in a fog computing cluster to minimize the total cost of the communication path in a way that total cost of cashing resources is minimized.

The majority of the aforementioned works consider non-distributed applications running in a single device. IoT applications are typically involved with processing streams of data that are generated by sensors (i.e. data streaming applications). Stream processing involves applications that are developed as dataflow graphs that include task vertices executing some user-defined logic and streaming the messages between the tasks, i.e., distributed multi-task applications [9]. Such applications are characterized by their continuous processing requirements and computation-intensive nature. Offloading computation tasks into the fog computing or cloud computing layer, i.e., distributed processing, is an efficient solution to cope with limited resources of the edge devices.

Application partitioning is a key technology in the development of distributed applications for distributed computing systems such as cloud and fog computing systems [13], [14]. The partitioning can be done in different levels of granularity. In coarse-grained granularity, applications are divided into a set of loosely coupled units, called application modules, that can be integrated to create larger applications. Each module of an application runs in a computing node (typically within a container or Virtual Machine (VM)) and communicates its output data to other involved modules, processing data streams collaboratively with the other modules of that application which may reside at several different computing nodes around the network [6]. In fine-grained granularity, applications or modules of applications are further divided into smaller sub-tasks or processes (e.g. threads) that can be executed on different components of a heterogeneous computing node such as Central Processing Unit (CPU), Graphics Processing Unit (GPU), and Field Programmable Gate Array (FPGA [6], [15]. Such distribution, if not properly managed, could be delay-inducing and resource-intensive in geographically distributed fog computing networks [16].

The authors in [14] propose a framework for partitioning of applications on Mobile Cloud Computing (MCC) platforms to maximize the performance of applications in terms of speed/throughput as well as optimally utilize cloud resources. The authors in [17] consider a coarse-grained application partitioning method based on virtual machines that run as in cloudlet-based MCC environments. In [18], the authors develop a resource allocation model that provides optimal partitioning and offloading the application partitions into MCC systems. They consider several system parameters such as network traffic and processing resources in mobile devices, mobile clouds, and the cloud to optimally decide where to run each partition of the application.

From the fog computing perspective, a distributed application composed of several modules needs to be mapped onto multiple computing nodes. The candidate processing nodes communicate the data related to that application among each other according to the data flow of the application. The authors in [19] study dynamic task module mapping, i.e., deployment, in a fog computing platform. They argue that, because of limited resources in fog computing platforms, splitting the requests from the users into smaller modules is a necessity. Their main goal is to find an optimal or near-optimal way to decompose the applications into task modules and efficiently map them onto the processing nodes.

It is evident that, in distributed processing, communication among the nodes will impose a delay that may affect the system's QoS and QoE. So, a very important challenge is how to assign applications' modules to a set of fog computing nodes resulting in minimal communication and execution delays, energy consumption, and network bandwidth usage.

In an IoT-fog system, an unpredictable number of applications with different sequences may require computing resources at any time. Therefore, allocation of computing resources to the arriving service requests needs to be done dynamically at run-time instead of design time. Satisfying power and performance constraints by allocation of resources for an application consisting of several communicating modules is a complex process. Moreover, the parallel execution of the task modules in the system adds to the complexity, potentially degrading the expected performance of the fog computing system if not properly managed. Determining a set of superior fog nodes to allocate an incoming distributed application with the least possible communication cost and delay is therefore crucial for ensuring high performance of the fog computing system. At run-time, a resource allocation unit (RTRA), manages this phase in two steps: i) *node selection*, that determines a set of neighboring nodes with adequate computing and storage capacity and bandwidth to be reserved for the new application, and ii) *task module mapping*, that forwards (distributes) the task modules of a particular application to the selected fog nodes.

The authors in [20] investigate the requirements of distributed task placement in fog computing networks. They point out that modeling both tasks and computing devices is necessary for studying task placement for fog computing. Also, development of distributed applications faces more challenges than that of monolithic applications, in terms of the communication complexity among each application's components that need to be addressed. They compare different algorithms for the tasks placement problem in distributed fog computing and conclude that greedy approaches, such as the Hill-Climbing algorithm, demonstrate better performance in solving the problem (higher speed), whereas genetic algorithms provide solutions that have better performance (higher quality). The authors in [21] propose an application placement technique for concurrent applications in a fog network. They present a weighted cost model for minimizing the execution time and energy consumption of multi-task applications in the fog computing context, supported by a pre-scheduling algorithm to maximize the number of parallel executions. The authors in [22] propose a module placement algorithm called MPC4.5 using the Markov Chain process in mobile fog computing networks. They consider an application consisting of multiple modules to be placed in a set of the most suitable fog nodes. Their proposed algorithm uses parameters such as authenticity, confidentiality, availability, capacity, integrity, cost, and speed as decision parameters for placing the application module in a fog node. In [23], the authors propose an application placement algorithm based on multidimensional QoE prioritization. They prioritize the incoming offloading requests based on three main domains: the environment runtime context, application usage, and user expectations, taking into account QoE and QoS. Then they choose a set of fog computing nodes for each requesting application based on proximity to the source of data, computing capabilities of fog nodes, and expected execution time for each application.

In [24], the authors present a cost-efficient resource management model for non-distributed applications in fog computing. They develop an optimal and heuristic (near-optimal) method to minimize the cost of offloading the applications in a fog computing platform. In their model, they consider the cost of deployment of a VM for each application as well as the communication delay for a given size of data. They conclude that an appropriate set of fog nodes to host the VMs for each application is a key factor for minimizing the cost of fog computing resource management. Inspired by their work, in this paper, we propose a novel resource management model for service placement of distributed multi-task applications in fog computing through mathematical modeling of this platform.

Our main design goal is to reduce the communication cost between the candidate nodes hosting different task modules of an application by selecting a group of nodes that are near to each other and also as close to the source of the data as possible. The communication cost considered in this paper is an abstract parameter to characterize the penalty of communication in distributed application execution, in terms of the size and range of involved communication events. The rationale is that the larger (smaller) this penalty is, the less (more) optimal the mapping is performance-wise. The adopted communication cost concept reflects (predicts) the fog system's behavior with respect to more concrete communication-related parameters such as communication delay or communication energy consumption.

