NTNU
Norwegian University of
Science and Technology

Trust Management in Fog Computing

Tuva Selstad Dybedokken

Master of Science in Communication Technology
Submission date: June 2017
Supervisor: Frank Alexander Krämer, IIK
Co-supervisor: Colin Alexander Boyd, IIK
Anders Eivind Braaten, IIK
David Palma, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

NTNU – Trondheim
Norwegian University of
Science and Technology

Trust Management in Fog Computing

Tuva Dybedokken

Submission date: June 2017
Responsible professor: Frank Alexander Kraemer, IIK
Supervisor: Anders Bråten and Colin Alexander Boyd, IIK

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Title: Trust Management in Fog Computing

Student: Tuva Dybedokken

Problem description:

The focus of this thesis will be issues related to trust management in fog computing.

Healthcare technology is one domain where "Internet of things" (IoT) has great potential. Wireless devices connected to the Internet are foreseen to have a significant impact on the treatment of patients. With continuous monitoring, caregivers gain a better overview of the whole health situation.

Today, processing of the data from the IoT devices often happens in the cloud. Many of the cloud servers are centralized. When more devices become connected, several problems may occur. Latency sensitive health applications will not work ideally, because of the distance. Problems like network delay and jitter will increase, due to the significant amount of data produced by the connected devices.

Because of these problems, the fog computing architecture has been introduced. The fog computing architecture consists of a range of devices, called nodes, located at the network's edge, between the devices producing data and the cloud servers. It can provide computation, storage, and networking. In fog computing, analysis of data can happen anywhere from the edge of the network to the core. In short fog computing offloads the cloud servers. Several issues are yet to be solved in fog computing.

The focus of this thesis will be issues related to trust management between an IoT device and fog node. Trust is more than security; it also includes characteristics like reliability and availability. Trust ensures user privacy and information security. This is especially important when dealing with health related data, as the information exchanged is highly personal. In our offline life, we are trained to see who and what we can trust. Using the same hints will not work in the fog computing architecture, which is why trust management is needed. Information should only be accessible to those who have rights to it. Damage may occur to patients if information is altered by a rogue node, for example. One of the goals of trust management is to make the user confident enough to use the resources offered by the IT service. Ideally, IoT devices should be able to connect to an arbitrary fog node, to perform computation and other services, and be sure that information delivered will not be misused. Fog nodes should also be able to accept IoT device, and be confidant its policy will be held. To do this trust between the device and the fog node must be established

The basis for this thesis will be a literature study, and an analysis of the existing technologies, like IoT, Cloud, Edge/Fog Computing and Trust Management. This will create the foundation for the further discussion of unsolved issues, research challenges, and future research directions regarding trust management in fog computing, between an IoT device and fog node. Specific use cases will be found in the healthcare domain, while trust management solutions in other domains will be used to gain insight into the relevant challenges. Solutions from these domains may not be applicable because of the resource constraints of the fog nodes.

Responsible professor: Frank Alexander Kraemer, IIK
Supervisor: Anders Bråten and Colin Alexander Boyd, IIK

Abstract

With the rise of fog computing, computation is moved away from a purely centralized approach. In the centralized approach devices have direct contact with the cloud servers. In fog computing allows for processing data along the path from the edge to the core of the network. The introduction of fog will have several benefits, such as reduced latency and increased reliability. The network topology changes continually in a fog computing network. New objects will join and leave. Since fog computing is a new field of study, there exists several open issues, one of them is how to establish trust. Trust is the level of assurance that an object will behave in a satisfactory manner. Trust can also be used to form autonomous collaboration.

This thesis explores the need for trust management in fog computing. Through a literature review, specific tasks for trust management in fog computing have been identified. Trust can provide assistance in the decision making, and allow autonomous connections to be established between fog nodes and resource constrained IoT devices, called fog clients. Requirements for the trust in fog computing has also been defined. Additionally, misbehaving nodes can be detected in a network as well. By surveying existing solutions in other domains, requirements for trust is fog computing were defined. Based on these requirements a trust management solution for fog computing has been devised. Finally, two main open issues are identified and discussed.

Sammendrag

Med inntoget av «fog computing» har det igjen blitt ett fokus på desentraliserte løsninger. Vi beveger oss vekk fra de sentraliserte sky tjenestene, og tilbyr prosessering på kanten av nettverket. Tjenester fordelt i et nettverk tilbyr distribuerte prosesseringsmuligher. Dette betyr at nettverketstopologi forandrer seg hele tiden. Nye noder vil bli koblet av og på kontinuerlig. Siden «fog computing» er ett nytt område er det flere uløste problemer, hvordan skape tillit mellom noder i nettverket er blant disse problemene. Tillit kan brukes for å starte samarbeid. Tillit tilsier hvor sikker man er på at ett annet objekt oppfører seg på ønsket måte.

Denne masteroppgaven undersøker behovet for tillit i et slik desentralisert nettverk. Gjennom en litteraturstudie har oppgaver for tillit blitt identifisert. Tillit kan brukes til beslutningstaking, og tillate autonome samarbeid mellom noder og enheter med lite prosesserings muligheter, også kalt «fog» klienter. Enheter som oppfører seg dårlig kan bli oppdaget kan gjennom et tillits system. Ved et studie av løsninger i andre domener, har det blitt spesifisert krav til tillit systemet. Basert på disse kravene har et tillits system for «fog computing» blitt utarbeidet. To åpne problemer knyttet til tillit i «fog computing» har blitt identifisert og diskutert.

Preface

This thesis is submitted as the final part of my master's degree at the Department of Information Security and Communication Technology at the Norwegian University of Science and Technology.

I would like to thank my supervisor Anders Bråten for his guidance and support throughout this final semester. Without his rapid and valuable feedback, this would not have been possible. I would also like to thank my Professor Frank Alexander Kraemer for his thoughts and advice which have helped form this thesis.

Thank you to Kjetil Aune and my father Bjørn Dybedokken for proof-reading this thesis. A special thanks to Marit Rudlang for staying with me to the very end, you have made this a great experience.

Finally, I would like to thank all the amazing people I have met during these years in Trondheim, it would not have been the same without you.

Trondheim, 12th of June 2017

Tuva Selstad Dybedokken

Contents

List of Figures

List of Tables

List of Acronyms

COI Community of interest.

IoT Internet of Things.

MANET Mobile ad-hoc networks.

MBAN Medical Body Area Network.

NTNU Norwegian University of Science and Technology.

PaaS Processing as a Serivce.

QoS Quality of Service.

RTT Round trip time.

SIoT Social Internet of things.

SP Service Provider.

SR Service Requester.

VI visually impaired.

VM Virtual Machine.

Chapter 1
Introduction

90 % of all existing data has been produced in the last two years [Dra13]. This data consists, for instance, of social media posts or data collected by sensors. Currently, the cloud is being used for storage and processing of all this data. However, several problems occur when everything is placed in the cloud. Bandwidth issues are the most pressing. Fog computing is introduced as a solution to these issues. A shift is happening from the centralized cloud to the distributed fog, with fog computing processing at the edge of the network. However, since fog computing is a new research topic several new challenges arises. One of these challenges is how to establish trust in the distributed fog network. This thesis explores the relative untouched topic of trust management in fog computing.

This chapter will introduce basic terms, more background information will be given in Chapter 2. The motivation for further exploration of trust in fog computing will be given in Section 1.2 and Section 1.3 will define the scope of the thesis.

1.1 Background

Internet of Things (IoT) is not a single technology, but the concept where anything can be accessible from everywhere. The idea is that sensors incorporated in devices sense the world around them. Data collected by the sensing devices is analyzed, processed and stored. Decisions can than be made based on the analyzed data. Hopefully, these decisions are more informed than decisions made today. In short, the IoT consists of the sensors, the data gathered by the sensors and the actions taken on the data.

Offloading is the action of delegating computational tasks to devices with more computational power. Drawing conclusions based on big data collected is a substantial computational task, tasks, which the resource-constrained devices are not able to perform with today's technology. Offloading computations have proven to increase

the performance of the applications running on the IoT devices [HCH+14], both in terms of latency and power consumption.

Edge computing utilizes the processing power located at the edge of the network. The edge of the network is the part of the network that connects the end-users to the network. Edge nodes are devices located close to the end users. Edge computing pushes the processing away from the centralized cloud. Processing the data at the edge implies processing the data physically closer to where the data is being produced by the IoT devices, rather than if the data was sent to the core of the network. Introducing edge computing will mitigate the dependency on the central nodes.

Fog computing

Fog is a subset of edge computing [GGD+]. Fog is a multi-layer architecture stretching from the cloud to the device [CW17]. Several advantages are gained with fog computing, and these are described in greater detail in Section 2.4.4. The goal of fog computing is to move the processing out to the edge of the network closer to where the data is sensed and collected. Fog nodes can be anything with processing power that provides service to fog clients, whereas fog clients utilize the processing power of the fog nodes.

Trust

Trust is the level of assurance that an object will behave in satisfactory manner [LS07]. This behavior can be regarding the quality of service or the security policies that the object has. Trust can help devices that have never encountered each other before establishing a collaboration. Nodes can be deemed safe or not, depending on the trust level. Trust management in the system and mechanisms in place to establish trust between two objects in a network.

1.2 Motivation

Fog computing brings several advantages to IoT. Though the use of fog computing applications can utilize third-party hardware and software to process user data. However, this leads to several interesting security concerns [VRM14]. Security and privacy concerns will lag the development of fog computing if not addressed [YQL15a]. This thesis will focus on one concern within the privacy and security concern, namely how to establish trust in fog computing.

Trust must be present in a network to allow for collaboration among the objects of the network. Without trust, objects will be hesitant to connect, because they will have no assurance about the future behavior of the counterpart.

Fog clients and fog nodes will join and leave networks continuously. Fog objects, fog clients and fog nodes, will constantly encounter unfamiliar objects. Trust management can provide mechanisms for determining whether or not to establish a collaboration with the unfamiliar node.

The flexibility of the deployment scenarios complicates the trust situation of fog computing [YQL15a]. Several vendors will deploy fog nodes. Each vendor may have different security policies [VRM14]. Additionally, fog nodes with the same vendor may also be maintained differently. Maintaining security is a problem in the fog environment [CW17]. The maintenance issue makes fog nodes vulnerable to attacks. Without prior knowledge of an unfamiliar fog node, it is hard for a fog client to assess the security level and behavior of the fog node.

The feasibility of a man-in-the-middle attack has already been proven in fog computing [SW14]. The existence of a rouge node in the network will have an impact on the security and privacy of user data. It is therefore important to be able to identify intruders in the network. Trust management can be used as a basis for intrusion detection systems [SHL08].

1.3 Scope and Objectives

The main objective of this thesis is to explore trust in fog computing. Trust in fog computing is still an open challenge. This thesis will focus on how trust between a fog node and a fog client can be established, and what advantages trust can provide in fog computing. Finally, still open issues will be identified.

A proposed solution is presented. However, this serves as an example for further discussion. Simulations have not been run on this solution.

1.4 Outline

The thesis has the following structure

Chapter 2 Background information about IoT domain and how fog computing fits in this domain. The motivation for the deployment of fog computing is also given here. A specific use case will be introduced to illustrate how trust can be used in fog computing.

Chapter 3 Trust and trust management, in general, are introduced here. Trust computation in IoT service management is studied in depth.

Chapter 4 Requirements and possible attacks for the trust management in fog computing are defined in Chapter 4

Chapter 5 Proposition and evaluation of a solution for trust management in fog computing.

Chapter 6 Open issues with trust in fog computing are identified and discussed.

Chapter 7 Conclusion of the thesis.

Chapter 2
Introduction to Fog Computing

Terms introduced in Section 1.1 will be further described in this chapter. Internet of Things (IoT), Social Internet of Things (SIoT), and finally, fog computing will be explained in detail.

2.1 Internet of things (IoT)

"Internet of things has the potential to change the world as much as the Internet did. Maybe even more so." - Kevin Ashton (2009)

Internet of things as a concept will be introduced in this section. Fog computing was introduced to deal with the ever-increasing number of devices. IoT must be understood to appreciate fog computing.

Gartner defined IoT as *"the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment"* [iot].

From this definition, we can say that the IoT is not a single technology. It is a concept of physical devices sensing the environment around them, and collecting a vast amount of data. Based on this data, applications can make better decisions. IoT consists of three things: small sensors, big data, and actions on this data.

The services that become available through the use of things are the most valuable aspect of the internet of things, not the things themselves [Howb]. The primary goal of IoT is to monitor anything from anywhere [RS17]. When devices are attached to the Internet, more information can be collected and sensed. Connecting devices will, therefore, enrich services. Connected devices produce a significant amount of data; it is by using these data companies hope to improve their services and make smarter decisions.

2.2 Social Internet of Things (SIoT)

This section will introduce Social Internet of Things (SIoT) as a concept. SIoT will be used as an example later when looking at trust in dynamic and distributed networks.

Internet of things and social networks are merging into one concept, called the Social Internet of Things (SIoT) [AIMN12]. Objects in the network form social relationship like humans do in social networks. These relationships ease the establishment of interactions among objects.

Social networks obtain better results than individual entities do alone. Using social networks will, therefore, make IoT achieve better results [AIMN12]: A network of objects collaborating will get better results than individual objects working alone. Relationships between objects in SIoT were defined by L. Atzori, A. Iera and G. Morabito [AIM11]. Relationships in SIoT are inspired by human relationships. Table 2.1 shows the different kind of relationship an object in a network can establish.

Through the relationship an object forms, it can find the service it needs without human intervention. The goal is to allow objects to form relationships and interact autonomously. Each object has an owner, and the owner sets the relationship rules, based on these rules the device forms a relationship to others [AIMN12].

2.3 Current problems of IoT

Today the data collected is stored and processed in the cloud [BMZA12]. Data is sensed by resource constrained devices and sent to centralized data servers for further processing. It is through this process this smart object becomes smart. The process today looks like the workflow in Equation 2.1.

$$capture \rightarrow store \rightarrow analyse \rightarrow act \rightarrow notify \tag{2.1}$$

Cloud servers have near unlimited processing power compared to the resource constrained IoT devices and can, therefore, provide additional resources for these devices [Ida16]. IoT has a connection to the cloud which handles all the processing of the data from these devices. IoT is used as a source for big data while the cloud has the means to process the big data. The capabilities of the cloud and IoT are therefore complementary [BdDPP14].

Storage and computing power has increased at a higher rate than network capacity the last couple of years [RS17]. Once the data is in the cloud centers, there is enough storage and computing power to handle the vast amount of produced data. However, it is getting the data to the cloud centers that is problematic. Bandwidth capacity is

Relationship	Description
Parental Object Relationship	The relationship is between objects that are produced by the same manufacturer in the same period. This relationship is between similar objects, formed during production.
Co-location Object Relationship	Is a relationship between objects always located in the same location. Objects that are always located in the same place can establish a co-location object relationship. An example is between sensors in a smart house, or in a hospital.
Co-work Object Relationship	Is a relationship between objects working on a common IoT application
Owner object relationship	Is a relationship between objects owned by the same owner. If two objects are owned by the same owner, they can form a relationship. For instance, a smartphone and smartwatch owned by the same owner from an owner object relationship.
Social Object Relationship	Is a relationship between objects that are in contact sporadically or continuously. This relationship can be established between objects that are in regular contact with each other, formed because the owners are in regular contact with each other. An example will be your smart watch and the router at your friend's house.

Table 2.1: Summary of the different relationships that can be established between objects in the SIoT defined by L. Atzori, A. Lera and G. Morabito [AIM11]

becoming a limiting factor. An enormous amount of data is feed to the cloud centers, and as of now, the network does not have the capacity to handle it.

Lack of bandwidth causes latency, which is a problem for the majority of IoT applications. Sending data to a remote location will result in unpredictable and significant latency, making it hard to predict the quality of service the application can deliver. Fog computing is being introduced to solve these latency issues.

Reliability is also a major problem. The centralized approach to cloud computing causes a single point of failure. Not being able to access the required resources could stop highly critical applications, such as traffic control or heart rate monitors.

2.4 Fog Computing

In this section, fog computing will be described in detail. So far only a general introduction of fog computing has been provided. A definition of fog computing is the topic of Section 2.4.1. Section 2.4.2 introduces the fog architecture. Different deployment models foreseen for fog computing will be described in Section 2.4.3. Characteristics of fog computing will be covered in Section 2.4.5.

2.4.1 Definition Fog Computing

OpenFog Consortium defines fog as *"a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum."* [CW17]

Fog computing moves the computation to the edge of the network. The goal is to utilize already existing computing resources located at the edge devices to reduce the amount of data sent to the cloud servers [BMZA12]. As defined in the introduction, fog computing is a subset of edge computing. Edge devices used in fog computing are called fog nodes and can be anything from small routers or switches, to high-end servers.

There are several advantages of moving the ability to process data close to the edge, where the data is being generated. Latency is reduced because the propagation time is shorter [BMZA12]. The bandwidth required is reduced since not all data needs to be sent to the cloud, which is located the core of the network [VRM14]. Reliability can be increased [YMSG+14a], by placing a large number of fog nodes at the edge of the network, there is no longer a single point of failure: If one node becomes unavailable, several other nodes can be reached, and redundancy is achieved.

It is important to note that fog is not a replacement for the cloud, but an extension. Data that can be processed at the edge is processed there, and the remaining data is sent to the cloud for long term storage and deeper analysis. Fog computing utilizes the computation of more nodes along the path between the edge and the cloud [VRM14].

