Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Julie Hodne Gundersen

Digital forensics on fog-based IoT devices

Master's thesis in Information Security
Supervisor: Jens-Petter Skjelvåg Sandvik
Co-supervisor: Katrin Franke
June 2022

NTNU
Kunnskap for en bedre verden

Julie Hodne Gundersen

Digital forensics on fog-based IoT devices

Master's thesis in Information Security
Supervisor: Jens-Petter Skjelvåg Sandvik
Co-supervisor: Katrin Franke
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Digital forensics on fog-based IoT devices

Julie Hodne Gundersen

June 1, 2022

Abstract

With the rapid, constant growth of new devices connected to the Internet, more data is stored and handled in the Cloud. With all these devices logging activity worldwide, law enforcement faces challenges where possible evidence can be overlooked or hard to collect. Fog computing is a specific technology that usually works in relation to a cloud, where it has been hard to create forensically sound methods for collecting evidence. Devices placed in this type of edge-computing environment can contain important data related to criminal activity which is essential for forensic investigators to analyze.

This thesis experiments with fog networks in different scenarios and evaluates existing digital forensic frameworks for IoT devices placed in a fog. The experiments are performed in a simulated fog environment where it is possible to see how the dynamic service movement can affect the evidence location and how the nodes can store data.

There was a dynamic movement of services across nodes in the network in all of the different scenarios. The results show that when the scale of the network increases, the number of service placements on each node decreases. The amounts of highly volatile data in fog networks become an issue when the process of locating the possible sources of evidence takes time. The initial location of data is likely to change between the time of the incident and the time of the investigation.

The simulation results were further discussed in the context of state-of-the-art forensics related to fog computing. Frameworks developed for fog forensics are mainly proactive solutions that focus on digital forensic readiness, where modules are implemented on the fog nodes prior to an incident. While there exist some specific frameworks for IoT forensics, they do not cover the topic of fog computing. Detecting abnormalities in the network traffic can prevent some types of cybercrime, but it is not feasible to implement in all fog networks. More research is necessary to ensure that all relevant evidence in fog networks is collected in a forensically sound manner.

Sammendrag

Den konstante økningen av nye enheter som blir koblet til internett resulterer i ekstreme mengder data som blir lagret og håndtert i en sky. Alle disse enhetene som logger aktivitet over hele verden, skaper utfordringer for etterforskere hvor digitalt bevis kan bli oversett eller vanskelig å innhente. *Fog computing* er en teknologi som vanligvis er satt i sammenheng med en sky, hvor det har vært vanskelig å lage forsvarlige metoder for å samle bevis uten å påvirke dens integritet. Enheter plassert i denne typen *edge computing*-miljøer kan inneholde viktig data relatert til kriminell aktivitet. Det er derfor viktig at etterforskere klarer å innhente og analysere denne typen data i en etterforskningsprosess.

I denne oppgaven har det blitt eksperimentert med fog nettverk i ulike scenarier, hvor eksisterende etterforskningsmetoder for IoT enheter plassert i en fog har blitt evaluert. Disse eksperimentene har blitt utført i et simulert fog-miljø, hvor det var mulig å se hvordan den dynamiske bevegelsen av tjenester påvirker lokasjonen av mulig bevis, og hvordan nodene i nettverkene lagrer data. Fra eksperimentene var det mulig å observere denne dynamiske plasseringen av tjenester mellom noder i nettverkene i de ulike scenariene. Resultatene viser at når størrelsen på nettverket øker, så minker antallet plasseringer av tjenester på hver node. Mengden data med kort levetid i fog nettverk blir et problem når prosessen av å lokalisere mulige kilder for bevis tar tid. Lokasjonen av data i nettverket kan fort endre seg mellom hendelsestidspunktet og starten på en etterforskningsprosess.

Resultatene av simuleringen ble videre diskutert i sammenheng med nyere etterforskningsmetoder relatert til fog computing. Rammeverk utviklet for etterforskning i en fog er i hovedsak proaktive løsninger som fokuserer på beredskap, hvor moduler implementeres på fog-nodene før en hendelse inntreffer. Det finnes ulike rammeverk for IoT-etterforskning, men disse dekker ikke fog computing. Oppdagelse av avvik i nettverkstrafikken kan forhindre enkelte typer digital kriminalitet, men det er ikke alltid mulig å implementere slike metoder i et fog nettverk. Det er derfor nødvendig med mer forskning for å sikre at alle relevante bevis i fog nettverk kan bli samlet inn på en forsvarlig måte som opprettholder bevisets integritet.

Preface

This thesis concludes my Master of Science degree in Information Security at the Norwegian University of Science and Technology (NTNU) in Gjøvik.

The topic was inspired by the Cyber Investigations compendium [1] used in the course IMT4130 Cybercrime Investigation, Spring 2021. The chapter on fog computing introduced me to the topic and its challenges within digital forensics. The gaps in current research and wanting to learn more about it were motivational factors for writing this thesis.

Acknowledgements

I want to thank my supervisor Jens-Petter Skjelvåg Sandvik for the great guidance on the topic of fog computing and general advice during the project. His technical skills and knowledge helped make this thesis possible.

I would also like to thank my co-supervisor Prof. Katrin Franke for her feedback and guidance along the way.

Lastly, I want to give a special thanks to my family for their support and encouragement throughout my studies.

Julie H. Gundersen

Julie Hodne Gundersen

Oslo, 01-06-2022

Contents

Abstract . . . iii
Sammendrag . . . v
Preface . . . vii
Contents . . . ix
Figures . . . xiii
Tables . . . xv
Acronyms . . . xvii
Glossary . . . xix
1 Introduction . . . **1**
- 1.1 Topic covered by the project . . . 1
- 1.2 Keywords . . . 1
- 1.3 Problem description . . . 1
- 1.4 Justification, motivation and benefits . . . 2
- 1.5 Research questions . . . 3
- 1.6 Scope and contributions . . . 3
- 1.7 Thesis outline . . . 3
- 1.8 Ethical and legal considerations . . . 4

2 Background . . . **5**
- 2.1 Cloud computing . . . 5
- 2.2 Edge computing . . . 6
- 2.3 Fog computing . . . 6
  - 2.3.1 Fog architecture . . . 7
  - 2.3.2 Service Placement . . . 7
  - 2.3.3 Software-Defined Networking (SDN) . . . 8
- 2.4 Digital forensics . . . 9
  - 2.4.1 Cloud forensics . . . 10
  - 2.4.2 IoT forensics . . . 11
  - 2.4.3 Fog forensics . . . 11
- 2.5 Related work . . . 11
  - 2.5.1 Fog computing and forensics . . . 12
  - 2.5.2 Fog simulation . . . 15
  - 2.5.3 Service placement in fog computing . . . 15

3 Methods . . . **19**
- 3.1 Research approach . . . 19

3.2 Literature study 19
3.3 Empirical study 21
3.3.1 Scenarios 21
3.3.2 Fog-network simulation using YAFS 22
3.3.3 Fog-network simulation using YAFS where the model includes volatility 23
3.3.4 Digital forensic analysis 24
3.3.5 Possibilities and restrictions 24
4 Experimental setup, implementation and results 25
4.1 YAFS simulator 26
4.1.1 Terminology 27
4.1.2 Experimental setup 29
4.2 Fog network simulation using YAFS 32
4.2.1 Scenario 1 - Witness 32
4.2.2 Scenario 2 - Tool 37
4.2.3 Scenario 3 - Target 41
4.3 Fog network simulation using YAFS where the model includes volatility 48
4.3.1 Results from scenario 1 - Witness 49
4.3.2 Results from scenario 2 - Tool 52
4.3.3 Results from scenario 3 - Target 54
4.4 Summary of results 57
4.5 Hypotheses discussion 57
5 Digital forensics analysis 59
5.1 Scenario 1 60
5.1.1 Proactive solutions 61
5.1.2 Identification 62
5.1.3 Collection 64
5.2 Scenario 2 65
5.2.1 Proactive solutions 65
5.2.2 Identification 66
5.2.3 Collection 67
5.3 Scenario 3 68
5.3.1 Proactive solutions 69
5.3.2 Identification 69
5.3.3 Collection 71
5.4 Summary 72
6 Discussion, conclusion and future work 75
6.1 Discussion on the SPP 75
6.2 Theoretical implications 76
6.3 Practical considerations 77
6.4 Conclusion 78
6.5 Future work 79
Bibliography 81

A Additional Material **87**
A.1 Service placements from Scenario 1 87
A.2 Service placements from Scenario 2 87
A.3 Service placements from Scenario 3 88

Figures

1.1 Fog architecture 2

2.1 Fog architecture with SDN 9

4.1 Architecture of YAFS 26
4.2 Scenario 1- Initial network topology 33
4.3 Module placement in each node 35
4.4 Scenario 2 - Initial network topology 38
4.5 Module placement in each node 40
4.6 Scenario 3 - Initial network topology 44
4.7 Module placement in each node 46

Tables

2.1 Summary of forensic frameworks applied for fog/IoT in research . . 14
2.2 Summary of service placement algorithms and approaches 17

4.1 Scenario 1 - Technical details 33
4.2 Scenario 2 - Technical details 38
4.3 Scenario 3 - Technical details 43
4.4 Scenario 1 - Proxy nodes statistics 49
4.5 Scenario 1 - Sink nodes statistics 49
4.6 Scenario 1 - Source nodes statistics 50
4.7 Scenario 2 - Proxy nodes statistics 52
4.8 Scenario 2 - Sink nodes statistics 52
4.9 Scenario 2 - Source nodes statistics 53
4.10 Scenario 3 - Proxy nodes statistics 54
4.11 Scenario 3 - Sink nodes statistics 55
4.12 Scenario 3 - Source nodes statistics 55
4.13 Hypothesis statistics 58

5.1 Beneficial frameworks for each scenario 72

Acronyms

DES	Discrete Event Simulation. xv
GiB	Gibibyte. xv
IDS	Intrusion Detection System. xv
IoT	Internet of Things. xv
IPS	Intrusion Prevention System. xv
MiB	Mebibyte. xv
QoS	Quality of Service. xv
VM	Virtual Machine. xv

Glossary

empirical research Type of research that includes observation and testing against hypotheses. xv

qualitative research Conducting research with analysis of non-numerical data. xv

quantitative research Conducting research with analysis of numerical data. xv

simulation Synthesizing a real system in a controlled environment. xv

volatility The lifetime of data. High volatility = lower lifetime, Low volatility = longer lifetime. xv

Chapter 1

Introduction

This chapter introduces the problem area that has been investigated in this thesis. After introducing the problem, the research questions are presented along with the scope of the project. The last section of the chapter describes the outline of the thesis.

1.1 Topic covered by the project

Fog computing introduces processing closer to the edge devices, which can provide lower latency in networks [2]. To ensure this benefit, fog networks include dynamic service movement and highly volatile data. A specific service placement algorithm defines the service movement in the networks, and there are numerous approaches to solve this.

The fog environment is similar to the cloud environment. However, it has a decentralized structure, which means that frameworks used for digital forensics in the cloud are not fully applicable in fog scenarios [3]. It is not easy to create clear evidence-gathering frameworks in this environment.

Therefore, this project experiments with and analyses the topic of fog computing in the context of digital forensics, where the results of this research hopefully benefit the development of new and better frameworks.

1.2 Keywords

Internet-of-Things forensics, Fog computing, Edge computing, Digital forensics, IoT-fog-environment, Fog evidence gathering, YAFS, Fog simulation

1.3 Problem description

Digital forensic investigators face new challenges every day, and with new technology, there must be developed frameworks and guidelines on how to collect evidence in a forensically sound manner. This requires extensive research to understand the technology and how to include it in the investigation process. One topic that has been important to understand is cloud-based IoT devices.

Society relies heavily on cloud-based information, and the chances are high that there exists some IoT device that can provide valuable information in an investigation process. This area has been well-researched, but a related and less researched topic is *fog-based IoT devices*. The fog environment can exist related to the cloud and could contain devices that process large amounts of data. This environment differentiates from the cloud as it has a decentralized structure that is directly connected to the devices. This makes it more efficient when it comes to collecting and processing data, but on the other hand, it makes it harder for digital investigators to collect evidence [3]. A visual representation of the fog computing architecture is presented i Figure 1.1.

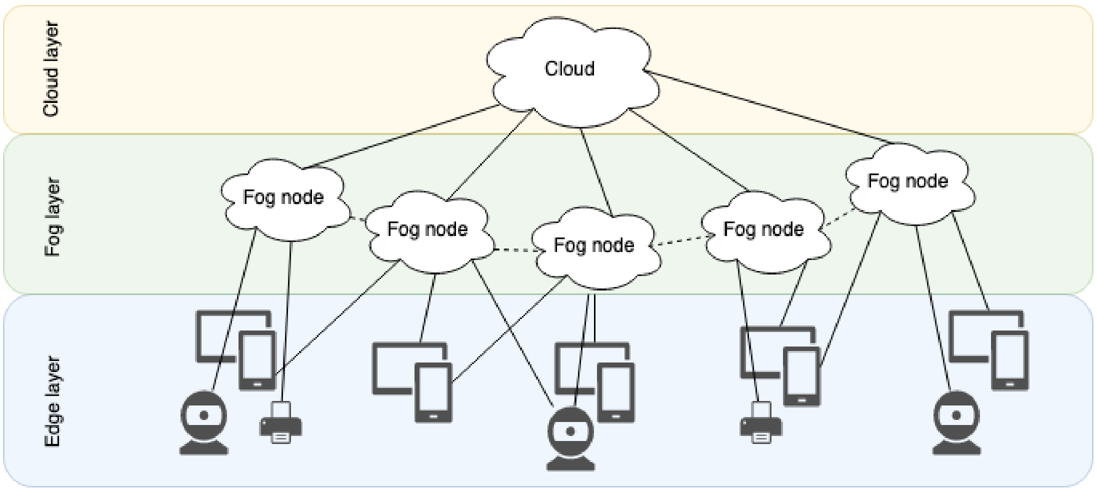

Figure 1.1: Fog architecture

There is no perfect framework for this process, and essential evidence could be left unnoticed. For the processes to improve, it is necessary to perform more experiments and research on the topic.

1.4 Justification, motivation and benefits

The thesis aims to provide valuable information to digital forensic investigators who face the difficult task of collecting evidence from fog-based IoT devices. As it is hard to develop frameworks and guidelines that will work in all cases, more experimenting and research can provide information that can improve current procedures. The aim of the project is for the result to contribute to creating optimized forensic processes in the fog-IoT environment in the future.

1.5 Research questions

The topic of IoT-fog-computing is broad, and it is therefore a need for specification and narrowing down the research for the thesis. The focus of the research evolves around forensics on IoT devices placed in a fog and how current frameworks for the investigation process are working for this purpose. For this to be specific enough and to ensure that there are measurable results, three research questions have been defined.

RQ1: Which types of service placement algorithms exist for fog computing?

RQ2: How does the dynamic service placement in fog-IoT systems affect the evidence location and the digital forensic process?

RQ3: How applicable are existing digital forensic methods for evidence gathering in fog-IoT systems?

1.6 Scope and contributions

The scope of this thesis is to understand and evaluate existing digital forensic methods and frameworks used in fog-IoT-environments. This is done by using a fog simulator to simulate fog networks. To make this possible, it was necessary to investigate the state-of-the-art in this area and attempt to define the challenges forensic investigators face today. A central part of fog computing is the service placement on the fog nodes, and **RQ1** has been defined to investigate this area. The overall aim of the thesis is to investigate how the dynamic service placement in fog networks affects the evidence location and the digital forensic process (**RQ2**) and evaluate how existing methods can apply in such scenarios (**RQ3**).

1.7 Thesis outline

This section introduces the chapters the thesis is divided into and a brief description of the contents. The thesis is divided into the following chapters:

Chapter 2 - Background introduces the main topics that are relevant for understanding the purpose of the research. The chapter first introduces cloud computing and edge computing, which are closely related to fog computing. Following these topics, fog computing is presented, where the general definition and architecture are described along with an introduction to service placement in fog networks. Software-Defined Networking is then presented as it is a highly relevant topic with fog computing. Further, the broad topic of Digital forensics is introduced. This section is divided into three parts: Cloud forensics, IoT forensics, and Fog forensics, and an explanation of each topic and their differences is provided. The final section presents related research in regards to Fog computing and forensics, Fog simulation, and Service placement in fog computing. This includes the state-of-the-art within the mentioned topics.

Chapter 3 - Methods includes the methods used to perform the research.

Chapter 4 - Experimental design and implementation covers the experiments that have been performed. This chapter starts with an overview of the simulator's terminology, followed by the experiment setup and details on running the simulator. The second section is dedicated to the first experiment, where there are three different scenarios used for generating results. The scenarios are explained with technical details along with the results for each of them. Further, there is a section for the second experiment where the same scenarios are used for generating results - this time where the module includes volatility. The last sections summarizes the most important findings from the experiments and the results of the hypothesis testing.

Chapter 5 - Digital forensic analysis is the chapter where the results of both experiments are put into the context of digital forensics. The chapter consists of four parts - one for each scenario and a summary at the end. Discussion on the state-of-the-art within relevant digital forensics is included in this chapter. The chapter includes the problem areas visible from the experiments and looks at the results in relation to the research questions.

Chapter 6 - Discussion, conclusion and future work firstly presents a discussion on the service placement problem. Further, the theoretical implications and practical considerations are presented, followed by the conclusion of the thesis and possibilities for future research.

1.8 Ethical and legal considerations

This thesis investigates how existing digital forensic methods can perform when there are IoT devices placed in fog networks. The experiments are performed in a simulated fog environment, which means that it is not necessary to access or use any sensitive data that exists. The scenarios in the experiments are entirely synthetic. This allows for a more straightforward analysis as no data needs to be censored or excluded in the results.

Chapter 2

Background

Cloud and fog forensics face many of the same challenges, but the topic of fog forensics is significantly less researched than cloud forensics. In order to understand the problems that exist within the field, it is important to understand the technology that creates these networks and what differentiates them. Another technology that is often discussed in relation to fog is edge computing, which is further presented in this chapter.

The chapter introduces the main areas this thesis focuses on. The chapter is separated into five main parts; Cloud computing, Edge computing, Fog computing, Digital forensics, and Related Work. The first parts looks into how Cloud and Fog-IoT computing work and aims to be a baseline for further understanding of the different digital forensic approaches. The digital forensics section includes Cloud forensics, IoT forensics, and Fog forensics. IoT devices can exist in both a cloud and a fog, and the topic of IoT forensics is therefore relevant in the thesis. As there are differences between cloud and fog, this chapter looks into how that affects the digital forensic methods that are to be used. The topics are also relevant for creating the simulated networks in the experiments.

2.1 Cloud computing

Fog computing is often seen in relation to a cloud, and the two technologies can appear similar. It is therefore necessary to understand the concept of cloud computing and how it differentiates from fog computing. In general, cloud computing is where "dynamically scalable and often virtualized resources are provided as services over the Internet" [4].

NIST (National Institute of Standards and Technology) specifies five essential characteristics of cloud computing [5]:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity

- Measured service

Combined, these characteristics mean that cloud computing will provide a user with automatic access to, e.g., storage or server time when needed (on-demand) from multiple access points. The cloud resources are pooled in a way that all clients will have their own physical and virtual resources based on their demands. The cloud capability for the consumers can be rapidly scaled to the demand, giving an elastic provisioning availability. The service will, in addition, provide measurements of resource usage, making it possible to optimize it for the consumer [5].

There are different ways that cloud computing services can be offered. The three most common models include Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (Iaas). These different models make it possible to adjust the services according to the consumers' requirements. IaaS would include the consumer having control over the OS, storage, and network components in some cases. While this might not be necessary for all consumers, PaaS or Saas are options with limited capabilities, whereas PaaS provides control over applications and configurations but not underlying cloud infrastructure. SaaS provides web-based access, and the consumer is not concerned about network maintenance, cloud infrastructure, and other operational factors [5].

2.2 Edge computing

Edge computing is a concept that often is confused with fog computing, which also encompasses end-devices and users [6]. IBM defines edge computing as "a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers" [7]. With this definition it is similar to fog computing, where the data is processed close to the edge of the network and therefore close to the connected devices. In edge computing, all the data that is processed at the edge is further sent to a connected cloud in data streams. This can include lots of unnecessary data, which is where fog computing can be a solution [8].

2.3 Fog computing

To answer the research questions defined for this thesis, it is important to understand what a fog network is and what it means when IoT devices are connected to fog nodes. When discussing the topic of fog computing, it is often in a setting where there are IoT devices involved [6]. Fog computing is considered to be a type of edge computing technology where fog nodes are placed between the cloud and the edge devices [2], providing additional benefits to the network. Cisco introduced the concept of fog computing in 2012, and it is considered a decentralized extension of the cloud where additional nodes are placed between the end users and the cloud. This can reduce latency as data processing happens in the fog nodes and, therefore, closer to the IoT devices, and it does not have to be sent to the centralized cloud [2]. This can ensure that only the data defined as important will be transmitted and stored long-term in the cloud, and the rest can be filtered out in the fog nodes.

To highlight these features of fog computing, NIST defines six essential characteristics: **1)** Contextual location awareness and low latency, **2)** Geographical distribution, **3)** Heterogeneity, **4)** Interoperability and federation, **5)** Real-time interactions and **6)** Scalability and agility of federated fog-node clusters [6].

2.3.1 Fog architecture

Fog computing typically involves three main layers: the *edge layer*, the *fog layer* and the *cloud layer*. The edge layer is closest to the users and consists of IoT devices. It is often distributed over larger geographical areas. The fog layer implements the devices that process and temporarily store the network's requests, including routers, gateways, switches, and servers. Lastly, the cloud layer exists on top of these layers and consists of cloud servers that can store larger amounts of data and provide more computing power [9]. Figure 1.1 provides a visual overview of the typical fog architecture. There is now an additional layer between the cloud and the devices compared to cloud computing.

The fog layer provides services closer to the edge, and the processing of data can happen with lower latency as there is no need to send all of the requests to the centralized cloud. As it sits between the edge devices and the cloud layer, only necessary data can be forwarded to the cloud for further processing and storage. This functionality creates the possibility to save bandwidth [9].