Figure 2 shows an overlay architecture of our resource management model in an IoT-fog computing system. The physical network underlay illustrates the sensors streaming their data to the fog nodes that are associated with them. The virtual network overlay shows an overlay network controlled by a virtual machine manager (VMM) eliminating the routers that provide physical interconnections between the

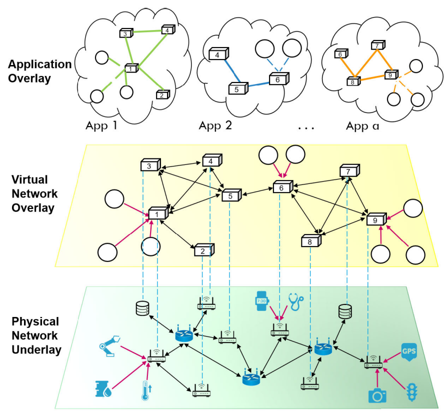


FIGURE 2. Overlay Architecture of fog computing network.

computing nodes. The task modules of each distributed application are mapped onto a set of fog computing nodes forming a separate overlay network, i.e., an application overlay, on top of the virtual network overlay.

The main contributions of this paper include:

- Mathematical modeling and presentation of distributed multi-task applications and fog computing systems.
- Formulating an optimization problem to reduce the overall communication costs in the system.
- Design and implementation of a heuristic greedy algorithm for the resource management problem for multi-tasking applications in fog computing networks.
- Performance evaluation of the proposed solution.

The rest of the paper is organized as follows. In Section 2, we present the research methodology of this study. In section 3 we present the mathematical modeling of the system and optimization problem formulation for multi-tasking resource management in the fog. In Section 4, we present our proposed algorithm for the problem. In Section 5, we provide performance evaluation for the proposed solution. Finally, in Section 6, we end with some concluding remarks.

II. METHODOLOGY

In this section, we briefly present the research methodology used in our study.

- **Modeling the applications and network:** We describe our approach for modeling an application as a composition of smaller task modules. In this study, we do not present a new method for application partitioning, but rather we consider an application comprising an ensemble of multiple task modules from the beginning. We also model our fog computing network consisting of nodes that are randomly distributed into a limited physical area.
- **Problem formulation:**We present the factors that constrain our model and formulate an optimization problem for resource management, in the context of task mapping, with the objective of minimizing the overall communication cost.
- **Heuristic method:** In order to validate our model, we propose a heuristic greedy algorithm for solving the formulated resource management problem.
- **Experimental analysis:** Finally, we investigate our proposed model from different points of view. We compare the properties of a method providing the absolute optimal mapping solution with the results obtained by the proposed heuristic greedy-based algorithm.

TABLE 1. List of notations.

Constants	
F	The fog node set
S	The sensor set
A	The application set
T	The task module set
ρ	Wireless communication range for each fog node or sensor
$\mathbf{A} \in \{0,1\}^{A\times T\times T}$	A binary-valued tensor that represents communication between task modules of an application
$\mathbf{R} \in \{0,1\}^{A\times T}$	A binary-valued incidence matrix representing the associations between task modules and applications
$\mathbf{Z} \in \{0,1\}^{S\times F}$	A binary-valued matrix that indicates if a fog node f is directly reachable from a sensor s
$\mathbf{N} \in \mathbb{N}^{F\times F}$	A distance matrix that represents the connections between fog nodes in the network.
$\mathbf{u} \in \mathbb{R}^{F}$	Vector of up-link costs of the fog nodes.
$\boldsymbol{\lambda} \in \mathbb{R}^{A}$	Vector of streaming rates of the applications.
$\mathbf{d} \in \mathbb{R}^{A}$	The length of data package uploaded to an application a
$\mathbf{c} \in \mathbb{R}^{T}$	Vector of container sizes of the task modules.
$\mathbf{h} \in \mathbb{R}^{F}$	A vector consisting of the hard disk (storage) capacities of fog nodes
Variables	
$\mathbf{M} \in \{0,1\}^{A\times T\times F}$	A binary-valued tensor variable that indicates if a task module t from an application a is mapped onto a fog node f

III. SYSTEM MODELING

In this section, we introduce our fog resource management model. For the convenience of the readers, the main notations used in this paper are listed in Table 1.

A. APPLICATION MODEL

We model an application in a fog computing system as an ensemble of task modules with inter-dependencies. Each task module $t \in T$ is a single function/portion of an application that receives input data, provided by precedent tasks, and

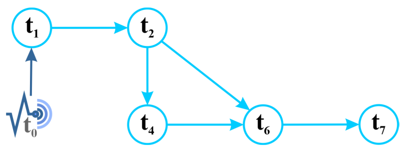


FIGURE 3. Task graph model of an application.

produces specific output data and sends it to the subsequent tasks. Hence, application mapping in a fog computing system follows a *many-to-many* function that should be calculated based on the topology of the network and the current locations of the sensors. We consider each task module developed as a bundle including a lightweight virtual machine, i.e., a container, and the source code. We define each application in the system as a directed graph.