2.4.2 Fog Architecture

Fog has a hierarchical architecture. Figure 2.2 shows this architecture. Computation can happen on every level of the hierarchy [CW17]. Different deployment scenarios will have different numbers of layers.

Nodes at different levels have different tasks in the fog [CW17]. Edge nodes focus on data collection, together with command and control of sensors. Higher level

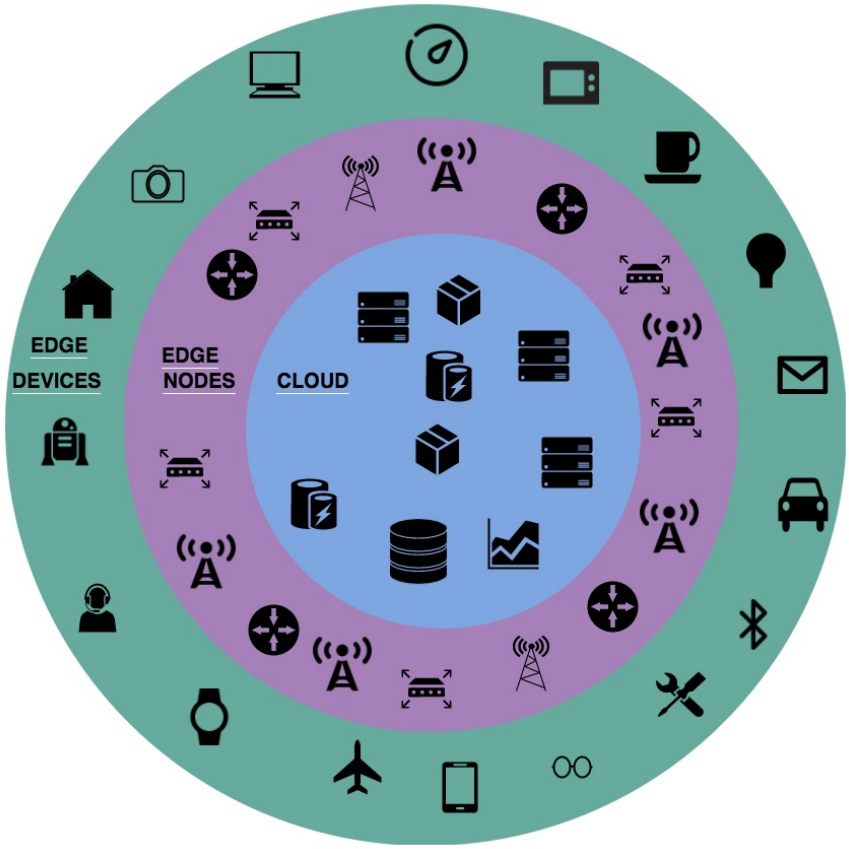


Figure 2.1: Fog computing utilizes the resources at the edge og the network. Here the cloud is in the center of the network, and the edge nodes represent fog nodes that are places further out in the network, close to the edge devices, which can, for instance, be fog clients in need of more resources.

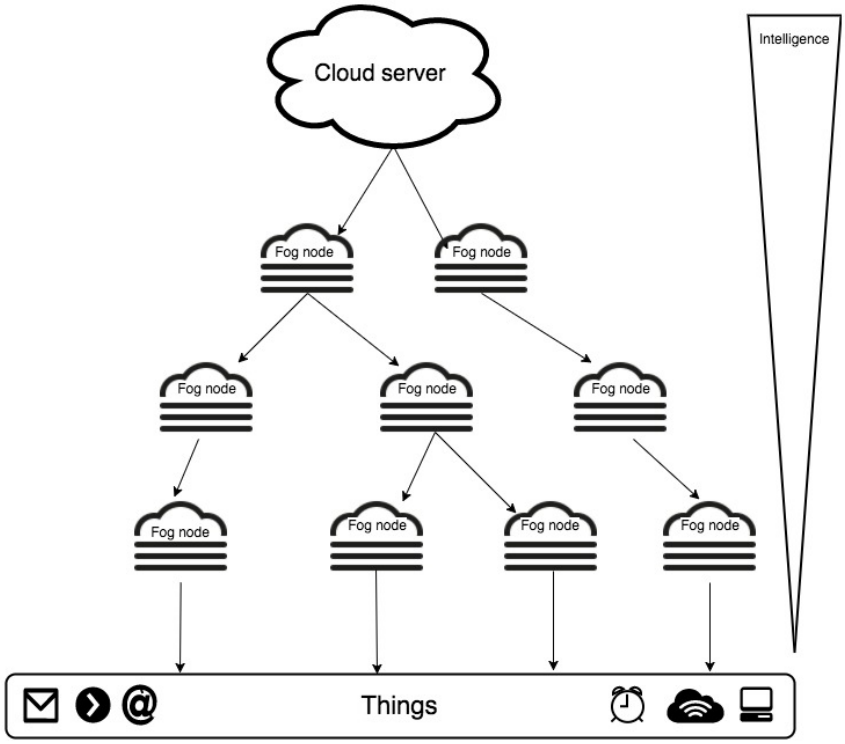


Figure 2.2: Fog computing is hierarchical. Nodes on the lower levels will capture and filter the data, before sending the more important data up. Nodes on the higher levels will aggregate and perform analysis on the data. Computational Intelligence increases with higher levels. Adapted from [CW17], figure 11

nodes will focus on data filtering, compression, and transformation. Computational Intelligence will increase further up in the hierarchy, meaning higher level nodes will have a greater overview of the network and be able to make more informed decisions. At the top of the hierarchy, there is typically a cloud server, and the relevant data is collected and turned into knowledge.

Nodes on the same level can be linked to form a mesh network to provide load sharing, resilience, fault tolerance and data sharing [CW17]. Peer to peer communication between nodes on the same level will be possible. Figure 2.3 shows one possible solution for a smart city. In this figure, each city level e.g., street or neighborhood is also a level in the fog hierarchy. This peer-to-peer communication will be important when discussing trust management in fog computing.

Within a street, the topology of the nodes may look like the network in Figure 2.4.

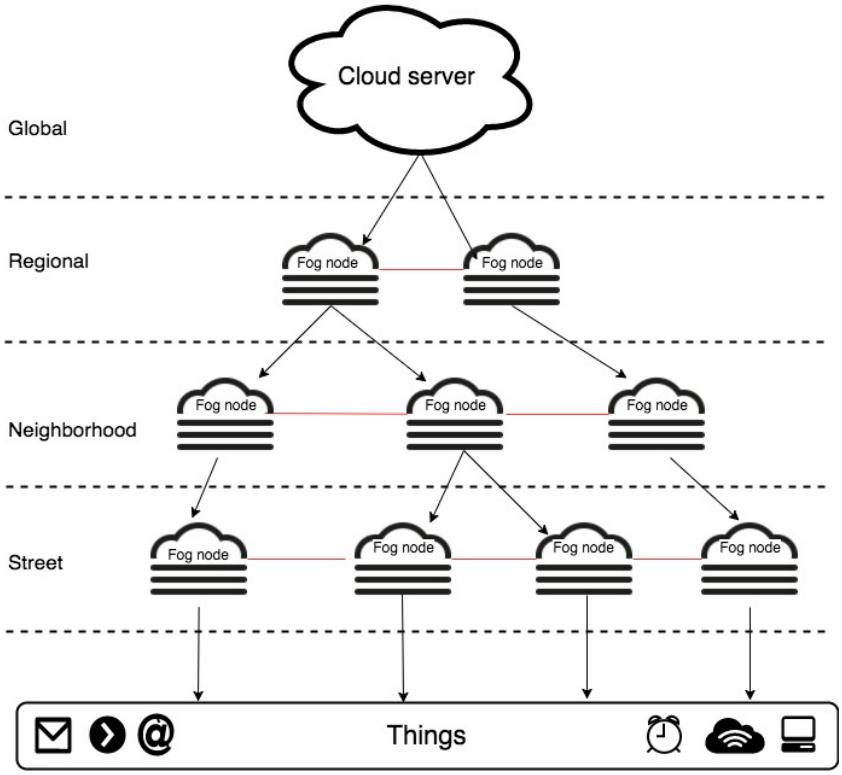


Figure 2.3: This figure shows a possible solution of the fog hierarchy in a smart city. Fog nodes on the same level form connection among themselves to provide additional services, shown here by red lines. Nodes in the same street, neighborhood and regional will know about each other. Adapted from [CW17], figure 12

However, the topology will change continuously. Nodes can coordinate handovers between themselves, or, if needed, the higher level can coordinate the handovers [CW17].

2.4.3 Fog Deployment models

Cloud computing has a public, private, hybrid and community deployment scenario. These are envisioned for fog computing as well [YQL15a]. However, no formal definition of the fogs different deployment models was found. Therefore the cloud deployment model will be used. This section will briefly describe these four deployment models, to gain further understanding of the fog computing domain.

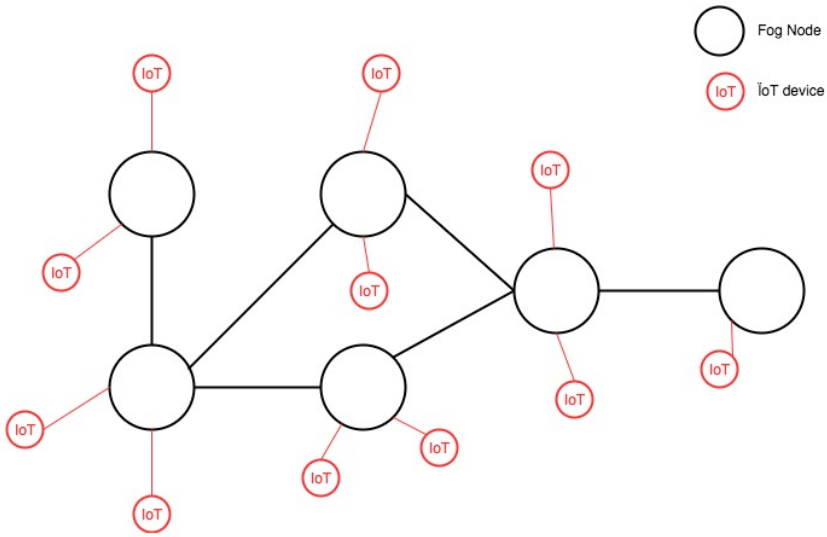


Figure 2.4: View of the connections between fog nodes on one layer and IoT devices. Fog nodes can connections among themselves. These can coordinate handovers and resource sharing to provide better QoS

- **Public fog computing** is an infrastructure that provides services for the general public. Members of the public can rent necessary resources. In cloud computing, the infrastructure is owned and managed by one organization [MG+11], one large organization managing the whole infrastructure may not be the case in fog computing. Large organizations, small organizations and possibly private individuals can collaborate to create an infrastructure.

- **Private fog computing** can be owned and operated by a single company like private clouds are today [MG+11]. This will be relevant for companies with high-security requirements but is more costly than public fog computing.

- **Hybrid fog computing** is a combination of public and private fog computing [MG+11]. Sensitive information can be handled by private fog computing, while public fog computing can handle the rest.

- **Community fog computing** will be shared by businesses that have common criteria in respect to the security and reliability [MG+11]. By deploying community fog computing, the cost will be shared between more companies than with private fog computing, but less than with public fog computing. More personalizing is allowed with community fog computing than with public fog computing.

Characteristics	Fog	Cloud
Architecture	Distributed	Centralized
Location awareness	Yes	No
Location of node	Edge of network	Core of network
Latency	Low	High
Number of nodes	Many	Few
Processing power of nodes	Low	Nearly Unlimited
Distance to nodes	Few hops	Multihop
Mobility support	Yes	No
Data analysis	Data in motion	Data in rest

Table 2.2: Table showing the main differences in characteristics of fog and cloud computing. Adapted from [RS17] table 6.1

2.4.4 Characteristics and Benefits of Fog computing

Characteristics that separate cloud and fog computing will be analyzed in this section. Benefits with fog computing will be derived based on these characteristics.

Table 2.2 highlights the most important differences between cloud and fog computing. Both cloud and fog can compute, store and network. However, the placement and number of the fog nodes constitute the main differences.

Several benefits are achieved because of these differences between fog and cloud computing. These benefits are outlined below.

- **Scalability** Scalability is a network's ability to grow in capacity. A fog network is a scalable network. If traffic in one area is too big to handle for the current fog nodes, another fog node can easily be placed to take off some workload. Additionally, nodes can be added to remote areas, where it has previously been difficult to provide coverage [VRM14].

- **Possible Increased Privacy** When data processing happens at the edge of the network, the data does no longer have to be stored in central entities. Users can take more control of their data [VRM14]. Processing of data happens closer to the clients. Information that can link a specific user to a piece of data can be removed at the edge, before sending it further to the centralized entity.

- **Dependability and Reliability** Dependability is increased in two ways by fog computing. First, clients will not depend on a centralized entity. If the centralized entity becomes, unavailable fog nodes can be utilized instead.

Second, several fog nodes can be deployed in one location. If one node becomes unavailable, several others can be utilized instead [YMSG+14b].

- **Reduction of Bandwidth and Storage** Fog nodes can filter out unnecessary data [VRM14]. Not all data collected by the IoT devices is useful. Filtering unnecessary data at the edge will reduce the amount of data sent into the cloud. The cloud can then focus on only relevant data i.e., data that will increase the knowledge of the system.
- **Location Awareness and Context** Location awareness is the ability to determine the geographical location of a device [Min04]. This type of information is useful. However, it has privacy concerns. A fog node may know where a fog client is geographical located because a fog client will connect to the closest fog node. Location awareness can be used for targeted advertising, or in a case of emergency.
- **Analyze Data in Motion** Fog computing has the ability to analyze data in motion, e.g., being able to draw conclusions in real-time [RS17]. Cloud computing does batch processing, while fog computing can analyze a stream of data. Being able to analyze a stream is beneficial because data is most valuable right after creation. For many IoT applications timing becomes a vital component for the device to function as intended. Sensor data does not work well in the original, capture, store and analyze method, which is how cloud computing works. The value is the greatest right after it is sensed, and should, therefore, be analyzed right after creation.

2.4.5 Instances of current fog computing technology

Several companies are developing fog computing environments. Some examples are described below.

Linux Foundation newest project called EdgeX Foundry [Edg] is an edge computing platform. EdgeX Foundry bridges the gap between the IoT and the cloud platform. The massive amount of data produced by IoT devices is aggregated by edge nodes, before being sent to the cloud [Lin]. EdgeX Foundry has the capability to collect data from IoT devices no matter the underlying technology. With EdgeX Foundry, Linux Foundation wants to provide a common platform for all edge devices. While IoT use different protocols, EdgeX Foundry standardizes the data before sending it to the cloud, making it possible for devices that run on any hardware or operating system to collaborate.

Amazon Web Services released AWS Greengrass [AWS] in November of 2016. AWS Greengrass is a service that extends AWS services to the devices. It can filter out unnecessary data generated by the devices, and only send necessary data to

the cloud. Management, Analytics, and durable storage can still done by the cloud. However, AWS Greengrass enables local computation.

Azure IoT Edge is Microsoft's answer to edge computing [Azu]. It is a service that delivers cloud capabilities to the edge. Azure IoT Edge claims to *"extend the intelligence and other benefits of cloud computing to edge devices"* [Azu]. Code that is currently running on cloud servers can with Azure IoT Edge be run on the devices themselves. Microsoft also claims Azure IoT Edge reduces bandwidth costs and increases the reliability of services.

2.5 Motivation for Fog Computing

Smart cities will be introduced in this section to provide a concrete example of where fog computing can be utilized. First smart cities will be introduced, followed by the motivation for fog computing in a smart city.

Smart cities are cities that use technology to better build and develop their cities, to provide better services for its residents [Thi]. Better utilization of resources and better quality of service for its citizens, while reducing its operational cost are common goals [ZBC+14]. As described in Section 2.1, the IoT collects vast amount of data. With the use of smart cities, cities can plan better and smarter, and place money and resources where it is most afficient.

Today, smart cities are becoming a reality, using for instance applications for traffic management and smart grids. A major challenge in the realization of smart cities is that each application deployed needs a designated network of gateways and sensors. Such designated networks are communally called silos. Silos are hard to set up, maintain, and control [YVJ+17] because of the complexity of the infrastructure. Silos also create a higher threshold for companies to deploy applications. Figure 2.5 shows the current scenario of a smart city environment. Each solution is completely separated from the others solutions [YVJ+17]. Existing solutions lead to fragmentation of services. Collecting and analyzing data for all applications, so that the city can fully take advantage of the smart city is hard, because there is no standard solution.

Fog computing can solve this by making a shared infrastructure for application deployed in a smart city. Instead of all applications having a dedicated network, one common network is offered by fog computing. A public fog can be deployed in a city. With a shared sensing and computing infrastructure third party companies are relieved from the task of building and maintaining their own infrastructure.

Cities can benefit in several ways from deploying a fog network. An improved and wider range of applications can be deployed utilizing the existing city infrastructure. Fog makes data collection easier and provides a common ground for accessing the data. In Barcelona, for instance, fog nodes have been put in street cabinets [YVJ+17], because these street cabinets where already been placed around the city. However, each city is different, and the processing power can be put wherever most convenient. The solutions may look like Figure 2.6.