2.3.2 Service Placement

In fog computing, the fog nodes consist of one or more physical devices that have the ability to store, process data and have some networking capabilities [10]. The nodes contain services that are a part of different applications running in the fog network. To deploy these services on the nodes, multiple algorithms exist with different priorities and focus areas. The placement of the services can affect the overall quality of the network, including performance, costs, and energy consumption [11]. The Service Placement Problem (SPP) was presented and discussed by Salaht et al. in [9], where the authors provide an extensive review of the existing problem areas and approaches to solve the SPP. In the paper, different types of classifications were used for the different algorithms. One of the ways one can classify the algorithms is by the resolution approach, where it is categorized based on the strategies used to solve the SPP. Salaht et al. present the following categories within this topic [9]:

> **Exact solution** - Algorithms in this category are often based on Integer Linear Programming where the problem is solved by linear functions and the variables in the function are integer. An exact solution could also be done

with an exhaustive search. This is not an optimal solution with fog computing as it works best for small problems and can be time-consuming.
Approximations - These algorithms make it possible to generate solutions to NP-hard optimization problems but will not ensure the best solution. Even if it does not provide the best solution, it is an efficient way of solving the problem.
Heuristics - Heuristic algorithms find a solution to the problem within a given time, with no guarantee regarding the performance. The aim is to have a solution that works, and it does not have to be the optimal solution.
Meta-heuristics - These algorithms aim at finding the best solution by being iterative and, in that way, improving the results. This also helps get past the local optima that often becomes the solution in heuristic algorithms.

Other ways of classifying the algorithms include focusing on the metrics or technical formulation [9], but further in the thesis, the classification based on the resolution approach will be used. There exist several different approaches in each category, which are further presented and described in section 2.5.

2.3.3 Software-Defined Networking (SDN)

A topic highly relevant to fog computing is Software-Defined Networking (SDN). SDN are networks where the physical devices such as switches or routers have software running on them, making it possible to set explicit rules for the network's traffic and, in this way, create complex rule sets. The software running on the devices is connected to a unit, such as a control unit, that manages all of the different network devices [12]. Compared to classic networking, SDN provides a way of centrally managing the network where new rules and policies do not have to be configured in each of the devices directly [13]. Looking at the architecture of SDN, it can be separated into three layers: 1) the forwarding layer, which consists of all the network devices, 2) the control layer, which is the centralized controller and 3) the application layer which provides access to the network [14].

In software-defined networking, OpenFlow is the communication protocol that is used the most. This protocol allows for access to the forwarding plane that exists in the network device and provides the ability to have an SDN controller [15]. As long as the network device supports OpenFlow, it is possible to connect to the controller, and rules and policies can be remotely added to the device.

With fog computing, SDN can be beneficial in the communication between the fog nodes [12]. One of the main characteristics of fog computing is reduced latency. It is beneficial that SDN also provides reduced time in adding new devices and configuring them correctly. Additionally, SDN can be used in fog networks to improve multiple security and privacy issues, such as within intrusion detection and authentication [14]. In the fog architecture, the fog nodes would be connected to the controller located on top of the fog layer, as shown in Figure 2.1, and with this controller, there would be more advanced traffic control for the entire fog network.

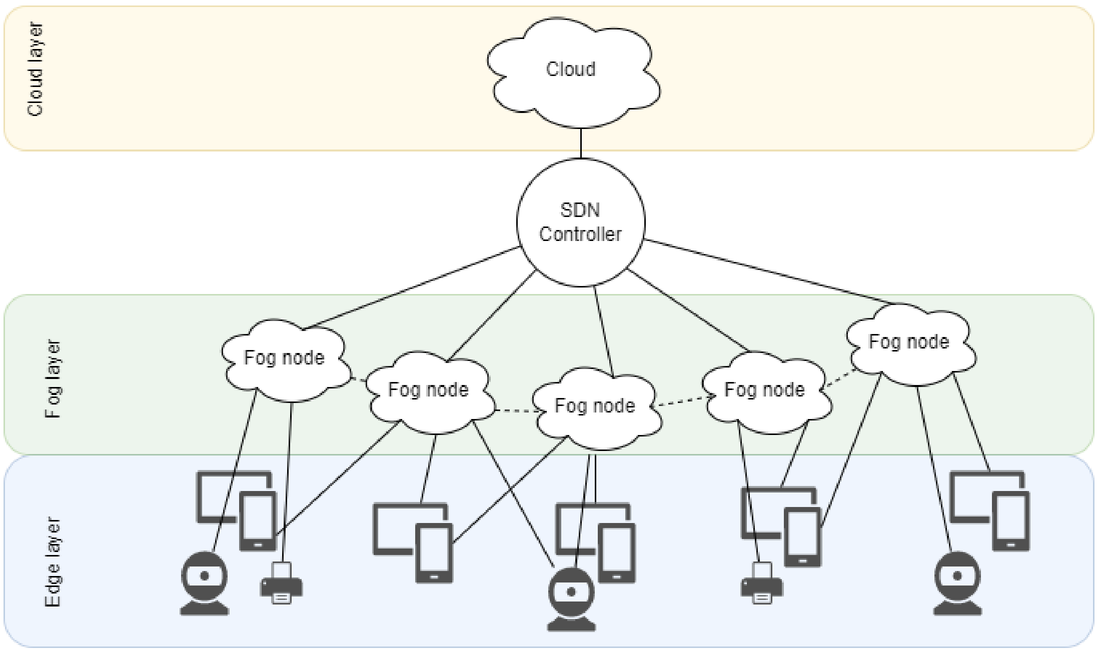

Figure 2.1: Fog architecture with SDN

2.4 Digital forensics

Digital forensics is considered to be a branch within forensic science that focuses on data that is stored electronically [16]. In general, forensic investigators will follow a digital forensics process to ensure that the investigation is *forensically sound*. This means that the investigation follows the different standards, principles, and processes that have been created within the field [17]. Multiple different processes exist, and one of them is the five-phase digital forensic process presented in [17]. The five phases introduced are **1)** identification, **2)** collection, **3)** examination, **4)** analysis and **5)** presentation - explained briefly below:

Identification - In this phase of the investigation, the focus is to identify the incident or criminal activity that has occurred. Additionally, it is important to identify the possible sources that can contain evidence or any information relevant to the case.

Collection - After identifying possible evidence, this following phase is where it is collected. There exist different methods to collect different data types, and the investigators must follow them in a forensically sound manner. An important aspect of collecting evidence is the *order of volatility*, where investigators prioritize the collection of the possible sources of evidence based on their volatility.

Examination - The next step after collecting the data is to examine it. There might be loads of data collected, where only parts of it are considered to be evidence. The examination phase includes pre-processing and structuring

of this data and makes it ready for the next phase.
Analysis - After the pre-processing is done in the previous phase, this phase is when the evidence is analyzed. The analysis has the goal of deciding the facts about the incident, such as the details of the event and the people involved.
Presentation - The final step in the process is to present the results of the investigation. It is important that all of the methods used have been in a forensically sound manner for it to be considered valid evidence. The presentation includes a report of the findings where descriptions of the tools, methods, and findings are presented [17].

The following sections introduce different fields within digital forensics that have been relevant for the research in this thesis. This includes Cloud forensics, IoT forensics, and Fog forensics.

2.4.1 Cloud forensics

The cloud has existed for a much longer time than fog, and as the two often are seen in relation to each other, the topic of cloud forensics is highly relevant for the research. Cloud computing has become a critical part of networking as we know it today. The amount of data generated each second means that it has been necessary to create digital forensic methods that can be applied in cloud networks. There are multiple ways a cloud network can be involved in criminal activity - some examples are storage of malicious files, cloud-connected devices that might be used in botnets and attacks, or it might be the target of an attack [18].

Cloud forensics often consists of multiple types of digital forensics as there are different systems and entities involved. An incident or a crime involving a cloud system might require a combination of network forensics, memory forensics, mobile forensics, and IoT forensics. The methods and procedures will all depend on the scale of the network and the type of cloud service involved. A private cloud hosted with IaaS will likely require different methods than a public cloud hosted with SaaS. The NIST Cloud Computing Forensic Science Working Group has defined a total of nine categories that spans the types of challenges that exist within cloud forensics in [19]:

1. Architecture
2. Data collection
3. Analysis
4. Anti-forensics
5. Incident first responders
6. Role management
7. Legal
8. Standards
9. Training

The categories include multiple challenges that have been found in cloud forensics by reviewing recent research performed within the field.

2.4.2 IoT forensics

IoT forensics is a branch of digital forensics that involves the many devices connected to the Internet. These devices are often connected to a cloud service, and IoT forensics will, therefore, in many cases, include cloud forensics in the process. As with cloud forensics, multiple types of forensics can be required for the investigation, such as network forensics, memory forensics, and mobile forensics.

IoT devices come in many different forms that require different approaches. Therefore, creating one methodology that works for all cases has been challenging. Performing IoT forensics will often include vast amounts of data, and it can be challenging to find relevant evidence [20]. Different types of evidence can be found in IoT devices, and one way to classify it is based on architecture [21]:

- **Sensor and Smart Device Level** - Evidence from logs and memory
- **Computer and Network Level** - Evidence from network device logs, digital device forensic images and memory
- **Internet and Cloud Storage Level** - Evidence from network logs, VM logs, VM snapshots and memory, and cloud storage

2.4.3 Fog forensics

The term "fog forensics" is not broadly used, and fog computing is a relatively new technology that has not been implemented in real systems at a large scale compared to technologies such as cloud computing. Most of the methodologies created for fog forensics are still at a research level and are further presented in Section 2.5. A majority of these forensic methods proposed for fog computing in research are proactive solutions where modules or mechanisms are added to the network prior to an incident, meaning that it, in reality, works similar to an IDS/IPS. This can detect abnormalities in the network traffic, preventing attacks against the network or other malicious activity. Detecting this unwanted traffic and storing data relevant to the incident creates a way for forensic investigators to get a hold of possible evidence that usually would be difficult to gather.

There are different types of digital evidence that can be found in fog systems, such as *encryption keys, network packets, metadata, malware binaries* and *user data* [22]. As fog systems usually involve IoT devices and cloud services, the evidence presented in the previous sections can also be found in fog networks.

2.5 Related work

The previous section provided an introduction to the topics necessary to understand before looking into related work. There is not much research done comparing forensic frameworks for IoT-fog environments, and it was somewhat difficult to find related work. Therefore, this chapter includes different forensic frameworks that are either directly created for fog or focused more on the IoT devices and cloud.

This chapter has been separated into three parts: one represents the related work that exists within fog computing and forensics - such as security challenges and developed frameworks, one section focuses on fog network simulators, and the last section presents related work within fog service placement.

2.5.1 Fog computing and forensics

There are several new challenges with the fog environment, which are essential to address to develop new methods and frameworks for forensics in the IoT-fog environment. Wang et al. [3] looked into the general challenges with fog computing and what sets it apart from cloud computing. The paper encourages more research and has much informational value that was beneficial in the analysis and experiments that were performed in this thesis. The authors identify the chain of dependency and logging as some of the key differences from cloud forensics. The paper describes different scenarios where IoT-fog is applied - like in SDNs, smart grids, and connected vehicles. Each scenario has a different structure and can have its own problem areas.

Mukherjee et al. [23] have looked into the security and privacy challenges that arise with fog computing. The authors mention research challenges within the topic, including how the number of fog nodes will affect the forensic process and that it is a main reason why it is different from cloud forensics. This is important to research as it is relevant to how a forensic investigation process will perform.

Huang et al. analyzed forensic approaches and challenges within vehicular fog computing in [24], and mentioned how it is not practical to physically investigate all of the deployed nodes in large-scale networks. One of the digital forensic approaches mentioned in the article is an *evidence-based digital forensic approach*, which is applicable in most fog networks, not only vehicular fog systems.

Looking more into specific frameworks that have been created for forensics on fog-based IoT devices, Al-Masri et al. propose FoBI [25]. This fog-based IoT forensic framework considers challenges investigators face when performing digital forensics on IoT devices placed in a fog. The framework focuses on how to mitigate cyber-attacks involving fog-based IoT devices, and consists of six modules that can run on a fog node: **1)** device monitoring manager, **2)** forensic analyzer, **3)** evidence recovery, **4)** case reporting, **5)** communication and **6)** storage.

Bandil and Al-Masri present another framework for fog forensics in [26]. Their framework VTA-IH uses a Complex Events Processing (CEP) model with a Degree of Abnormality (DA), which makes it possible to detect abnormality in real-time events.

Duan et al. present their research on "trusted, encrypted, and queryable network archives" for IoT applications that are connected to a fog network in [27]. The approach involves a hardware enclave and a searchable encryption technique.

It is possible to create dynamic private query handling from a cloud with these components. Compared with a baseline construction that has been challenged by the large amounts of traffic in fog networks, this new dynamic approach performed significantly better.

As few specific frameworks are developed for fog computing, a combination of IoT forensics and cloud forensics may also be applicable. In [28], Meffert et al. present Forensic State Acquisition from Internet of Things (FSAIoT). This framework focuses on digital forensics on IoT devices and discusses the challenges with such devices. FSAIoT includes a centralized controller that can acquire state data with precise logging and secure storage. One of the challenges with this framework and IoT forensics, in general, is the lack of possibility to retrieve deleted or historical data [28]. The paper does not discuss the topic of fog, but there could be possibilities for implementing this framework on fog nodes.

In [29], Oriwoh and Sant describe the Forensics Edge Management System (FEMS), which provides autonomous security and forensic services in the context of smart homes. The model was developed in 2013 and is based on anomaly detection and logging with network monitoring tools. Similar to [28], the model is created for IoT devices, but concerning fog computing, it could possibly be located in a fog node.

Kebande and Ray presented a holistic approach for digital forensics investigation in [30] where the framework includes proactive functionality, IoT forensics, and reactive functionality. Their framework DFIF-IoT is an alternative to the typical digital forensic investigation process [17] and complies with ISO/IEC 27043:2015[1], which still remains as the current standard today. The proactive process is used for Digital Forensic Readiness (DFR), IoT forensics makes it possible to retrieve evidence, and the reactive process allows for investigation after an incident. The forensics used in the framework combines cloud forensics, network forensics, and device-level forensics.

Salama et al. proposed the MCF2I Framework in [21]. This framework is an approach that considers IoT devices in the investigation process. Similar to the digital forensic process used in [17], the MCF2I is also based on five phases. However, there are slight differences between the two. Salama et al. includes digital forensic readiness as its own phase in the process, which leads to the following phases: **1)** Manage/Prepare, **2)** Identification, **3)** Acquisition, **4)** Analysis and **5)** Presentation. The main point of the framework is to provide detailed multi-levels for investigators to use in an IoT-focused investigation. The results of the evaluation showed that the majority of the participants(law enforcement personnel) testing the framework believed that it could be beneficial in their investigations [21].

The frameworks presented in this section have been summarized in table 2.1.

[1]International standard for "Information technology - Security techniques - Incident investigation principles and processes" [31]

Framework	Environment	Summary
FoBI [25]	Fog/IoT	Mitigation of attacks against the fog network with monitoring and forensic analysis. Possibilities of evidence recovery.
VTA-IH [26]	Fog/IoT	Detecting abnormalities in real-time events in the fog network.
Secure Network Archives [27]	Fog/IoT	Queryable, secure network archives for IoT data
FSAIoT [28]	IoT	Centralized controller used to capture the state of an IoT device
FEMS [29]	IoT	Network monitoring, IDS/IPS and data storage for smaller networks such as a smart home
DFIF-IoT [30]	IoT	Holistic approach for digital forensics investigation involving IoT devices. Focus on digital forensic readiness, evidence retrieval and post-incident investigation
MCF2I [21]	IoT	Framework for the digital forensic investigation process that considers IoT devices

Table 2.1: Summary of forensic frameworks applied for fog/IoT in research

The topic of fog forensics is slim in research compared to cloud forensics and IoT forensics. As seen in this section, most of the related works within fog forensics are proactive solutions and not methods applied in an investigation after an incident. A combination of other types of forensics, such as IoT forensics and cloud forensics, provides some solutions. However, there are gaps in the current research where investigators may struggle to collect all available relevant artifacts. The related work presents some frameworks and relevant papers, but it still lacks concrete solutions in some areas.

2.5.2 Fog simulation

Simulation of a fog network can be performed using multiple different simulators. Using a simulator to conduct experiments in research can be beneficial as it creates a controlled environment. There exists several simulators that can be used for this purpose, such as iFogSim [32], EdgeCloudSim [33] and FogNetSim++ [34].

A simulator that has been found to be relevant for the research in this thesis is YAFS [11]. YAFS is different from other fog simulators in the way that it provides "dynamic allocation of new application modules, dynamic failures of network nodes, and user mobility along with the topology" [11]. Essential functions it provides are network visualization, post-simulation analysis, and the ability to add customized processes. The python-based project is available on GitHub [35], and the functionality and availability of this simulator were found to be the most relevant of the ones mentioned regarding the research questions.

2.5.3 Service placement in fog computing

Deciding the location of the services in a fog network can be based on different types of service placement algorithms and approaches. As presented in 2.3.2, Salaht et al. described the SPP and a review of the existing algorithms proposed to solve it [9]. Another survey on solutions to the SPP was presented by Raghavendrea et al. in [36]. Mahmud et al. wrote an extensive survey on application management, including the placement problem in [37]. A more recent review on the existing fog service placement algorithms was presented by Smolka and Mann in [38]. In the paper, the authors review 99 different articles, including multiple algorithms mentioned in [9]. The most recent survey is presented by Guerrero et al. in [39], where the authors reviewed 70 different optimization papers within fog computing with a focus on genetic-based solutions. Some of the reviewed papers were specifically focused on the service placement problem.

This thesis will not include all of the solutions reviewed by the different papers but focuses on the different categories and types of approaches that exist. The importance is not to go into detail in each one, but rather to see the types that exist and what issue they are designed to solve. The categorization of the solutions is based on the theory presented in Section 2.3.2 by Salaht et al. [9].

This section introduces some of these approaches presented in research and what the aim of each of them are.

Santos et al. introduced an approach using Integer Linear Programming (ILP) in [40]. The proposed solution is iterative, ensuring that the new results are more optimal than the previous results. The approach was designed for fog systems in a setting of smart cities and aimed to optimize aspects of the SPP such as latency and energy consumption. The results prove that the services with the highest dependency on low latency are processed in the nearest fog node as it is the shortest path. The services that are not as dependent on low latency can be placed further away from the IoT device and focus on energy consumption instead of latency.

In [41], Skarlat et al. present three different approaches to the SPP in fog systems: one with a greedy first fit heuristic, one with an exact optimization method, and lastly using a Genetic Algorithm. The results show that the exact optimization approach gave lower execution costs than the GA, but the GA produced lower deployment delay.

In [42], two different approaches are presented. The approaches use application profiles in the placement process, where they are guided by popularity. One of them is based on a Genetic Algorithm and the other on graph partitions, both of which were tested against a baseline First Fit algorithm. The results proved that the algorithm based on graph partitioning outperformed both the GA and the First Fit algorithm.

An autonomic service placement method was presented in [43] called Neighborhood Exchange Local (NE-L), which is a version of the Neighbor Exchange algorithm. The approach optimized energy consumption and application costs with the distributed solution, and the results showed that it was competitive compared to centralized heuristic approaches.

Yu et. al proposed an approximation algorithm in [44], where the focus is on application provisioning. The problem is formulated as a Mixed Integer Quadratic Program (MIQP). The solution presented is a polynomial-time approximation algorithm that finds the minimum congestion ratio.

Another approximation approach was developed by Shah-Mansouri and Wong [45], using the Nash Equilibrium (NE) and proposing a near-optimal resource allocation mechanism. The method is based on the users maximizing their own Quality of Experience (QoE), and the results of the algorithm showed reduced computation delay.

A heuristic algorithm was proposed by Brogi and Forti in [46], where the approach is based on fail-first and fail-last to find an appropriate solution as fast as possible. These methods find the "best" node for service placement based on spatial proximity and how powerful the node is, considering that a powerful node will be an appropriate solution.

Mann et al. proposed another heuristic method in [47], where they created an algorithm called FOGPART. The performance was compared to an exact solution using ILP and another heuristic method following the first-fit principle. The FOGPART algorithm performed better than the exact solution and was close to the optimum with moderately higher costs.

Multiple proposals exist to solve the service placement problem with meta-heuristic methods such as using a Genetic Algorithm (GA). Yadav et al. presented a solution where a GA was used in combination with Particle Swarm Optimization (PSO) [48]. Brogi et al. proposed another combination, consisting of a GA with a Monte Carlo method in [49]. Mehran et al. wanted to solve the SPP with the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [50].

The service placement algorithms and approaches presented in this section have been summarized in table 2.2.

Category	Authors	Approach Type
Exact solution	Santos et al. [40]	ILP
	Skarlat et al. [41]	ILP
Approximations	Yu et al. [44]	MIQP
	Kayal et al. [43]	NE-L
	Shah-Mansouri and Wong [45]	NE
Heuristics	Brogi and Forti. [46]	Mathematical model
	Mann et al. [47]	Graph partitioning
	Velasquez et al. [51]	Graph partitioning
Meta-heuristics	Skarlat et al. [41]	GA
	Velasquez et al. [51]	GA
	Yadav et al. [48]	GA-PSO
	Brogi et al. [49]	GA
	Mehran et al. [50]	NSGA-II

Table 2.2: Summary of service placement algorithms and approaches

Chapter 3

Methods

This chapter includes the methods that have been used to perform the research and answer the research questions. The first section explains the research approach, and further sections go into details about the specific methods.

3.1 Research approach

The thesis consists of a mixed method, specifically, an embedded approach. A mixed method consists of both qualitative and quantitative methods, and with the embedded approach, we put an emphasis on one of the two [52]. The research in this thesis is mainly based on quantitative research in the form of empirical research consisting of two experiments.

The specific approach in the empirical research is Analysis by Synthesis. In this case, the topic of fog computing was investigated, and a fog network was synthesized using a simulator. The results of the simulations are further analyzed regarding the related research questions. The details of the empirical research are further presented in section 3.3. Some qualitative research was necessary to get a fulfilling answer to all of the research questions. The specific method used in the qualitative research is literature study, which is introduced in the following section.