We employ a simple mathematical model for representing applications running in the system. Let us denote by $\mathbf{A} \in \{0,1\}^{A\times T\times T}$ the tensor, whose each horizontal slice $\mathbf{A}_{a,:,:} \in \{0,1\}^{T\times T}$ is an *adjacency matrix* that represents the task graph of application a. Each entry $\mathbf{A}_{a,t_s,t_d}$ in the tensor represents communication between pairs of tasks (i.e., source and destination tasks) within the task graph of the application a that receives data from a particular sensor. To mark the first task module that receives the data from the sensor, we consider a dummy task module that we call the source node in the application's task graph, i.e., t_0, to represent the streaming sensor in $\mathbf{A}$. In this case, the task module that receives the data from t_0 will represent the first task module in the application task graph. Accordingly, an incidence matrix $\mathbf{R} \in \{0,1\}^{A\times T}$ can be obtained that represents the set of task modules composing the application. The following example (Figure 3) shows an application task graph for a given application and a set of associated sensors. The application a_1 is composed of following five task modules: t_1, t_2, t_4, t_6, and t_7, i.e., $\mathbf{R}_{a_1,:} = [1,1,1,0,1,0,1,1]$, where the involved tasks are a subset of $T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. The adjacency matrix $\mathbf{A}_{a_1,:,:}$ represents the task graph of this application in our system:

$$\mathbf{A}_{a_1,:,:} = \begin{bmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0 \end{bmatrix} \quad (1)$$

B. NETWORK MODEL

A fog computing network comprises heterogeneous computing nodes that communicate with each other directly or indirectly through network routers forming multi-hop paths. We denote by $\mathbf{N}$ a *distance matrix* to represent our network. Each element in this matrix represents the connection between a pair of fog computing nodes. Here, the values equal to 1 indicate a direct connection, i.e., the nodes are either physically connected or within the wireless communication range of each other. On the other hand, the values greater than 1 represent indirect connections through intermediate fog nodes or routers. In this case, the value is equal to the hop distance in the shortest path between two nodes. Two fog nodes are *reachable* from each other if there is a direct or indirect (multi-hop) connection between them. Without loss of generality, we assume that all the fog nodes in the network are reachable from each other.

1) SENSOR-NODE COMMUNICATION

Without loss of generality, we assume that all fog nodes and sensors are using wireless technology for communication. We define a radius ρ for each fog node and sensor that determines its communication range. Fog nodes $f \in F$ and sensors $s \in S$ can communicate with each other only if they are within the communication range of each other. Let us denote by $\mathbf{Z} \in \{0,1\}^{S\times F}$ a binary *incidence matrix* between sensors' set S and fog nodes' set F that indicates if a sensor $s \in S$ and a fog node $f \in F$ are within the communication range of each other, i.e., directly reachable from each other.

For the sake of simplicity, we assume that each application receives data from only one sensor, and the other way around, each sensor streams the data only to one application. That means that there is one and only one application associated with each sensor and vice versa. So, there is a one-to-one relationship between an application and its corresponding sensor. This being the case, we can refer to the application and sensor interchangeably in our model. Sensors stream their data as packages with the length of $\mathbf{d}_a, \forall a \in A$, to the fog nodes $f \in F$ they are associated with, in specific intervals with an up-link cost of $\mathbf{u}_f, \forall f \in F$ and an up-link streaming rate of $\boldsymbol{\lambda}_a, \forall a \in A$. Sensors $s \in S$ can stream their data only to the nodes $f \in F$ that are directly reachable from them, i.e., to the nodes for which $Z_{sf} = 1$ holds. Each sensor $s \in S$ should be associated to one and only one fog node at the time to ensure that the network traffic is not overloaded redundantly.

C. MOTIVATION

Let us consider a toy example to demonstrate the main motivation for this work. Figure 4 illustrates how a distributed application is mapped onto a network. For simplicity and for the sake of illustration, we consider a network with a 4×4 grid-mesh network topology. A wearable ECG sensor needs to offload its data to the fog network for processing. The sensor is within the communication range of the nodes 1 and 2 in the fog network. The ECG processing application is composed of 5 task modules, i.e., $\mathbf{R}_{a,:} = (1,1,0,1,0,1,1,1)$. First, the sensor will be associated with one of the fog nodes within its range. In our example, the sensor gets associated with the fog node 1. The resource mapping algorithm needs then to select a group of fog computing nodes that can accommodate and process the containers of the task modules of the ECG processing application, providing them

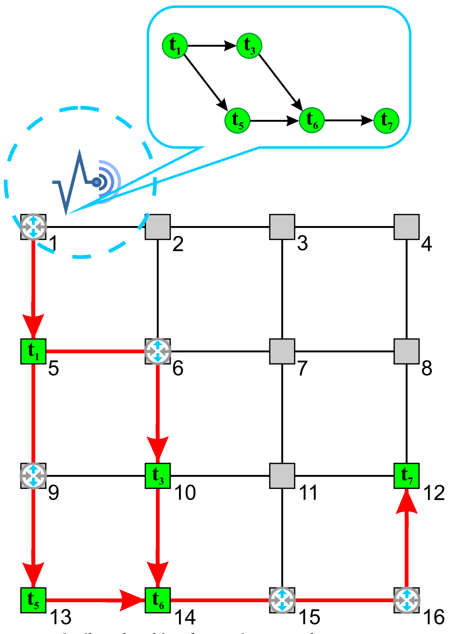


FIGURE 4. Distributed Multi-Task mapping example.

with appropriate communication paths. In our example, the fog nodes 5, 10, 12, 13, 14 are the selected nodes. Since fog nodes in general have limited resources in terms of storage capacity and processing power, some nodes cannot be chosen to host the containers of the task modules. These nodes, instead, can act as intermediate routing nodes. In the example, the nodes 6, 9, 15, 16 are such routers.

In this work, our optimization goal is to reduce the communication costs and delays between the task modules of an application by choosing the nodes that are as close as possible to the involved sensors and in the proximity of each other.

D. PROBLEM FORMULATION

In this section, we define and formulate the resource management and task mapping problem for distributed applications in fog computing systems.