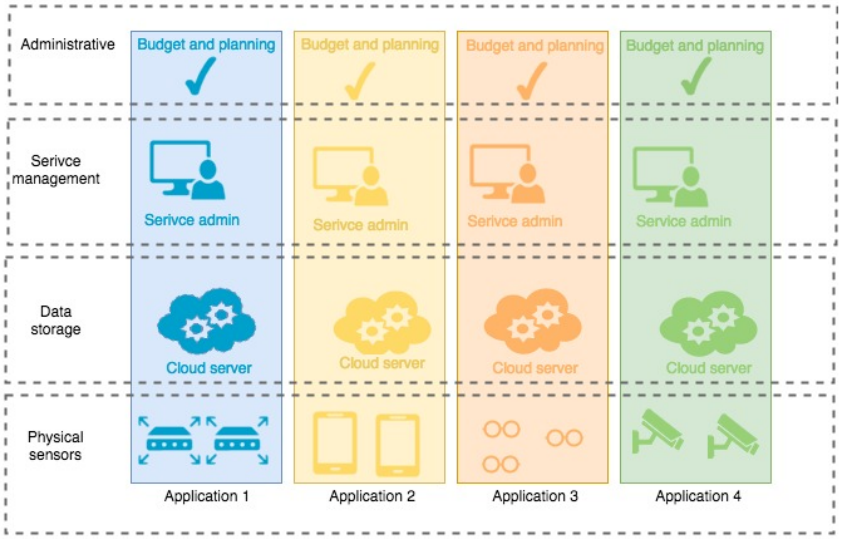


Figure 2.5: In a smart city each application has a different setup and environment. As the figure shows, these applications use the same infrastructure. When an application is deployed, a dedicated network also needs to be deployed. Maintaining this network is a tedious task. It would be easier to set up and maintain a shared network for all smart city applications. Adapted from [YVJ+17]

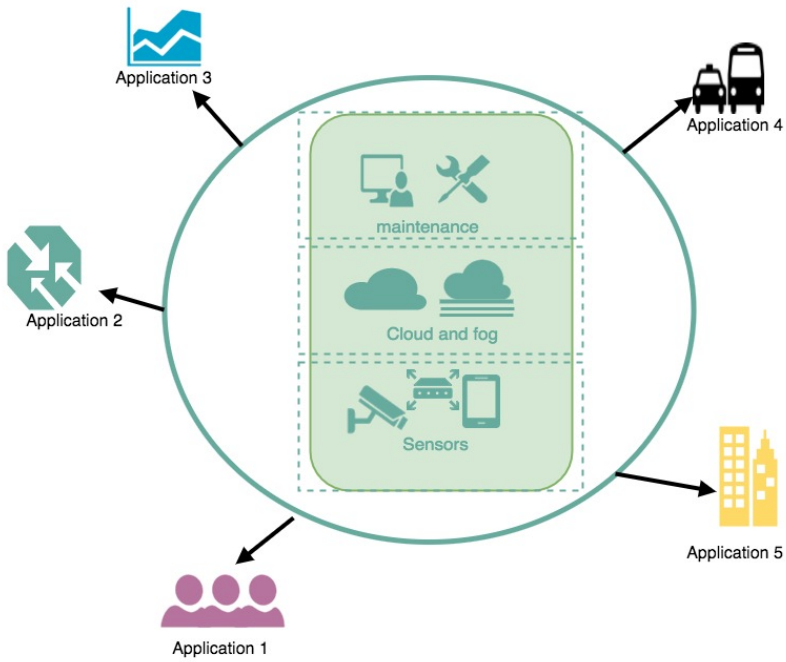


Figure 2.6: Fog computing can deploy a shared common infrastructure for all applications in a smart city. In the figure all sensors, computation and maintenance is put into one common system. Applications can these use the resources they need to provide services to the city. Application developers can focus on developing applications and take advantage of the capabilities provided by the fog and cloud computing. Adapted from [YVJ+17]

2.6 Use Case

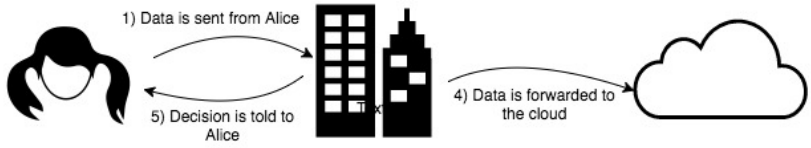
In this section, a use case is described. An application utilizing the fog computing infrastructure has been deployed to help the visually impaired (VI) gain more person mobility. This use case exemplifies a specific resource-constrained device, a fog client, in need of offloading. Reliability and latency are important for the application described, and therefore cloud services cannot be utilized. However, to be sure the resources needed can be provided by the fog node, trust needs to be established between the fog node and the fog client.

Imagine that a city has deployed a fog computing network. All around the city, fog nodes are placed with a varying degree of processing power and link stability. All fog nodes can provide a service for any fog client no matter the protocol or hardware of the device. One kind of service can be Processing as a Serivce (PaaS). With PaaS, devices that are resource constrained, can offload the computationally heavy tasks to devices with more resources. All devices and fog nodes have a unique and distinct identification.

Now to the specific use case, Alice is a 22-year-old student, due to an illness, she is slowly losing her sight. As a consequence, she is losing her personal mobility, and therefore more of her social life. She does not want to use a cane or service dog because this draws attention to the fact she is visually impaired. Familiar places are not a problem for Alice. However, unfamiliar places such as new streets or indoor navigation at subway stations or malls pose difficulties for Alice. An application that could sense the environment around her, and provide necessary feedback would increase her personal mobility. Necessary feedback could be about the traffic, or to provide navigation on a crowded street. These applications can also provide help with reading road signs or web pages.

Computer vision may be a solution for those VI, such as Alice. K. Ha, Z. Chen, W. Hu et al. [HCH+14] describe a system that uses Google glasses [Goo], and computer vision to provide assistance to VI. With such a system, Alice can use the Google glasses to sense her surroundings and the information can be given back to her in a convenient way. She can be warned about crossovers in the street, people she knows and obstacles in her way through other senses than her sight. This may increase her personal mobility and independence.

However, computer vision algorithms are heavy to run and require at lot of computational power which the resource constrained fog client does not have. Wearable devices, such as Google glasses, are not capable of running these kinds of applications themselves. Wearable devices should be light, small and have long battery life. Therefore the processing power is restricted. A possible solution for the resource problem is offloading. When using offloading, an external server is used to execute



(a) Simplified cloud deployment of a smart city

(b) Simplied fog computing deployment of a smart city

Figure 2.7: **(a)** shows a simplified cloud computing deployment of a smart city. All data has to be sent to the centralized cloud servers for processing. **(b)** shows a simplified fog computing deployment of a smart city. The collected of fog nodes, here illustrated by the buildings, in the smart city can make a decision based on the data collected from Alice, without the need for the centralized cloud server.

the heavy back-end computations for the application and return the output to the wearable device and the user.

K. Ha, Z. Chen, W. Hu et al. [HCH+14] have compared the power consumption and time spent when running an assistant application on the Google Glass itself and when using external services. Offloading resulted in ten times lower power consumption and time spent executing computations. These results show that offloading is beneficial for resource constrained applications. Using external servers for offloading causes less energy consumption, which increases battery life and reduce heating problems.

Humans process information from senses extremely quickly. High latency in these kinds of systems can therefore not be tolerated. In normal light conditions, humans take from 370 ms to 620 ms to detect a face [LE03], depending on the familiarity of the face. Virtual Reality applications need a latency of less than 16 ms to achieve perceptual stability [EMAH04]. The latency requirement for assistant applications is the same.

Latency and processing power is a challenge. Cloud servers are located far from the edge devices, and the data spends a lot of time traveling. Results conducted by Ha. et al. [HCH+14] show that by using cloudlets i.e., a mini-cloud server placed at

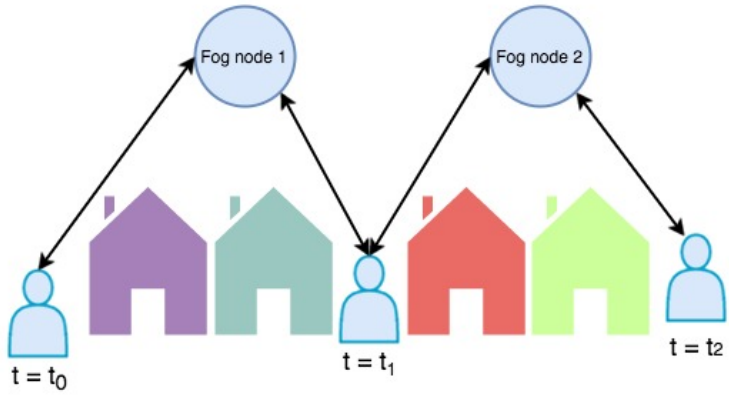


Figure 2.8: A handover is put into motion when a fog node senses that better service can be provided by another node in the fog network. Trust should be established between the new fog node and the IoT device to be sure that the fog node can provide the quality of service needed.

the edge [SBCD09], the latency and power consumption was reduced compared to using cloud server. Less latency means less time spent in states that require a lot of energy. Devices offloading to edge nodes used 30-40% less energy compared to offloading to the cloud servers [HCH+14].

Alice needs a continuous and stable connection to a processing unit. Without connection, she will not have any assistance. Therefore, mobility needs to be supported. Fog nodes placed all around the city could provide offloading for this type of applications. When Alice walks around the city, her glasses – the fog clients – can connect to the deployed fog nodes. Reliability is increased when the fog computing network is used. Even if the cloud servers are down, Alice can access processing power. Thus, the single point of failure is avoided.

However, some problems still exist. Assistance applications have strict service requirements. Alice has to trust that the fog node can handle the computing tasks she needs offloaded. Fog nodes can as mentioned have different and varying processing requirements. Some fog nodes will be able to process the computer vision tasks, while others will not. This can also change according to other environmental conditions such as the amount of other fog clients they are currently working for, a number of neighboring nodes, or and the signal strength.

Trust can be used as a measure to gain assurance that the level of service that is required can be provided by the fog node. Trust can assist in choosing the best node if several nodes are available. The goal is to achieve a level of assurance without the

need for Alice to intervene in the trust process. Her assistance application should be able to autonomously set up a connection with fog nodes never before encountered, and be assured that this node can handle the processing complexity. This means that when Alice comes to a new place, she is assured safe and reliable service.

Chapter 3
Introduction to Trust

This chapter will focus on trust in IT environments. Trust has so far been introduced as *the level of assurance that an object will behave satisfactorily.* Why trust is necessary for IT systems will be explored in Section 3.1. Trust is systematized by trust management. Trust management will be defined in Section 3.1.1. Finally, trust management in service management IoT will be addressed in detail in Section 3.2. Trust management in service-oriented IoT will be discussed in detail because of the similarities between the desired behavior of objects in service-oriented IoT systems and fog computing objects.

3.1 Trust in IT Environments

Trust in IT environments relates to more than security, it is also about the reliability, integrity, dependability, and ability to perform a service [YZV14]. Trust also motivates for collaboration between parties [SF99]. Best results on the tasks are achieved when all the participants work in an honest manner [ZX15]. Trust establishment in IT systems gains assurance that an object will work in a manner you deem satisfactory.

Trust may be defined in many ways. *Dictionary.com* defines trust as *"reliance on the integrity, strength, ability, surety, etc., of a person or thing; confidence"* [Tru]. This means your counterpart will behave in a manner you find appropriate. In IT environments, this means that your counterpart acts according to defined protocols.

Predicting the future behavior of an entity is essential for the establishment of trust [GCT17]. However, it is challenging, the behavior of an object may change over time, and therefore only relying on past behavior may lead to potentially dangerous situations. Getting the necessary information to establish trust is hard in the IoT domain, because of small sample sizes and uncertainty of data origin.

To establish trust, assurance that an object will act according to a set of policies must be obtained, assurance in the sense that the entity will behave in a matter you

find satisfactory [LS07]. How this assurance is obtained will depend on the application in question. It depends on the deployment environment, network applications, and the level of security required [ZX15]. Some applications, for instance, health and safety critical applications, will need a high degree of trust, while other applications – where reliability might not be such an issue – will require a lower level of trust.

Distributed networks, such as the fog network, are vulnerable to attacks, making trust necessary. Objects in these kinds of networks will encounter other objects they have not interacted with earlier. Malicious objects will exist in the network and exploit unsuspecting objects. Cohesion must, therefore, be taken before collaborating with others. However, collaboration is necessary for the system to deliver the ultimate results. Trust is therefore highly relevant in distributed systems because it enables collaboration among entities.

Y. Sun, Z. Han and K. Liu [SHL08] defined three ways where trust is beneficial in a network. Firstly, it can provide assistance in decision making. When a node can predict the future behavior of another, it can avoid collaborating with ones that will not behave in a satisfactory manner. Secondly, the security can be increased if the risk is high. If a node is highly trustworthy, less security may be used, making the computations less complex, and possibly reducing the latency in the system. Finally, misbehavior of devices in the network can be detected. If the trustworthiness of an object shows to be very low, actions can be taken to investigate and remove the device from the network.

Reputation is often used interchangeably with trust. These concepts are similar but not the same [LS07]. A reputation is an opinion someone has about something. Trust may be derived from this opinion, but trust may also take other properties into account. Trust is the prediction of the future behavior of an entity, while the reputation is based on the past behavior.

Security is often used when trust is involved. Security provides protection against malicious entities. However, there has to be pre-installed or pre-configured information to be able to protect objects in the network. From a trust perspective, security measures can be seen as a way to transport trust from where it is established to where it is needed [JKD05]. If two communicating parties share a key, they have already established trust with each other.

Jøsang [JKD05] describes trust as a directional relationship between two parties, the trustee, and the trustor. The trustor is the party that wants something done, and is relying on the trustee to do this task. According to Jøsang, the trustor has to be a thinking party, in other words, be able to make decisions based on a set of evidence. Se Figure 3.1 for a visualization of this concept.

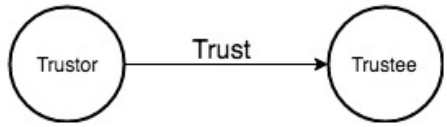


Figure 3.1: Graph showing the relationshiip between the trustor and the trustee of a trust relationship

Since there are two parties in a trust establishment – trustee and trustor – there are two separate goals. The trustor wants to correctly assess the trustworthiness of the trustee, how good the future behavior is. Connecting with an untrustworthy node can lead to problems and should therefore be avoided. On the other hand, the trustee wants to have the best possible trust values.

3.1.1 Trust Management

Trust management enables entities to establish trust with others. It describes the systems and mechanisms in an application that facilitates the trust to be established. Trust management was first introduced by M. Blaze, J. Feigenbaum, and J. Lacy [BFL96]. They define the trust management problem as *"the problem of figuring based on formulated security policies and security credentials if a set of security credentials of an entity satisfies the security policies".* Trust management considers what information to collect and how to collect this information, to gain assurance that an entity is trustworthy. Trust management deals with trust establishment, trust update and trust revokement [CSC11].

3.2 Trust Computing in Service-oriented Internet of Things

Trust computing is a subcategory of trust management. Trust computing means how trust values are gathered, which aspects of trust is used, and how these are put together to find the final trust values before they are delivered to the network again.

In a service-oriented IoT an Service Requester (SR) requests a service from an Service Provider (SP) [GCT17]. This can be view as a peer-to-peer system. To be able to provide the best services it is crucial to evaluate trustworthiness of the SP.

The following section will explore trust computing in service-oriented IoT, because of the similarities to fog computing. The goal of service-oriented IoT is allowing devices to request service from each other [GCT17]. A thorough introduction to the state-of-the-art research in trust management in service-oriented IoT will be needed to form a trust management scheme for fog computing later. A introduction to trust

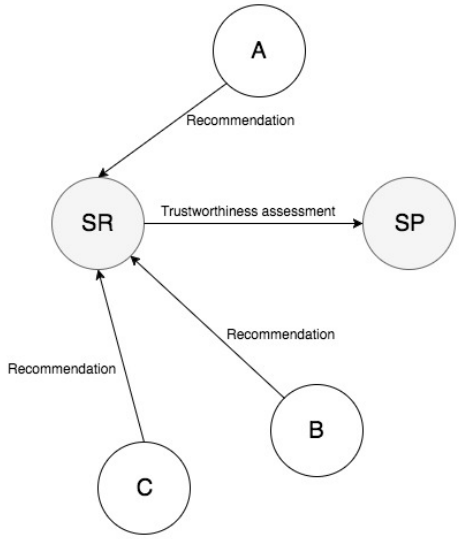


Figure 3.2: Graph showing how trust between nodes may occur. The Service Requester(SR) receives recommendations about the trust level of the Service Provider (SP). Based on these recommendations the SR forms an opinion about the level of trust it can put on the SP.

in service-oriented IoT will be given in Section 3.2.1. Section 3.2.3 to Section 3.2.7 treats different components of the trust establishment between two objects in IoT.

3.2.1 Introduction to trust in Service-oriented Internet of Things

An example on how trust can be established in a service-oriented IoT network is given in Figure 3.2. A service requestor (SR) wants to connect to a service provider (SP), which it has no previous experiences with. Therefore the SR asks for recommendations from A, B, and C. Through previous interactions the SR has built up a level of trust to A, B and C. In other words, a belief in whether the information provided by A, B and C is trustworthy or not. The SR may have different level of trust to A, B, and C. By combining Alice's own experiences with the recommendations, here A, B, and C, the trust level of the recommenders, the SR can assess the trustworthiness of SP.