3.2 Literature study

To investigate state-of-the-art and how current digital forensic frameworks perform in fog-IoT environments today, it was necessary to conduct qualitative research. This part consisted of literature study and was especially important in answering *RQ1 - Which types of service placement algorithms exist for fog computing?*

Literature search and study were performed throughout the research. At the start of the project, it was important to gather enough information to see the whole picture and understand how the experiments would add value to current research. This process was ongoing throughout the whole research period.

As the topic of fog forensics does not resolve to that many research papers, it was necessary to search for other work surrounding the topic. This mainly involved cloud forensics, IoT forensics, and fog technology in general. With these additional topics, it was easier to perform a literature study and better understand the problem areas.

When conducting the literature study, the research started with Google to find short and concrete definitions of terms used with fog computing. This included terms such as *fog node, fog architecture, fog network, IoT fog* and *fog-based IoT-devices*. It was important to include terms in the searches that indicated that the search was for the term fog concerning computing and not the natural phenomenon.

After understanding the basic terms, it proceeded with searches within more specific terms and research within fog forensics. Keywords used in this search included *fog forensics, fog-IoT forensics, fog forensic challenges* and *cloud vs fog forensics*. This started with the use of Google Scholar. This gave a broad overview of how much research has been performed previously on fog computing and forensics. This method provided research articles that seemed relevant to the topic, and further literature study started with reading the abstract of the papers. This was an efficient way to find the relevant research that could be used in this thesis. In many cases, it turned out to be irrelevant for the research in this thesis, and it was, therefore, no point in reading it further. If the abstract appeared relevant, it was reasonable to continue reading the introduction and conclusion of the paper. This ensured that multiple papers were filtered out, as many were not that relevant after further reading. If it was relevant after this process, it was saved to a reference list along with a short description of the main points. This made it easier to have control over the studied papers and their contents. The reference list was further used when needed.

As Google Scholar did not always provide relevant research, Scopus was also used in the literature study. The results included many of the same papers found previously in the study, but it was easier to filter it based on different parameters. With the same keywords as used previously, it was possible to find articles and research that were more relevant. Understanding the state-of-the-art was more manageable when the results were sorted on publication date or amount of citations. This ensured that both new and credible sources of information were found.

One of the challenging parts of the problem area of this thesis is the lack of research. An example of this is a search for "fog forensics" in Scopus resolved to only 56 papers, where the oldest relevant paper is from 2015 [3]. Of these results, approximately 20 of them were relevant to the topic. This allowed more time to be spent on each relevant paper. Deciding what made a paper relevant could range from explanations of specific frameworks within fog forensics to more high-level reviews on the challenges with fog computing and forensics. In general, the snowball method [53] was often used when a highly relevant paper was discovered.

With this method, the reference list in the specific paper was investigated to find additional relevant papers. An example of this was the paper that introduced the simulator that would be used in this thesis [11], where many of the references related to information relevant for the research within fog computing.

Most of the results of the literature study were presented in Chapter 2. However, it was also necessary for the empirical study to understand how to implement the experiments and understand the results.

3.3 Empirical study

The purpose of performing experiments was to simulate a fog network in different scenarios and review the post-simulation data. The research method is Analysis by Synthesis, where a fog network has been created to simulate or synthesize a real fog network.

The simulator provides information about traffic flow in the network, which applications are generating messages, and which nodes are handling the messages. With some adjustments to the original simulation source code, it was possible to customize it to the scenarios.

Two experiments were performed:

1. Fog network simulation using YAFS
2. Fog network simulation using YAFS where the model includes volatility

Both of the experiments were performed using Yet Another Fog Simulator (YAFS) [11] which is a Discrete Event Simulator (DES). This made it possible to simulate the network traffic in a fog environment. As it is not an emulator, it was not possible to evaluate the performance of different digital forensic frameworks regarding retrieving data, which was one of the reasons for having a second experiment. YAFS allows for post-simulation analysis, which was relevant in understanding how forensic frameworks can perform. The following section introduces the scenarios created for the simulation and the differences between the two experiments that were performed. The details of the experimental setup and implementation are provided in Chapter 4.

3.3.1 Scenarios

To simulate a fog network that could be similar to a real fog network and be used for further analysis, it was necessary to define a scenario in the context of criminal activity. Creating only one scenario would not be representative enough for the topic of fog forensics, and it was decided to create three different scenarios - each for a different way an IoT device in a fog network could be involved with criminal activity.

When addressing a digital device regarding cybercrime, it can be involved in three different ways: a tool used to perform the criminal activity, a target of the criminal activity, or a witness to the criminal activity. With IoT devices, the category where the device has been a *witness* of criminal activity, either digital or physical [54], is highly relevant. Devices such as sensors and cameras can pick up activity related to criminal activity and, in that way, become a witness. These three categories have been the outline for creating the scenarios, where each represents one of the three ways an IoT device can be involved in criminal activity [21]. The scenarios are of a different scale to create more variety in the experiments.

Scenario 1 - Witness

In the first scenario, a surveillance camera is placed in the fog network and captures and stores footage of someone committing criminal activity. In this scenario, the network is small - consisting of 10 nodes where all of them are known to the owner of the network and placed within the range of a small building. This scenario aims to create a clear understanding of the network structure and the possible devices that are placed in it.

Investigators have identified that there may exist some evidence in this fog network as there were surveillance cameras placed at the crime scene. This was discovered two days after the incident, meaning that there have been changes in the fog network structure and that the investigators do not initially know where the evidence is located.

Scenario 2 - Tool

The second scenario includes an IoT device in the fog network that has been infected with malware and was therefore part of a cyberattack from a larger botnet. This is considered a medium-scaled network, where we know what some of the nodes are but not all of them. The placement of the nodes are spread across a small town.

Investigators are aware of the cyberattack but have yet to discover how the attacker got access to the network and further details of the scope of the attack.

Scenario 3 - Target

In the third scenario, an IoT-device is placed in a fog that logs and stores sensitive data, and a criminal has now successfully stolen some of this data. In this case, we do not know what all the nodes do, which simulates how a large-scale fog network can be. This network has nodes spread over national borders.

The data was discovered stolen after it was leaked online. The investigators are to discover which nodes generated the data, where it was retrieved from, and the scope of the intrusion.

3.3.2 Fog-network simulation using YAFS

In the first experiment, Yet Another Fog Simulator (YAFS) [11] was used to understand the behavior of a fog network where there is a dynamic service movement within the fog nodes. Using the scenarios created, it was possible to see the behavior in three different types of fog networks. We ensure that the results are comparable and more reliable with three different scenarios.

To conduct this experiment, it was necessary to understand how the simulator is built and what the different attributes represent. This was a time-consuming process that included testing parts of the simulator and reading the documentation.

After getting a more in-depth understanding of the simulator and fog computing, it was possible to start testing with data that was more fitting to the scenarios. A natural starting point was the first scenario where the network was small-scaled, and it was easy to get a complete overview of the results. The testing ensured that the simulator operated the way it was planned to do in the experiments. The following simulation could then include the data created for the scenarios. After running the simulator for the specific scenario, the results were collected and analyzed, and the findings were documented. This included information on how the packets in the network had been routed, how the services had been moved between nodes, and statistics on the involvement of each node. This process was repeated for each scenario.

3.3.3 Fog-network simulation using YAFS where the model includes volatility

As an addition to the original YAFS simulator, including volatility in the model makes it possible to simulate how and where data is stored and how quickly it disappears. The extension used in the experiments was developed by Jens-Petter Sandvik and is available as a forked repository from the original YAFS on GitHub [55].

The second experiment is therefore conducted with volatility included in the fog simulation. With this model, it was easier to understand how the possible evidence may be stored in a fog environment and what it demands from a forensic investigator when attempting to collect them.

As volatility is the main focus, it is necessary to understand how the volatility is decided for the nodes in the simulations. The volatility is based on the network's three different node types: source, sink, and proxy. Source nodes are typically IoT devices with low capacity that often contain high volatility data. Sink nodes usually are more powerful than the source nodes, which means higher capacity and lower volatility. Proxy nodes generally have a high capacity but not necessarily low volatility.

After conducting the first experiment, there was no need to understand the basic principles of the YAFS simulator. However, as this second experiment included additional functionality where volatility is considered, it was necessary to understand this new functionality. The functionality was developed by the supervisor of this thesis, Jens-Petter Sandvik, and some time was therefore spent on getting an introduction and explanation of the model in our meetings before it was implemented in the experiment.

The experiment was conducted in the same manner as the previous experiment, but the focus of the results and analysis was on the volatility and not the service movement.

3.3.4 Digital forensic analysis

After conducting the experiments, a digital forensics analysis was performed. This analysis is based on the results from the experiments and related research discovered by performing the literature study. The purpose of the analysis was to put the results from the network simulations into the context of digital forensics and see how a digital forensic process is affected by the dynamic nature of fog nodes.

3.3.5 Possibilities and restrictions

By choosing to perform the experiments with a simulator, some important points should be mentioned. While a simulated environment is an effective way to create, understand and modify a fog network, some restrictions also apply. A fog network is complex and involves multiple outside factors that can be difficult to simulate. The simulator provides possibilities for adjustments and customization, but not to the extent of a real fog network. It is, however, a good solution to conduct an experiment in this manner as it is in a completely controlled environment where all the activity in the simulation is logged.

It also must be considered that performing an experiment of this type might not be feasible in a live, existing fog network that processes real data. The time and effort it would take to perform such an experiment in an uncontrolled and real environment is an essential factor as to why this research was performed using a simulator. This also made it possible to use synthetic data, where no results needed to be excluded from the results for privacy concerns.

This simulator made it possible to create a network from scratch with attributes as desired. There was complete control over the networks during the experiments. Simulating the nodes and creating a synthetic dataset involves that some of the attributes are fixed and directly affect the results. In a real network, events might be affected by outside forces, which are not that easy to simulate. However, with some fixed values, it was possible to simulate networks suitable for each of the scenarios and therefore have them be highly relevant for the experiment.

It is challenging to simulate reality, but the simulated networks encompass traits that are known to occur in fog networks - such as service movement, user movement, and topology changes.

Chapter 4

Experimental setup, implementation and results

This chapter explains the experimental setup and describes the process of running the simulator. The technical details of the scenarios are included, and the results of the simulations are presented. The chapter is separated into three main sections, where the first section introduces YAFS where the terminology and details of the setup are explained. The second and third section is one for each of the two experiments, where the data used to create the networks is presented, followed by a summary of the results of each scenario.

This part of the thesis aims to mainly answer **RQ2:** *How does the dynamic service placement in fog-IoT systems affect the evidence location and the digital forensic process?* and the following hypotheses were created for the experiments.

Null hypothesis (H_0): The evidence location in simulated fog-IoT systems is not affected by the dynamic service placement
Alternative hypothesis (H_1): The evidence location in simulated fog-IoT systems is affected by the dynamic service placement

The hypotheses are tested against a theoretical baseline cloud network which contains one centralized node for cloud storage. This means that there is one node available for service placement. The hypotheses can therefore be defined as:

(H_0): $x = 1$
(H_1): $x > 1$

Where x = number of nodes used for service movement

RQ3 is also partially based on the results of the experiments, but is difficult to answer with exclusively the results from the simulations. It must therefore be further put in the context of digital forensics, provided in Chapter 5.

4.1 YAFS simulator

Yet Another Fog Simulator (YAFS) is used in the experiments to create the fog network. This section introduces the functionalities and elements of the simulator and the terminology used.

YAFS is a simulator where **all elements** in the network are represented as *nodes*. This can be network devices, cloud abstractions, software modules, workloads, etc. [35]. The simulator makes it possible to see which nodes are sending and receiving messages in the post-simulation files that are created. The purpose of using YAFS is to see how the dynamic service placement and data storage can be in a fog network, and how it differentiates from a standard centralized cloud network. The architecture of YAFS is presented in figure 4.1 which is retrieved from Lera et al. [11]. The simulator consists of six main classes that perform the core functionality:

Application consists of modules that run the services and creates messages
Placement defines the allocation of application modules
Selection defines the orchestration of services
Population encompasses the workload generators
Topology defines the network topology based on graph structure
Core is the main class and manages the overall execution of the simulator

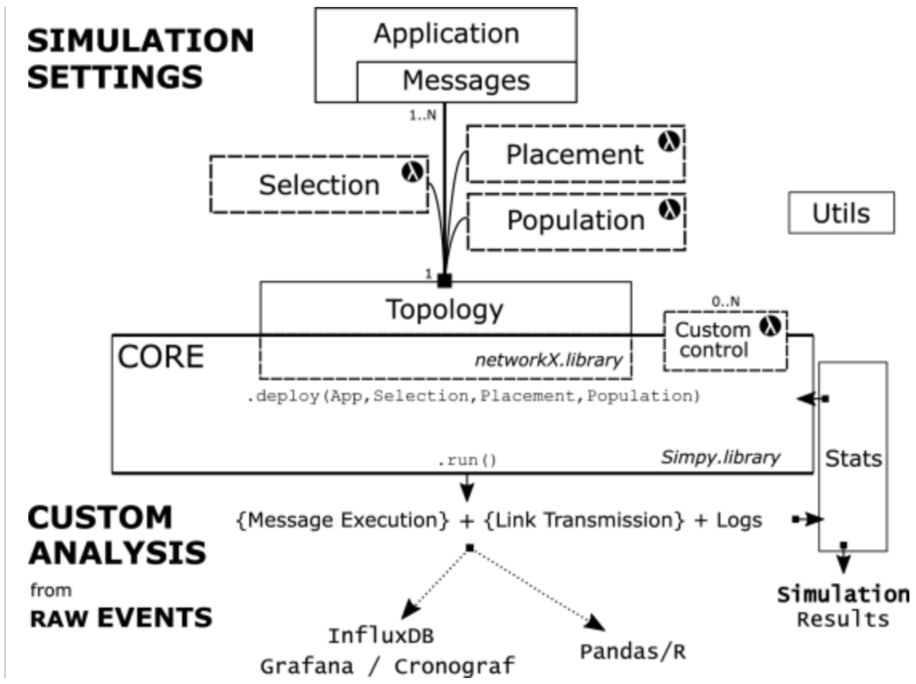

Figure 4.1: Architecture of YAFS

4.1.1 Terminology

To understand the results that are presented in sections 4.2 and 4.3, it is necessary to understand the terms that are used. The following definitions of terms is from the glossary provided in the YAFS documentation [56].

Module is a part of the application that can be deployed in a device
Entity is the abstraction of any type of device
Link is a network connection between two entities
Pure Source is a special type of node who only generates messages (i.e. sensor devices)
Pure Sink is a special type of node who only receives messages (i.e. actuator devices, servers)
Message is the representation of a network package that enables the action of a application module

The nodes in the simulation are created with the following characteristics [56]:

IPT Instructions per simulation time
RAM Memory available
COST Cost per unit time

In regards to performance, a link between two entities have the following characteristics [56]:

BW Channel Bandwidth: in Bytes
PR Channel Propagation speed
Latency is dynamically computed using: (Message.size.bits / BW) + PR)

In total, there are four post-simulation files that have been used to read and understand the results of the simulations:

sim_trace.csv
sim_trace_link.csv
logFile.log
sim_trace_volatility.csv (Only in experiment 2)

Each of these files contains attributes whose values are presented in the results. Therefore, it is important to understand what the files are and what the attributes represent. The following sections provides a short description of all the attributes in the different files. Some of them are used in multiple files, and are therefore mentioned several times. This is done to not only understand the attributes, but also to see the context in each file.

sim_trace.csv

This file includes the task execution events that occurred in the simulation [11]. The attributes recorded for each event are defined in the YAFS documentation [56] and are the following:

type represents the entity who run the taks: a module (COMP_M) or an actuator (SINK_M)
app is the application name
module is the module or service who manages it
service is the service time
message is the message name
DES.src is the DES process who send this message
DES.dst is the DES process who receive this message (the previous module)
TOPO.src is the ID of the node where the DES.src module is deployed
TOPO.dst is the ID of the node where the DES.dst module is deployed
module.src is the module or service who send this message
service represents the service time
time_in represents the time when the module accepts the message
time_out is the time when the module finishes its process
time_emit is the time when the message was sent
time_reception is the time when the message is accepted by the module

sim_trace_link.csv

This file contains the network transmissions in the simulation and records the following attributes for each event [56]:

type is the link type
src is the source of the message - ID node topology
dst is the destination of the message - ID node topology
app is the application name
latency is the the time taken to transmit the message between both nodes
message is the name of the message
ctime is the simulation time
size is the size of the message
buffer represents the number of waiting messages in all the links

logFile.log

The logfile provides information about which custom processes occur in the simulation, e.g. when a new node is added to the network or a service has been moved to a new node. Example of output in the logfile:

```
2022-03-22 12:42:40,657 - root - INFO - Moving Service 4_01 from 9 to 3 node
```

This example shows that during the simulation the module/service named 4_01 was moved from the node with ID=9, to the node with ID=3, simulating service movement. The logfile was an important resource in understanding the behaviour of the processes in the simulation.

sim_trace_volatility.csv

This file contains information about volatility for the events and records the following attributes:

id is the message id
type is set to *volatility*
src is the source node for the event
dst is the destination node for the event
node is which node the event occurs in
app is which app handles the event
message is which message is generated or sent/received
mem is the memory on the node
time_created is when the message is created (in proxy and sink this is when the message is received as it can not generate messages)
time_unlink is when the message is deleted/unlinked from the application. It is deleted or rewritten to memory
time_erase is when the message is deleted by the operating system, and when the app is no longer in control of the message. The operating system is now handling it and deals with deletion at this point, e.g. garbage collection
delta_unlink is time between time_created and time_unlink, meaning how long it exists in the application
delta_erase is the time between time_unlink and time_erase, meaning how long the message exists in the node after the application has unlinked it
vtype is the type of node that handles the event, either SRC (source), PRX (proxy) or SNK (sink)

4.1.2 Experimental setup

Requirements and technical setup

The experiments was performed in a virtual environment using Python 3.9.5 with the *venv* module [57]. The simulator has the following requirements for Python libraries and frameworks, retrieved from the *requirements.txt*-file in the YAFS GitHub repository [35]:

```
decorator==4.4.2
networkx==2.5
```

```
numpy==1.19.3
simpy==4.0.1
pandas==0.25.3
```

The code-editor used for adjusting the source code of the simulator to the scenarios was Atom [58], and the simulation was executed from the terminal on MacOS. The editor was connected to a personal forked GitHub repository to keep track of changes.

The network topology graphs created in the simulations were visualized using the open-source software Gephi [59].

Simulation details

The simulator setup included other technical details, such as deciding the graph structure for the topology and the node types used in the network.

There are mainly three different types of nodes that exists in the simulator; *source, sink* and *proxy*. The source nodes mainly generates the messages, the sink nodes are mostly actuators and the proxy nodes can provide a gateway between the nodes. Some of the nodes with additional storage capacity also acts as servers.

To simulate a fog network, a topology structure had to be defined. This structure was created using the Python package NetworkX [60]. The graph structure used for the network was initially set to be a binomial tree by default in the simulator. A binomial tree consists of a structure where if a tree is of order 0 it has 1 node and a tree of order k consists of two binomial trees of order k-1. One of the trees is then the leftmost child of the other [61]. As an example, a binomial tree of order 2 consists of 4 nodes, a tree of order 3 has 8 nodes and so on.

However, to create a network structure that resembles a fog network, it was decided to use the Barabási–Albert graph. This structure is based on *preferential attachment*, where the nodes with a high degree of network links get even more links, and the nodes with fewer links does not get that many new links. This is often found in real networks, where popular nodes become more popular as the network expands. Another step that is included in the Barabási–Albert model is *growth*, where the network will expand over time and new nodes connects to existing nodes in the network - which is relevant for fog computing [62]. The probability p_i of the attachment to node i depends on the degree k_i and is denoted as:

$$p_i = \frac{k_i}{\sum_j k_j} \tag{4.1}$$

With these two steps, it means that when new nodes are added to the network they can connect to any node, but is more likely to connect to nodes with a higher degree of links.

The simulator also required a definition of the service placement. For these experiments, the services was placed on the nodes by definition in JSON-format.

This allocation definition placed the services at a specific node, which could further be moved in the simulation run-time. Moving the services in the simulation was added as a custom process that was invoked multiple times. Several of the services was therefore moved around between nodes. This was based on a random selection of services and nodes, not on a specific service placement algorithm. The results show the services that have been moved and to which destination node. As it is random, it is not a result of a specific algorithm typically seen in a real network. The analysis of the results therefore considers that the movement is random, and addresses how it could be the result of specific service movement algorithms.

4.2 Fog network simulation using YAFS

This section includes the technical details and results of the first experiment. It is separated into three main parts - one for each scenario. These sections are introduced with adjustments made to the simulator source-code prior to running it and the technical details for the specific scenario. Finally, the results of the simulations are presented based on the post-analysis files that are generated.

4.2.1 Scenario 1 - Witness

In the first scenario the simulator was ran with a class for service movement, where there are only a few nodes and applications running. With this simple structure, it was easier to identify all the nodes and get a full overview of the network's purpose.

Technical details

Table 4.1 provides an overview of the elements of the simulation, including which applications the modules belonged to, what user it was connected to and the name of the message it could generate. As the scenario includes a security camera, some of the messages in transmission during the simulation represents video files. Other messages represent data such as logs with temperature changes, motion sensor activity or device instructions. The network was generated by the Topology-class based on the Barabasi-Albert graph and is presented in Figure 4.2.