An application is partitioned into smaller task modules that can be deployed as containers at fog computing nodes in the system. Each task module within an application needs to be mapped onto a selected fog node. We define a binary-valued tensor variable $\mathbf{M} \in \{0, 1\}^{A\times T\times F}$, that indicates whether the task module $t \in T$ from the application $a \in A$ is mapped onto the fog node $f \in F$, i.e.,

$$\mathbf{M}_{a,t,f} = \begin{cases} 1, & \text{task module } t \text{ from application } a \text{ is mapped} \\ & \text{onto fog node } f \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

An obvious base requirement for this mapping is that the considered task module t is part of the considered application a. This condition can be expressed as:

$$\sum_{f\in F} \mathbf{M}_{a,t,f} = \mathbf{R}_{a,t}, \quad \forall a \in A,\ t \in T\ . \tag{3}$$

Since the applications are entering the fog computing system dynamically at run-time, assigning multiple task modules of an application to the nearest fog node will decrease the QoS of applications that will join later. In this case, the nearest fog nodes will soon be fully loaded with the tasks of the earlier applications, and, consequently, newly arriving applications will need to be mapped onto the fog nodes far away from their data sources, resulting in higher communication costs and delays. To avoid this condition and to effectively granulate the QoS of all involved applications, our premise is that one and only one task module of an application a can be mapped onto any given fog node. However, this constraint does not apply to a (dummy) source task t_0 representing a gateway that passes sensor data to an application. In other words, such a gateway node could also be a candidate for processing one of the computational task modules of this application. The constraint is formulated as follows:

$$\sum_{t\in T\setminus\{t_0\}} \mathbf{M}_{a,t,f} \le 1, \quad \forall a \in A,\ f \in F\ . \tag{4}$$

A sensor can be associated to a fog node, if and only if the fog node is reachable from the sensor. To ensure this, we define the following constraint:

$$\mathbf{M}_{a,t_0,f} \le \mathbf{Z}_{s_a,f} \quad \forall a \in A,\ f \in F \tag{5}$$

Mapping and deployment of containers of the task modules is possible only if there is enough disk/storage space in the candidate fog nodes. This condition can be expressed as:

$$\sum_{a\in A}\sum_{t\in T} \mathbf{M}_{a,t,f}\mathbf{c}_t + \mathbf{M}_{a,t,f}\mathbf{d}_a \le \mathbf{h}_f \quad \forall f \in F\ , \tag{6}$$

where $\mathbf{c}_t$ is the container size of a running task module t, and $\mathbf{h}_f$ is the hard disk (storage) capacity of a fog node f.

E. MIQP FORMULATION

In this work, we associate resource management and task mapping in fog computing with the objective of minimizing the overall communication cost in the system. In the optimum case, the data is processed in the fog nodes that are in the proximity of each other and as near as possible to the sensors providing the data streams. The total communication cost in the system is calculated as the sum of the up-link and inter-node communication costs for every application that receives data from sensors, i.e.,

$$C_{Com} = C_{Up-link} + C_{Inter-Node} \tag{7}$$

The communication cost between two points in the network is determined by the amount and rate of the data transferred between the two points as well as the hop distance between the two respective nodes. In the case of up-link communication, since the sensors are directly connected to the receiving fog nodes, we consider the hop distance equal to 1. For inter-node communication, we use the hop distances stored in **N**. Hence, the total communication cost can be calculated as:

$$C_{Com} = \sum_{a\in A}\sum_{f\in F} \mathbf{u}_f \mathbf{M}_{a,t_0,f} \boldsymbol{\lambda}_a \mathbf{d}_a + \sum_{a\in A}\sum_{t_s\in T}\sum_{t_d\in T}\sum_{f_s\in F}\sum_{f_d\in F} \mathbf{M}_{a,t_s,f_s}\mathbf{N}_{f_s,f_d}\mathbf{M}_{a,t_d,f_d}\mathbf{A}_{a,t_s,t_d}\boldsymbol{\lambda}_a\mathbf{d}_a \tag{8}$$

Our goal is to minimize the overall communication cost by choosing the best setting for **M**. Since the distance dependence in the inter-node communication cost function is quadratic with respect to **M**, we can formulate a mixed-integer-quadratic-programming (MIQP) problem as follows:

$$\begin{aligned} &MIQP: \\ &\text{min}:\ (8), \\ &\text{subject to}:\ (3), (4), (5), (6) \\ &\text{and}:\ \mathbf{M} \in \{0,1\}^{A\times T\times F} \end{aligned} \tag{9}$$

IV. SOLUTIONS

The MIQP problem defined in the previous section can be solved in different ways. To achieve an optimum solution for a given resource management task, we implement our multi-task resource management model using the Gurobi optimizer. However, since the search space grows exponentially by adding more fog computing or sensor nodes to the system, an optimum solution may not be feasible for real-world scenarios. The fog computing nodes are resource-constrained, and the dynamic decisions for resource mapping need to be done swiftly to reduce the service delivery time and to contribute to better QoS. Hence, to evaluate our model's efficiency and performance, we propose a greedy algorithm as a heuristic method that guarantees a near-optimal solution which is less computationally intensive and faster to achieve, more suitable for a resource-constrained computing environment.

A. PROPOSED GREEDY ALGORITHM

We propose a two-phase distributed greedy algorithm as a heuristic solution for our resource management model. In the first phase presented in Algorithm 1, the system greedily associates the sensors to the fog nodes considering the constraints (5) and minimizing the system's overall up-link cost. In the second phase, at each iteration, the greedy algorithm assigns a fog node with the minimum communication cost for one task module from each application considering the constraints (3), (4), (6). At each iteration, the algorithm stores the next node(s) in a queue based on the sequence defined in each application's task graph. Once all the task modules have been mapped onto fog nodes, a communication graph from the selected fog nodes for processing each multitask application is formed, and the overall cost is calculated based on Equation (8). The greedy algorithm is presented in more detail in Algorithm 1.

Most of the tensors and matrices used in our approach are very sparse, meaning that they mostly comprise zero values. Consequently, to improve the performance and simplify the complexity of the algorithm, we only consider the non-zero values in the tensor M that together represent the total number of tasks modules from all the applications that are allocated to the fog nodes. This corresponds to the total number of elements that are pushed into the queue in Algorithm 1. Similarly, we consider only the non-zero values in the matrix Z that represent the number of all possible connections that one sensor could have with the fog nodes within its range. So, if we denote τ to correspond to the number of non-zero values in the tensor M and Z where $a \leq \tau \leq at, \forall a \in A, t \in T$, the complexity of the first and second phases could be calculated as $O(\tau)$, and $O(f\tau), \forall f \in F$ respectively. So the total complexity of our algorithm in the worst case is $O(f\tau), \forall f \in F$.