Trust allows for autonomous communication between objects, here the SR and SP, without the need for the owners to intervene [AIM11]. The goal of trust management in service-oriented IoT is to know if an SP providing a service is safe to connect to, regarding reliability and availability. Objects with no previous experience of each other can gain confidence from collecting information about each others behavior. This information can be stored in a central server, or in a distributed manner, such

as the scenario in Figure 3.2. Autonomous establishment of communication between devices with no previous knowledge of each other is also desirable in fog computing.

3.2.2 Five trust dimensions

J. Guo, I. Chen and J. Tsai [GCT17] surveyed existing techniques for trust computation in the service-oriented IoT. They classified trust computing methods into five dimensions: trust composition, trust propagation, trust update, trust formation and trust aggregating. This section will describe each of these dimensions in detail.

- **Trust Composition** Trust Composition decides what components are included in the trust computation. The quality of Service and Social Trust are the two main components. Trust Composition is described in Section 3.2.3.
- **Trust Propagation** Whether or not the system uses a centralized or distributed manner to store and compute trust is decided by the Trust Propagation dimension. Trust propagation is explained in Section 3.2.4. The solution in Figure 3.2 is a distributed approach.
- **Trust Update** Trust Update indicates how often the trust values are updated this is discussed in Section 3.2.5. Event driven and periodical updates are the two main approaches for Trust Update.
- **Trust Formation** Trust Formation describes how the trust properties determined by trust composition are combined. Some schemes only consider one property, and some schemes consider a combination of several properties. Section 3.2.6 covers trust formation.
- **Trust Aggregation** Combining recommendations from others with own experiences in the trust computation may be necessary. Trust Aggregation decides how this is done. Two aggregation schemes are introduced in Section 3.2.7.

All these dimensions are necessary for the establishment of trust values in service-oriented IoT environments [GCT17].

Trust Formation, Trust Composition, and Trust Aggregation may be seen as the same dimension, however, they are not. Trust Composition defines a set of trust properties. Trust Formation decides how to combine these properties. Some trust properties may be more important than others. Determining which trust properties should be weight heaviest is the Trust Compositions task. The Trust Aggregation can collect these properties from own experiences, or from the experiences of other nodes. Trust Formation, Trust Composition and Trust Aggregation will be described in detail in the following sections.

3.2.3 Trust Composistion

Trust Composition defines what components are considered in the trust computation. All trust components can be divided between Quality of Service (QoS) trust components and social trust components. An object in the following section will refer to a SR or a SP in the network.

QoS trust

Belief in the QoS provided by object is defined by the QoS trust i.e. the level of assurance that a node can deliver the service requested. Variables related to performance might include the amount of throughput, latency or errors occurring. S. Namal and H. Gamaarachchi and G. MyoungLee and T. W. Um [NGMU15] considers availability, reliability, irregularity, and capacity as QoS parameters. M. Li and H. Wang and D. Ross [LWR09] proposed using experience, knowledge, and recommendation. How these variables are used and combined is described in trust formation, see Section 3.2.6.

Social trust

Social relationships between the owners of objects form the social trust [GCT17]. Recall from Section 2.2 that different relationships exist in SIoT. Objects in the service-oriented IoT can form relationships. The social relationship between two humans will influence the amount of trust established, e.g., you will have a greater amount of trust to a friend than you will to a stranger. This will also be the case in IoT. Objects with the same owner will have a higher level of trust, than two objects which are produced by the same manufacturer. Objects that are in regular contact with each other will develop a relationship because the owners will most likely have a relationship in the physical world. When you are at your friend's house, their router will recognize your watch as an object they have encountered before.

Other values can also be considered in addition to the relationships between two objects when dealing with social trust. Social networks can be considered as a network. Other values can for instance be:

- **centrality** how important an object is a network, how many interaction it has with other objects
- **credibility** how believable the information provided is.
- **social similarity** how similar the objects are in interaction

Chen et al. [CGB16] measures social similarity in friendship, social contact similarity, and Community of interest (COI) similarity. Friendship is defined by how

many interactions have occurred between the objects. The more direct interactions, the higher degree of friendship, and the higher degree of trust among those two objects. However, the degree of friendship will impact the credibility of a recommendation of an object [NGA14]. Social contact similarity is a measurement of the degree they trust the same objects. If two objects are in the same COI, they will have the same interest, making the trust relationship more beneficial to each other, because they are working towards the same goal.

Recall Figure 3.2, if the SR and A have a friendship, while B and SR do not, SR will put more weight on the recommendation coming from A than B. Another example is if the SR and SP have the same owner, a higher degree of trust can be established between the two objects.

3.2.4 Trust Propagation

This part of trust computation describes how trust values are stored in the network. Two main schemes are described the literature; distributed and centralized.

Centralized Trust Propagation

A centralized entity, for instance, the cloud or a trusted entity has responsibility for storing trust values to all objects in network. Having a centralized entity in charge of the trust values, makes information sharing easier [GCT17]. All entities in the network will have access to the same information; this will free memory in the resource constrained objects. Processing of trust values can also happen in the centralized entity. However, this leads to a undesired dependency on availability of the central node which may become a single point of failure. If the trust center becomes unavailable, no object can access the trust values. M. Nitti, R. Girau, and L. Atzori [NGA14] use a distributed hash table to store the trust values of the objects in the network. All trust queries are sent to this entity, and the central entity responds to trust queries for all objects in the network. A centralized entity will not always be available in the IoT environment, which is why a distributed approach is the most common in current literature [GCT17].

Following the example in Figure 3.2, if the SR contacted a central server instead of asking for recommendations from the objects around, it would be a centralized solution.

Distributed Trust Propagation

In a distributed approach every object has the responsibility of storing and computing the necessary trust values, and provide recommendations to others. This is the original scenario described in Figure 3.2. Monitoring of the network is necessary in a

distributed trust approach. Based on the information the SR received it will form an opinion about the SP. No centralized entity is used, and objects are responsible for sharing information. Attacks are possible with a distributed approach, such as bad-mouthing, see Section 4.3. Also in energy constrained objects, monitoring the network constantly requires a lot of energy [KH].

3.2.5 Trust Update

This dimension describes when trust values are updated. In either centralized or distributed trust propagation, trust needs to be updated. Figure 3.2 Again, two main approaches are described: time-driven and event-driven.

Event-driven

In this approach trust update takes place after an event. For instance, the quality of service may be computed after a transaction has taken place and the experience of the service recorded. It can be sent to the centralized unit or stored locally and shared when required. J. Guo et al. [GCT17] states that this approach is more suitable in a centralized topology because it is more energy-intensive. Updating the trust value after every event requires energy, which is less suitable for resource-constrained devices.

Figure 3.2 shows event-driven approach. Before connecting, the SR asks around and updates its trust value for the SP.

Time-driven

Trust values are updated periodically, with fixed time intervals. This approach is considered more suitable in a distributed system because it is energy persevering compared to an event-driven approach [GCT17]. However, the trust value of an entity can change during a period, and the time-driven approach is therefore considered less accurate. The optimal interval for both the maximum accuracy and the minimum energy consumption needs to be determined. However, in a time-driven approach objects may need to store trust values for objects it may not need.

If the solution in Figure 3.2 was time-driven, the SR would have received information about the trust of SP at the previous time interval.

3.2.6 Trust formation

The main task of trust formation is to weight trust properties according to relevance. Trust formation describes how to form trust out of multiple trust properties. Some of the possible properties are described in Section 3.2.3. Trust can be formed by

only considering one trust property (single-trust) or by combining several properties (multi-trust).

Single-trust

In schemes where only one trust property is considered, it is usually the most important one, namely the QoS. Trust is seen as multi-dimensional [GCT17], therefore single-trust formation is one-sided. However, if the objects resources are very limited it could be considered as a viable option since less storage and computation is required.

Multi-trust

Trust is seen as multi-dimensional, in multi-trust formation several trust properties are taken into account to compute the trustworthiness of an object. There are two ways this can be done:

- First option is to set a threshold value for every trust property. In this approach, several trust properties are considered, but they are not combined into one. Each trust property is assigned a threshold value. The threshold value reflects the importance of the property in the application. If the value of one of the properties is below its set threshold, the whole object is seen as untrustworthy.
- Second option is to combine the trust properties into one. Here a weighted sum can be used. If a trust value is more important than another, it will be assigned a heavier weight. The weights assigned can be adjusted according to the context, and be altered. However, how to dynamically adjust the weight is a topic of further research [GCT17].

3.2.7 Trust Aggregation

After the properties have been defined, trust needs to be collected from the members of the network. How to collect experiences from other objects in the network is defined by the trust aggregation. Some objects experiences may be more useful and truthful than others – the trust aggregation dimension handles this. Collecting experiences need to happen in both a centralized and distributed manner. However, since a distributed approach is the most explored in current literature, this section will focus on distributed approaches.

Combining trust values from both own experiences and the experiences of others into a single value happens in this step. Guo et al. [GCT17] discusses several techniques, where weighted sum is the most common method in the current literature. In reputations systems, weighted sum can be used in such a way that the users with

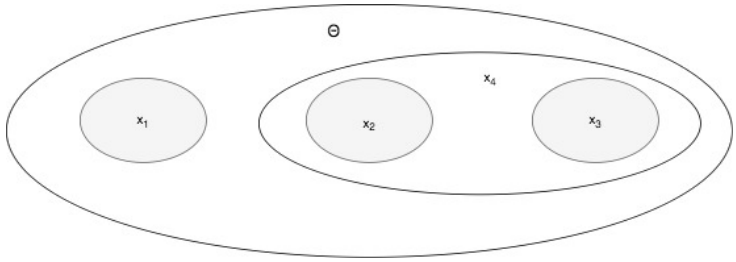


Figure 3.3: A model of a system is described in this figure. This model has three atomic stats, x_1, x_2, x_3. Only one of the atomic states can be true at once. x_4 is a non-atomic state, also called a super-state. If x_2 or x_3 is true, x_4 will also be true.

the highest reputations, will have the most impact on the total trust value. Other techniques mentioned by Guo et al. [GCT17] are fuzzy logic, belief theory, Bayesian inference and regression analysis.

Subjective logic, which is a special form of belief theory and regressional analysis will be further described below, as they are seen as the most appropriate for fog computing. Using subjective logic has been described in many studies [GCT17]. Using logistic Regression analysis to estimate trust is a relatively new concept. The following section will discusses these two topics.

Subjective logic

Subjective logic builds on the belief that all trust is individual i.e. subjective, and everyone will experience trust differently [JØS01]. This is the reason each object in the network has to compute its own trust values. Not all trust evidence may be available, and sometimes a conclusion has to be drawn despite a degree of uncertainty. Jøsang in his paper *"A Logic for Uncertain Probabilities"*[JØS01] proposed the subjective logic solution, which makes it possible to draw a conclusion, despite the lack of evidence.

The belief model is the foundation for subjective logic. The belief model works in the following way. Given a system, the model describes all possible states of the system, where only one state can be true at a time. The goal of belief theory is to say which state is true. A believe mass is given to each state, which defines the possibility that a state can be true. An example is shown in Figure 3.3. A state can be atomic, as state x_1, x_2 and x_3, or non-atomic as state x_4. x_4 is a superstate, consisting of state x_2 and x_3. Only one atomic state at a time can be true if one of the atomic states of a superstate is true, the superstate is also true.

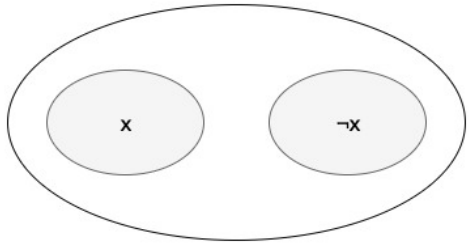


Figure 3.4: A system model considering the trust of an object will only have two states, trustworthy, x and not trustworthy, $\neg x$.

When dealing with trust, a model only contains two states. A system can be trustworthy or not trustworthy. Shown in the Figure 3.4

Jøsang [JØS01] defines four variables, belief b, disbelief d, uncertainty u and relative atomicity a. Belief b is the likelihood that an observer thinks that an object is in the trust state. Disbelief d is the belief that the object is not in the trust state. The uncertainty fills the void between the belief and disbelief so that $b + d + u = 1$. It models the lack of evidence that is present in such a model. Relative atomicity a denotes how likely it is that the degree of uncertainty results in belief or disbelief. The relative atomicity a will not be considered in the further description.

The degree of trust for an object i is given by the tuple $w_i = (b_i, d_i, u_i)$. Figure 3.5, shows the relationship between the three variables, where the degree of trust for the object i is a point in the triangle. If the uncertainty is low, it will be close to the bottom of the triangle. The uncertainty is low when there is a lot of experience with node i. Less experience will lead to a greater uncertainty, and the point of trust will be higher in the triangle.

One way to find the number for b,d,u, is to take the positive and negative experience into account. What is defined as a positive and negative experience can be defined by the trust formatting described in Section 3.2.6. In Jøsang and Knapskog [Kna98] introduced the following metric to find b,d,u, shown in Equation 3.1 to 3.3, where p describes the number of positive experiences and n describes the number of negative experiences.

$$b = \frac{p}{p + n + 1} \tag{3.1}$$

$$d = \frac{n}{p + n + 1} \tag{3.2}$$

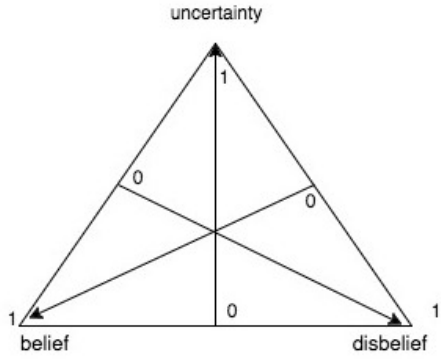


Figure 3.5: Triangle showing the relation between the trust values of Jøsang subjective logic, belief, disbelief and uncertainty. Trust is in subjective logic given by the tuple $w_i = (b_i, d_i, u_i)$, this tuple is located in the triangle.

$$u = \frac{1}{p + n + 1} \tag{3.3}$$

As mentioned, subjective logic can be used to assure that an object is in a trustworthy state before establishing collaboration. To use subjective logic the tuple *b,d,u* is collected from self-experiences and the experiences of others called recommendations. However, there are some challenges with collecting recommendations from others. Some recommenders may not always tell the truth. To make the most accurate prediction of the trustworthiness, recommendations have to be computed. Therefore weighting recommendations to determine the accuracy of the recommendation is the task of the trust aggregation dimension. In subjective logic this is done in two ways, discounting and consensus.

Discounting Not all recommenders will give an honest opinion. False recommendations can be given to perform attacks, described in Section 4.3. To defend against this, the discounting-operator $\otimes$ is used. If A, wishes to compute the trustworthiness of C, it can take recommendations from other objects in the network, for example, B. When A receives a recommendation from B, it will take its own trust towards B into account, when computing the trust from C. This makes the total trust of C a combination of the trust B has to C, and the trust A has to B.

The trust A has to B is given by:

$$w_B^A = (b_B^A, d_B^A, u_B^A) \tag{3.4}$$

B's trust to C is given by:

$$w_C^B = (b_C^B, d_C^B, u_C^B) \tag{3.5}$$

The trust A has to C, using B's recommendation is given by Equation 3.6

$$w_C^A = w_B^A \otimes w_C^B = (b_B^A b_C^B), (b_B^A d_C^B), (d_B^A + u_B^A + b_B^A u_C^B) \tag{3.6}$$

Consensus Recommendation from several objects should be taken into consideration. This is done by using the consensus operator $\oplus$. Each recommendation is treated in a fair and equal way when using this operator. The order is not important. If A and B both produce a recommendation about C in the given form:

$$w_C^A = (b_C^A, d_C^A, u_C^A) \tag{3.7}$$

$$w_C^B = (b_C^B, d_C^B, u_C^B) \tag{3.8}$$

the result will be

$$w_C^{AB} = w_C^A \oplus w_C^B = (\frac{(b_C^A u_C^B + b_C^B u_C^A)}{\kappa}, \frac{(d_C^A u_C^B + d_C^B u_C^A)}{\kappa}, \frac{(u_C^A u_C^B)}{\kappa}) \tag{3.9}$$

κ is equal to $u_C^A + u_C^B - u_C^A u_C^B$.

Using a combination of consensus and discounting will lead to the optimal results. Recommenders with high trust will have a bigger impact on the final trust value, than the recommendations of those with low trust [JØS01].

Applying this to our use-case, Alice is walking around town and needs to connect to a new fog node, R, which she has not encountered before. She will get recommendations from other fog nodes; let's say B, C, and D about fog node R. Alice has a different degree of trust with the separate recommenders. By combining the trust she has for the recommender and the recommendations, she will find the appropriate trust level of R. This is shown in Figure 3.6

The final computation of Alice's trust to fog node R is given by:

$$w_R^A = (w_B^A \otimes w_R^B) \oplus (w_C^A \otimes w_R^C) \oplus (w_D^A \otimes w_R^D) \tag{3.10}$$

Subjective logic conclusion The necessary computations to derive the degree of trust by using subjective logic are not intensive, making them suitable for resource

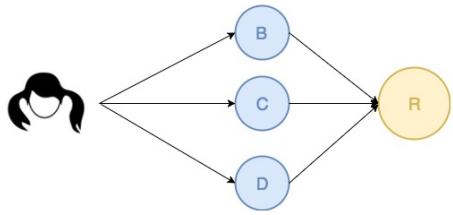


Figure 3.6: Alice has an opinion about the trust level of B, C and D. Alice uses the opinion from the recommenders to get the trust value of the router, R.

constrained devices [KH]. However, there are some problems with the subjective logic approach. The storing of the belief, disbelief, and uncertainty tuple may cause problems if an object has to store these values for every object it encounters. As mentioned, it has been suggested to only store the trust values for objects with high trust and the trust values for the nodes recently encountered. Continuous network monitoring is necessary for subjective logic, which is energy consuming for the objects [KH].