App	Message	Module	Allocation	User
0	M.USER.APP.0	0_01	Module 0_01: App 0	1
1	M.USER.APP.1	1_01	Module 1_01: App 1	2
2	M.USER.APP.2	2_01	Module 2_01: App 2	3
3	M.USER.APP.3	3_01	Module 3_01: App 3	4
4	M.USER.APP.4	4_01	Module 4_01: App 4	5
5	M.USER.APP.5	5_01	Module 5_01: App 5	6
6	M.USER.APP.6	6_01	Module 6_01: App 6	7

Table 4.1: Scenario 1 - Technical details

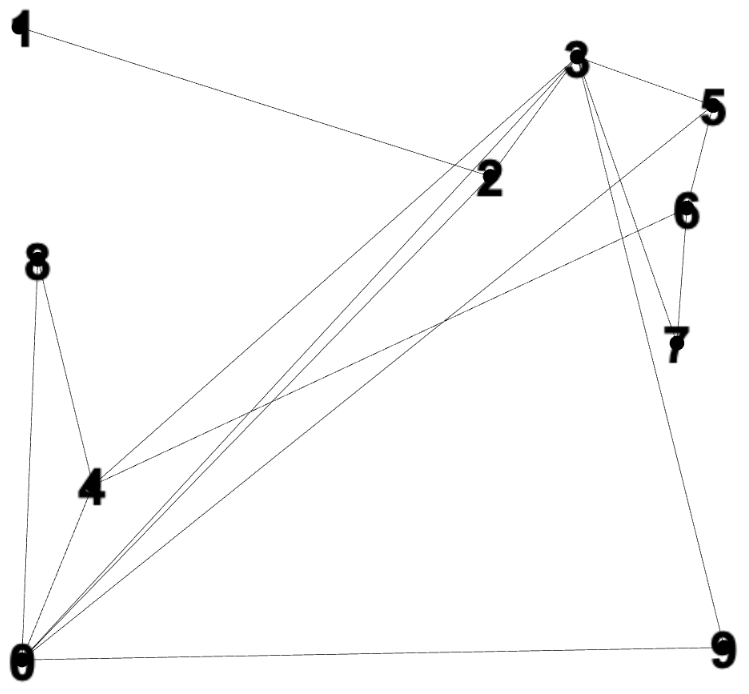

Figure 4.2: Scenario 1- Initial network topology

The following sections include the simulation results of scenario 1, separated into sections for each of the different post-simulation files.

Results from logfile

From the results, it is observed that within the given time period, the simulation performed service movements that changed the placement of modules on the different nodes a total of 15 times. This means that the data the services are processing have been moved as well. All of the services start at node with id 0, and are then spread to different nodes after the first service movement.

As an example, Module 0_01 started in node with id 0, and following this event the module has been placed in the following nodes:

8, 5, 8, 9, 3, 2, 1, 8, 1, 3, 1, 8, 4, 1, 5

This shows that the service/module has been placed in almost all nodes that exist in the network with an exception of node 6 and 7. The service was placed several times on node 8 and 1 - in this case it was random, but in reality this could mean that these nodes have a lower latency or more space for the service to run efficiently.

Another example is Module 4_01: The module started in node with id 0, and following this event, the module has been placed in the following nodes:

1, 7, 8, 7, 3, 8, 1, 5, 7, 2, 1, 6, 9, 3

As observed with module 0_01, module 4_01 is almost placed on every node that exists in the network at some point during the simulation time.

The logs show that that multiple services can be placed on the same node, such as:

```
2022-03-22 12:42:39,880 - root - INFO - Moving Service 4_01 from 0 to 1 node
2022-03-22 12:42:39,881 - root - INFO - Moving Service 5_01 from 0 to 1 node
2022-03-22 12:42:40,123 - root - INFO - Moving Service 3_01 from 5 to 7 node
2022-03-22 12:42:40,123 - root - INFO - Moving Service 6_01 from 6 to 7 node
```

This means that both service 4_01 and 5_01 were placed on node 1 and that service 3_01 and 6_01 were placed on node 7 at the same time during the simulation. This happens several times during the simulation and some nodes are also running three different services at the same time in a short period, such as node 3:

```
2022-03-22 12:42:40,656 - root - INFO - Moving Service 3_01 from 1 to 3 node
2022-03-22 12:42:40,656 - root - INFO - Moving Service 5_01 from 5 to 3 node
2022-03-22 12:42:40,657 - root - INFO - Moving Service 4_01 from 9 to 3 node
```

The results of the service movement has been summarized in Figure 4.3, which represents how many times the nodes hosts a specific module/service during the simulation. The graph shows that node 1 was the most used, and node 6 was the least used node. The results show that all nodes were used to host services.

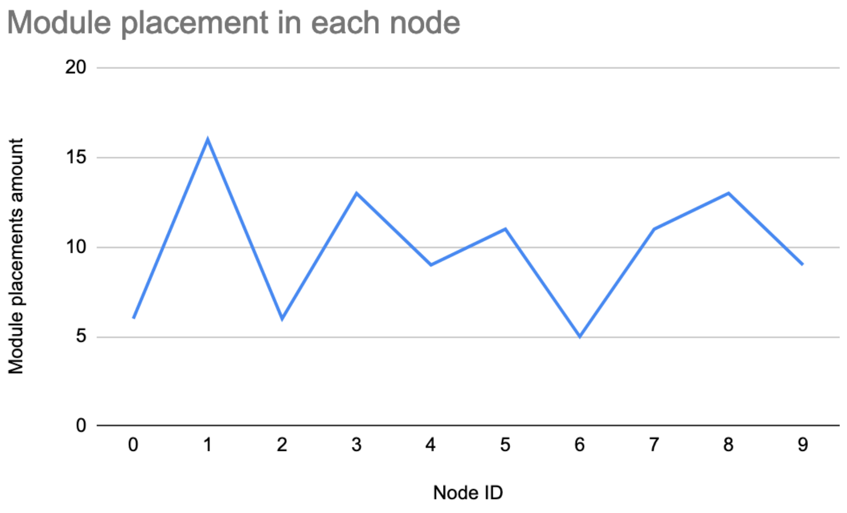

Figure 4.3: Module placement in each node

To summarize what is observed in the logfile: multiple services are running on the same node and that all services are being moved approximately 15 times throughout the simulation. The amount of times the services are moved is defined prior to running the simulator and is not random. All of the services have been placed at almost all of the different nodes in the network, which is most likely a cause of the network being of such a small scale. The complete data on service placements is added as Appendix A.1.

Results from sim_trace-file

The sim_trace-file show all of the task execution events from the simulation. The results show that the TOPO.src-field and DES-src-field are the same for all events for the specific event that is listed. E.g. App 0 with Module 0_01 and message M.USER.APP.0 has a TOPO.src value of 1 and a DES.src value of 0. The TOPO.dst-field value is however somewhat different for each of the events. Some of them are repetitive, where it is possible to see some of the same nodes as the events' destination.

The DES.dst-field is more interesting as it has a lot of repetitive events, but by removing duplicate events (ignoring all time-fields), there are different values for each event. The number of events for each module ranges from 14 to 16.

The TOPO.dst values are repeated between 5 and 16 times for each node, meaning that some nodes are more frequently involved in events than others. In this scenario node 1 is the only node involved 16 times, making it the most used node. On the contrary, node 6 is the least involved, only 5 times. This data is from the "optimized" spreadsheet where the time-fields are omitted. The reason for removing the time fields is to get a better overview of the data, where the involvement of each node become more visible. In the original data, node 0 is the most used node with 403 events, where node 6 is still the least used with 50 events.

Results from sim_trace_link-file

The results from the sim_trace_link-file was analyzed by removing duplicate events based on the *id* and *ctime*-fields, which left 94 rows of events. The node that is most used as source is node 0 which is used 27 times. This makes sense as all services start at this node before being spread to other nodes. The least used node as source is node 1 which is only used 1 time.

The most used destination is node 0 with 17 events, and the least used destination node is node 6 with 4 events. These values reflects what was seen in the sim_trace-file, and the results show that the same nodes are more used than others.

The latency of sending and receiving the messages between nodes is directly connected to the size of the message. The larger messages have a higher latency, and the smaller ones have a lower latency. The messages with a size of 10 bytes have a latency of 1.00001 and the messages with a size of 200 has a latency of 1.0002.

The buffer represents how many messages are waiting in all of the links. The mode is 3 and 2, which occurs 19 times each in the simulation. The least common value is 6 which occurs 4 times in the simulation. This means that it is more likely that 2 or 3 messages are pending in the traffic queue.

4.2.2 Scenario 2 - Tool

In this scenario, the simulation runs with a class for service movement as used in scenario 1, and an additional class for topology changes. This means that the module/service of the apps changes the allocation on the nodes randomly and that there are new nodes added to the network as well. In this case, the number of iterations is set to 1, as the script changes randomly and can simulate a dynamic allocation change without needing to run multiple iterations.

Technical details

Table 4.2 provides an overview of the elements of the simulation, including which applications the modules belonged to, what user it was connected to and the name of the message it could generate. The network in this scenario is larger than previously seen in scenario 1, and it is therefore more data and traffic in transmission. The messages represent all sorts of IoT-traffic such as logs with temperature changes, motion sensor activity, device instructions and connection data. The network was generated by the Topology-class based on the Barabasi-Albert graph and is presented in Figure 4.4.

App	Message	Module	Allocation	User
0	M.USER.APP.0	0_01	Module 0_01: App 0	1
1	M.USER.APP.1	1_01	Module 1_01: App 1	2
2	M.USER.APP.2	2_01	Module 2_01: App 2	3
3	M.USER.APP.3	3_01	Module 3_01: App 3	4
4	M.USER.APP.4	4_01	Module 4_01: App 4	5
5	M.USER.APP.5	5_01	Module 5_01: App 5	6
6	M.USER.APP.6	6_01	Module 6_01: App 6	7
7	M.USER.APP.7	7_01	Module 7_01: App 7	8
8	M.USER.APP.8	8_01	Module 8_01: App 8	9
9	M.USER.APP.9	9_01	Module 9_01: App 9	10
10	M.USER.APP.10	10_01	Module 10_01: App 10	11
11	M.USER.APP.11	11_01	Module 11_01: App 11	12
12	M.USER.APP.12	12_01	Module 12_01: App 12	13
13	M.USER.APP.13	13_01	Module 13_01: App 13	14

Table 4.2: Scenario 2 - Technical details

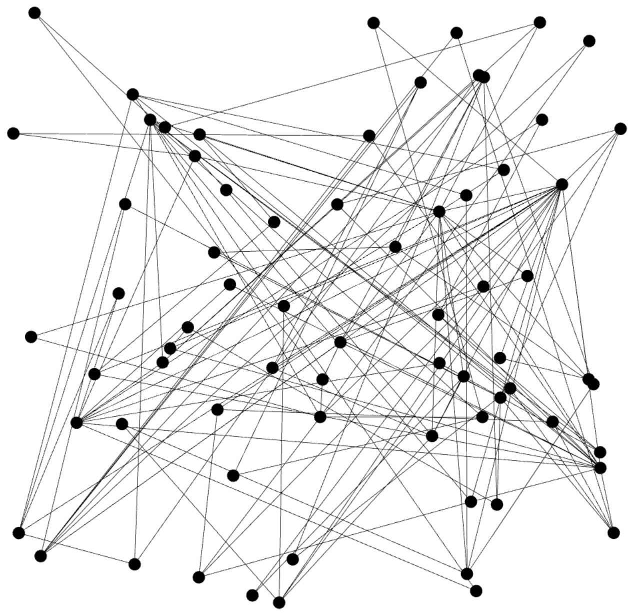

Figure 4.4: Scenario 2 - Initial network topology

The following sections include the simulation results of scenario 2, separated into sections for each of the different post-simulation files.

Results from logfile

The results show that within the given time period, the simulation performed service movements that changed the placement of modules on the different nodes a total of 15 times. This means that the data the services are processing have been moved as well. All the services start at node with id 0, and are then spread to different nodes after the first service movement.

As an example, Module 12_01 started in node with id 0, and following this event the module has been placed in the following nodes:

34, 36, 22, 10, 58, 0, 25, 5, 22, 35, 53, 21, 1, 34

In this scenario there are more nodes in the network, which means that there are more possible placements for the services. Module 12_01 was placed in some nodes multiple times (22 and 34), but not as many times in the same node as seen with Module 0_01 in the first scenario. With more nodes, the service placement for each node is more varied, but this will depend on the specific algorithm implemented for the service placement.

Another example is Module 2_01. Running the simulator, it is observed that the module being placed in the following nodes:

24, 25, 15, 72, 14, 21, 1, 39, 41, 19, 5, 22, 56, 68, 59

This module has not been placed in the same node twice and all of the placements are therefore unique for the module. Compared to the first scenario, there are not many instances where a node contains multiple modules simultaneously. There are a few example where this occurs, where the data below presents some of them:

```
2022-03-21 14:17:16,511 - root - INFO - Moving Service 1_01 from 0 to 2 node
2022-03-21 14:17:16,513 - root - INFO - Moving Service 8_01 from 0 to 2 node
2022-03-21 14:17:16,683 - root - INFO - Moving Service 11_01 from 5 to 20 node
2022-03-21 14:17:16,684 - root - INFO - Moving Service 0_01 from 7 to 20 node
2022-03-21 14:17:16,902 - root - INFO - Moving Service 5_01 from 5 to 54 node
2022-03-21 14:17:16,909 - root - INFO - Moving Service 4_01 from 70 to 54 node
2022-03-21 14:17:17,002 - root - INFO - Moving Service 12_01 from 10 to 58 node
2022-03-21 14:17:17,003 - root - INFO - Moving Service 10_01 from 31 to 58 node
```

This data shows that node 2, 20, 54 and 58 are the only ones that hosted two services at once. There are no instances where a node contains more than two services simultaneously. The results of the service movement has been summarized in Figure 4.5, which represents how many times the nodes hosts a specific module/service during the simulation. The graph shows that node 5 was the most used node for service placement, and that not all nodes were used to host services.

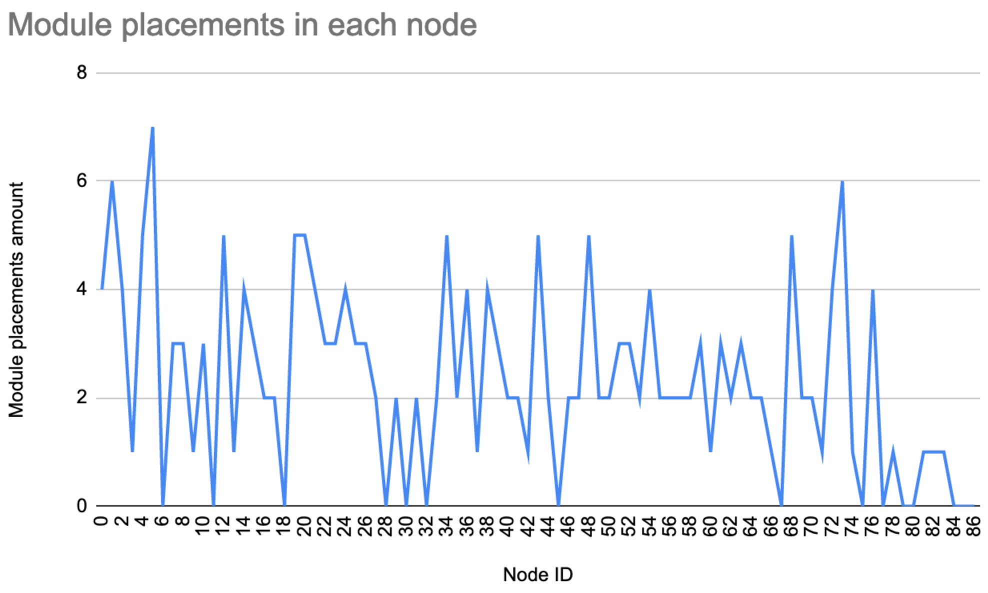

Figure 4.5: Module placement in each node

To summarize what was observed in the logfile: some nodes contained two services at the same time, but in most cases, there was only one service on each node. All services are being moved approximately 15 times throughout the simulation, defined prior to running the simulator. As this is a medium-scaled network, more nodes are available for the service placement, and not all were used. This contrasts to scenario 1, where all of the nodes were used multiple times throughout the simulation. The complete data on service placements is added as Appendix A.2.

Results from sim_trace file

In this file, the first 686 events all has a TOPO.dst value of 0. All of the values for TOPO.src is between 1 and 14, meaning that all event are coming from these 14 different nodes. After the first 686 events, the TOPO.dst is more varied, and the other nodes in the network are also used.

Looking at the statistics for TOPO.dst, it is possible to see that amount of times nodes are used ranges from 10 to 70. Using an optimized spreadsheet where duplicate entries are removed and time-fields are ignored, the TOPO.dst values change into a frequency range from 1 to 18. The most used node is node 0 with 18 times, followed by node 5 with 7 times. At the bottom 14 different nodes are only used once:

Least used nodes: 13, 3, 37, 42, 44, 60, 66, 71, 74, 78, 81, 82, 83, 9

The DES.dst-field has many repetitive events, and by removing duplicate events there are 214 different events based on this variable. The number of events for each module ranges from 10 to 16, where module 7_01 stands out with only 10 events, while the rest of the modules has 15 or 16 events.

Results from sim_trace_link file

After removing duplicates in this file (ignoring id and ctime) there are 376 rows of data left. The node that is **most used as source is node 3** which is used 73 times, followed up by node 0 that is used 47 times. The least used nodes as source are 7 different nodes which are only used 1 time.

The **most used destination is node 0** with 27 times, followed up by node 3 with 25 times. The least used destination nodes are 10 different nodes with 1 event. These values reflects what was seen in the event trace-file. The values in between does not perfectly line up between the two files, but there seems to be a similar trend where the same nodes are more used than others.

The latency between sending and receiving the messages between nodes is directly connected to the size of the message. The larger messages have a higher latency, and the smaller messages have a lower latency. The messages with a size of 10 bytes have a latency of 1.00001 and the messages with a size of 700 has a latency of 1.0007. The latency is calculated by subtracting *time-emit* from *time-reception*.

In some of the links the latency has increased. For messages with the size of 120 bytes, the latency has gone from 1.00012 to 10.000012. This happens only a few times for each of the different packet sizes, but not for packets with the size of 600 bytes - this latency stays the same throughout the simulation.

The buffer represents how many messages are waiting in all of the links. The mode is 11, which occurs 57 times in the simulation, and the least common value is 0 which occurs 3 times in the simulation. This means that there is a higher chance that 11 messages are awaiting to be handled, and that it is not common to not have any messages in queue.

4.2.3 Scenario 3 - Target

In this scenario, the simulation runs with a class for service movement, a class for topology changes, and an additional class for user movement. This means that the modules/services of the apps changes the allocation on the nodes randomly and that there are new nodes added to the network as well. In addition, there are new users added and existing users can be moved between nodes throughout the simulation. In this case, the number of iterations is set to 1, as the script changes randomly and can simulate a dynamic allocation change without needing to run multiple iterations.

Technical details

Table 4.3 provides an overview of the elements of the simulation, including which applications the modules belonged to, what user it was connected to and the name of the message it could generate. The network in this scenario is the largest of the three scenarios, and it is therefore huge amounts of data and traffic in transmission. The network in the scenario spans over a large geographical area, and it is not possible to have a complete overview of the different nodes and traffic. The messages represent all sorts of IoT, fog and cloud traffic. The network was generated by the Topology-class based on the Barabási-Albert graph and is presented in Figure 4.6.

App	Message	Module	Allocation	User
0	M.USER.APP.0	0_01	Module 0_01: App 0	1
1	M.USER.APP.1	1_01	Module 1_01: App 1	2
2	M.USER.APP.2	2_01	Module 2_01: App 2	3
3	M.USER.APP.3	3_01	Module 3_01: App 3	4
4	M.USER.APP.4	4_01	Module 4_01: App 4	5
5	M.USER.APP.5	5_01	Module 5_01: App 5	6
6	M.USER.APP.6	6_01	Module 6_01: App 6	7
7	M.USER.APP.7	7_01	Module 7_01: App 7	8
8	M.USER.APP.8	8_01	Module 8_01: App 8	9
9	M.USER.APP.9	9_01	Module 9_01: App 9	10
10	M.USER.APP.10	10_01	Module 10_01: App 10	11
11	M.USER.APP.11	11_01	Module 11_01: App 11	12
12	M.USER.APP.12	12_01	Module 12_01: App 12	13
13	M.USER.APP.13	13_01	Module 13_01: App 13	14
14	M.USER.APP.14	14_01	Module 14_01: App 14	15
15	M.USER.APP.15	15_01	Module 15_01: App 15	16
16	M.USER.APP.16	16_01	Module 16_01: App 16	17
17	M.USER.APP.17	17_01	Module 17_01: App 17	18
18	M.USER.APP.18	18_01	Module 18_01: App 18	19
19	M.USER.APP.19	19_01	Module 19_01: App 19	20
20	M.USER.APP.20	20_01	Module 20_01: App 20	21
21	M.USER.APP.21	21_01	Module 21_01: App 21	22
22	M.USER.APP.22	22_01	Module 22_01: App 22	23
23	M.USER.APP.23	23_01	Module 23_01: App 23	24
24	M.USER.APP.24	24_01	Module 24_01: App 24	25
25	M.USER.APP.25	25_01	Module 25_01: App 25	26
26	M.USER.APP.26	26_01	Module 26_01: App 26	27
27	M.USER.APP.27	27_01	Module 27_01: App 27	28
28	M.USER.APP.28	28_01	Module 28_01: App 28	29
29	M.USER.APP.29	29_01	Module 29_01: App 29	30
30	M.USER.APP.30	30_01	Module 30_01: App 30	31
31	M.USER.APP.31	31_01	Module 31_01: App 31	32
32	M.USER.APP.32	32_01	Module 32_01: App 32	33
33	M.USER.APP.33	33_01	Module 33_01: App 33	34
34	M.USER.APP.34	34_01	Module 34_01: App 34	35
35	M.USER.APP.35	35_01	Module 35_01: App 35	36
36	M.USER.APP.36	36_01	Module 36_01: App 36	37
37	M.USER.APP.37	37_01	Module 37_01: App 37	38
38	M.USER.APP.38	38_01	Module 38_01: App 38	39
39	M.USER.APP.39	39_01	Module 39_01: App 39	40
40	M.USER.APP.40	40_01	Module 40_01: App 40	41
41	M.USER.APP.41	41_01	Module 41_01: App 41	42
42	M.USER.APP.42	42_01	Module 42_01: App 42	43
43	M.USER.APP.43	43_01	Module 43_01: App 43	44
44	M.USER.APP.44	44_01	Module 44_01: App 44	45
45	M.USER.APP.45	45_01	Module 45_01: App 45	46
46	M.USER.APP.46	46_01	Module 46_01: App 46	47
47	M.USER.APP.47	47_01	Module 47_01: App 47	48
48	M.USER.APP.48	48_01	Module 48_01: App 48	49
49	M.USER.APP.49	49_01	Module 49_01: App 49	50
50	M.USER.APP.50	50_01	Module 50_01: App 50	51

Table 4.3: Scenario 3 - Technical details

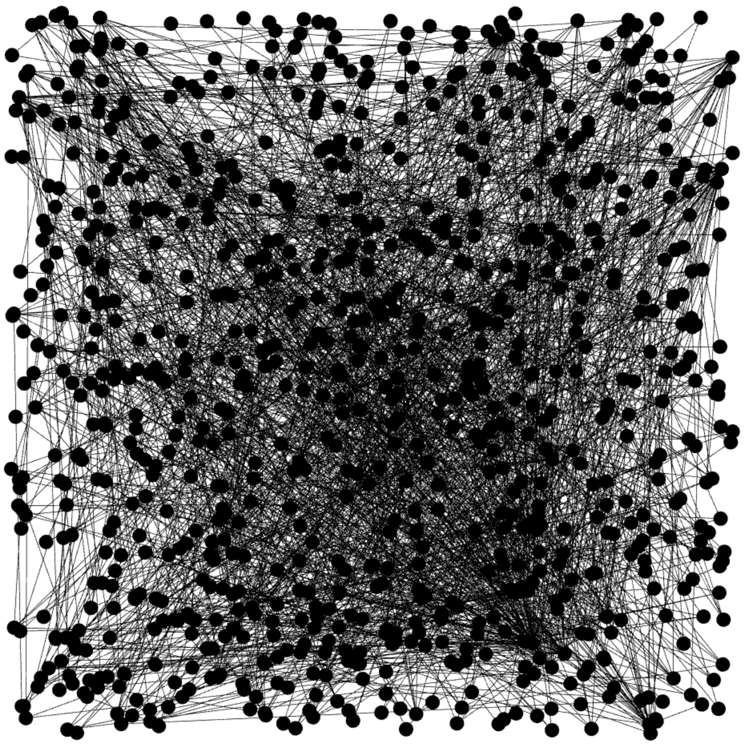

Figure 4.6: Scenario 3 - Initial network topology

The following sections includes the results from the simulation of scenario 3, separated into sections for each of the different post-simulation files.