Algorithm 1 Greedy Algorithm

```
Data: Communication radius and coordinates of fog
      nodes and sensors
Result: $\mathbf{M}_{a,t_0,f}$
▷ Phase 1: Sensor association;
for each non-zero value in $Z_{sf}$ do
    Calculate the up-link cost;
    $f$ ←FogNode with the minimum up-link cost from $s$;
    $\mathbf{M}_{a,t_0,f}$ ← 1
    $Queue.PUSH$ ← [a, 0 and, the candidate gateway
    fog node]
end
▷ Phase 2: Task module mapping;
$L$ = [ ] // a list that contains the Fog Nodes that are
already assigned for a task module of the application
while There is an item in the Queue do
    $a, t_s, f_s$ ← $Queue.POP$
    for each unallocated task module $t_d$ acting as a
    destination of $t_s$ in application $a$ do
        $f_d$ ← find a fog node closest to $f_s$ that is not in $L$
        $Queue.PUSH$ ([$a$, $t_d$, $f_d$])
        $L.append(f_d)$
        update $\mathbf{M}_{a,t_d,f_d}$ ← 1
    end
end
```

V. EXPERIMENTAL SETUP AND EVALUATION

To evaluate the efficiency of the proposed solutions in a small-scale network of 100 × 100 meters, we consider 10 fog nodes and 3 sensor nodes in a field with a uniform distribution. The wireless coverage for the fog nodes and sensor nodes is set to 40 and 20 meters, respectively. Each sensor is associated with a multi-task application with the task graph

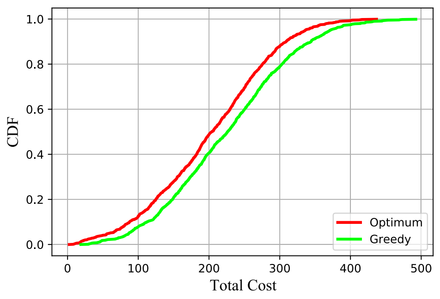


FIGURE 5. Cumulative Distribution Function of total cost for Optimum solution and Greedy algorithm.

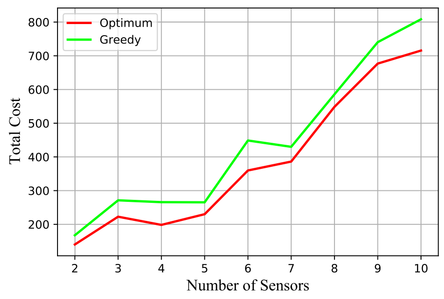


FIGURE 7. Effects of number of sensors on total cost.

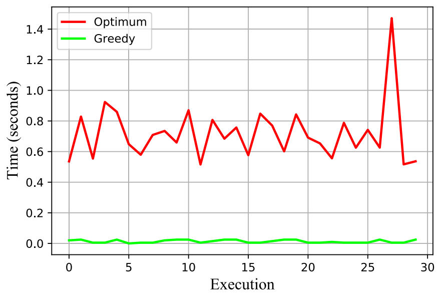


FIGURE 6. Execution time.

presented in Figure 3. The data streaming rate from each sensor is assigned based on a Gaussian distribution with a mean and standard deviation of 1.2 and 0.8, respectively. The container size for each task module is set to 0.4, and the storage capacity of each fog node is normalized to 1. The up-link cost associated with each fog node is also assigned based on a Gaussian distribution with a mean and standard deviation of 1.2 and 0.8, respectively.

We use a commercial solver, Gurobi 9.1, for solving the MIQP problem for the optimal case. We run 1000 simulation instances with different random seeds and plot the Cumulative Distribution Function (CDF) of both algorithms' total costs in Figure 5. According to this figure, our greedy solution achieves near-optimal results, with the CDF reaching on average 93.2% of the optimal value. This validates the correctness and efficiency of our algorithm.

To validate the performance of our greedy algorithm, we also record the execution time for 30 simulations with the same setup. Figure.6 illustrates the comparison of the execution time for both optimal and greedy methods. Our greedy heuristic algorithm achieves considerably lower and more predictable execution time than the optimal method, cutting the execution time for resource mapping to 0.97% of the time required by Gurobi-based optimization on average.

We visualize the task mapping for the optimal and greedy algorithms for 3 sensor nodes with the same setting that we described above. Figure 15 visualizes the solution by our greedy algorithm, and Figure 16 visualizes the optimal solution obtained by the Gurobi optimizer.

To evaluate the effects of a network setting on the average total cost in both greedy and optimal solutions, we run 10 simulations for each network setting and obtain the average values for the total cost. Figure 7 illustrates the average total cost under different settings for a group of sensors, varying from 2 to 10 sensors. To make the model feasible for our larger experiments, we set the storage capacity of each fog node to 10, a relatively large arbitrary value. We keep the rest of the setup the same as in our earlier experiments. It is evident, based on this diagram, that the total cost generally increases when increasing the number of the sensor nodes, as can be expected. However, the somewhat surprising local decline in the diagram in the case with 4 sensors in the optimum cost curve, and with 7 sensors in the greedy cost curve, can result from a placement of sensors in an area where the density of fog node distribution is higher. In such conditions, the majority of inter-node communication is either direct involving no intermediate nodes, or indirect involving only few intermediate nodes, leading to a lower total cost due to a lower average communication cost between task modules of applications associated with the sensors.

Figure 8 illustrates the average total cost with respect to varying rates of arriving data. In this experiment, the number of fog nods and sensors are set to 10 and 3, respectively. As can be seen, the average total cost for the greedy algorithm is very near to the optimal solution and increases with the arrival data rate.