Logistic Regression

This section will introduce the second trust aggregation technique, logistic regression. Compared to subjective logic logistic regression is more computation heavier, however more accurate results have been achieved [WLC+15].

Logistic regression builds a prediction model. Based on this model predicts the probability of an event occurring. Events are binary: either they happen, or they do not happen. Using independent variables, the model can provide insight into which variables are relevant to the event, and which are not.

Logistic regression can be used in trust computing. Trust computing is a binary decision, i.e., trust computing has the goal of predicting if an object is trustworthy or not. Since logistic regression can predict the probability of a binary event, it can be used in trust computation.

We can find the probability, p, of true or false given any linear combination of variables, also called conditions. These variables are independent. In trust computing variables can be environment conditions. For instance, how the network is currently behaving or how much computing resources an object needs. Based on these variables – the environmental conditions, the model predicts the trustworthiness of an object.

Figure 3.7 shows the plot of the logistic regression function. We can see that it is only defined between 0 and 1. Equation 3.11 expresses the logistic regression equation,

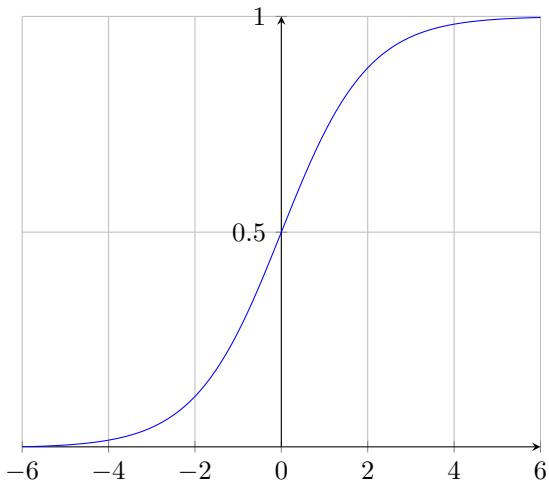


Figure 3.7: Plot of the logistic regression function

where α denotes the combination of independent variables i.e., how important each variable is for the trust estimation.

$$logit^{-1}(\alpha) = \frac{e^{\alpha}}{(1 + e^{\alpha})} \tag{3.11}$$

To compute the relevance of each independent variable, machine learning is used. A model is trained using a training set, where the training set consists of a set of variables and their outcomes. Patterns found in the training set are used to predict future behavior. When the learning process is complete, the logistic regression model will look as Equation 3.12 where β are the coefficients, and represent the importance of each variable. $x_1, ...x_n$ are the independent variables.

$$logit(p) = \ln(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \tag{3.12}$$

To estimate p, the probability, we use $\hat{p}$.

$$\hat{p} = \frac{e^{\beta_0 + \beta_1 x_1 + ... + \beta_n x_n}}{(1 + e^{\beta_0 + \beta_1 x_1 + ... + \beta_n x_n})} \tag{3.13}$$

Wang et al. [WLC+15] proposed a trust estimation system using logistic regression, called LogitTrust. Given environmental conditions in a mobile ad-hoc

network (MANET), an SR can predict the future behaviour of an SP. In LogitTrust, relevant environmental conditions are energy-sensitivity, capability-limitations and profit-awareness.

LogitTrust combines an object's own experiences with the experience of other objects in the network to build the model. Each SR creates its subjective logistic regression model, in this context individual, and is not to be confused with the subjective logic model described in the previous section. Each SP it wants to connect to with the help of the surrounding objects.

Malicious objects can provide false and misleading information. When less than 50 % of the nodes are malicious, this wrongful information will be marked as outliers. However, when more than 50 % of objects providing recommendation are malicious, this has a serious impact on the correctness of the model. Wang et al. [WLC+15] solved this by using social trust as well. Only objects that have a relationship with the SR can provide recommendations.

Logistic regression is an optimization problem, and there is no guarantee that the model you have generated is the optimal solution. Training the model takes $O(NC^2)$, N being the number of data points, the number of experience collected, and C being the number of variables. However, for small data sets, building the logistic regression model is only dependent on the number of variables [Iye15]. Nevertheless, after the model is built only lookups are required. Lookups in a logistic regression model are done in constant time.

Wang et al. [WLC+15] showed that logistic regression analysis produced better results than Bayesian inference. Every device needs to compute a personal model of each object it wants to communicate with. For a resource constrained device it might take minutes to generate the logistic regression model [WLC+15].

Chapter 4
Trust in Fog computing

In this chapter the need for trust management in fog computing will be discussed. Five requirements for trust in fog computing have been defined in Section 4.2. Several attacks on trust will be discussed in Section 4.3.

4.1 Why Trust in Fog Computing

Fog computing has a goal of increasing the reliability of the network. Trust can further improve the reliability by providing assistance for object selection. Applying trust management in fog computing architecture will allow fog clients, resource constrained devices, and fog nodes to predict future behaviour of each other. When predicting future behaviour is possible, fog clients can select a fog node in the vicinity that will provide the best service.

Trust management also allows for monitoring of the network. Intrusion detection systems can be built on trust management [SHL08]. Most likely, be used for the same purposes in the fog architecture. Intrusion detection is important to avoid malicious objects impacting the reliability of the network.

Trust motivates for collaboration among objects [SF99]. Collaboration among object is important to improve the performance of a network. In fog computing, fog clients offload and take advantage of the higher processing power in the fog nodes. This results in less energy usage and lower latency for the fog client. Fog nodes can charge for the service provided, and obtain profit from the fog clients using their services. In other words, collaboration between object is beneficial for both fog clients and fog nodes.

From a fog clients perspective, fog nodes represents a potential threat. A fog client is a device utilizing the service offered by the fog nodes. Fog nodes are maintained independently and are owned and operated by different organizations. Fog clients therefore need to be extra cautious when using fog nodes. Different

owners maintain security differently, and the security among fog nodes deployed by the same organization may also be different. Trust can be used as an assessment of the security level of the fog node.

Fog nodes can be deployed by anyone, making it easy to set up a rouge node.[SW14], and a threat for the whole network. Fog nodes are also deployed in locations that are easily physically accessible, and therefore – unlike cloud centers – can be tampered with on-site. If a node's hardware or software has been tampered with, it is a potential threat. Private data that is shared with the fog node can be exposed or shared with unauthorized entities. A rouge node can also send wrongful information, and overflow the network, negatively affecting the network performance and increased packed loss. Trust can be used to detect a rouge fog node in the network.

From a fog node's point of view the fog client also poses as a potential big threat. The fog nodes perform services for the fog clients. These services can hide scripts or code with damaging effect on the nodes hardware or software. Containers, or using Virtual Machines (VMs) can help the node to protect itself against malicious scripts. However it is still important to evaluate the trust level.

Data collected from fog clients are further used to gain knowledge by higher level systems. If the data collected from the fog client is corrupt or wrong, this may propagate through the network. Trust is therefore an important part of the evaluations scheme for fog nodes. Determining if the data collected from a fog client is truthful can be done through trust.

As a consequence of the issues presented trust management solution for fog computing needs to detect intentional and accidental misbehaviour, and conduct punishment and redemption of trust [YQL15b].

4.2 Trust scheme requirements in fog computing

This section describes requirements for trust in fog computing. J. Cho, A. Swami and I. Chen [CSC11] defined five trust requirements for mobile ad-hoc networks (MANET), which will also hold for fog computing.

1. **Trust is dynamic** Trust in fog computing needs to be dynamic for two reasons. Firstly the network topology in fog computing will continuously change. New objects will join and leave the network. Secondly, the objects in the network may alter their behaviour. Therefore, trust needs to be computed continuously. Things change rapidly in the fog network, therefore the trust values of an entity needs to be dynamic and not static.

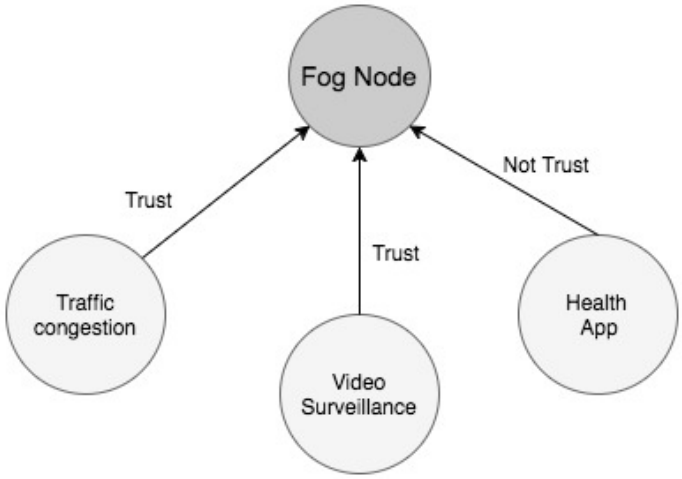


Figure 4.1: Trust in fog computing is subjective. Different applications will have different trust policies, and the trust management schemes needs to take this into consideration. Here the Health application has stronger requirements to the service needed from a fog node than the node illustrated can deliver, therefore the Health App does not trust the fog node. The service offered by the fog node fit the requirements of the Video Surveillance and Traffic Congestion application, and therefore they chose to trust the fog node.

2. **Trust is subjective** Each object may have different security requirements compared to other objects in a network. Each object in the network will have different trust properties that are important and carry more weigh than other properties. Because of the different trust policies of objects in the network, trust is subjective. Figure 4.1 demonstrates this property.

3. **Trust is not necessarily transitive** This is connected to the previous point. Each object has its own security policy. This is shown in Figure 4.2. If fog client E, trusts fog node A, and fog node A trusts fog node B, fog client E will not necessarily trust fog node B.

4. **Trust is asymmetric** If fog client E finds fog node A trustworthy, fog node A can find fog client E not trustworthy. Figure 4.3 exemplifies this.

5. **Trust is context dependent** Context is important in fog computing [KBTP17], and is therefore also important in trust computing. An example from the real world is that you may trust a friend to keep a secret, but not to cook food for you. The same goes for fog computing: one fog node can be trusted to perform a specific task for a fog client, for example storage. However, if the

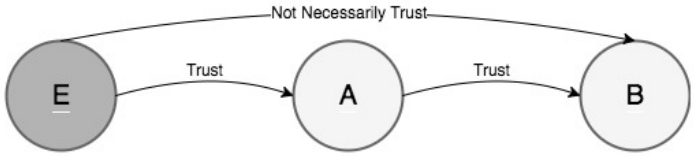


Figure 4.2: Trust is not always transitive. Even if **E**, trusts **Node A**, one cannot assume **E** will automatically trust **Node B**.

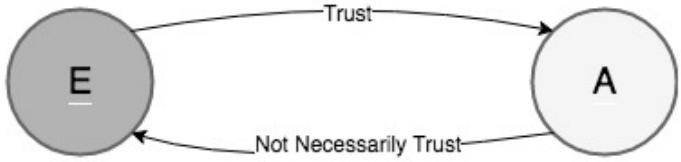


Figure 4.3: Trust is asymmetric. If **E** trusts **A**, **A** should not automatically trust **E**. The trust from one device to another are independent.

fog client later requires real-time processing the fog node may be deemed as not trustworthy.

4.3 Attacks on the trust computation

Fog nodes and fog clients have much to gain by having a high trust value in the network. Fog nodes with high trust values will be selected more times than fog nodes with lower trust value, making them more profitable. Fog clients with high trust value will be accepted more than fog clients with low trust value [GCT17]. Malicious nodes may try different attacks to get a higher trust value then they should have. These attacks are described in this section.

Self-promotion attack (SPA) When a malicious node gives good recommendations about itself.

Bad-mouthing attack (BMA) Malicious nodes can go together and give bad recommendations about a good node. This will decrease the good node's reputation. This is a form of collusion attack, which is the term used when several nodes work together to spread false information.

Ballot-stuffing attack (BSA) Another form of collusion attack. A malicious node sends good recommendations about another malicious node to increase the reputation of the malicious node.

Opportunistic service attacks (OSA) When a malicious node senses that its reputation has dropped, it can perform good services to regain its reputation.

On-off attack (OOA) A node can perform good and bad services randomly to avoid being labeled as a bad node.

Chapter 5
Proposed Solution

In this chapter a solution for trust management in fog computing is proposed and evaluated. The solution is introduced as an example for further discussion. The proposed solution described in this chapter is built on the five dimensions discussed in Section 3.2.2. The proposed solution is evaluated according to the requirements and attacks introduced in Chapter 4, in Section 5.5 of this chapter.

5.1 Solution briefly explained

An overview of the solutions is given below as background for the more detailed descriptions that follows.

Recall the five dimensions defined in 3.2, this proposed solutions uses:

Trust composition: Both QoS and Social trust

Trust Propagation: Distributed and Centralized

Trust Update: Both Time-driven and Event-driven

Trust Formation: Multi-trust

Trust Aggregation: Subjective logic and Logistic regression

In short, it is the fog nodes task to compute trust values in the proposed solution. Fog clients query other fog nodes for recommendations, and the fog nodes answer the query. The fog nodes also compute and forward the trust level to the requesting fog client. The proposed solution is a combination of a distributed and a centralized trust management system.

Fog nodes use logistic regression, introduced by Wang et al. [WLC+15], described in Section 3.2.7, to build a model for the trust of the other fog nodes in the network.

This model takes environmental variables into account. These variables – also called conditions – describe the environment around the fog node. These conditions can, for instance, be how many fog clients it is currently serving, and what service requirements the fog client has. By using logistic regression, trust is computed not only based on the experiences of others, but also on what is currently happening in the network at the time. Fog clients in need of real-time processing require more of power than an application, for instance measuring or collecting temperatures. Trust is the level of assurance that the fog node can deliver the service requested: since the level of service is different, this should be taken into consideration when computing the trust.

Fog clients store the experiences with fog nodes, and it is based on these experiences the models in the fog nodes are built. The hierarchical property of fog computing is used. The experiences of fog clients are propagated higher in the hierarchy. This propagation makes it possible for nodes up in the hierarchy to have a better overview and detect fog client and fog nodes with low trust value. Note that this solution assumes a persistent, unique and distinct identity of each object, which this is still an open research question [YQL15b].

5.2 Notation

Each **fog node** holds a logistic regression model, s_j, representing the trust in the other fog nodes in the network. Instead of the resource constrained fog clients building the trust model, as LogitTrust [WLC+15] requires, this task is offloaded to the fog nodes that have more computational power. The set of logistic regression models is $\boldsymbol{s}_i = \{s_j, j = 1....N\}$, N being the number of fog nodes the fog node i has a model for in the network. This model is built on a data set of experiences, collected by fog clients that share their experiences with fog node i. $\boldsymbol{E}_F = [t_k, \boldsymbol{x}_k, k = T_1...T_m]$ is the experiences collected by the fog clients about fog nodes in the network, where the binary number t_k, marks if the client rated the fog node as trustworthy or not, and $\boldsymbol{x}_k$ is the conditions the application had in the period the experienced happen. Also, each fog node has a database that contains the number of positive and negative experiences with each fog client it has encountered recently.

Each **fog client**, here considered as a resource constrained device that is requesting a service from the fog node, has $\mathbf{T} = \{T_i, i = 1....M\}$ which is a collection of trust values for each fog node it has recently been in contact with, M being the number of fog nodes stored. The storage capacity of the fog client determines how many node interactions are stored. Each fog client also stores the experienced service level every interaction. $\boldsymbol{E}_I = [t_i, \boldsymbol{x}_i, i = T_1...T_H]$, t_i is a binary number, marking if a node has had acceptable service, and $\boldsymbol{x}_i$, being the environment conditions in the network at

the time of service. Each experience is connected to a specific fog node. H is the number of experiences it has stored.

$\boldsymbol{E}_F$, which is stored in the fog node, is a collection of $\boldsymbol{E}_I$ from different Fog clients.

Conditions are the variables that are inputted into the logistic regression model. See Section 3.2.7 for more information.

5.3 Trust Estimation

In this section, the proposed solution will be described in detail. It will be divided between fog nodes and fog clients as they have different solutions for the trust assessment.

5.3.1 Fog client to Fog node trust estimation

Fog clients use **T** to find other fog nodes to perform the trust estimation for it. By using **T**, each fog client can build up its own trust network based on its previous experiences.

Trust estimation from a fog client to a fog node has four steps. Figure 5.1 shows these steps. A fog client, i wishes to connect to j, which is a fog node with more processing power. By using **T** it is able to identify fog nodes it trusts, and can request recommendations from these fog nodes. In Figure 5.1 these recommendor fog nodes are F_1, F_2, F_3, F_4.

Step 1: fog client i asks fog nodes F_1, F_2, F_3, F_4 for recommendations on the trustworthiness of fog node j under a set of conditions. When asking for recommendation it sends a message in the format m = $\{j, \boldsymbol{x}\}$, where $\boldsymbol{x}$ is the environmental conditions that the trust is dependent on.

Step 2: Fog nodes F_1, F_2, F_3, F_4 all have a trust model s_j, which is a logistic regression trust model for fog node j. A trust estimation for a fog node j under a given set of conditions is produces by s_j. Details about logistic model are described in Section 3.2.7. $T_{F_k j}$, the trust recommendation, is either 0 or 1. 1 being that the node j is trustworthy under the given conditions, and 0 meaning node j is not trustworthy under the given conditions. As described in Section 3.2.7, the model will produce the probability that the node is trustworthy. If the probability is over 50 %, $T_{F_k j}$ will be 1. If the probability is under 50 %, $T_{F_k j}$ will be 0. This recommendation is binary to simplify the messages sent.