Results from logfile

The results from the logfile shows that within the given time period, the simulation performed service movements that changed the placement of modules on the different nodes a total of 15 times. This means that the data the services are processing have been moved as well. All of the services start at node with id 0, and are then spread to different node after the first service movement.

In this scenario, there are 14 new users added along the simulation and some users are randomly moved as well.

The following logs were observed in the logfile:

```
2022-03-21 14:24:08,787 - root - INFO - Creating a new user 107 on node 646
2022-03-21 14:24:09,317 - root - INFO - Creating a new user 108 on node 699
2022-03-21 14:24:09,765 - root - INFO - Creating a new user 109 on node 135
2022-03-21 14:24:11,220 - root - INFO - Creating a new user 212 on node 234
2022-03-21 14:24:12,099 - root - INFO - Creating a new user 264 on node 235
2022-03-21 14:24:14,194 - root - INFO - Creating a new user 418 on node 44
```

```
2022-03-21 14:24:14,740 - root - INFO - Creating a new user 470 on node 710
2022-03-21 14:24:15,326 - root - INFO - Creating a new user 522 on node 246
2022-03-21 14:24:15,838 - root - INFO - Creating a new user 573 on node 731
2022-03-21 14:24:16,831 - root - INFO - Creating a new user 673 on node 788
2022-03-21 14:24:17,586 - root - INFO - Creating a new user 724 on node 442
2022-03-21 14:24:18,098 - root - INFO - Creating a new user 775 on node 860
2022-03-21 14:24:18,645 - root - INFO - Creating a new user 825 on node 91
2022-03-21 14:24:19,204 - root - INFO - Creating a new user 876 on node 729
```

From these logs, it is possible to see that all of the new users have been added to different nodes in the network.

In addition, there are two users being removed during the simulation:

```
2022-03-21 14:24:10,351 - root - INFO - Removing a user 108 on node 699
2022-03-21 14:24:13,625 - root - INFO - Removing a user 106 on node 644
```

The user "108" was one of the users added during the simulation and it did not exist for a long time before it was removed from the network. It is also possible to see the following users being moved to new nodes during the simulation:

```
2022-03-21 14:24:09,034 - root - INFO - Moving a user 107 from node 646 to 303
2022-03-21 14:24:12,900 - root - INFO - Moving a user 107 from node 646 to 232
2022-03-21 14:24:16,363 - root - INFO - Moving a user 470 from node 710 to 869
```

As an example, Module 35_01 started in node with id 0, and following this event the module was placed in the following nodes:

- 734, 10, 577, 844, 728, 442, 0, 145, 670, 885, 208, 112, 395, 283, 293

In this scenario there are approximately 1000 nodes in the network, which means that there are a lot more possible placements for the services than what was seen in the two previous scenarios. Even with all the different nodes, Module 35_01 was placed in one node more than once - node 0. However, this is not as many times in the same node as observed with the modules in scenarios 1 and 2.

Another example is Module 21_01. The module started in node with id 0, and following this event the module has been placed in the following nodes:

- 173, 157, 849, 618, 853, 269, 533, 580, 171, 681, 390, 393, 529, 791, 615

This module is never placed in the same node twice. With this large number of nodes, the service placement for each node is more varied, but this will depend on the specific algorithm implemented for the service placement as mentioned in scenario 2.

Figure 4.7 represents the amount of times a service has been placed in each of the nodes during the simulation. The nodes that have not been used for service placement are not included in the graph.

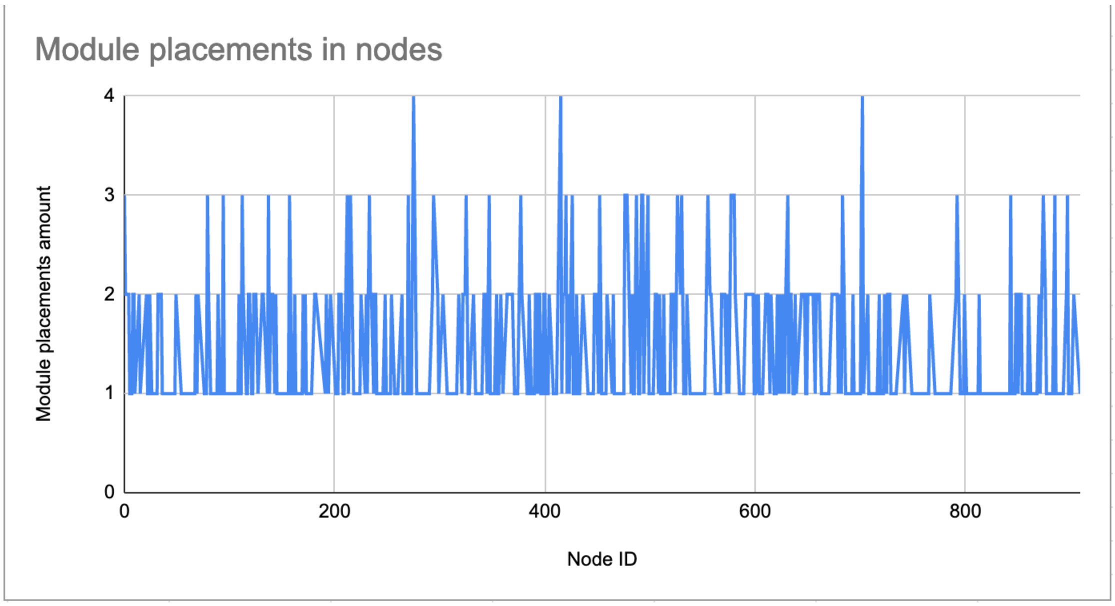

Figure 4.7: Module placement in each node

The nodes with the highest frequency of service placement are nodes with ID 275, 415 and 702, where a service was placed on the node a total of four times. From the graph it is possible to see that there is a higher amount of nodes being used one, two or three times. There are 391 nodes that are excluded from the graph, meaning that 391 nodes were not used for any services in the simulation. The ratio of modules to nodes and the limited simulation time means that it is not possible to use all nodes, which is reflected in the results. Compared to the two previous scenarios, it is obvious that when the amount of nodes increases in the network, the amount of service placement in each node decreases. The complete data on service placements is added as Appendix A.3.

Results from sim_trace-file

From the post-simulation file, it is possible to see that the first 2864 events all has a TOPO.dst value of 0. All events following these events have a more varied destination.

The statistics for TOPO.dst shows that the amount of times nodes are used as a destination ranges from 120 to 10. When removing duplicate entries where time-fields are ignored, the TOPO.dst values now changes into a frequency range from 1 to 59. The most used node is node 0 with 59 times, followed by node 275 with 6 times.

Looking at the DES.dst-field, there are many repetitive events. When removing duplicate events (ignoring all time-fields), there are 919 *different* events based on this variable.

The number of events for each module ranges from 8 to 32, where module

4_01 stands out with only 8 events, while the rest of the modules mostly has 15 or 16 events. At the top is module 8_01 with 32 events, followed by 30_01, 49_01 and 1_01 with 30 events each.

Results from sim_trace_link-file

After removing duplicate events (ignoring id and ctime) there are 6239 rows of data left. The node that is most used as source is node 3 which is used 918 times, followed up by node 0 that is used 376 times. These are the same most used nodes as seen in scenario 2. The least used nodes as source are used 2 times and regards several nodes.

The most used destination was node 0 with 297 times, followed by node 3 with 256 times. The least used destination nodes are only used 1 time and regards to multiple different nodes.

The latency between sending and receiving the messages between nodes is directly connected to the size of the message. The larger messages has a higher latency, and the smaller messages has a lower latency. The messages with a size of 10 bytes have a latency of 1.00001 and the messages with a size of 700 has a latency of 1.0007. The latency is calculated by subtracting time-emit from time-reception. As observed in scenario 2, some of the links the latency increased along the way. For messages with the size of 600 bytes, the latency went from 1.0006 to 10.00006. This happens only a few times for half of the different packet sizes (600, 30, 20 and 310 bytes), but not for packets with the size of 10, 120, 700 and 40 bytes - this latency stays the same throughout the simulation.

4.3 Fog network simulation using YAFS where the model includes volatility

This experiment was performed as an extension to the first experiment, and implementing a class for volatility is what makes the experiment unique. This experiment is based on the same scenarios as used in the first experiment with the same technical details. The volatility class makes it possible to focus more on how the data is stored in the nodes and not only the traffic and service movement. The simulation generates the same files as in the first experiment and an additional file for volatility data.

In this experiment, the results presented are mainly from the volatility file. This is because the experiment aims to look at the volatility in contrast to the first experiment where the purpose was to see the results of the service movement.

Technical details

The simulation in this experiment created a network with nodes with additional attributes for memory. The memory is defined by the type of node (source, sink or proxy). The volatility is defined by the following rules:

SOURCE:

- Time between creation and unlink of data is decided by a negative exponential distribution of 1/300 seconds (**delta_unlink**)
- Time between unlink and earsure is decided by a negative exponential distribution of 1/60 seconds (**delta_erase**)

SINK:

- Time between creation and unlink of data is decided by a negative exponential distribution of 1/300 seconds (**delta_unlink**)
- Time between unlink and earsure is decided by a negative exponential distribution of 1/600 seconds (**delta_erase**)

PROXY:

- Time between creation and unlink of data is decided by a negative exponential distribution of 1/300 seconds (**delta_unlink**)
- Time between unlink and earsure is decided by a negative exponential distribution of 1/10 seconds (**delta_erase**)

Events with type SRC have the same node as both *src* and *node* as it generates the message. PRX-events have different values as it receives the message from one node (src), has its own ID (node) and a different destination (dst). SNK-events have one *src*, and then the same *dst* and *node* as this is where the message ends up.

4.3.1 Results from scenario 1 - Witness

There are 3 sources deployed in the simulation. This amount was decided as it was relevant to the small size of the network. The following statistics are the results of running the simulation with scenario 1:

PRX			
	Memory	Unlink time	Erase time
count	30.00000	30.000000	30.000000
mean	8935.50000	275.555006	3.366062
std	3947.40136	398.475086	3.206849
min	6022.00000	3.654010	0.036501
25%	6022.00000	42.452248	0.980196
50%	6371.50000	136.784288	1.959844
75%	14413.00000	305.901184	5.345024
max	14413.00000	1861.853621	10.792072

Table 4.4: Scenario 1 - Proxy nodes statistics

SNK			
	Memory	Unlink time	Erase time
count	105.000000	105.000000	105.000000
mean	8365.428571	330.071308	257.014020
std	4030.622048	299.365507	296.517455
min	3664.000000	2.371567	0.208046
25%	6022.000000	101.401794	52.024132
50%	6721.000000	227.926505	153.226579
75%	14413.000000	443.933770	339.095392
max	14413.000000	1334.898924	1594.516046

Table 4.5: Scenario 1 - Sink nodes statistics

SRC			
	Memory	Unlink time	Erase time
count	90.000000	90.000000	90.000000
mean	3178.722222	254.099858	14.552874
std	2416.314047	278.262362	19.007565
min	395.000000	0.836544	0.096942
25%	395.000000	72.456861	2.869872
50%	3664.000000	146.393697	8.167930
75%	6022.000000	356.805166	18.544150
max	6022.000000	1387.209775	125.952682

Table 4.6: Scenario 1 - Source nodes statistics

From these statistics some findings should be mentioned. There are 90 events created by source nodes, 105 events sent to sink nodes, and 30 events going through a proxy node.

The mean of the statistics shows that:

- The average **memory** of source nodes is 3178.722222 M, which is approximately 3.1 GiB.
- The average **memory** of sink nodes is 8365.428571 MiB, which is approximately 8.17 GiB.
- The average **memory** of proxy nodes is 8935.50000 MiB, which is approximately 8.7 GiB.

The proxy nodes are on average more accessible regarding memory available on the node, whereas the source nodes have the least amount of memory. Considering that source nodes often are the IoT devices, such as sensors and cameras, it is realistic that there is less memory available on theses devices. Looking at the statistics for unlink time, it is possible to see the following:

- The average **unlink** time for source nodes is 254.099858
- The average **unlink** time for sink nodes is 330.071308
- The average **unlink** time for proxy nodes is 275.555006

This means that on average, the messages exists longer in the sink nodes than they do in the proxy and source nodes. However, there is not a significant difference between the three different types of nodes in regards to the time it takes messages to be removed from the application. There are more differences in the statistics for erase time:

- The average erase time for source nodes is 14.552874
- The average erase time for sink nodes is 257.014020
- The average erase time for proxy nodes is 3.366062

This means that messages exist in the sink nodes after being removed from the application significantly longer than in the other nodes. The messages exist the shortest in the proxy nodes. This is also possible to observe in the other statistics, where the highest duration of a message to exist in a sink node is 1594.516046 versus 10.792072 in a proxy node. In the simulation, the results are based on the distribution data set before initiating the simulation, and the results directly reflect this.

Results from volatility-file The volatility class separates this experiment from the first experiment, and the results saved to the volatility-file are the most relevant to analyze.

Looking at **all of the events**: the node with most events as src is node with ID 4 which has 90 events, following is node 0 with 70 events, and the node with the least events as src is node 5 which has 65 events.

Looking at only **source events**: the most events comes from node 4 which generates 35 messages, and then node 0 which generates 30 messages, and finally node 5 with 25 messages.

Looking at only **sink events**: Instead of reflecting the amount of events from the source nodes, there exists some events where the sink nodes generates a message, has itself as the destination, and handles the message as well. This occurs 10 times for node 5 and 5 times for node 0. This creates an equal amount of events to 35 from each source node. Node 4 is not a part of the sink nodes, indicating that node 4 is a *pure source*. The node which has received most events is node 3 and 1 with 30 events each, followed by node 5 with 20 events. This can indicate that they are devices that are frequently used in the network. The least amount of events has gone to node 0 with 10 events, which means that it is probably a device that is not actively used in the network - at least not for the given time period. Node 2 receives 15 events.

Looking only at the **proxy events**: there are three nodes used as proxy: node 0, 1 and 3. There are 30 events where a message goes through one of these proxy nodes. There are 15 events with node 0, 10 with node 1 and 5 with node 3. The most used application that has messages going through a proxy node are app 4 and 1 with 10 events each, followed by app 6 and 2 with 5 events each. Apps 0, 3 and 5 do not generate any messages via a proxy node. The destinations of these messages are to three different nodes: 2, 3 and 5. The source of the events are from nodes 0, 4 and 5, where most of them are from node 4 with 20 events, and least from node 0 and 5 with 5 events each.

This means that the three nodes used as proxy in the simulation also have generated a few events and are used as destination. This is likely due to the network only consisting of 10 nodes in total and there are 7 applications running. From the results it seems that node 4 is a pure source node, and that node 5 and 2 are *pure sink nodes*.

4.3.2 Results from scenario 2 - Tool

There are 10 sources deployed in the simulation. This amount was decided as it was relevant to the medium size of the network. The following statistics are the results of running the simulation with scenario 2:

PRX			
	Memory	Unlink time	Erase time
count	415.000000	415.000000	415.000000
mean	10962.746988	291.941127	2.917503
std	7435.712755	303.392722	3.391380
min	1330.000000	2.445672	0.005705
25%	3610.000000	81.465385	0.723040?
50%	10021.000000	177.413784	1.761711
75%	17451.000000	386.990333	3.688454
max	25478.000000	1666.450598	19.706395

Table 4.7: Scenario 2 - Proxy nodes statistics

SNK			
	Memory	Unlink time	Erase time
count	700.000000	700.000000	700.000000
mean	12100.642857	309.541237	182.324885
std	7696.329398	299.781240	205.268530
min	1292.000000	0.821127	0.089811
25%	6505.500000	100.136797	50.445066
50%	10658.000000	216.210892	118.110079
75%	18378.000000	446.616505	245.185212
max	26898.000000	2199.998040	2053.017289

Table 4.8: Scenario 2 - Sink nodes statistics

SRC			
	Memory	Unlink time	Erase time
count	700.00000	700.00000	700.00000
mean	442.80000	313.911934	6.547315
std	281.76582	327.837342	6.635133
min	6.00000	0.358388	0.005495
25%	203.00000	90.035522	1.837062
50%	476.00000	204.057771	4.486005
75%	653.00000	411.176986	9.322882?
max	851.00000	2133.082992	60.774306

Table 4.9: Scenario 2 - Source nodes statistics

From these statistics some findings should be mentioned. There are 700 events created by source nodes, 700 events sent to sink nodes, and 415 events going through a proxy node. The mean of the statistics shows that:

- The average memory of source nodes is 442.80000 MiB, approximately 0.4 GiB.
- The average memory of sink nodes is 12100.642857 MiB, approximately 11.8 GiB.
- The average memory of proxy nodes is 10962.746988 MiB, approximately 10.7 GiB.

The average memory of sink and proxy nodes are not that far apart, whereas the average memory of source nodes stands out and is significantly lower than the others. In contrast to the results in scenario 1, the memory in this case is noticeably lower, making it more realistic to be IoT devices with low capacity. The statistics for unlink time shows:

- The average unlink time for source nodes is 313.911934
- The average unlink time for sink nodes is 309.541237
- The average unlink time for proxy nodes is 291.941127

There are no large differences in the unlink time of the different node types, and the standard deviation for each type is around 300. Looking at the statistics for erase time, the following data are presented:

- The average erase time for source nodes is 6.547315
- The average erase time for sink nodes is 182.324885
- The average erase time for proxy nodes is 2.917503

The erase time for sources and proxy nodes is similar, but the erase time in sink nodes stands out. The highest erase time in a sink node is 2053.017289, which is the message "M.USER.APP.11" coming from node 29 and being sent to node 11. There is therefore (on average) a higher chance of locating the data when it is in a sink node in this network.

Results from volatility-file Looking at **all of the events** from the volatility file, it is possible to see that the node with most events as src is node 57 which has 210 events. The node with the least events as src is node 4 which has 145 events. Further, the least amount of events is regarding node 23 with 170 events, the rest has 175-200 events each.

Looking at only **source events**, it is observed that 10 different nodes have been used as source nodes: 4, 8, 23, 29, 45, 57, 59, 74, 76 and 92. All of them generated 70 messages each. In contrast to scenario 1, all of them are now *pure sources*.

When observing only **sink events**, it is possible to see that there are 38 different sink nodes. The most messages have been sent to node 0, with 115 events. Node 0 might therefore be a commonly used device in the network. The least events are for multiple nodes with only 5 events, and they might be a device of the similar type that is not frequently used. All of them are *pure sink nodes*.

From observing the **proxy events** only, it is possible to see that there are 22 nodes being used as proxy nodes. The most used proxy node is node 5 with 65 events, and the least used are nodes 2, 7 10, 15, 18, 24, 25, 38 and 7 with 5 events. All of the different applications have generated messages that go through a proxy node.

4.3.3 Results from scenario 3 - Target

There are 100 sources deployed in the simulation. This amount was decided as it was relevant to the large size of the network. The following statistics are the results of running the simulation with scenario 3:

	PRX		
	Memory	Unlink time	Erase time
count	15200.000000	15200.000000	15200.000000
mean	9155.116447	300.081235	2.258095
std	8671.972225	298.989948	2.833579
min	80.000000	0.007966	0.000025
25%	1926.250000	86.402383	0.535673
50%	6552.000000	208.828849	1.334811
75%	13606.000000	416.620790	2.913415
max	56719.000000	3116.725435	45.140120

Table 4.10: Scenario 3 - Proxy nodes statistics

SNK			
	Memory	Unlink time	Erase time
count	25500.000000	25500.000000	25500.000000
mean	10895.800980	297.131676	149.729456
std	8750.269313	299.231371	179.976301
min	1051.000000	0.012184	0.002518
25%	4239.000000	84.822956	36.110061
50%	9245.000000	203.489310	90.596652
75%	16858.000000	410.880265	194.164295
max	59129.000000	2877.666527	3502.967291

Table 4.11: Scenario 3 - Sink nodes statistics

SRC			
	Memory	Unlink time	Erase time
count	25500.000000	25500.000000	25500.000000
mean	507.110000	302.594140	6.408334
std	270.195968	301.845445	6.447273
min	11.000000	0.000264	0.000074
25%	288.250000	86.127957	1.853965
50%	542.000000	211.040971	4.455867
75%	729.250000	418.437852?	8.891102
max	976.000000	3629.600217	68.093768

Table 4.12: Scenario 3 - Source nodes statistics

From these statistics, some findings should be mentioned. There are 25500 events created by source nodes, 25500 events sent to sink nodes, and 15200 events going through a proxy node. The mean of the statistics shows that:

- The average memory of source nodes is 507.110000 MiB, approximately 0.5 GiB.
- The average memory of sink nodes is 10895.800980 MiB, approximately 10.6 GiB.
- The average memory of proxy nodes is 9155.116447 MiB, approximately 8.9 GiB.