To investigate the scalability of our model, we run a set of experiments to examine the CPU time and maximum used memory under different network settings. Figures 9 and 10 illustrate the CPU time and maximum used memory, respectively, with the number of sensors varying from 10 to 100. In this experiment, we set the number of fog nodes to 50. It is evident that our greedy algorithm outperforms the Gurobi solver, that computes the optimal solution, by requiring

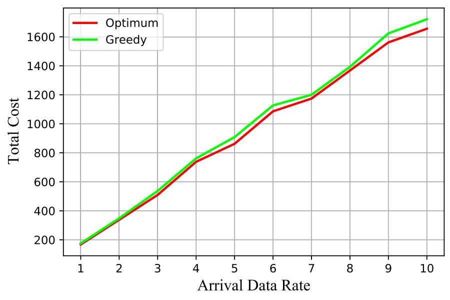


FIGURE 8. Effects of arrival data rate on total cost.

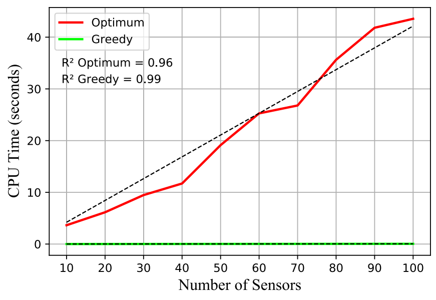


FIGURE 9. Effects of number of sensors on CPU execution time.

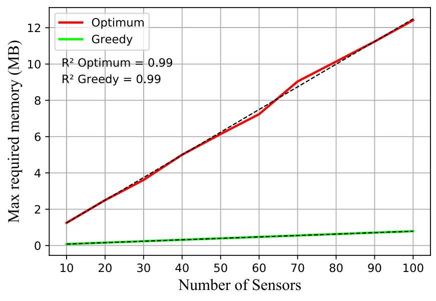


FIGURE 10. Effects of number of sensors on maximum required memory.

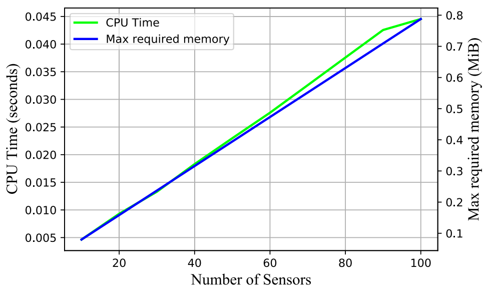


FIGURE 11. CPU time vs maximum required memory of greedy algorithm with respect to the number of sensors.

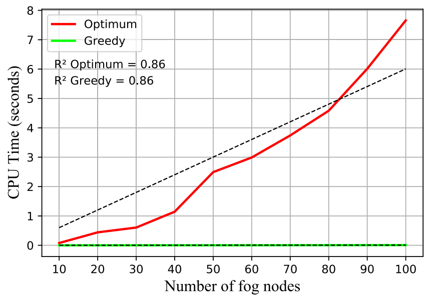


FIGURE 12. Effects of number of fog nodes on CPU execution time.

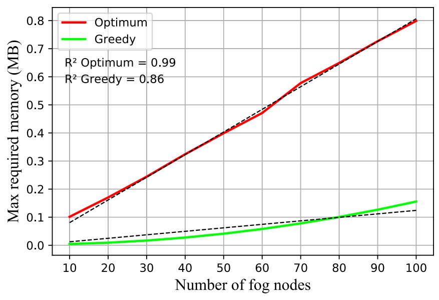


FIGURE 13. Effects of number of fog nodes on maximum required memory.

significantly less resources (i.e., on average 99.88% less CPU usage and 93.67 % less memory usage) for computing a task mapping solution. The resource usage in Gurobi-based optimization shows near-linear dependence on the number of sensors with a much steeper slope than in the case of the greedy method. Since the CPU time of the greedy method is almost constant (i.e., a horizontal line) in this experiment, we take a closer look into the resource usage of our algorithm in Figure 11, where we can see that for 10-100 sensors, the CPU time remains below 45 ms, and the memory need remains below 1 MB, indicating a relatively slow increase of the requirements. According to the experiment, for Gurobi, every additional set of 10 sensors increases the required CPU time by 4434 ms and the memory need by 1.24 MB on average. In comparison, for our greedy algorithm, these values are 4 ms and 0.078 MB, respectively. This provides evidence on the scalability of our algorithm with respect to the number of sensors in the network.

Figures 12 and 13 illustrate the CPU time and maximum used memory, respectively, with the number of fog nodes varying from 10 to 100 and the number of sensors having the constant value of 3. Also in this experiment, our greedy

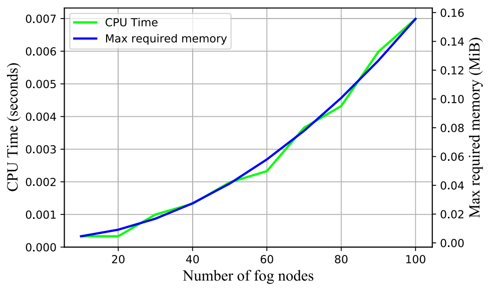


FIGURE 14. CPU time vs maximum required memory of greedy algorithm with respect to the number of fog nodes.

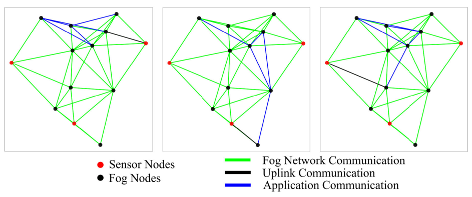


FIGURE 15. Task mapping examples (solved by our Greedy Algorithm).