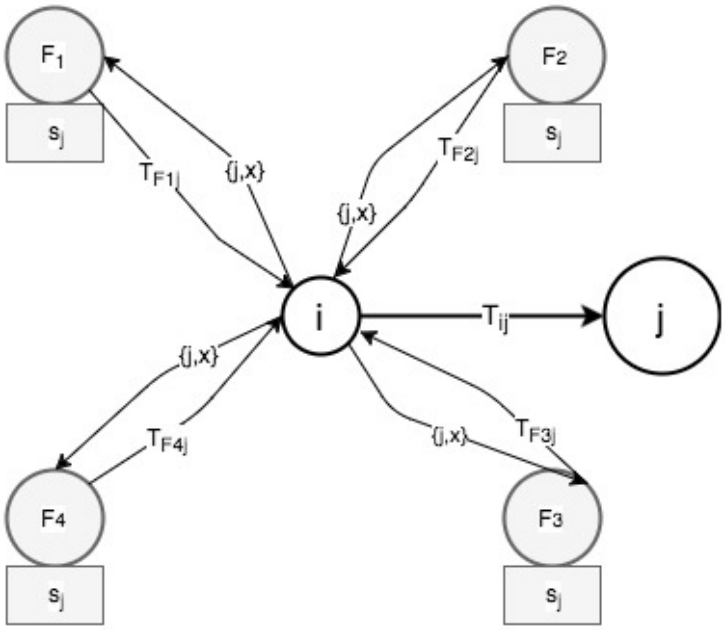


Figure 5.1: Fog client i wants to estimate the trustworthiness of fog node j. It asks the surrounding fog nodes to estimate the trustworthiness of node j under a set of conditions. Fog node F_1, F_2, F_3, F_4 all have a regression model for the trustworthiness of fog node j. The received recommendation is scaled according to fog client i trust to the recommendation.

Step 3: Fog nodes F_1, F_2, F_3, F_4 return the trust level computed T_{F_kj} to the fog client i.

Step 4: Fog client i finds the trust estimation for node j by combining the trust it has for fog node F_1, F_2, F_3, F_4, and the result returned from each. The trust the fog client i has for the fog nodes F_1, F_2, F_3, F_4 is found using $\mathbf{T}$. The trust fog client i has to fog node j is found through $T_{ij} = (T_{F_1} \wedge T_{F_1j}) \wedge (T_{F_2} \wedge T_{F_2j}) \wedge (T_{F_3} \wedge T_{F_3j}) \wedge (T_{F_4} \wedge T_{F_4j})$. This is the combination of the recommendation and the trust fog client i has to the recommendors. Based on this computation, the fog client will decide if the fog node can deliver the service it requires or not. T_{ij} is the probability that j will act in a way i finds satisfactory. The probability needed to conclude that j is trustworthy will depend on the application deployed on i. Some application conclude on 50 %, others need 80 %.

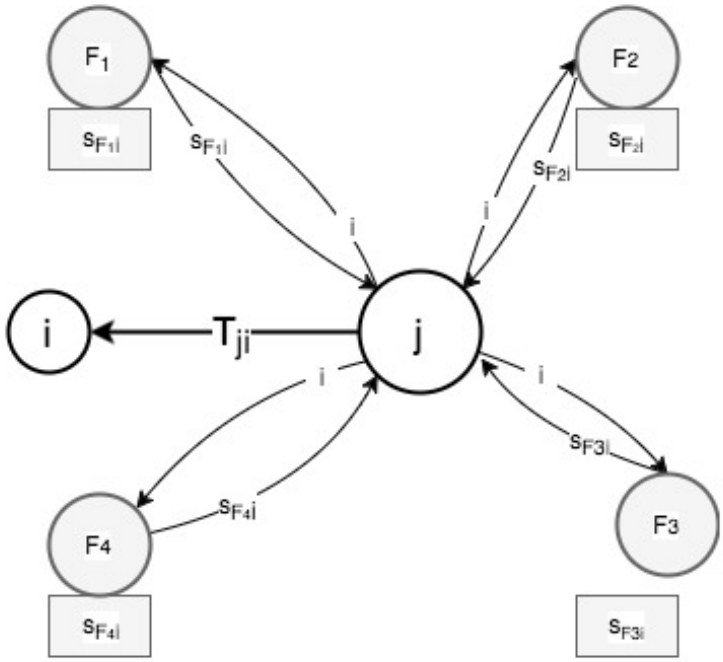


Figure 5.2: Trust from fog node j perspective. Fog node j asks the surrounding nodes for a recommendations about fog client i. Based on these recommendations and own experiences, fog node j forms a estimation about the trust level of the fog client i.

5.3.2 Fog node to Fog client trust estimation

Figure 5.2 describes the same scenario as figure 5.1, but from fog node j perspective. A fog node's trust to a fog client will not be dependent on the service it wants. This solution assumes that the fog client has requested the same service from other fog nodes. Logistic regression will not be used to compute the fog trust for the fog clients. Logistic regression is computational heavy, and there is not necessary to build a model for each fog client. A limited amount of data will be collected about the fog client's behavior, and the trust from the fog node to the fog client will not depend on the environmental conditions. Subjective logic is therefore used to estimate the trust the fog node can have to the fog client. More on subjective logic can be found in section 3.2.7. However, the logistic regression model built for other fog nodes in the network will be used to scale the recommendations from other nodes.

The following defines the steps needed for the fog node to determine the trust of a fog client.

Step 1 Fog node j sends out a request for trust recommendations on fog client i to other nodes in the network. If none of the fog nodes on the same level has any previous experience with the fog client, i, fog node j will ask the nodes higher up in the hierarchy.

Step 2 Fog nodes F_1, F_2, F_3, F_4 reply to the request, as they have experience with the fog client i. The reply is in the format $s_{F_m i} = (b_{F_m i}, d_{F_m i}, u_{F_m i})$, which is the subjective logic tuple explained in section 3.2.7.

Step 3 Fog node j scales the recommendations from each node, using the logistic regression model. Variables inputted into the model are static. The final recommendation from fog node F_1 has the form $T_{F_1 i} = (s_{K_1 i} \wedge s_1)$. s_1 is the result of the trust model fog node j already has stored, and $s_{K_1 i}$ being the subjective logic recommendation received from F_1.

Step 4 Fog node j combines its own subjective tuple of the fog client, if one exists, with the scaled recommendations. The final computation will be $T_{ji} = s_{ji} \oplus T_{F_1 i} \oplus T_{F_2 i} \oplus T_{F_3 i} \oplus T_{F_4 i}$. s_{ji} being fog nodes j the own experience of fog client i. The final result will be a tuple of the degree of belief, disbelief and uncertainty. How much belief and disbelief about the behaviour of the fog client is allowed is defined by the individual fog node.

5.3.3 Argumentation

Two different Trust Aggregation methods have been chosen for several reasons. Logistic regression is used to predict the future behavior of the fog nodes because more information about their will be available. Environment conditions will most likely have a bigger impact on the fog node's ability to deliver a service than a fog clients. Subjective logic requires fewer data and fewer computations, which means less would have to be less to store about each fog clients behavior. There will be a larger number of fog clients in the fog network than fog nodes, building a model for all fog clients is a substantial task. Building a logistic regression model is view as an unnecessary task.

5.4 Trust Update

Trust needs to be updated for the solution to be dynamic. Objects can change their behavior, and the trust management system needs to take this into consideration. The trust updates in the proposed solution are described in this section.

As mentioned, fog clients store the recent activity in $\boldsymbol{E}_I$. The size of $\boldsymbol{E}_I$, will depend on the resources of the fog client in question, but the recent activity will be the most important. This is because fog nodes may change their behavior over time,

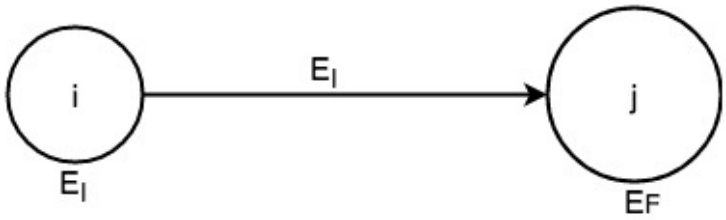


Figure 5.3: After complication of a service the fog client, here node i transfer the previous experiences to the fog node j. Node j uses these to update the logistic regression model.

and recent activity is more valuable than older behavior. $\boldsymbol{E}_I$ is sent by the fog client to the fog nodes after a service is completed. Object i would send $\boldsymbol{E}_I$ to object j in the previous example, this is also show in Figure 5.3.

Fog nodes use $\boldsymbol{E}_F$ to build their regression model. For the model to be as good as possible and secure against attacks, experiences from multiple fog clients need to be taken into consideration. $\boldsymbol{E}_F$ is a collection of $\boldsymbol{E}_I$s from different fog clients. Figure 5.3 shows the transferring of $\boldsymbol{E}_I$.

Updating the regression models can accrue at any point. When enough new data has been collected, a fog node can update its model. Updating the model implies retraining the model. From Section 3.2.7 we know that training the logistic regression model takes $O(NC^2)$, N being the number of data points, and C is the number of variables.

Higher level fog nodes also update their trust models. Nodes lower in the hierarchy are in direct contact with the fog clients and get experiences from these. Nodes in higher levels need these experiences in the same way as the lower level nodes. $\boldsymbol{E}_F$ is sent from nodes on the lower levels and combined at the higher level nodes. Figure 5.4 shows this solution.

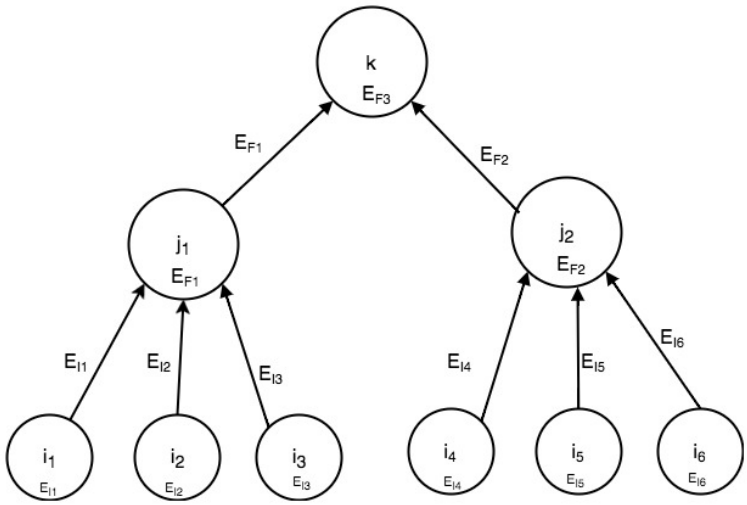


Figure 5.4: Experiences of the resource constrained fog clients, here marked as $i_1, i_2, i_3, i_4, i_5, i_6$ are propagated through the hierarchy. Node k will make its own trust models based on the experiences of the resource constrained devices.

5.5 Evaluation

In this section, the proposed solution is evaluated according to the trust requirements and robustness to withstand attacks defined in Chapter 4. No simulations have been performed; this is a purely theoretical evaluation.

5.5.1 Evaluation trust requirements

This section describes how requirements named in Section 4.2 are upheld.

Dynamic The trust model is recomputed with regular time intervals. More weight is put on short-term activity, than on long-term activity to prevent an object from building up a reputation, only to start acting maliciously. Fog nodes recompute the model for other fog nodes in the network periodically. How often may depend on the amount of traffic currently in the network, and how often the malicious fog nodes are foreseen to change their behavior.

Subjective Each object computes the trustworthiness of the other party, making the trust subjective. This is beneficial because each object is different. Every object may have a different set of security requirements and functional requirements. Functional requirements, meaning requirements such as latency or storage capacity. Combining recommendations that all depend on the environment the application is running in, and scaling these recommendations accordingly will make the trust level reached subjectively.

Not Necessary Transitive In the proposed solution an object does not fully rely on the recommendation from others. Figure 4.2 describes the transitive requirement. **A** and **B** may provide different processing power, so even if **E** receives appropriate service from **A**, and **A** trusts **B**, **E** can not directly expect the same level of service from **B**. Both the environmental variables and the scaling of the recommendations makes the not necessary requirement true. However, if **A** and **B** provide the same service levels, and **E** trust **A** fully, the trust is transitive.

Asymmetric In the proposed solution, fog client **E** and fog node **A** compute the level of trust independent of each other. The trust computed is based on different algorithms, and on a different set of recommendations. So even if a fog client **B**, find **A** trustworthy, **A** may find that **E** is not trustworthy, based on previous experiences.

Context Dependent Each application running on a fog client may have different requirements, and the fog nodes operate in different environments. The proposed solution makes it possible to determine the level of trust with regards to these conditions.

5.5.2 Evaluation based on protection against attacks

Two behaviors are possible with regards to trust in a fog network, normal or misbehaving. Normal objects will behave under all conditions and are always trustworthy. Misbehaving objects may alter their behavior. A misbehaving object can either produce false recommendations or feedback in some or all cases or not provide the service required. Further discussion of malicious behavior can be found in Section 6.5. Accidentally and intentional misbehavior will in this solution be treated as the same.

In Section 4.3, several trust attacks were described. How the solution provides protection against these attacks are described below. The evaluation is split in two, because of the nature of the attacks. One section describes attacks based on recommendations, while the other is on an object's changing behavior.

Protection against recommendation attacks

Malicious nodes can provide false information to promote themselves or other nodes in the network. How the solution handles these kinds of attacks is described below.

Self-promotion attack (SPA) In this attack objects can give false good recommendations about themselves. In the proposed solution objects do not provide recommendations about themselves, therefore it is not possible to perform self-promotion attacks.

Bad-mouthing attack (BMA) and Ballot-stuffing attack (BSA) These are collaborating attacks. Multiple nodes or clients can provide wrongful trust information on purpose.

Recommenders provide false recommendations on purpose to manipulate the trust level of an object. Only one recommendation is excepted from each recommender in this solution. When only one recommender provides a false recommendation, this will not impact the decision because recommendations are combined from multiple recommenders.

Recommendations are rated according to the trust level associated with the recommended. If an object produces a wrong recommendation, it will get a lower recommender trust. When combining recommendation, half of the recommenders need to provide the wrong recommendation to provide a wrong answer [WLC+15]. However, in hostile environments where the majority of objects are malicious, this may lead to some problems.

Wang et al. [WLC+15] solve this problem by introducing social relationships. In hostile environments, only recommendations from friends will be taken into account. This is not included in the proposed solution but can be included as a part of the environmental variables.

Social similarity can also be used to rate the recommender. If the social similarity is used, each object holds a list of friends, and when rating the recommendation, an object sees how many friends they have in common, in other words, how likely they are to trust the same people. This should only be used in hostile environments because it adds to the computational complexity. Knowing if an environment is hostile or not is another challenging task.

Protection against malicious behavior attacks

An object's behavior can change over time, both in the long term and short term. Trust management needs to be able to handle these changes. How the solution handles, these type of attacks is described below.

Opportunistic service attacks (OSA) Fog nodes will record all activity of the other objects, both long term, and short term. Long term is weighted less than short term. If an object has been known to misbehave in the past, it should be marked. In this solution, it is hard for an object to regain its trust in the network.

On-off attacks (OOA) A node can perform a good or bad service at random. This attack has not been considered in the proposed solution. Mendoza et al. [MK15] proposed a reward and punishment scheme to protect against these types of attacks in a distributed sensor network. Rewards were given when a service was provided, and punishment was given when a service was not delivered. The punishment has twice the weight of the reward, making it harder to regain the trust. This can also be implemented in the solution.

Chapter 6
Discussion

This chapter will explore still open questions in trust management for fog computing. Two central problems are identified: First, how trust information should be collected and verified, to prevent the attack described in Section 4.3. Second, whether a distributed or centralized storage and trust computation approach should be taken. To be able to rate the behaviour of objects, malicious behaviour needs to be defined. The previous chapter introduced a combination of a centralized and distributed approach and is recommendations based. However, other solutions are possible, this chapter discusses other solutions.

6.1 Trust in different deployment models

Section 2.4.3 describes the different deployment models for fog computing, notably public, private, hybrid and community. Since public and private fog computing infrastructure have different properties, there might be a need for different trust management schemes, meaning that the trust is exchanged differently in the two deployment models. This section highlights the main differences needed for the trust management in public and private fog computing.

With public fog computing, the general public will have access to the infrastructure. Unlike cloud computing, the infrastructure may not be owned by only one organization. Several organizations or even individuals can set up a fog node. Together these fog nodes will create the public fog infrastructure. Because of the multiple ownership, a common trust management system should be defined. With a standardized system anyone can join the network.

With private fog computing the owner of the infrastructure will likely have some control of the infrastructure [Bit]. Therefore they will be able to define and implement their own trust management system. In cloud computing the owner can configure the security requirements to their wishes. It is not unthinkable that this will be the case in fog computing as well. This may result in less unknown nodes. Maintaining a

direct trust table for all nodes in a private fog computing network may be plausible. In cloud computing it is assumed that all objects in the private cloud have the same context [CdSJdCdOA12], whether this also holds for fog computing is still unknown. However, if it does hold for private fog computing, a general trust value can be determined for every object in the network. This trust value can be stored in a centralized entity, and be utilized by everyone without the need for extra trust computations.