The average memory of sink and proxy nodes is not that far apart, whereas the average memory of source nodes stands out and is significantly lower than the others. This is similar to what was observed in scenario 2. The statistics regarding unlink time show that:

- The average unlink time for source nodes is 302.594140
- The average unlink time for sink nodes is 297.131676
- The average unlink time for proxy nodes is 300.081235

There are no large differences in the unlink time of the different node types, and the standard deviation for each type is around 300, which is the same as observed in scenario 2. Looking at the erase time in the experiment, it is possible to see the following average values:

- The average erase time for source nodes is 6.408334
- The average erase time for sink nodes is 149.729456
- The average erase time for proxy nodes is 2.258095

The erase time for sources and proxy nodes are similar, but the erase time in sink nodes stands out, which is the same as seen in the two previous scenarios. The highest erase time in a sink node is 3502.967291, which is the message "M.USER.APP.36" coming from node 78 and being sent to node 43. There is therefore (on average) a higher chance of locating the data when it is in a sink node in this network.

Results from volatility-file By looking at **all of the events** in this file, the node with most events as src is the node with ID 827 which has 860 events, followed by node 953 with 830 events. The node with the least events as src is the node with ID 8 with 510 events. Further, the least amount of events is regarding node 37 with 515 events, the rest has 520-765 events each.

When addressing only **source events**, 100 different nodes have been used as source nodes. All of them generated 255 messages each and all of them are *pure sources*.

By looking at only **sink events**, it can be observed that there are 424 different sink nodes. Most messages have been sent to node 2, with 1335 events, followed by node 3 with 1220 events. Nodes 2 and 3 might therefore be commonly used devices in the network. The least events are multiple nodes with 5 events and might be a device of a similar type that is not frequently used.

Lastly, 180 nodes have been used as **proxy nodes**. The most used proxy node is node 3 with 865 events, and the least used are multiple nodes with 5 events. Both node 3 and 5 have been used as sink nodes as well in the simulation, meaning that not all of the sink nodes are pure sink nodes. All of the different applications have generated messages that go through a proxy node.

4.4 Summary of results

The first experiment provided results mainly regarding the service placement on the nodes in the network and how the services were moved throughout the simulation. The second experiment made it possible to see the volatility for the different node types, and provide data on where it might be possible to locate evidence. From the two experiments some findings should be highlighted:

- The source nodes in scenarios 2 and 3 were all pure source nodes. In scenario 1 this was not the case as only two of the nodes were pure source nodes. This is likely due to the size of the network, where the smallest network has more devices that have several purposes such as both generating data and acting.
- When the size of the network increases, each node is less likely to be used for service placement multiple times. Each node in scenario 1 were used significantly more than the nodes in scenarios 2 and 3.
- In each scenario there are nodes being used particularly more than others, giving an indication of which devices and services are the most used.
- Some nodes have multiple services running simultaneously, but the amount and frequency of this decrease when the network size increases.
- The memory is higher and the existence of data is longer in the service/sink nodes, indicating a better chance of locating evidence in nodes of this type.

The findings shows how the dynamic service placement might be in a fog network, and how volatility differentiates within the types of entities. This provides an idea of how possible evidence can exist and move around in a live system.

4.5 Hypotheses discussion

The following hypotheses were created for the experiments:

Null hypothesis (H_0): The evidence location in simulated fog-IoT systems is not affected by the dynamic service placement
Alternative hypothesis (H_1): The evidence location in simulated fog-IoT systems is affected by the dynamic service placement

The hypotheses were also defined as:

(H_0): $x = 1$
(H_1): $x > 1$

Where x = number of nodes used for service placement.

Because of the the nature of the hypotheses, it is defined as a one-tailed test. From the experiments it was possible to see where the different services were placed during the simulation. Specifically, the results that shows how many different nodes each of the services in i each scenario was placed in is the relevant data. The statistics are presented in Table 4.13.

Scenario	Services deployed	Mean of unique nodes	Std.
Scenario 1	7	8	1
Scenario 2	14	14.07	1.7
Scenario 3	51	14.3	1.02

Table 4.13: Hypothesis statistics

The confidence level is set to 95% and the alpha value α is therefore .05. The statistical method used is the *p-value approach* [63].The statistics were used to find the *z-score*. This is defined by:

$$Z_c = \frac{(\bar{X} - \mu)}{(\sigma_x)} \tag{4.2}$$

$\bar{X}$ = Mean of unique nodes
μ = Mean of the population
σ = Standard Error with sample

The hypothesis is tested against the three samples from the scenarios with the sizes 7, 16 and 51, where the following z-scores were calculated:

Scenario 1: Z-score of 18.6
Scenario 2: Z-score of 29
Scenario 3: Z-score of 93

Based on the one-tailed test with a confidence level of 95%, the rejection region of the distribution is above 1.645. From the z-score, the p-value was calculated for the three sample populations. With the extreme z-values, all of the p-values for the scenarios were of **< .00001.**

This result shows that for all of the scenarios, the result is *significant*. The z-scores in this case are extreme, which is due to the comparison with the cloud network where there is only one node. Normal distribution is therefore *not* an applicable model for this data. However, considering that a "normal" network can consist of a centralized cloud, this is still a relevant measure.

Based on these findings in the experiments, it is possible to reject the null hypotheses and keep the alternative hypothesis. With the observations made in the experiments, it is possible to see that the location of the possible evidence can be at different locations based on the dynamic service movement, supporting H_1.

To further answer RQ2 and RQ3, the impact these findings have on the digital forensic process is addressed and discussed in the analysis in Chapter 5.

Chapter 5

Digital forensics analysis

The experiments aim to understand how existing digital forensic methods work with IoT devices placed in a fog network. The chapter is separated into four parts - one for each scenario and a summary of the results and analysis. The previous chapter has given several artifacts for each scenario that have been further used for analysis in this chapter. The analysis considers the technical details of each scenario and looks at the artifacts in context with different digital forensics approaches - specifically within cloud forensics, IoT forensics, and fog forensics.

The five phases of the digital forensics process were previously presented in Chapter 2. In this analysis, the focus is on the *identification* and *collection* phases, as these are where most of the challenges related to fog forensics exist.

Before conducting the forensic investigation, there exists multiple proactive solutions that can be used for digital forensic readiness. Therefore, it is necessary to address the applicability of these mechanisms in each of the scenarios and discuss how it could have affected the incident.

Identifying the criminal activity and relevant evidence in the scenarios is the first phase of the investigation. The aim is for the investigators to create hypotheses about the incident and understand what evidence that could exist. An essential step in the first phase is to have proper preparations in order before initiating the investigation. As fog computing is different from a regular network, the preparations must be suitable for the structure of the fog network. This includes having a lab setup with appropriate tools and a prepared team [17]. The dynamic routing nature of a fog network can affect the identification phase of the investigation as more devices than usual may be involved regarding the same evidence. This was an important aspect of the analysis in the following sections.
Considering that all the scenarios are live systems running while investigators are to retrieve evidence, the collecting phase must happen accordingly. The order of volatility is an important factor in the analysis where evidence is collected from the fog nodes, and there are multiple different types of devices creating and storing data. Acquiring data from one node may impact the data stored in other nodes, and is therefore important to consider. The investigators must prioritize what data to collect and when [17].

It is difficult to cover all parts that must be considered in an investigation, especially when the scenarios are in a simulated environment and a real-life scenario would be very complex. The analysis is performed at the best ability with the results from the simulation and the technical details defined for each scenario.

Scenario 1: In this first scenario there was a surveillance camera that captured footage of a criminal activity. This device was placed within a fog network.

Scenario 2: In the second scenario, an IoT device placed in a fog network was infected with malware and further used as a part of a botnet attack.

Scenario 3: In the third scenario, an IoT device placed in a large-scale fog network logged sensitive data, and a criminal intentionally found and stole this data.

Each of the scenarios are of a different scale which comes with different challenges and opportunities. Scenario 1 is a small network and does not require too much effort to identify all devices and nodes. Scenario 2 shows a larger network where it might become difficult to identify all connected devices and follow the trace between all of the nodes. In scenario 3, the scale of the fog network is the largest issue for forensic investigators. Having this amount of nodes connected can imply that evidence is being spread between a high amount of nodes, and not only a few as seen in scenarios 1 and 2.

The investigators that are to collect evidence from the incidents do not know in advance which specific device is infected and will need to follow a digital forensic approach to locate it and retrieve evidence. As there are not that many fully developed frameworks for IoT devices located in a fog, a combination of IoT forensics and cloud forensics might be applicable in the scenarios. With fog computing, it is no longer only the specific IoT device or cloud servers involved in the forensic process. The forensic approach must consider all of the devices and nodes in the network, which can quickly become complex [22].

In [3], the authors introduce some challenges in fog forensics such as the limitation of bandwidth, higher criticality of logging, and higher criticality in regards to the dependability of chain of custody. These challenges are not unique to fog forensics but are mostly of higher criticality than seen in cloud computing. There are many techniques used in cloud forensics that can be implemented in fog forensics as well. However, as there are key differences between fog and cloud, it is important to have evidence-gathering methods that are appropriate for the fog paradigm.

5.1 Scenario 1

In the simulation of scenario 1, it was observed that multiple services were running on some of the nodes and that most services have been located in all of the different nodes. The evidence in the scenario could be stored at multiple locations. The evidence can include the footage of the criminal activity and surrounding logs that provide more information about the incident. Some important questions the investigators need to consider are:

- Where was the surveillance camera located at the time of the incident?
- How has the evidence moved within the network from the time of the incident until now?
- Where is the footage located in the fog network now?
- Where in the network can relevant log data be located?
- How can evidence be retrieved and still keep its integrity?
- How can the evidence be retrieved with the least impact on the network functionality?
- What is the volatility of data?

5.1.1 Proactive solutions

Similar to the use case involving a smart refrigerator presented in [25], this scenario also includes a network of a small scale. With this framework, the FoBI management software detects unusual traffic coming from an unfamiliar location, flags the activity, and starts a monitoring and analyzing process. With the unfamiliar properties, the traffic was blocked in this use case. Concerning the scenario in this thesis, the data that the investigator is looking for is not malicious in itself but contains valuable footage of a crime. FoBI will therefore not be relevant for the scenario as there was no unusual traffic to detect in the first place. If there were any other evidence of this crime logged by another device such as a sensor, FoBI might have been applicable. This could include a movement sensor detecting unusual traffic, or perhaps a door that is being opened at an unusual time. However, even if the framework [25] was implemented prior to the incident, it would not provide the footage that the investigators are looking for. The same regards to VTA-IH, the framework proposed by Bandil and Al-Masri [26]. The framework performs detection with a complex event processing model (CEP) on the data coming from the IoT devices. As with FoBI [25], unless there were any other traffic in the network surrounding the crime, the framework would not be relevant for this scenario.

The secure network archives for IoT presented by Duan et al. [27] requires the fog nodes to be equipped with Software Guard Extensions (SGX). The model captures traffic coming from the IoT devices and stores it in the SGX enclave, which is further stored in a queryable archive in the cloud. The investigator can privately retrieve this data. If this model was present before the incident, there is a possibility that some data was captured and stored. In contrast to FoBI [25] and VTA-IH [26], this model is not based on detecting abnormal activity. Therefore, it is a higher chance that the evidence from the criminal activity is available for the investigator and its integrity is kept.

The Forensic State Acquisition from IoT proposed by Meffert et al. [28] implements a centralized controller that captures a state from the IoT device. In this scenario it could be any motion detected by the camera. The model is not considered for fog systems, but using fog nodes as the controller could be a possibility, where the relevant state changes are further sent to the cloud. If the camera were to detect the crime and capture a state change, it would not provide the footage, but could help create a timeline of events. The paper did not consider forensic soundness, which would be a crucial point if the investigators were to collect evidence and present it in a case.

FEMS [29] is a relevant framework in this scenario as it was created for smaller networks involving IoT devices. The model can be placed in devices such as gateways, meaning that even if it was not designed for fog computing, there is a possibility to implement it in a fog node. In this way, network monitoring can be done with intrusion detection and prevention functionalities. The data will be stored for a selected time, and the occurrence of any incident will treat future data as relevant for the case [29]. The model is similar to the more recent frameworks presented in [25] and [26], but is specifically designed for small networks such as a smart home. However, such as in FoBI and VTA-IH, it is not likely to be significantly beneficial in terms of the crime in this scenario.

5.1.2 Identification

As the first step in the digital forensic process, the investigator must identify the possible evidence that can exist in the network [17]. The criminal activity has been identified, and it is now necessary to figure out what and where evidence can be found to understand more about the act.

The scenario is in a small-scale local fog network, so it does not involve many users. Therefore, it is not a large issue in this case that a great number of users could be affected by the investigation. The investigators must still consider that it is a live system and that there are some active users that should not be affected by the investigation unless they are connected to the case.

In the first experiment, it was observed that most of the services had been placed in all of the different nodes and that the evidence in theory could be stored anywhere in the network. It was observed that node 1 was the most used node, indicating that this node is a popular node based on the Barabási-Albert model used to create the network topology. There were also multiple services running on this node simultaneously at some point during the simulation. This node might have a higher capacity than the other nodes or it could be the most optimal node regarding latency. This would be decided by the service placement algorithm in a real network. With these results, there is a higher chance to find some sort of evidence in node 1. The investigators might get information from the owner of the network regarding which nodes are the most used. It is possible that it is not completely accurate information, but as there are few nodes in the network, the owner should have an indication of which devices are used more than others.

In the second experiment, it was observed that the average source node had a memory of 3.1 GiB, the average sink node had 8.17 GiB and the average proxy node had 8.7 GiB. This means that the sink and proxy nodes have more capacity than the source nodes. It was also possible to see that the *unlink* and *erase times* were significantly higher for the sink nodes. The proxy nodes had high capacity, but the messages were quickly erased after being unlinked from the application. However, it is important to mention that these values are within the scale of predefined limits and that the results are somewhat as expected.

The investigators have less chance of finding any evidence still existing in the proxy nodes or the source nodes. As the network is small, it is easy to know what the different nodes are, and in this way locate the most relevant nodes quickly.

The results also indicated that node 4 was the only pure source in the network, which means that this is an IoT device, likely a sensor. This node was also the node that generated the most messages. In addition, node 0 and 5 were used more than the other nodes to generate messages but did also receive some messages. This indicates that these IoT devices can generate data and perform an action based on the messages received. In this scenario, it *could* be the camera that recorded the criminal activity. 30 events were going through a proxy node, and these events can therefore have more trails left behind than those that have not gone through a proxy node.

With the small scale of the network in this scenario, it is less likely to have new devices added or removed. There is movement of services in the network, but the initial devices are still the devices connected at the investigation time.

DFIF-IoT proposed by Kebande and Ray [30] is a holistic framework that was developed for the cases where IoT devices are involved. Their depiction of DFR and the Identification phase is a *Proactive process*. This process can consist of proactive solutions such as those previously discussed in Section 5.1.1. It is mainly for all the activities that happens *before* the incident is identified [30]. The process specifically focuses on identifying the sources of evidence related to IoT devices and defining IoT scenarios. The focus on the devices is relevant for scenario 1 as IoT devices are involved. It is however not necessarily crucial to use the framework as the scale of the network is small and the devices are known by the owner. With the prior knowledge, the investigators would easily know what devices to examine. The incident in the scenario is not defined as a *cybercrime* and there is not a lot of IoT data that needs to be identified. The investigators are looking for the footage that captured the criminal activity as evidence in the ongoing investigation. Based on this and the scale of the network, the identification phase could therefore be successful by proceeding with a standard five-phase process.

A more recent approach to the involvement of IoT devices in digital forensic investigation was proposed by Salama et al. [21]. The phases relevant to the *identification* in MCF2I are defined as the two phases *Manage/Prepare* and *Identify*. The multilevel approach provides a detailed description of tools and activities that can be performed when identifying possible evidence sources and defining the incident. With a specific focus on the IoT devices, this framework is also relevant for the scenario. As it is more similar to a the five-phase process used in

[17], it might be more applicable in this scenario than DFIF-IoT [30] because of the type of crime and evidence.

The identification phase could benefit from these new frameworks as the focus is on the IoT devices. However, the dynamic service movement of fog computing is not considered and must also be considered in the process. As the network in this scenario is small, it is less likely to create issues for the investigation and none of the nodes should be difficult to identify.

5.1.3 Collection

In [64] the authors mention the limited volatile storage and processing capability as one of the challenges with IoT devices in digital forensics, where evidence may only exist for a short period of time. These devices can also add another level of complexity regarding the collection as they may use different operating systems. Therefore, the investigators must consider this when collecting evidence from the nodes. Short-term storage is one of the characteristics of devices in a fog network, and if the specific data is not sent to a cloud server, it could be difficult to retrieve it. In the second experiment, the volatility was observed where it was possible to see that the messages did not exist long in the different nodes. Combining the erase time and unlink time, the messages had a longer average existence in the sink nodes.

DFIF-IoT [30] includes a separate process for IoT forensics in the digital forensics investigation process. The phase is specifically based on cloud forensics, network forensics and device-level forensics. The focus on these types of forensics is relevant for scenario 1 as there are some IoT devices involved. The network in the scenario is however small, and as discussed in the previous section it does not appear that the framework would provide any particular benefits compared to the usual digital forensic investigation process. There is no need for comprehensive investigation of the entire network, and the collection of the evidence would likely be satisfactory without using DFIF-IoT.

As discussed in the previous section, MCF2I [21] seems to be an applicable approach for the scenario. The *collection* phase is defined as *Aquire/Image* in the framework, and the multilevel approach includes more specific steps and guides for the investigators in regards to collecting evidence from IoT devices. As with the identification phase, MCF2I appears to be a plausible approach to collect IoT evidence in the scenario. It does not consider the network to involve fog computing, and the investigation phase must therefore address the investigation with this in mind. It might not be the optimal framework for fog-IoT, but it is a step in the right direction.

The evidence collection must be performed in a forensically sound manner, and the short-term memory and highly volatile data is the biggest challenge in this phase of the investigation. Using a framework specifically created for IoT might not be necessary for this scenario, but the incident still includes an IoT device which must be considered. Having the right guidelines and procedures for collecting IoT data will be important in order to retrieve the evidence and keep its integrity. It is a live system, and the activities of the forensic investigator will affect the system if not performed carefully.

5.2 Scenario 2

In the simulation of scenario 2, it is observed that some nodes have two services running simultaneously, but mostly there is only one service running in each node. Not all nodes have been used for service placement, but the evidence in the scenario could be stored at multiple locations. The evidence can include the infected device, surrounding log data on the infected device, and how it became connected to the botnet. Important questions the investigators need to consider are:

- How did the attacker get access to the IoT device?
- Where was the device located at the time of the incident?
- Where in the network is relevant log data located?
- What was the source of the intrusion?
- How has the device been used in the botnet attack?
- How can evidence be retrieved and still keep its integrity?
- How can the evidence be retrieved with the least impact on the network functionality?
- What is the volatility of data?

5.2.1 Proactive solutions

Another use case involving a smart city presented with the FoBI framework [25] is similar to the network in scenario 2, which involves a larger fog network. In this use case, an IoT device is not responding, which is further flagged and the forensic analyzer module starts examining the activity of the device. If FoBI was implemented in the fog system in scenario 2 prior to the incident, the activity might have been detected before the attack was launched. The monitoring manager in the framework could detect some unusual behavior coming from the infected device, and further block all traffic from this device. As it is a botnet attack it would likely include devices from multiple networks, and therefore it would still be performed even if the device in this specific network was disconnected. However, it could ensure that the spreading of malware inside the specific network was stopped and that no further connected devices would be compromised.

VTA-IH is also a relevant framework in this scenario, where the CEP model could detect if there were any abnormal traffic in the network coming from the compromised device [26]. The probability of detecting the traffic is based on the degree of abnormality defined prior to the incident. If the traffic was above the threshold, the Evidence Recovery layer would extract important data from the logs surrounding the suspicious traffic. This would further initiate a process in the Threat and Attack Handling layer to defend the system, where the compromised device could be detached. The gathered evidence could then be provided to an investigator if needed.

Both FoBI and VTA-IH are frameworks that could have ensured evidence from the incident, but another model that would have been relevant is the secure network archives proposed by Duan et al. [27]. The model captures all of the traffic from the IoT devices, not only traffic that is flagged as abnormal. This means that the amount of data available is a lot higher than in the other frameworks, but the use of keywords in queries can ensure that the investigator locates the relevant evidence efficiently. With the encrypted data and private queries, the model is important for keeping evidence integrity, but is limited to the fog nodes' capacity. The model does not initiate any defending processes, but can be resourceful in a post-mortem investigation [27].

The Forensic State Aquisition from IoT proposed by [28] captures a state from the IoT device and could detect any state changes in the infected device. This could provide an indication of there being abnormal behaviour in the network, but it is not guaranteed to be detected. If the incident is already under investigation, the model could help create a timeline of events in the system.

5.2.2 Identification

Scenario 2 is of a larger scale than previously seen in scenario 1, which means that there are other challenges that must be considered in the identification phase. It is a live system with more users involved, which means that the investigation could significantly affect the network.

In the first experiment, it was observed that some services were being placed on the same nodes multiple times, and that some of the nodes hosted more than one service at once. However, there were no instances where a node hosted more than two services at the same time. Some nodes were not used for service placement, meaning that these nodes are not likely to contain any evidence. The most popular node for service placement was node 4. In a real network, this node might have a higher capacity than the other nodes or be the most optimal node based on the service placement algorithm. Node 4 is a highly relevant node to investigate in the network, as there is statistically a higher chance of finding evidence here. With the network being of medium-scale, the investigators might not get all the information needed from the owner of the network. There could be difficulties finding out which nodes are the most used, and assumptions might be the best solution.