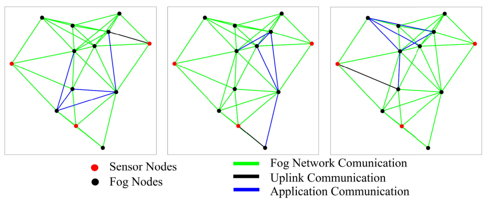


FIGURE 16. Task mapping examples (solved by Gurobi).

method clearly outperforms the Gurobi solver in terms of computing resource needs, requiring on average 99.9% less CPU time and 86.14 % less memory for solving a task mapping problem. Figure 14 shows that for 10-100 fog nodes, the CPU time and memory need do not exceed 7 ms and 0.16 MB, respectively. Moreover, for each additional group of 10 fog nodes, our greedy algorithm requires, on average, only 0.7 ms of more CPU time and 0.016 MB of more memory, whereas Gurobi requires 842 ms and 0.077 MB, respectively. Hence, compared with Gurobi, our method is highly scalable also with respect to the number of fog nodes in the network, facilitating dynamic run-time task mapping in networks of different sizes.

VI. CONCLUSION

We introduced a model for handling IoT requests with multi-tasking applications in a fog computing network and an analytical model to formulate the resource management problem from a communication cost perspective. We also proposed an algorithm based on a greedy principle to minimize the cost. Our proposed algorithm demonstrated a near-optimal efficiency, i.e., 93%, with respect to the communication cost (solution quality), while outperforming the considered optimal method in terms of the computing speed (solution latency), cutting the execution time to less than 1% of the execution time of the commercial Gurobi optimizer providing the absolute optimal solution. We showed that our proposed model is highly scalable. However, since the communication cost employed in this paper is an abstract parameter, not fully covering all communication-related cost factors in real networks, there is a need to further investigate concrete aspects such as communication delays and energy consumption when considering the resource management problem in fog computing systems. As part of the future work, we plan to extend our proposed model by considering the communication bandwidth between network nodes to calculate realistic communication delays. Moreover, we plan to take into account the deployment costs of task module containers in resource management problems.

REFERENCES

[1] S. Yi, Z. Hao, Z. Qin, and Q. Li, “Fog computing: Platform and applications,” in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[2] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, “A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing,” *IEEE Access*, vol. 7, pp. 115760–115773, 2019.

[3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[4] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for Internet of Things and analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence* (Studies in Computational Intelligence), vol. 546. Cham, Switzerland: Springer, 2014, pp. 169–186.

[5] I. Stojmenovic, “Fog computing: A cloud to the ground support for smart things and machine-to-machine networks,” in *Proc. Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.

[6] W. Zhang, Z. Zhang, and H.-C. Chao, “Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management,” *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 60–67, Dec. 2017.

[7] F. Hosseinpour, Y. Meng, T. Westerlund, J. Plosila, R. Liu, and H. Tenhunen, “A review on Fog Computing technology,” *Int. J. Advancements Comput. Technol.*, vol. 8, pp. 1525–1530, Oct. 2016.

[8] F. Hosseinpour, A. S. Siddiqui, J. Plosila, and H. Tenhunen, “A security framework for fog networks based on role-based access control and trust models,” in *Research and Practical Issues of Enterprise Information Systems*, A. M. Tjoa, L.-R. Zheng, Z. Zou, M. Raffai, L. D. Xu, and N. M. Novak, Eds. Cham, Switzerland: Springer, 2018, pp. 168–180.

[9] Y. Sahni, J. Cao, S. Zhang, and L. Yang, “Edge mesh: A new paradigm to enable distributed intelligence in Internet of Things,” *IEEE Access*, vol. 5, pp. 16441–16458, 2017.

[10] R. Deng, R. Lu, C. Lai, and T. H. Luan, “Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3909–3914.

[11] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, “A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing,” in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2015, pp. 324–329.

[12] J. Su, F. Lin, X. Zhou, and X. Lü, "Steiner tree based optimal resource caching scheme in fog computing," *China Commun.*, vol. 12, no. 8, pp. 161–168, Aug. 2015.
[13] J. Liu, E. Ahmed, M. Shiraz, A. Gani, R. Buyya, and A. Qureshi, "Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 99–117, Feb. 2015.
[14] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 4, pp. 23–32, Mar. 2013.
[15] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading inference for delivering applications in pervasive computing environments," in *Proc. 1st IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2003, pp. 107–114.
[16] E. Ahmed, A. Gani, M. Sookhak, S. H. A. Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *J. Netw. Comput. Appl.*, vol. 52, pp. 52–68, Jun. 2015.
[17] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile cloud Comput. Services (MCS)*, New York, NY, USA, 2012, pp. 29–36.
[18] Y. Liu, M. J. Lee, and Y. Zheng, "Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system," *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 2398–2410, Oct. 2016.
[19] H.-J. Hong, P.-H. Tsai, and C.-H. Hsu, "Dynamic module deployment in a fog computing platform," in *Proc. 18th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Oct. 2016, pp. 1–6.
[20] R. Eyckerman, S. Mercelis, J. Marquez-Barja, and P. Hellinckx, "Requirements for distributed task placement in the fog," *Internet Things*, vol. 12, Dec. 2020, Art. no. 100237.
[21] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.
[22] R. Patro, S. S. Patra, L. Barik, A. D. Prusty, and R. K. Barik, "Module placement scheme using MPC4.5 with Markov chain process for mobile fog computing environment," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Feb. 2021, pp. 304–309.
[23] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.
[24] L. Gu, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Dec. 2017.

FARHOUD HOSSEINPOUR (Member, IEEE) received the M.S. degree in information security from the University of Technology Malaysia (UTM). He is currently pursuing the Ph.D. degree in embedded computing architectures with the Department of Computing, University of Turku, Finland. He has served as a Lecturer and a Producer of several MOOCs for the European Institute of Innovation and Technology (EIT Digital), University of Turku. His main research interests include fog computing, the Internet of Things, intrusion detection systems, and modeling and optimization in embedded computing systems.