More unknown users are present in the public fog. Security within the fog nodes may also vary more in the public fog than in the private fog.

6.2 Trust in regards to security issues

Authentication is a problem in fog computing, where trust may provide a solution. Stojmenovic et al. [SW14] referred to authentication as the primary security concern in the fog architecture. Trust can be used to ensure that the authentication is valid [SF99].

Intrusion detection can be facilitated by trust management if correctly implemented [GCT17]. Trust management can be used to monitor the system state, if an object has a low trust value, it may be an intruder. Subjective logic, described in Section 3.2.7 is used in intrusion detection for IoT [KH]. Therefore, assuming that trust management can be used for intrusion detection for fog computing is reasonable. One major concern with this is the reliability of the trust information, which is a concern connected to the second main issue identified.

6.3 Other possible trust tasks

Trust can also have other duties than facilitating node selection.

Scalability is a characteristic of fog computing. New fog nodes can be added in locations where they are needed. Actions can be taken if several fog nodes in one area have a low trust value. Reduced trust value in an area can be a sign of nodes misbehaving due to insufficient processing power. Additional fog nodes can be added to relieve the workload of the current fog nodes in that location. Reduced trust value can also insinuate an attacker in the vicinity.

Handovers are essential when providing mobility. Handovers should happen when needed [CN16]. If a fog node senses that a fog client can receive better service from another node, it should provide a handover. Trust management can help in determining whether a fog node should execute a handover: By evaluating the trust of the surrounding nodes, the best surrounding node can be selected.

Ensuring user privacy can be done by a trust management system. Location can reveal personal information [AA16]. Fog computing has location awareness [BMZA12], therefore it is important to protect the location of fog clients. If, for instance, the smart grid is employed, location can be used to detect whether a person is home or not. If a client always selects the closest fog node, the node may reveal where the fog client is located. Fog nodes can be selected by considering for instance latency, reliability and reputations instead of location. If trust is evaluated on the grounds of what fog node can perform the best service, one can not assume that the closest fog node will be the best fit. Fog nodes will know that the client is in the vicinity, but not the exact location. Trust may therefore be one way to protect the fog clients location.

6.4 Other possible trust management systems for computing

A recommendation based solution is not the only viable solution. This section describes blockchain and hardware components, which are two other suggested solutions.

6.4.1 Blockchain

Recently blockchain has been introduced to deal with trust in a decentralised network. IBM Watson has used blockchain to compute the trust of unknown IoT devices [IBM]. The blockchain is, in short, a ledger which stores all transactions [FF16]. Each block holds a set of transactions, and all blocks are linked together. All participants of the network have access to the same information stored in the blocks. Data on the chain is immutable, appending new data is the only action allowed. When a transaction is added, all the participants in the network will verify it. Majority rules when deciding if a transaction is valid. By using a blockchain you can trace an objects life-cycle, its creditability and authenticity [Howa]. What these transactions are, and how they are created and stored is not yet solved. Scalability and storage introduce problems when the blockchain technology is introduced to establish trust [ASe]. There will be a huge set of transactions stored on the ledger since the blockchain ledger will trace everything in the IoT domain. The ledger of transactions will grow, and resource constrained fog clients will not be able to store it. However, fog nodes may be able to store the ledger. When a fog client requires information on the ledger, it can ask the fog nodes for that particular information.

6.4.2 Hardware trust

Building trust models on hardware is also a possibility [YQL15a]. OpenFog Consortium [CW17] recommends a Hardware-Root-of-Trust. The Hardware-Root-of-Trust is a trusted hardware component which receives control at power-on. It then ex-

tends the chain of trust to other hardware, firmware and software components. The Hardware-Root-of-Trust can be useful in confirming the identity of an object [Har]. If an object has been altered in any part of the software, hardware or other important components that form the chain, the chain of trust will be broken, and the object is no longer seen as trustworthy. Hardware-Root-of-Trust can provide assurance that an object has not been tampered with. However, the hardware root of trust can not provide assurance that a node can perform a specific task.

6.5 Malicious Behaviour

Malicious objects should cause an object to be marked as untrustworthy. However, malicious behavior is not defined for fog computing. Fog clients and fog nodes can act normal or malicious [SW14]. Khari and Saini [SK11] defined malicious behavior in an ad-hoc network. Since ad-hoc networks interactions and fog client - fog node interactions are similar, these definitions may be applicable in fog computing as well. In ad-hoc networks, the definition of normal behavior is that the object performs operations while maintaining the security principles [SK11]. A malicious object in an ad-hoc network is an object that breaches any of the security principles.

However security principles for fog computing have not been defined. Confidentiality, Integrity, Availability, Authenticity, and Non-Repudiation are standard security principles used in multiple environments [Sta10]. Since these are standard principles used in multiple environments, one can assume they will be important in fog computing as well.

The possible malicious behavior for fog nodes is defined below. The malicious behavior is based on Khari and Saini [SK11] malicious behavior for ah-hoc networks.

- **Packet Drop** - A fog node can drop packets on purpose.
- **Battery Drain** - A fog node can drain the battery of a fog client on purpose, for instance by keeping it in a highly energy consuming state.
- **Nodes with no authentication** - A rouge node with no authentication can enter the network. It can try to listen on the network and gain access to private information.
- **Packet Delay** - After completing a computation, a fog node can on purpose delay sending the answer back to the fog client.
- **Message Tampering** - Tampering with data before sending it to the cloud for further storage and analysis.

- **Bandwidth Consumption** - Overflowing the bandwidth available with unnecessary data.

- **Information Stealing** - Storing information about fog clients that it is not supposed to keep.

A fog client can also behave maliciously. Malicious attacks that can be performed by fog clients are described below.

- **Buffer Overflow** Fill the buffer of a fog node with fake updates, making it unavailable for others.

- **Bandwidth Consumption** Same as with fog nodes, this makes the network unavailable for other clients.

- **Message Tampering** Sending wrong and false information to the fog node. For instance, be invalid temperature measurements.

Attacks described above need to be detected in the network, and malicious nodes need to be isolated for the network to work optimally. Discovering this kind of behavior can be done by a trust management system.

Bad behavior can be intentional and unintentional and are often hard to tell apart. Most misbehaving nodes will have a motivation behind the behavior. However, unintentional misbehavior can happen, due to lack of resources, or placement in the network [BLB03]. Unintentional behavior can happen because an object is malfunctioning. An object should be marked as a malicious if it behaves badly, not based on the reasons behind it [BLB03].

How to motivate for normal behavior among objects is still an open question.

6.6 Trust when all devices are known

In this section trust establishment between two devices that are known in the network will be discussed. We are assuming that the fog node, fog client, and the owners, are known in the network, and other objects can provide information about the trust regarding that object. A network is a collection of objects that have knowledge about each other. Problems regarding trust when a new object enters the network will be discussed in Section 6.7. Figure 2.4 can be viewed as an example of this network.

6.6.1 Fog node

This section will focus on problems fog nodes encounter when determining the trustworthiness of other objects in the network. Fog nodes should determine the trustworthiness of the fog clients and peer fog nodes.

Fog Client Evaluation

Fog client may not always produce and send the correct data. Decisions are made based on the data collected, and therefore it needs to be correct. Several things may influence a fog clients ability to collect the correct information. This section focuses on remaining challenges before trust from a fog node to a fog client can be computed.

The consistency of data collected can be used as a factor to evaluate the truthfulness of data collected by a fog client [ML15]. If the data collected is consistent, there is a higher chance that the data is correct. Highly inconsistent data will occur when the device is broken [ZLTV10].

Virtualization techniques are envisioned for fog computing [VRM14]. Containers can be handed from fog node to fog node. In these containers, the application for the fog client is held. Tracing the behavior of the container reveals previous behavior. Each fog node will hold experiences with the container, instead of the fog client. Tracking containers can also conceal the identity of the fog client.

Each container has an application running inside it. Evaluating this application can also be a possible solution for the trust evaluation of a fog client. Some applications can be known to be less trustworthy, possibility because of poor coding, while others application are seen as safe. Evaluating the application reputation together with the client's reputation may lead to more accurate conclusions.

Applications also have updates. If applications are used to evaluate the trust level, updates may impact the trust level of an application. To what degree is not known. Whether a fog client with updated software is more, less, or equality trustworthy compared to previous interactions with the fog client is not yet known.

Some manufactures, i.e., the company that produces the client, may have devices that are known to misbehave more often than others. Manufactures may have known security issues. Some can be known for having good security on their produces, others not. If little information is available about a fog client, this may be a possible measurement. A manufactures reputation will likely be more relevant in the early stages of interaction. If enough experience is collected, newer experiences will likely be more relevant.

To gain insight into which trust variable reveals most about a fog clients future

behavior, trust information from the fog client need to be collected and analyzed. Trust variables here relate to the ones discussed above, such as manufactures, applications, and computing resources. By determining which set of variables provides the most accurate predictions, the trust management will gain better results. In Chapter 5 the proposed solution uses a machine learning technique to rate the relevant trust variables. However, not all possible variables should be collected and stored. Analyzation of the connection between trust variables and trust value is therefore important to determine a finite set of variables that should be collected.

Fog client behavior may change during transmission or connection. Trust values are computed when nodes initially establish communication. However, it might be a possibility that a fog client turns malicious during connection. It is necessary for the trust management system to predict the future behavior for the whole connection period. The trust values should, therefore, have a period, when this period experience the trust should be recomputed.

Peer Evaluation

Peer-to-peer communication is possible in fog computing [CW17], and trust needs to be established between peer fog nodes as well. The proposed solution in Chapter 5, the fog nodes collect trust information from fog clients. False trust information is a possibility in this scenario, e.g., fog clients can perform trust attacks. A central entity can provide the trust information for establishing peer-to-peer trust among devices. However, a trusted central entity is a single point of failure. Problems regarding this will be discussed in Section 6.9.

6.6.2 Fog Client

This section identifies several issues regarding the establishment of trust from the fog client to the fog node. Among the issues is the most central issue is, which trust properties to consider when computing the trust value of a fog node. The proposed solution from Chapter 5 allows for each client to set their security policies, in the form of environmental variables. There might be a trade-off between the amount of environmental variables available and the size of this set. As with the fog nodes, it is important to define the most relevant trust properties.

Fog node Evaluation

Latency, quality of service, response time, computing power, failure probability, prices, the correctness of computation and energy status can be variables of the fog node that can affect the trust level. Some fog clients may value low latency, while other clients are more concerned with the correctness of computation. Verifiable Computing can be used to check the work of a fog node [YQL15a], leading to a

measurement of computational correctness. If the verification returns true the trust level is increased, else it is reduced.

Trust may be based on the capabilities of the device. E. D. Canedo et al. [CdSJdCdOA12] performed node selection and trust evaluation in the private cloud based on the nodes storage space, operation system link and processing capacity. Some operating systems may be seen as more secure than others. While a larger link means that a fog node has a higher ability to receive and transmit information. Evaluating the capabilities of the fog nodes may be useful in determining the level of trust to put on a fog node.

In the proposed solution, described in Chapter 5, fog clients store interactions with recently encountered nodes. The fog client remembers some fog nodes and gains trust to new fog nodes via the recommendations of these fog nodes. For memory constrained fog clients storing all encountered fog nodes is unfeasible [GCT17]. J. Guo, I. Chen and J. Tsai [GCT17], suggest that objects can store the objects with the highest trust value. This solution can also be used for fog clients. Fog clients can store fog nodes with highest trust value, together with a reasonable number of recently encountered fog nodes. Storing recently encountered fog nodes will help a fog client determine trust levels to other fog nodes in close vicinity in a distributed trust management system. However, if a centralized approach is taken, fog clients may share the experience with the central entity right away, and no storage is necessary. In a centralized approach, the fog nodes only need to remember the central entity.

Trust management can help in the decision making process [SHL08]. If a fog client decides that a node is trustworthy, it will connect – else it will not. However, a fog client may conclude that a whole network of fog nodes is untrustworthy. In other words, no fog node in the network is assumed to be trustworthy. In this scenario, the fog client can process the data collected itself, or send information direct to the cloud. Increasing the security is also a possibility for establishing trust.

6.6.3 Owner

Each object has an owner, no matter if it is a fog node or a fog client. An owner can be a company or an individual, such Alice in the use case. An object's owner, if known, can be taken into consideration when computing the trust level of an object. The weight of the owner relationship on the trust level computation is not known.

An owner may be malicious. An owner can be marked as malicious if one of his objects are marked as malicious by the network. If one of the owners objects misbehave, a natural conclusion is that the owner's other objects will be likely to misbehave as well. However, the relationship between an owner's objects behavior in fog computing is not known.

6.7 New members of the network

In this section, we will focus on when a new object enters the network. Challenges regarding the trust evaluation raised in the previous section are still relevant, but only questions regarding unknown objects will be discussed in this section. A new object can be a fog node, a fog client or an owner. A network here is still defined as a collection of nodes that has knowledge about each other.

6.7.1 New Fog Node

This section will focus on when a new fog node enters a network. Three scenarios are considered when evaluating a new node in the network. First, has this fog node been in the network before under a different ID? Second, has this fog node been in other networks, and can we get information about it from them? Finally, how do we handle entirely new fog nodes?

Scalability is, as mentioned in Section 2.4.4 an important benefit of fog computing. Both companies and individuals can add new nodes. If the traffic in one area is too big to handle for the current fog nodes, another fog node can easily be placed to offload the others. Therefore, it is important to evaluate the trust challenges when a new fog node is added to the network. Network topology changes constantly. Meaning that fog nodes can change networks. If a recommendation based trust management system is used the fog nodes should be most concerned with establishing trust to other fog nodes in the network.

Is it a new fog node?

A possible attack can be that a fog node vanishes from the network after being marked as a malicious node, to appear later disguised as another node. This is possible if the IDs are mutable. To protect against this attack, an immutable ID scheme must be found. An identification scheme has not been found for neither edge or fog computing [Sat16]. If it is a possibility that nodes can alter their ID, they will be marked as entirely new nodes; this will be discussed on the next page.

New in that network

New fog nodes join and leave networks constantly. Individuals can, for instance, have their personal fog node that they carry around, such as a smart phone. When a node enters the network, it will not have any knowledge of the surrounding nodes. The main challenge in this scenario is to establish the initial trust values between the new node and the new network.

Recall from the introduction of this discussion, that distributed versus centralized is one of the two major questions regarding trust in fog computing. Centralised

versus distributed is discussed in depth in Section 6.9. The centralized approach will have advantages in this scenario. The new fog node and the new network can query the central entity for the trust value.

Another approach to solving this problem is a decentralized approach. In Section 6.9.3 we will discuss whether the hierarchical structure of the fog network can be used to solve this problem. In a decentralized approach, there will be multiple entities the control the trust exchange. These entities can communicate with each other, to update and rate each other.

Completely new fog node

In the two previous scenarios some information is known about the new fog node. However, in this scenario nothing about the previous behavior of the fog node is known. Establishing the initial trust must, therefore, rely on other parameters than previous experiences.

One solution is to assign an initial trust value to each fog node. All new nodes can be assigned the same initial trust value, and from this increase or decrease the trust value based on the behavior. The new node, in turn, must initially trust the network, and build up experiences. After enough experiences have been collected the trustworthiness can be reevaluated.

Variables discussed in Section 6.6 can be used to form the initial trust value. If the owner is known in the network, the trust value from the owner can contribute to the initial trust value. A new node with unknown owners is discussed in Section 6.7.3. If it is found that the computational capabilities have an impact on the trust value, the capabilities of the new node will contribute to the initial value.

6.7.2 New Fog Client

A fog client can enter a new network, as an entirely new fog client with no previous interactions, or an old client disguised as a new client in a network. For instance, Alice's glasses from the use case in Section 2.6 are fog clients. These glasses can appear in new networks if Alice moves to a location she has never been before. The first time Alice uses the glasses, she will be a completely new fog client.

The following section will focus on challenges that occur when a new fog client appears in the network.

Is it a new fog client?

A client can try to reconnect to the network with a new ID, similar to fog nodes. Old clients disguised as new should be detected by the trust management scheme. A

unique identification scheme for fog clients is necessary to mitigate.

New in that network

Network topology changes constantly [Sat16], objects join and leave the network. A client is considered new in a network when it appears in a network for the first time or joins after an extended period so that all evidence is no longer relevant. However, in this section only clients that have previously visited other networks are considered. Let us say Alice is in a new location in the city where she has never been. Alice's glasses need to establish trust to the network, and the network needs to establish trust to the glasses. Both the client's trust establishment to the network, and the networks trust establishment to the client is considered below.

- **From fog client side** New fog clients need to establish trust to a network to be able to start a collaboration. As with fog nodes, Section 6.7.1, the problem relates to one of the two main challenges, centralized versus distributed. If a centralized approach is taken, trust establishment for new fog clients is easy. A central entity can be queried. However if a distributed approach is taken, and the network will have no information about the new fog client.
- **From fog node side** When a new client enters the network, the network needs to establish assurance that the new client is trustworthy. If a central entity is used, all trust values are stored there. This centrally stored trust value can be computed based on all interactions all fog nodes have previously experienced. A problem here is whether all this information collected can be trusted. A global and local solution can also be deployed. The network can query a central entity for a global trust value, after receiving this global value, it can be modified based on the behavior of the client in the network.

Issues regarding centralized versus distributed storage and trust computation which is a central issue here will be raised again in Section 6.7.1.