In the second experiment, it was observed that the average source node had a memory of 0.4 GiB, the average sink node had 11.8 GiB and the average proxy node had 10.7 GiB. This means that the sink and proxy nodes have a significantly higher capacity than the source nodes. The source nodes have noticeably less memory compared to the same node type in scenario 1, which indicates that the IoT devices in this network consist of more pure sources such as sensors. There were no large differences in regards to average *unlink time* between the different node types, but the average *erase time* was significantly higher for sink nodes. Similar to scenario 1, the proxy nodes had high capacity but the messages were quickly erased after being unlinked from the application. The investigators have less chance of finding any evidence still existing in the proxy nodes or the source nodes. As it might not be possible to have an overview of all the nodes, it becomes challenging to know what the different nodes are, and therefore spend more time locating the most relevant nodes quickly.

The results also showed 10 different source nodes in the network, all of which were pure sources. This matches with the results proving that the average memory of the source nodes is low. It was also possible to see that all of the sink nodes in the scenario were pure sink nodes. With the scale of the network in this scenario, it is likely to have new devices added or removed which creates difficulties in finding relevant devices.

DFIF-IoT [30] was discussed to not be particularly relevant in the first scenario. The focus on the devices is highly relevant for scenario 2 as numerous IoT devices are involved. It could be crucial to use the framework or a framework of similar type in the investigation, as the scale of the network is quite large and there are possibilities of unknown devices. The incident in the scenario is defined as *cybercrime* and there are lots of IoT data that needs to be identified. The investigators are looking for any evidence that can help identifying the incident and locate possible sources of evidence. Based on this and the scale of the network, the identification phase could therefore benefit from using DFIF-IoT [30] or another holistic framework of similar approach. MCF2I [21] proposed by Salama et al. is also a relevant model for this scenario. The multilevel approach would benefit the investigators, providing guidelines and procedures to prepare for the IoT investigation and identify possible sources of evidence.

The identification phase would likely benefit from these new frameworks focusing on the IoT devices. The dynamic service movement of fog computing is not considered, and must also be evaluated in the process. As the network in this scenario is larger than seen in scenario 1, it might become difficult to have an overview of all the nodes that exist. There does not seem to be any frameworks that considers the identification phase with fog computing specifically, which means that the challenge of the dynamic service movement is likely to negatively affect the investigation of this scenario.

5.2.3 Collection

As previously stated in Section 5.1.3, Jones and Vidalis [64] mention the limited volatile storage and processing capability as one of the challenges with IoT devices in digital forensics, where evidence may only exist for a short period of time. Short-term storage is one of the characteristics of devices in a fog network, and this was possible to see especially on the source nodes in this scenario. In the second experiment, it was observed that the messages did not exist long in the different nodes. Like the first scenario, the messages had a longer average existence in the sink nodes. This makes sink nodes the best nodes to retrieve evidence from regarding time.

DFIF-IoT proposed by Kebande and Ray [30] includes a separate process for IoT forensics in the digital forensics investigation process. The phase is specifically based on cloud forensics, network forensics and device-level forensics. The focus on these types of forensics is relevant for scenario 2 as numerous IoT devices are involved. The network in the scenario is larger than in scenario 1, and all the devices are not necessarily known to the owner. The incident in the scenario is defined as *cybercrime* and the investigation is mainly regarding digital evidence from IoT devices. Focusing on the devices with standardized processes can help ensure the integrity of the evidence and that it is collected in a forensically sound manner. The framework is not developed for fog-IoT systems, but the key aspects of the framework are highly relevant for the scenario. There are possibilities of further development where fog is considered as well.

MCF2I [21] also seems to be an applicable approach for the scenario. The *Aquire/Image* phase in the framework provides a multilevel approach with specific steps and guides for the investigators in regards to collecting evidence from the IoT devices. As with DFIF-IoT [30], it does not consider the network to involve fog computing, and the investigation phase must therefore address the investigation with this in mind as well.

The evidence collection must be performed in a forensically sound manner, and the short-term memory and highly volatile data becomes a bigger challenge in this scenario than in scenario 1. More devices are involved, more data is in in transmission, and there is more widespread service placement on the nodes. Using a framework specifically created for IoT will be highly valuable for this scenario. Without following developed procedures for IoT devices, the evidence is at risk of losing its integrity.

5.3 Scenario 3

In the simulation of scenario 3, it is observed that there is only one module that is placed in the same node twice, all of the other modules are moving between unique nodes. The evidence in the scenario is likely to be stored at multiple locations. The evidence can include both the actual location of the stolen data and the traces of how the intruder got a hold of it. Essential questions the investigators need to consider are:

- How did the intruder get access to the fog network?
- Where was the sensitive data located at the time of the incident?
- How has the data moved through the network?
- What was the source of the data?
- How can evidence be retrieved and still keep its integrity?
- How can the evidence be retrieved with the least impact on the network functionality?

- What is the volatility of data?

5.3.1 Proactive solutions

The use case involving a smart city presented with the FoBI framework [25] is somewhat similar to the network in scenario 3 as well. The smart city use case involves a citywide network that involves a large number of nodes but is not entirely of the same scale as this network. As previously stated, in the use case used in [25], there is an IoT device that is not responding, which is further flagged, and the forensic analyzer module starts examining the activity of the device. If FoBI had been implemented in the fog system in scenario 3 prior to the incident, the activity might have been detected before the attack was launched. The monitoring manager in the framework could *perhaps* detect that there was some unusual behavior coming from the device that was compromised. It is not certain that it would be detected as the attacker might proceed with stealth and imitate normal behavior, making it difficult for FoBI to flag it. If the attacker performed the intrusion without any specific precautions, there is a high chance of seeing abnormalities in traffic. If detected, further traffic from this device could be blocked, but it is not sure that the sensitive data would be safe. It could, however, stop further devices and data from being compromised.

VTA-IH is a relevant framework in this scenario as well [26]. How well it would perform is based on the traffic from the compromised device being above or below the threshold for abnormality. If it were above the threshold, the Evidence Recovery layer would extract essential data from the logs surrounding the suspicious traffic. Initiating the Threat and Attack Handling layer processes could further defend the system and detach the compromised device. It is a high chance of collecting evidence before it is lost based on the real-time analysis of events. The gathered evidence could then be provided to an investigator if needed.

The secure network archives proposed by Duan et al. [27] are relevant in this scenario as well. The model captures all traffic from the IoT devices and focuses on evidence integrity. The model does not initiate any defending processes but can be resourceful in finding information about the intrusion.

The Forensic State Aquisition from IoT proposed by [28] could help provide an indication of there being abnormal behavior in the network by logging the state changes of the device. However, as in scenario 2, this is not guaranteed to be detected. If the incident is already under investigation, the model could help create a timeline of events in the system.

5.3.2 Identification

Scenario 3 is of the largest scale among all of the scenarios used in the experiments, which means that the challenges related to size are highest for this case. It is a live system with more users and nodes involved, which means that the investigation could significantly impact the network.

In the first experiment, it was observed that there are few instances where services are placed on the same node more than once. The placement of the modules/services are more widespread than observed in previous scenarios. There were three nodes with a higher amount of service placements - node 275, 415, and 702. Numerous nodes were not used for service placement, meaning that these nodes are not likely to contain any evidence. In a real network, the three most used nodes might therefore have a higher capacity than the other nodes, or it could be the most optimal node based on the service placement algorithm. However, there are not significant differences between the amounts, and other nodes can therefore be highly relevant. With the network being of large scale, the investigators have a minimal chance of getting all the information needed from the owner of the network. It is highly likely to be challenging to find out which nodes are the most used, and assumptions might be the best solution.

In the second experiment, it was observed that the average source node had a memory of 0.5 GiB, the average sink node had 10.6 GiB, and the average proxy node had 8.9 GiB. These results are close to what was observed in scenario 2, where the sink nodes and proxy nodes had significantly higher capacity than the source nodes. The low capacity of the source nodes indicates that there are more potential low-resource IoT devices used as source nodes. The differences in regards to average *unlink time* between the different node types are even smaller than observed in scenario 2, and the average *erase time* was significantly higher for sink nodes. As observed in the two other scenarios, the proxy nodes had high capacity, but the messages were quickly erased after being unlinked from the application. The investigators have less chance of finding any evidence still existing in the proxy nodes or the source nodes. As the scale and geographical distribution of nodes in the network is larger in this scenario, it becomes extremely difficult to have an overview of all the nodes in the network. The identification phase of the investigation will therefore be demanding in terms of locating relevant nodes quickly. In the scenario, there is an intruder with access to the network, and if there are possibilities of causing more damage, the investigators must take this into consideration. This will impact the process as it might be necessary to take the network or parts of the network offline to prevent further compromise. A set of standard operating procedures (SOP) should be in place before the first responder proceeds with the investigation process, and both *evidence integrity* and *chain of custody* must be considered at this point [17].

Huang et al. analyzed forensic approaches and challenges within vehicular fog computing in [24] and mentioned how it is not practical to physically investigate all of the deployed nodes in large-scale networks. One of the digital forensic approaches mentioned in the article is an *evidence-based digital forensic approach*, which is applicable in most fog networks, not only vehicular fog systems. To locate the possible compromised fog nodes, the process should start by forensically analyzing artifacts from devices or nodes that are believed to be compromised. To do so, there must exist a system on the node initially that can detect abnormal behavior, such as the mechanisms presented by Al-Masri et al. [25] and Bandil and Al-Masri [26].

Further investigation can lead to the compromised device or node and ensure that it gets restricted access to the network. Monitoring the believed compromised device or node could further lead to evidence of the incident [24]. The topology of a vehicular fog network can be similar to the network in this scenario, and with a compromised device, this approach is highly relevant. The investigator should assess whether it is more beneficial to keep the device connected to the network and generate evidence or if it should be completely disconnected in order to prevent further damage. This is a critical consideration as the network in the scenario is a live system [17].

DFIF-IoT [30] was discussed to not be particularly relevant in the first scenario, but more relevant in the second scenario due to the type of the crime and scale of the network. The focus on the devices in the proactive process is highly relevant for this scenario as well because of the numerous IoT devices involved. It could be crucial to use the framework or a framework of a similar type in the investigation, as the scale of the network is immense, and there are several unknown IoT devices. The incident in the scenario is defined as *cybercrime*, and there is more IoT data that needs to be identified than in the previous scenarios. The investigators are looking for any evidence that can help identify the incident and locate possible sources of evidence. Based on this and the scale of the network, the identification phase could therefore benefit from using DFIF-IoT [30] or another holistic framework of a similar approach. MCF2I [21] is also relevant in this scenario. Similar to the other scenarios, the multilevel approach would be beneficial for the investigators in providing guidelines and procedures to prepare for an IoT investigation and identify the possible sources of evidence.

It is highly likely that the identification phase would gain multiple benefits from these new frameworks. As mentioned, the traits of fog computing are not considered in some of the specific frameworks. As the network in this scenario is the largest of the three, it will be nearly impossible to have an overview of all the nodes that exist. The approach mentioned by Huang et al. [24] seems to be one of the few that considers fog networks in the digital forensic process. The challenge of the dynamic service movement will be one of the most prominent issues for the investigator in this scenario, as there are many nodes involved. Identifying the possible sources of evidence will be an incredibly difficult task.

5.3.3 Collection

Similar to the previous scenario, the short-term storage was possible to see especially on the source nodes in this scenario. In the second experiment, it was observed that the messages did not exist long in the different nodes. As with the two other scenarios, the messages had a longer average existence in the sink nodes. This makes sink nodes the best nodes to retrieve evidence from regarding time.

DFIF-IoT [30] includes a separate process for IoT forensics in the digital forensics investigation process. The focus on the types of forensics is relevant for scenario 3 because of the involvement of IoT devices. The network in the scenario is large and the investigators can not be provided with information about all of the devices. The incident in the scenario is defined as *cybercrime* and the investigation is mainly regarding digital evidence from IoT devices. The benefits of having standardized processes can help ensure the integrity of the evidence and that it is collected in a forensically sound manner. The framework is not developed for fog-IoT systems, but the key aspects of the framework are highly relevant for the scenario, similar to scenario 2. Due to the scale of the network, it is arguably more relevant to use a framework such as DFIF-IoT [30] as a majority of the evidence will be related to the IoT devices, if not all of it. This means that a framework like MCF2I [21] is also relevant in the way that it provides specific steps and guides for the investigators in regard to collecting evidence from the IoT devices.

Short-term memory and highly volatile data become an even more significant challenge in this scenario compared to the two previous scenarios. The number of devices involved, data in transmission, and widespread service placement on the nodes make this the most complex network to retrieve data from. Using a framework created explicitly for IoT will be highly valuable for this scenario, but the key characteristics of fog computing are crucial to consider in the process. Without procedures for IoT devices placed in a fog network, there may be little to no success in retrieving evidence, and it is also at risk of losing its integrity.

5.4 Summary

From the analysis and discussion presented in this chapter, it has been possible to understand more about different frameworks and approaches within fog-Iot forensics. To summarize the main points from the analysis, the frameworks, along with their ability to be used in each scenario have been added to Table 5.1. The x symbolizes where the framework could be used.

Frameworks (by author)	Scenario 1	Scenario 2	Scenario 3
Al-Masri et al. [25]		X	X
Bandil and Al-Masri [26]		X	X
Duan et al. [27]	X	X	X
Meffert et al. [28]	X	X	X
Huang et al. [24]			X
Kebande et al. [30]		X	X
Salama et al. [21]	X	X	X

Table 5.1: Beneficial frameworks for each scenario

It is possible to see that the type of scenario and scale of the network affects the applicability of the frameworks. Scenario 3 consists of the most complex network and is the only scenario where all of the mentioned approaches were discussed to be relevant. Similarly, Scenario 2 could benefit from most of the frameworks. Scenario 1 stands out, where most of the discussed approaches were not particularly relevant. This does not mean that they are not applicable, but rather that the investigation would not gain any significant benefits from the framework.

Chapter 6

Discussion, conclusion and future work

The previous chapters have shown the results of the literature study in terms of theory and related work that was important for the research in this thesis. Further, the methodology for answering the research questions was presented. The following chapter introduced the experiments and then presented the results of the performed experiments with each of the scenarios. The previous chapter presented the discussion on the results in the context of digital forensics. This chapter presents a discussion on the SPP, the theoretical implications, and the practical considerations of the thesis. Lastly, the thesis results are concluded, and areas of future research are suggested.

6.1 Discussion on the SPP

The previous chapter analyzed and discussed the experiments and the digital forensic aspect. Therefore, the discussion in this section focuses on RQ1 regarding the different types of service placement algorithms that exist for fog computing. The categories used in for approaches were based on the approach to solve the SPP, and consisted of the following four types: **1)** Exact solutions, **2)** Approximations, **3)** Heuristics and **4)** Meta-heuristics [9].

From the related work and the presented selection of approaches, it was possible to see multiple solutions based on the same type of approach. Most of the exact solutions were based on presenting the problem with ILP. Although the solution could provide an optimal solution for the specific problem, the process is very time-consuming compared to other approaches. Less time-consuming approaches can be found within approximation solutions. In this category, there are different types of approaches, where the overall aim is to find a solution that is good enough. It does not provide the optimal solution but will ensure a more efficient way of getting an acceptable solution. An example is the solution provided by Shah-Mansouri and Wong [45], which resulted in reduced computation delay.

Within the category of heuristic strategies, there are efficient methods for solving the SPP. Instead of focusing on an exact solution, these strategies focus on finding a satisfactory solution as fast as possible. With this type of strategy, it was typical to see graph partitioning being used, such as in the algorithm FOGPART proposed by Mann et al. [47] that outperformed an exact solution using ILP.

Lastly, the meta-heuristic solutions reviewed were mostly based on a Genetic Algorithm. This included types that were based on a standard GA, or others that included other elements, such as a GA with PSO proposed by Yadav et al. [48]. Such hybrid solutions seem to be an area of opportunity for better solutions, where there are possibilities of outperforming the results of the standard GA.

To summarize, there are multiple ways of solving the service placement problem, where one way of categorizing them is based on the strategy type as presented in this thesis. The "No Free Lunch" theorem [65] is highly relevant to solving the SPP of fog computing because of the vast differences in scale and usage. As seen in the experiments in this thesis, the challenges within the different networks are not entirely the same. They are of the same type, but a service placement algorithm used for the network in scenario 1 would likely not be the optimal solution for scenario 3. Therefore, the continuous development of fog computing requires new and efficient algorithms to ensure that fog networks of all types can deliver their key characteristics.

6.2 Theoretical implications

The overall aim of the thesis was to see how the dynamic service placement in fog computing affects the digital forensics process. The experiments were performed with the approach Analysis by Synthesis, which is a common way to conduct experiments. Based on the literature study, there seems to be no previous work performed in the same manner and with a focus on digital forensics. The data used for the research was specifically developed for the three scenarios, which encompasses a broad area within fog computing with its diversity. Additionally, the second experiment was conducted with a unique model for volatility.

RQ1: *Which types of service placement algorithms exists for fog computing?*

To answer RQ1, a literature study was performed on the topic. Studying previous reviews on the Service Placement Problem provided an understanding of the different types of approaches that exists. There are multiple ways to classify the types of approaches, but the focus in this thesis was on the type of strategy used to solve the SPP. As there are several papers on the topic, the categories used for the classification in this thesis was based on a previous study. It was decided that it was not necessary to create new categories. From these categories, the literature study was conducted with these in mind, and multiple different types of approaches were presented in the Related work-section. Further discussion on these approaches was performed in the previous section of this chapter.

RQ2: *How does the dynamic service placement in fog-IoT systems affect the evidence location and the digital forensic process?*

The conducted experiments synthesized fog networks of different scale, and provided results on how the dynamic service placement could be in a fog network. It was possible to see the services being moved throughout the simulation time. The experiments additionally presented statistics on how the volatility could be for different types of nodes in the network using a recently developed model that has not been tested in research before. This volatility is essential to consider when performing a digital forensic investigation, where the order of volatility must be decided. This is specifically important for the research as most data in fog networks are highly volatile. The experiments tested hypotheses regarding the evidence location in the networks, where it was possible to see that the placement of services matters in regards to where the evidence is stored compared to a centralized cloud network.

The results of the experiments provided quantitative data that could further be used in the analysis of the digital forensic context. The analysis showed that dynamic service movement does affect the digital forensic process, specifically in the identification phase of an investigation where more extensive fog networks are incredibly dynamic and complex.

RQ3: *How applicable are existing digital forensic methods for evidence gathering in fog-IoT systems?*

It was necessary to conduct an extensive literature study on the state-of-the-art fog forensics to answer this research question. As there were not many search results on this topic, it was important to understand the methods used in digital forensics involving cloud and IoT devices. The results of the literature study were further used in the context of the experiment results and discussed in the digital forensics analysis. From this analysis, it was possible to see that there are challenges with fog computing that are not solvable with current methods and frameworks. Multiple challenges also exist within IoT forensics and cloud forensics, but for fog forensics, it is specifically the dynamic service movement and the highly volatile data that is difficult to manage.

6.3 Practical considerations

All of the tools used are open source, and they are easily available if the experiments were to be reproduced. Once the documentation of the simulator was read, and after testing it a few times, YAFS [35] was relatively easy to use with the examples provided by the developers. The first attempts at customizing the simulator to the scenarios were not completely successful, and it is recommended to understand all of the components of YAFS [35] before further changing the source code. Testing the simulator with the different examples provided made it easier to understand the functionality in practice. The post-simulation files provide a lot of information, and it is recommended to go in-depth in understanding how the different values are calculated and what each attribute represents.

The experiments were limited to a specific amount of custom processes for movement in the network, and with fewer restrictions, there could have been even more movement.

The experiments were conducted with a dataset in JSON-format that was explicitly created for the scenarios. The specifications in the dataset were based on typical capacity of different entities in a fog network, such as different types of IoT devices, gateways and routers. It should therefore be considered that the data is not created from a deployed network or existing dataset.

6.4 Conclusion

This thesis has looked into fog computing and the challenges with digital forensics. While the closely related topics of cloud forensics and IoT forensics have been discussed in numerous papers, there is little research done on fog forensics. With the dynamic service movement and highly volatile data, there are different challenges that must be tackled.

By creating three different scenarios, fog networks of different scales were presented in the conducted experiments. From the results, it was possible to see the differences in service movement in the scenarios. When the scale of the network increased, the number of service placements on each node decreased. The movement of these services makes it more challenging to retrieve evidence in the fog nodes as it is first necessary to locate it. With smaller networks, such as in scenario 1, it is easier to locate the evidence as the process of identifying the nodes is not as demanding as it is in more extensive networks - such as in scenario 2 and 3.

Specific frameworks that focus on fog-IoT forensics are mainly proactive solutions where the modules are implemented on the fog nodes, such as FoBI [25] and VTA-IH [26]. These are aimed at detecting abnormalities in the network and preventing incidents from occurring in the first place. While this might be reasonable solutions in some cases, it might not be applicable in all fog-IoT networks.

There is a lack of practical digital forensic frameworks applicable to fog-IoT networks. It is therefore necessary to develop such frameworks that consider fog. The holistic frameworks presented by Kebande et al. [30] and Salama et al. [21] are created with IoT devices as the main focus of an investigation process. They do, however, not consider fog networks. They might not be the optimal framework for fog-IoT, but it is a step in the right direction.

6.5 Future work

Based on the experimental results presented in this thesis and the discussion on digital forensics involving fog-IoT, there are some research areas that are proposed for further research.

Experiment based on the structure of a deployed fog network The network structure in this thesis was not based on a deployed fog network but rather on assumptions and theory. Future research on the topic could analyze a deployed fog network and use the observed details as the base of the structure for creating a network with a simulator.

Experiment with a deployed fog system As this thesis was based on analysis by synthesis, further research should perform a similar type of experiment on a real, deployed fog network. This would be a less controlled environment than what was possible in this thesis, but could provide more realistic results. This would ensure that the results of the service movement approach is observed and that the volatility of the entities is real.

Implement and test the different frameworks/models created for fog-IoT forensics The forensic aspect in this thesis was based on the results of the experiments, theory and previous work. Future research should test the discussed frameworks in practice to see how they would perform when attempting to conduct an investigation process.