AHMAD NAEBI was born in Saray, Tabriz, Iran, in 1981. He received the B.Sc. degree from Tractor Applied Science and Technology University, Tabriz, in 2006, and the M.Sc. degree from Qazvin Islamic Azad University, Qazvin, Iran, in 2011. He is currently pursuing the Ph.D. degree with Xi'an Jiaotong University, Xi'an, China, in 2021. His research interests include programming languages, optimization, modeling, simulation, analyzing, designing, implementing, bond graph methodology, machine vision, signal processing, cognitive science, neuroscience, deep learning, and artificial intelligence.

SEPPO VIRTANEN (Senior Member, IEEE) received the D.Sc.(Tech.) degree in communication systems from the University of Turku, Finland, in 2004. He is currently an Associate Professor of cyber security engineering with the Department of Computing, University of Turku, where he is also the Vice Head of the Department. His current research interests include cyber security issues in communication and network technology, especially in the smart environment context.

TAPIO PAHIKKALA received the Ph.D. degree from the University of Turku, Finland, in 2008. He currently holds an Associate Professorship of machine learning with the University of Turku. He has led many research projects, supervised ten Ph.D. theses, held several positions of trust in academia, and served in the program committees of numerous international conferences. He has authored more than 150 peer-reviewed scientific articles and participated in the winning teams of several international scientific competitions/challenges. His current research interests include theory and algorithmics of machine learning, data analysis, and computational intelligence, as well as their applications on various different fields.

HANNU TENHUNEN received the M.Sc.EE. degree from the Helsinki University of Technology, in 1982, the Ph.D. degree from Cornell University, in 1986, and the Honorary Ph.D. degree. Since 1986, he was with the Tampere University of Technology as an Associate Professor and the coordinator of the National Microelectronics Program, from 1987 to 1991. Since January 1992, he has also been the Chair Professor of electronic system design with the KTH Royal Institute of Technology, Stockholm. He was the dean of the New School of ICT, KTH Royal Institute of Technology, from 2002 to 2005. For the period 2006–2011, he was the Director of the Turku Centre of Computer Science (TUCS), Finland. He has been a part-time Invited Professor at the University of Turku, Finland, since 2006. He has published over 940 reviewed publications and holds nine international patents. His current research interests include embedded electronics for autonomic and smart systems, and the IoT. He was the Education Director of the European Institute of Innovation and Technology, EIT Digital, for the period 2006–2011, at European level. He received the title of an Honorary Professor and the Mangolian Silver Award and Metal from the city of Shanghai for his contributions there.

JUHA PLOSILA (Member, IEEE) received the Ph.D. degree in electronics and communication technology from the University of Turku (UTU), Finland, in 1999. He is currently a Professor in autonomous systems and robotics with the Department of Computing, UTU. He is also the Head of the Autonomous Systems Laboratory (ASL) Research Group and the Smart Systems Unit, UTU. He leads the EIT Digital Master Programme in Embedded Systems at the EIT Digital Master School (European Institute of Innovation and Technology) and is a member of the Node Strategy Committee of the EIT Digital Finland Node. His research interests include adaptive multi-processing systems and platforms, intelligent multi-agent monitoring and control architectures, machine learning and optimization approaches, and application of heterogeneous energy efficient architectures to new computational challenges in the cyber-physical systems and the Internet-of-Things domains, with a recent focus on fog/edge computing (edge intelligence) and autonomous multi-robot systems.

1035. YAHYA, MANAL. Augmented reality based on human needs. 2022. Diss.

1036. KARUPPANNAN GOPALRAJ, SANKAR. Impacts of recycling carbon fibre and glass fibre as sustainable raw materials for thermosetting composites. 2022. Diss.

1037. UDOKWU, CHIBUZOR JOSEPH. A modelling approach for building blockchain applications that enables trustable inter-organizational collaborations. 2022. Diss.

1038. INGMAN, JONNY. Evaluation of failure mechanisms in electronics using X-ray imaging. 2022. Diss.

1039. LIPIÄINEN, SATU. The role of the forest industry in mitigating global change: towards energy efficient and low-carbon operation. 2022. Diss.

1040. AFKHAMI, SHAHRIAR. Laser powder-bed fusion of steels: case studies on microstructures, mechanical properties, and notch-load interactions. 2022. Diss.

1041. SHEVELEVA, NADEZHDA. NMR studies of functionalized peptide dendrimers. 2022. Diss.

1042. SOUSA DE SENA, ARTHUR. Intelligent reflecting surfaces and advanced multiple access techniques for multi-antenna wireless communication systems. 2022. Diss.

1043. MOLINARI, ANDREA. Integration between eLearning platforms and information systems: a new generation of tools for virtual communities. 2022. Diss.

1044. AGHAJANIAN MIANKOUH, SOHEIL. Reactive crystallisation studies of CaCO3 processing via a CO2 capture process: real-time crystallisation monitoring, fault detection, and hydrodynamic modelling. 2022. Diss.

1045. RYYNÄNEN, MARKO. A forecasting model of packaging costs: case plain packaging. 2022. Diss.

1046. MAILAGAHA KUMBURE, MAHINDA. Novel fuzzy k-nearest neighbor methods for effective classification and regression. 2022. Diss.

1047. RUMKY, JANNATUL. Valorization of sludge materials after chemical and electrochemical treatment. 2022. Diss.

1048. KARJUNEN, HANNU. Analysis and design of carbon dioxide utilization systems and infrastructures. 2022. Diss.

1049. VEHMAANPERÄ, PAULA. Dissolution of magnetite and hematite in acid mixtures. 2022. Diss.

1050. GOLOVLEVA, MARIA. Numerical simulations of defect modeling in semiconductor radiation detectors. 2022. Diss.

1051. TREVES, LUKE. A connected future: The influence of the Internet of Things on business models and their innovation. 2022. Diss.

1052. TSERING, TENZIN. Research advancements and future needs of microplastic analytics: microplastics in the shore sediment of the freshwater sources of the Indian Himalaya. 2022. Diss.

ISBN 978-952-335-885-0
ISBN 978-952-335-886-7 (PDF)
ISSN 1456-4491 (Print)
ISSN 2814-5518 (Online)
Lappeenranta 2022