Experiment with the implementation of service placement in the context of digital forensics Future research on the topic can experiment with the different types of service placement approaches in the context of digital forensics and see if there are any differences or changes in impact on the investigation process.

Experiment with SDN Software-Defined Networking was briefly introduced in this thesis, but it is a highly relevant topic within fog computing and forensics. SDN is likely to be used more with implementations of fog networks in the future based on the benefits regarding privacy, security, and latency. Further work should investigate how a fog network with SDN affects the digital forensics process.

Experiment with different fog simulators The experiment in this thesis was done using YAFS [35], and future research could perform similar experiments with different simulators for fog computing and see if there are any significant differences. This would be relevant for future research when the researcher has the option to choose between the many different simulators that exists for fog computing.

Bibliography

[1] N. Sunde, I. M. Sunde, P. C. Bjelland, J.-P. Sandvik, K. Franke and L. Øverlier, *Cyber Investigations*, A. Årnes, Ed., ser. Compendium. NTNU, CYBER CRIME INVESTIGATION COURSE, 2021, vol. DRAFT V0.708.

[2] *What is edge computing?* Oct. 2021. [Online]. Available: `https://www.cisco.com/c/en/us/solutions/computing/what-is-edge-computing.html#~benefits`.

[3] Y. Wang, T. Uehara and R. Sasaki, 'Fog computing: Issues and challenges in security and forensics,' Jul. 2015. DOI: `10.1109/compsac.2015.173`.

[4] B. Furht and A. Escalante, *Handbook of cloud computing*. Springer, 2010.

[5] P. Mell and T. Grance, 'The NIST definition of cloud computing,' U.S. Department of Commerce, Washington, D.C., Tech. Rep. NIST Special Publication 800-145, 2011. DOI: `10.6028/NIST.SP.800-145`.

[6] M. Iorga, L. Feldman, R. Barton, M. J.Martin, N. Goren and C. Mahmoudi, 'Fog computing conceptual model,' U.S. Department of Commerce, Washington, D.C., Tech. Rep. National Institute of Standards and Technology Special Publication 500-325, 15 pages (March 2018), 2018. DOI: `10.6028/NIST.SP.500-325`.

[7] *What is edge computing*. [Online]. Available: `https://www.ibm.com/cloud/what-is-edge-computing`.

[8] *Difference between edge computing and fog computing*, Nov. 2021. [Online]. Available: `https://www.geeksforgeeks.org/difference-between-edge-computing-and-fog-computing/`.

[9] F. A. Salaht, F. Desprez and A. Lebre, 'An overview of service placement problem in fog and edge computing,' pp. 1–43, 2019. [Online]. Available: `https://hal.inria.fr/hal-02313711/document`.

[10] E. Marin-Tordera, X. Masip, J. Garcia Almiñana, A. Jukan, G.-J. Ren, J. Zhu and J. Farre, 'What is a fog node a tutorial on current concepts towards a common definition,' Nov. 2016.

[11] I. Lera, C. Guerrero and C. Juiz, 'YAFS: A simulator for IoT scenarios in fog computing,' *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019. DOI: `10.1109/access.2019.2927895`.

[12] A. Alomari, S. K. Subramaniam, N. Samian, R. Latip and Z. Zukarnain, 'Resource management in sdn-based cloud and sdn-based fog computing: Taxonomy study,' *Symmetry*, vol. 13, no. 5, 2021, ISSN: 2073-8994. DOI: 10.3390/sym13050734. [Online]. Available: https://www.mdpi.com/2073-8994/13/5/734.

[13] E. Ahvar, S. Ahvar, S. M. Raza, J. Manuel Sanchez Vilchez and G. M. Lee, 'Next generation of sdn in cloud-fog for 5g and beyond-enabled applications: Opportunities and challenges,' *Network*, vol. 1, no. 1, pp. 28–49, 2021, ISSN: 2673-8732. DOI: 10.3390/network1010004. [Online]. Available: https://www.mdpi.com/2673-8732/1/1/4.

[14] Y. Karim and R. Hasan, 'Securing the fog using software-defined networking: A study of challenges, approaches, and open problems,' in *2020 SoutheastCon*, 2020, pp. 1–8. DOI: 10.1109/SoutheastCon44009.2020.9249658.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, 'Openflow: Enabling innovation in campus networks,' *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008. DOI: 10.1145/1355734.1355746.

[16] *Digital forensics*. [Online]. Available: https://www.interpol.int/How-we-work/Innovation/Digital-forensics.

[17] S. Axelsson, P. C. Bjelland, A. Dilijonaite, A. O. Flaglien, K. Franke, J. Hamm, J.-P. Sandvik and I. M. Sunde, *Digital Forensics*, A. Årnes, Ed. Wiley, 2018.

[18] B. Manral, G. Somani, K.-K. R. Choo, M. Conti and M. S. Gaur, 'A systematic survey on cloud forensics challenges, solutions, and future directions,' *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–38, 2019. DOI: 10.1145/3361216.

[19] M. Herman, M. Iorga, A. M. Salim, R. H. Jackson, M. R. Hurst, R. Leo, R. Lee, N. M. Landreville, A. K. Mishra, Y. Wang and et al., 'NIST cloud computing forensic science challenges,' 2020. DOI: 10.6028/nist.ir.8006.

[20] S. Mrdovic, 'Iot forensics,' in *Avoine, G., Hernandez-Castro, J. (eds) Security of Ubiquitous Computing Systems*. Springer, Cham, 2021. DOI: 10.1007/978-3-030-10591-4_13.

[21] U. Salama, L. Yao and H.-Y. Paik, 'A multilevel collective framework for internet of things digital forensic investigation,' *Computer*, vol. 55, no. 2, pp. 44–53, 2022. DOI: 10.1109/MC.2021.3095492.

[22] R. Hegarty and M. Taylor, 'Digital evidence in fog computing systems,' *Computer Law & Security Review*, vol. 41, p. 105 576, 2021, ISSN: 0267-3649. DOI: https://doi.org/10.1016/j.clsr.2021.105576. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0267364921000492.

[23] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury and V. Kumar, 'Security and privacy in fog computing: Challenges,' *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017. DOI: 10.1109/access.2017.2749422.

[24] C. Huang, R. Lu and K.-K. R. Choo, 'Vehicular fog computing: Architecture, use case, and security and forensic challenges,' *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, 2017. DOI: 10.1109/MCOM.2017.1700322.

[25] E. Al-Masri, Y. Bai and J. Li, 'A fog-based digital forensics investigation framework for IoT systems,' in *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, IEEE, Sep. 2018. DOI: 10.1109/smartcloud.2018.00040.

[26] A. Bandil and E. Al-Masri, 'VTA-IH: A fog-based digital forensics framework,' in *2020 6th International Conference on Science in Information Technology (ICSITech)*, IEEE, Oct. 2020. DOI: 10.1109/icsitech49800.2020.9392064.

[27] H. Duan, Y. Zheng, C. Wang and X. Yuan, 'Treasure collection on foggy islands: Building secure network archives for internet of things,' *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2637–2650, 2019. DOI: 10.1109/JIOT.2018.2872461.

[28] C. Meffert, D. Clark, I. Baggili and F. Breitinger, 'Forensic state acquisition from Internet of Things (FSAIoT),' *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017. DOI: 10.1145/3098954.3104053.

[29] E. Oriwoh and P. Sant, 'The forensics edge management system: A concept and design,' in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, 2013, pp. 544–550. DOI: 10.1109/UIC-ATC.2013.71.

[30] V. R. Kebande and I. Ray, 'A Generic Digital Forensic Investigation Framework for Internet of Things (IoT),' in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016, pp. 356–362. DOI: 10.1109/FiCloud.2016.57.

[31] *Iso/iec 27043:2015*, Sep. 2020. [Online]. Available: https://www.iso.org/standard/44407.html.

[32] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh and R. Buyya, 'Ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments,' *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. DOI: https://doi.org/10.1002/spe.2509. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2509. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509.

[33] C. Sonmez, A. Ozgovde and C. Ersoy, 'Edgecloudsim: An environment for performance evaluation of edge computing systems,' *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, e3493, 2018, e3493 ett.3493. DOI: `https://doi.org/10.1002/ett.3493`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3493`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3493`.

[34] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid and S. U. Khan, 'Fognetsim++: A toolkit for modeling and simulation of distributed fog environment,' *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018. DOI: `10.1109/ACCESS.2018.2877696`.

[35] I. Lera, C. Guerrero and C. Juiz, *YAFS*, `https://github.com/acsicuib/YAFS`, 2019.

[36] M. S. Raghavendra, P. Chawla and A. Rana, 'A survey of optimization algorithms for fog computing service placement,' in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2020, pp. 259–262. DOI: `10.1109/ICRITO48877.2020.9197885`.

[37] R. Mahmud, K. Ramamohanarao and R. Buyya, 'Application management in fog computing environments: A taxonomy, review and future directions,' vol. 53, no. 4, 2020, ISSN: 0360-0300. DOI: `10.1145/3403955`. [Online]. Available: `https://doi.org/10.1145/3403955`.

[38] S. Smolka and Z. Á. Mann, 'Evaluation of fog application placement algorithms: A survey,' *Computing*, Feb. 2022. DOI: `10.1007/s00607-021-01031-8`.

[39] C. Guerrero, I. Lera and C. Juiz, 'Genetic-based optimization in fog computing: Current trends and research opportunities,' *Swarm and Evolutionary Computation*, vol. 72, p. 101 094, May 2022. DOI: `10.1016/j.swevo.2022.101094`.

[40] J. Santos, T. Wauters, B. Volckaert and F. De Turck, 'Resource provisioning for iot application services in smart cities,' in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–9. DOI: `10.23919/CNSM.2017.8255974`.

[41] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski and P. Leitner, 'Optimized iot service placement in the fog,' *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017. DOI: `10.1007/s11761-017-0219-8`.

[42] K. Velasquez, D. P. Abreu, M. Curado and E. Monteiro, 'Service placement for latency reduction in the fog using application profiles,' *IEEE Access*, vol. 9, pp. 80 821–80 834, 2021. DOI: `10.1109/ACCESS.2021.3085370`.

[43] P. Kayal and J. Liebeherr, 'Autonomic service placement in fog computing,' in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–9. DOI: 10.1109/WoWMoM.2019.8792989.

[44] R. Yu, G. Xue and X. Zhang, 'Application provisioning in fog computing-enabled internet-of-things: A network perspective,' in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 783–791. DOI: 10.1109/INFOCOM.2018.8486269.

[45] H. Shah-Mansouri and V. W. S. Wong, 'Hierarchical fog-cloud computing for iot systems: A computation offloading game,' *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018. DOI: 10.1109/JIOT.2018.2838022.

[46] A. Brogi and S. Forti, 'Qos-aware deployment of iot applications through the fog,' *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017. DOI: 10.1109/JIOT.2017.2701408.

[47] Z. Á. Mann, A. Metzger, J. Prade and R. Seidl, 'Optimized application deployment in the fog,' *Service-Oriented Computing*, pp. 283–298, 2019. DOI: 10.1007/978-3-030-33702-5_22.

[48] V. Yadav, B. V. Natesha and R. M. R. Guddeti, 'Ga-pso: Service allocation in fog computing environment using hybrid bio-inspired algorithm,' in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, 2019, pp. 1280–1285. DOI: 10.1109/TENCON.2019.8929234.

[49] A. Brogi, S. Forti, C. Guerrero and I. Lera, 'Meet genetic algorithms in monte carlo: Optimised placement of multi-service applications in the fog,' in *2019 IEEE International Conference on Edge Computing (EDGE)*, 2019, pp. 13–17. DOI: 10.1109/EDGE.2019.00016.

[50] N. Mehran, D. Kimovski and R. Prodan, 'Mapo: A multi-objective model for iot application placement in a fog environment,' in *Proceedings of the 9th International Conference on the Internet of Things*, ser. IoT 2019, Bilbao, Spain: Association for Computing Machinery, 2019, ISBN: 9781450372077. DOI: 10.1145/3365871.3365892. [Online]. Available: https://doi.org/10.1145/3365871.3365892.

[51] K. Velasquez, D. Perez Abreu, M. Curado and E. Monteiro, 'Service placement for latency reduction in the internet of things,' *Annals of Telecommunications*, vol. 72, Jun. 2016. DOI: 10.1007/s12243-016-0524-9.

[52] W. A. Edmonds and T. D. Kennedy, 'Chapter 16 | embedded approach,' in *An applied guide to research designs: Quantitative, qualitative, and mixed methods*. Sage, 2017.

[53] C. Wohlin, 'Guidelines for snowballing in systematic literature studies and a replication in software engineering,' *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 2014. DOI: 10.1145/2601248.2601268.

[54] F. Servida and E. Casey, 'IoT forensic challenges and opportunities for digital traces,' *Digital Investigation*, vol. 28, 2019. DOI: 10.1016/j.diin.2019.01.012.

[55] J.-P. Sandvik, I. Lera, C. Guerrero and C. Juiz, *YAFS*, https://github.com/jenspets/YAFS/tree/volatility, 2022.

[56] I. Lera and C. Guerrero, *Documentation for YAFS*, 2017. [Online]. Available: https://yafs.readthedocs.io/en/latest/contents.html.

[57] *Venv - creation of virtual environments*. [Online]. Available: https://docs.python.org/3/library/venv.html.

[58] *Atom*. [Online]. Available: https://atom.io/.

[59] M. Bastian, S. Heymann and M. Jacomy, *Gephi: An open source software for exploring and manipulating networks*, 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

[60] A. A. Hagberg, D. A. Schult and P. J. Swart, 'Exploring network structure, dynamics, and function using networkx,' in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.

[61] *Binomial heap*, Nov. 2020. [Online]. Available: https://www.geeksforgeeks.org/binomial-heap-2/.

[62] A.-L. Barabási, *Network Science - The Barabási-Albert model*, 2014. [Online]. Available: https://barabasi.com/f/622.pdf.

[63] K. Hartmann, J. Krois and B. Waske, *Critical value and the p-value*, 2018. [Online]. Available: https://www.geo.fu-berlin.de/en/v/soga/Basics-of-statistics/Hypothesis-Tests/Introduction-to-Hypothesis-Testing/Critical-Value-and-the-p-Value-Approach/index.html.

[64] A. Jones and S. Vidalis, 'Rethinking digital forensics,' *Annals of Emerging Technologies in Computing*, vol. 3, no. 2, pp. 41–53, 2019. DOI: 10.33166/AETiC.2019.02.005. [Online]. Available: http://aetic.theiaer.org/archive/v3/v3n2/p5.html.

[65] *What is no free lunch theorem*, Jun. 2021. [Online]. Available: https://www.geeksforgeeks.org/what-is-no-free-lunch-theorem/.

Appendix A

Additional Material

A.1 Service placements from Scenario 1

All nodes used for placement of the specific Service/Module, values separated by spaces

Service/Module 0: 0 8 5 8 9 3 2 1 8 1 3 1 8 4 1 5
Service/Module 1: 0 9 0 8 9 5 8 9 0 4 8 7 9 1
Service/Module 2: 0 2 1 3 0 5 1 8 9 5 2 7 3 9 7 4
Service/Module 3: 0 4 5 7 6 1 3 4 8 6 3 2 8 7 1 3
Service/Module 4: 0 1 7 8 7 3 8 1 5 7 2 1 6 9 3
Service/Module 5: 0 1 8 4 1 5 1 7 9 0 1 3 5 3
Service/Module 6: 0 5 6 7 0 3 7 4 3 4 2 4 6 5 0

A.2 Service placements from Scenario 2

All nodes used for placement of the specific Service/Module, values separated by space

Service/Module 0: 0 4 7 20 1 31 50 72 68 14 2 48 55 82 76 15
Service/Module 1: 0 2 56 61 31 43 68 0 78 48 52 13 65 59 3
Service/Module 2: 0 24 25 15 72 4 21 1 39 41 19 5 22 56 68 59
Service/Module 3: 0 23 51 1 19 24 34 52 2 14 39 72 12 14 7 76
Service/Module 4: 0 15 12 44 70 54 0 54 36 34 1 81 63 74 64 69
Service/Module 5: 0 62 73 40 5 54 4 24 61 5 69 16 68 50 73 25
Service/Module 6: 0 26 34 57 17 0 5 71 33 26 38 76 73 83 27 57
Service/Module 7: 0 7 65 66 62 73 42 54 58 73 44
Service/Module 8: 0 2 23 33 43 29 20 16 12 36 17 4 40 43 52 8
Service/Module 9: 0 70 21 68 46 58 64 1 36 72 43 38 43 19 51 14
Service/Module 10: 0 55 61 73 37 31 58 41 10 19 76 38 39 4 29 24
Service/Module 11: 0 12 5 20 59 49 63 27 51 47 35 19 9 63 38 47
Service/Module 12: 0 34 46 22 10 58 0 25 5 22 35 53 21 1 34
Service/Module 13: 0 8 60 53 36 20 8 10 23 5 26 20 39 12 21

A.3 Service placements from Scenario 3

All nodes used for placement of the specific Service/Module, values separated by space

Service/Module 0: 0 112 483 603 140 349 476 391 482 827 506 601 498 887 281 719
Service/Module 1: 0 628 707 519 96 508 531 356 522 192 586 422 520 264 358 870
Service/Module 2: 0 366 530 452 144 79 298 638 254 535 229 555 324 557 85 856
Service/Module 3: 0 169 701 238 581 629 324 68 212 152 707 621 64 278 876 600
Service/Module 4: 0 31 366 727 573 368 581 722 795
Service/Module 5: 0 441 716 467 369 883 744 658 360 531 692 558 601 101 206 303
Service/Module 6: 0 591 273 901 574 289 112 119 327 89 445 212 610 429 577 325
Service/Module 7: 0 408 715 896 485 762 591 377 459 342 620 4 831 535 0 426
Service/Module 8: 0 475 209 871 708 364 526 598 841 700 436 416 766 106 626 564
Service/Module 9: 0 77 355 624 574 394 409 487 387 648 111 774 446 165 358 275
Service/Module 10: 0 658 172 660 7 510 702 465 427 158 634 139 157 298 205 178
Service/Module 11: 0 728 36 566 58 318 399 426 562 422 0 480 607 577 336 657
Service/Module 12: 0 725 814 275 571 802 95 132 206 615 678 316 862 885 731 532
Service/Module 13: 0 698 385 449 303 377 634 848 6 259 33 68 605 631 726 874
Service/Module 14: 0 79 400 702 415 237 512 275 487 164 506 585 580 354 852 287
Service/Module 15: 0 797 741 492 833 234 346 181 521 696 529 897 136 354 322 860
Service/Module 16: 0 436 664 415 235 231 191 204 123 233 452 89 526 340 221 246
Service/Module 17: 0 558 526 175 490 796 41 392 182 498 454 877 683 257 741 143
Service/Module 18: 0 744 118 172 70 281 234 202 513 480 63 225 903 35 713 729
Service/Module 19: 0 568 113 521 676 122 457 447 131 290 248 215 459 843 117 125
Service/Module 20: 0 141 420 270 420 294 699 695 493 786 293 723 396 14 727 712
Service/Module 21: 0 173 157 849 618 853 269 533 580 171 681 390 393 529 791 615
Service/Module 22: 0 156 417 415 836 673 232 478 260 702 663 502 270 765 629 673
Service/Module 23: 0 212 151 26 374 778 791 612 843 852 703 80 888 603 897 230
Service/Module 24: 0 168 100 79 874 649 404 392 466 638 702 67 28 32 181 332
Service/Module 25: 0 233 5 162 468 204 249 1 637 48 610 215 285 848 170 476
Service/Module 26: 0 30 240 170 70 230 763 137 432 351 325 47 890 131 766 265
Service/Module 27: 0 538 619 823 775 483 238 870 415 125 155 771 24 253 275 693
Service/Module 28: 0 138 313 530 886 443 499 632 465 893 167 159 264 194 161 611
Service/Module 29: 0 839 110 878 109 364 473 215 575 196 614 650 492 655 35 548
Service/Module 30: 0 505 612 385 271 758 49 549 518 400 750 897 498 683 807
Service/Module 31: 0 366 109 801 718 513 1 8 676 808 701 552 158 384 131 82
Service/Module 32: 0 639 751 636 203 433 318 868 567 509 898 430 233 826 399
Service/Module 33: 0 754 371 678 884 368 850 94 677 824 346 21 631 779 14 516
Service/Module 34: 0 294 611 54 309 391 693 345 127 452 44 150 478 22 222 647
Service/Module 35: 0 734 10 577 844 728 442 0 145 670 885 208 112 395 283 293
Service/Module 36: 0 813 493 137 147 312 394 232 108 144 650 544 45 647 332 8
Service/Module 37: 0 343 651 661 33 860 205 83 588 401 201 476 847 76 410 78
Service/Module 38: 0 701 87 849 503 677 842 530 80 487 137 116 378 347 478 681
Service/Module 39: 0 121 524 398 494 493 322 426 347 307 500 423 348 61 682 49
Service/Module 40: 0 822 325 419 113 522 94 4 792 780 651 660 447 509 798 57
Service/Module 41: 0 876 464 21 555 903 182 622 239 854 196 404 395 568 842 430
Service/Module 42: 0 859 694 342 94 9 492 42 389 691 777 505 91 270 563 321
Service/Module 43: 0 419 799 104 853 383 225 760 421 60 598 555 644 69 235 449
Service/Module 44: 0 792 838 520 141 93 123 843 742 347 866 772 119 118 718 254
Service/Module 45: 0 884 378 213 403 148 352 239 711 792 909 633 541 140 451 896
Service/Module 46: 0 268 599 580 683 411 515 29 139 369 571 812 624 813 299 32
Service/Module 47: 0 162 834 855 733 891 334 551 343 192 218 726 749 587 315 23
Service/Module 48: 0 350 723 40 163 589 484 785 24 489 248 377 224 633 804 809

Service/Module 49: 0 644 631 88 625 799 417 828 458 37 874 15 244 655 86 420
Service/Module 50: 0 514 661 604 602 132 157 621 885 534 626 686 328 557 679 294

Julie Hodne Gundersen

Digital forensics on fog-based IoT devices