Completely new client

A new fog client with a known owner can appear in the network. For instance, if Alice gets a heart rate monitor that also sends data to the fog network. New fog clients with unknown owners are discussed in Section 6.7.3.

How to establish the initial trust value is the main obstacle, as with fog nodes in Section 6.7.1. Similarly standardizing the initial trust value could be one option. All new client get the same trust value. Or the trust value can be scaled according to properties such as application or hardware components.

The initial trust can be established based on the owner's previous behavior in the network. This technique is based on social trust, introduced in Section 3.2.3, and will be discussed in greater detail in Section 6.8.

6.7.3 New Owner

When Alice first receives her glasses, and she is assumed to have no other objects in any network she is considered a new owner. It is assumed that all fog objects, both nodes, and client, are unknown if the owner is unknown. Two possible solutions for establishing trust to new owners are: through an owners fog objects, or through the owner's social network.

Assuming all fog clients are unknown if the user is unknown, establishing this initial trust of these objects can be hard. Several techniques for trust establishment not connected to the owner has been described so far in this discussion. These techniques can be used in the initial trust establishment. Trust can be established based on energy status, storage space, hardware components. After the trust of an owners object is found, the owner can be assigned the aggregated trust value of his objects.

The second possible solution is to use the social network of the owners. For more on social network see Section 2.2. For more discussion on using social networks, and social trust see Section 6.8. By utilizing the social network, the owner's trust can be found first, before establishing trust to the owner's objects.

6.8 Social Trust

Up til now the discussion has, for the most part, focused on the quality of service trust i.e., how well a service can be delivered, however, social relationships will exist in the network as well. Social trust was introduced in Section 3.2.3, and social relationships among objects in a network was introduced in Section 2.2. This section will focus on social relationships effect on trust.

Trust management for IoT needs to consider social relationships [GCT17]. Since fog computing and IoT is so closely coupled, it may be assumed that social relationships will affect the trust in fog computing as well. A central yet still unsolved question in IoT, and therefore fog computing, is how trust and relationships affect each other.

Social relationships may depend on what goal of trust establishment is, e.g., what trust is measuring. The social relationship will not determine the amount of free processing power in a node, nor the nodes ability to provide a service. Other variables such as free bandwidth, the number of neighboring nodes and failure probability may

have a bigger impact on the capability of providing a level of service than the social relationship.

However, to rate recommendations, social trust may be a good solution. The proposed solution in Chapter 5 uses recommendations to gain assurance that the fog node can provide the required level of service. The accuracy of the final trust score is based on how the fog client rates the different recommendations. If social relationships are built with objects that give the right recommendations, they can also be used to rate recommendations afterward.

In L. Atzori et al. [AIMN12] the owner sets the parameters for the objects social relationships. A relationship based on establishment of trust is possible. If a node provides good service or good recommendations the strength of the relationship is increased. This may again lead to easier trust establishment. If an object has a strong relationship with an other object in the network, it is possible to only rely on that object for recommendations. This may ease the connection process. However, if that object is down, it is not a single point of failure, the object can in that case query other objects in the network.

As mentioned in Section 6.7.3, an owners trust can be found through his or hers social network. Examples of this are found in works by I. Chen et al. [CGB16] and J. Cho et al. [CSC11]. I. Chen et al..[CGB16] uses social contact i.e., the closeness of two social networks to establish trust. If a node detects that a owners social network is similar to itself, it might be more willing to trust.

Social relationships will likely affect the trust establishment, and trust will be more likely to impact the relationship establishment. However other variables will impact these decisions as well. It is not possible to base the trust establishment solely on the social relationships in fog computing.

6.9 Centralised versus distributed

Advantages and disadvantages with both centralized and distributed trust storage and computation will be emphasized in this section. This is one of the major questions regarding trust management in fog computing. Challenges regarding the location of storage and computation of trust have already been discussed in this chapter, especially in Section 6.7. They will be described further in this section.

6.9.1 Distributed

Distributed trust allows objects to autonomously exchange trust observations between each other [GCT17]. This allows for highly subjective trust establishment [GCT17]. Distributed approach delegates the trust computation task to individual entities. No

reliance are put on a central entity. Hence the reliability is increased. A distributed approach requires more processing on the object's side.

However, by putting more processing on the object side, resource constrained fog clients will have to store and process more data than with a centralized approach. Decisions on how much emphasis to put on own observations versus the recommendations from others is also a challenge. In the proposed solution in Chapter 5, the fog client gets recommendations from the fog nodes, and much of the work has to be done by the fog nodes, but the fog client still needs to combine these recommendations.

Additionally, an object will have to determine which object to trust for recommendations. As mentioned in Section 6.7, establishing this initial trust value is a challenge.

6.9.2 Centralised

In the centralized approach, one central entity has the complete overview of the whole system. J. Guo, I.R Chen and J. Tsai [GCT17] states that *"a centralized approach makes more sense"*.

Centralized servers collect all information and perform more informed decisions. Collecting all the information in one entity has several advantages. Since all information is collected in one place, a full overview of the network health can be obtained. This may also be a better solution when mobility is introduced. Problems discussed in Section 6.7 are solved with a centralized approach. All objects query the central entity no matter what, if more than 50 % of the network is trustworthy, this will make the information collected correct as well [WLC+15]. All objects need to trust the central entity.

Single point of failure is a challenge here. Reliability is important in fog computing. Introducing a single entity makes the network less reliable. So far making a centralized entity available for all IoT devices has been a challenge [GCT17]. However, with fog computing a hierarchical structure is being deployed. This hierarchical structure can be used for trust management.

6.9.3 Hierarchy Trust Evaluation

Section 2.4.2 introduces fog as a hierarchical architecture. This architecture can be useful in trust management. In this section a hierarchical solution for trust management is explored as well.

L. Ma and G. Liu [ML15] proposed a hierarchical trust solution for hierarchical cluster of wireless sensor nodes. Both centralized and distributed trust computation

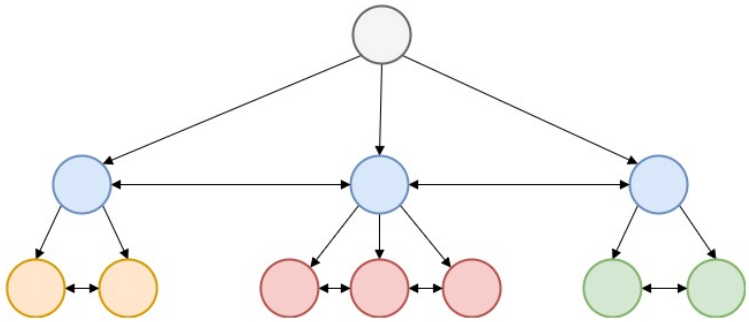


Figure 6.1: Figure shows fog nodes in the fog hierarchy. The arrow shows which nodes have trust values about each other. Nodes on the same level, and in the same sub tree have trust knowledge about each other. The head, the top level node will have overview of everything in the network.

and storage are used in L. Ma and G. Liu's solution. Each cluster has a cluster head (CH), which stores trust values for the sensor nodes (SN) in the cluster in a centralized manner. All CH rate each other, in a distributed manner. In L. Ma and G. Liu's solutions only two levels exist, expanding L. Ma and G. Liu's solution is possible to fit the hierarchy of fog computing.

J. Guo, I.R Chen and J. Tsai [GCT17] called for a hierarchical solution for trust computation in the IoT. They envisioned a hierarchical cloud solution, where the lower level cloud servers keep track of the IoT devices in its direct service area. While the higher level cloud servers keep track of the all the local cloud servers the IoT devices covered by the local servers below it.

Section 2.4.2 introduced fog as a hierarchical architecture, which makes it a possible infrastructure for the kind of solution proposed above. Figure 6.1 shows a hierarchical trust model. A lower level fog node can have overview of the trust level of fog clients in its coverage area and for the fog node on the same level, covered by the same higher level node. We can see that the blue nodes, have a trust level for the nodes in their coverage area, and to each other. While the higher level nodes, in Figure 6.1 this is the gray nodes, will have an overview of both the fog nodes and the fog clients trust values in its coverage area, together with the trust level of the peer level fog nodes.

The proposed solution in Chapter 5 can employ this structure. If a fog client enters a new network, which has a connection to the old network through the hierarchy, the trust values can be handed over.

Chain of trust is used for web-pages. Certificates are signed to make authentication

possible. This chain of trust may also be applied to fog computing. If a higher level node is trustworthy and says one of its lower level node is trustworthy, this may be trusted. Since you trust the higher level node, you implicit trust the lower level node. By going up in the hierarchy as far as you need to find a node you trust, as you do when checking certificates for a web page. Chain of trust employs a decentralized approach. Higher level nodes need to have an overview of the nodes on the next level, but not all trust values are stored in one central entity.

Another aspect to consider with fog computing is that computation can happen anywhere along the path from the fog client to the cloud server. Up to now, the discussion has been based on the evaluation of trust between a fog client and one singular fog node. However, since fog computing has a hierarchical architecture, the network architecture might need to be taken into consideration. If a fog client and fog node establish collaboration, the fog node might send the data up the hierarchy. Therefore if a fog client trusts a fog node, it can trust the whole hierarchy.

Trust can move from the lower layers, up to the higher levels, or from the higher levels down to the lower. If a fog client trusts a higher level node, it can trust the node inn the lower levels. Which solution works best for fog computing needs further investigation.

6.10 Discussion Summary

The malicious behavior for nodes in an ad-hoc network was used as an example of what can be defined as malicious behavior in the fog domain as well.

One central aspect is how to evaluate the trust values of an object. Several properties have been mentioned. Defining the most relevant trust properties is important for to gain a correct trust assessment.

The centralized or distributed management system has been measured up against each other. Both have advantages and disadvantages. A hierarchical solution that combines both has been proposed.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

This thesis explores the need for a trust management system in fog computing. The need for trust management in fog computing has been found by analyzing trust management tasks in similar domains, such as service-orient IoT.

Trust management in fog computing is needed to provide assurance to fog clients that a fog node can provide the necessary services. Fog nodes need to be sure the fog clients will behave according to the fog node's policies. Trust can provide assistance in decision making in fog computing. It can also be used as basis for intrusion detection systems.

Requirements for a trust management solutions have been defined. These requirements are necessary to be able to provide trust in fog computing. A possible solution was introduced that fulfill the requirements.

Recommendation based trust management solutions are the most popular in current literature for distributed networks such as fog computing. However, other solutions are also available. Blockchain and hardware solutions can also be used for trust management in fog computing.

Two open questions have been identified. First is to identify what trust values are most relevant in regards to trust computing. Second is the trade-off between a centralized and distributed approach. A possible solution is to combine both and take advantage of the hierarchical structure the fog computing network.

7.2 Future Work

There are still many open questions related to fog computing and trust management in fog computing.

This thesis introduced a proposed solution, based on existing solutions in other domains. Simulations should be conducted to check the correctness of the proposed solution. Simulations can also determine which of the trust properties that should be which trust properties have should be included in the future trust management solution.

Malicious behavior defines when a fog node is trustworthy or not. However malicious behavior for a fog object has not been defined. It is therefore important to define what classifies as malicious behavior in fog computing, to be able to determine if a fog object is trustworthy or not.

As a mean to gain trust in fog computing and IoT leveraging the hierarchical structure of the fog should be explored. More research should be conducted on a possible hierarchical solution that combines a distributed and centralized solution, as discussed in the discussion. A centralized entity has been hard to establish in IoT up till now, but with fog computing, a central entity can be possible. Therefore a centralized trust manager in the IoT may be possible for the IoT with the use of fog computing, however, more research is needed.

References

[AA16] F. Alrayes and A. Abdelmoty. Towards location privacy awareness on geo-social networks. In *2016 10th International Conference on Next Generation Mobile Applications, Security and Technologies (NGMAST)*, pages 105–114, Aug 2016.

[AIM11] Luigi Atzori, Antonio Iera, and Giacomo Morabito. SIoT: Giving a social structure to the internet of things. *IEEE Communications Letters*, 15(11):1193–1195, 2011.

[AIMN12] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization. *Computer Networks*, 56(16):3594–3608, 2012.

[ASe] A secure model of iot with blockchain - openmind. https://www.bbvaopenmind.com/en/a-secure-model-of-iot-with-blockchain/?utm_source=views&utm_medium=article06&utm_campaign=MITcompany&utm_content=banafa-jan07. (Accessed on 05/15/2017).

[AWS] Aws greengrass – embedded lambda compute in connected devices - amazon web services. https://aws.amazon.com/greengrass/. (Accessed on 05/16/2017).

[Azu] Azure iot edge | microsoft azure. https://azure.microsoft.com/en-us/campaigns/iot-edge/. (Accessed on 05/16/2017).

[BdDPP14] A. Botta, W. de Donato, V. Persico, and A. Pescapé. On the integration of cloud computing and internet of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 23–30, Aug 2014.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. 4:164, 1996.

[Bit] Thomas Bittman. Clarifying private cloud computing - thomas bittman. http://blogs.gartner.com/thomas_bittman/2010/05/18/clarifying-private-cloud-computing/. (Accessed on 05/27/2017).

[BLB03] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for mobile ad-hoc networks. Technical report, 2003.

[BMZA12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, page 13, 2012.

[CdSJdCdOA12] E. D. Canedo, R. T. de Sousa Junior, R. R. de Carvalho, and R. de Oliveira Albuquerque. Trust model for private cloud. In *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, pages 128–132, June 2012.

[CGB16] Ing Ray Chen, Jia Guo, and Fenye Bao. Trust Management for SOA-Based IoT and Its Application to Service Composition. *IEEE Transactions on Services Computing*, 9(3):482–495, 2016.

[CN16] X. Chen and M. Ni. Seamless handover for high mobility environments. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 281–286, Sept 2016.

[CSC11] Jin-Hee Cho, Ananthram Swami, and Ing-Ray Chen. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials Tutorials*, 13(4):562–583, 2011.

[CW17] Openfog Consortium and Architecture Working. OpenFog Reference Architecture for Fog Computing. (February):1–162, 2017.

[Dra13] AAse Dragland. Big data, for better or worse: 90% of world's data generated over last two years – sciencedaily. https://www.sciencedaily.com/releases/2013/05/130522085217.htm, 05 2013. (Accessed on 06/08/2017).

[Edg] Edgex foundry. https://www.edgexfoundry.org/. (Accessed on 05/03/2017).

[EMAH04] Stephen R Ellis, Katerina Mania, Bernard D Adelstein, and Michael I Hill. Generalizeability of latency detection in a variety of virtual environments. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 2632–2636. SAGE Publications Sage CA: Los Angeles, CA, 2004.

[FF16] Dongqi Fu and Liri Fang. Blockchain-based trusted computing in social network. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 19–22, Oct 2016.

[GCT17] Jia Guo, Ing Ray Chen, and Jeffrey J P Tsai. A survey of trust computation models for service management in internet of things systems. *Computer Communications*, 97:1–14, 2017.

[GGD+] Nelson Mimura Gonzalez, Walter Akio Goya, Rosangela De, Fátima Pereira, Karen Langona, Erico Augusto Silva, Cristina Melo De Brito Carvalho, Charles Christian Miers, Jan-Erik Mångs, and Azimeh Sefidcon. Fog computing: Data Analytics and Cloud Distributed Processing on the Network Edges.

[Goo] Google glass. https://www.google.com/glass/start/. (Accessed on 03/30/2017).

[Har] Hardware roots of trust for iot security. http://www.techdesignforums.com/practice/technique/hardware-roots-of-trust-for-iot-security/. (Accessed on 05/19/2017).

[HCH+14] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. Towards wearable cognitive assistance. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, pages 68–81, 2014.

[Howa] How blockchain can help solve the problem of trust in iot - dzone iot. https://dzone.com/articles/blockchain-and-enterprise-iot. (Accessed on 05/15/2017).

[Howb] How people are actually using the internet of things. https://hbr.org/2015/10/how-people-are-actually-using-the-internet-of-things. (Accessed on 06/12/2017).

[IBM] Ibm watson iot - private blockchain. https://www.ibm.com/internet-of-things/platform/private-blockchain/. (Accessed on 05/15/2017).

[Ida16] Imen B E N Ida. A survey on security of IoT in the context of eHealth and clouds. pages 25–30, 2016.

[iot] Internet of things defined - tech definitions by gartner. http://www.gartner.com/it-glossary/internet-of-things/. (Accessed on 04/26/2017).

[Iye15] Karthik Iyer. COMPUTATIONAL COMPLEXITY OF DATA MINING ALGORITHMS USED IN FRAUD DETECTION. Master's thesis, The Pennsylvania State University, 2015.

[JKD05] Audun Jøsang, Claudia Keser, and Theo Dimitrakos. Can We Manage Trust? *Trust Management*, 3477(May 2005):93–107, 2005.

[JØS01] AUDUN JØSANG. a Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 09(03):279–311, 2001.

[KBTP17] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma. Fog computing in healthcare x2013; a review and discussion. *IEEE Access*, PP(99):1–1, 2017.

[KH] Z. Khan and P. Herrmann. How to secure internet of things devices in an energy efficient way. https://ercim-news.ercim.eu/en109/r-i/how-to-secure-internet-of-things-devices-in-an-energy-efficient-way. (Accessed on 04/07/2017).

[Kna98] S J Knapskog. A metric for trusted systems. *Proceedings of the 21st National Security Conference*, pages 16–29, 1998.

[LE03] Michael B Lewis and Andrew J Edmonds. Face detection: Mapping human performance. *Perception*, 32(8):903–920, 2003.

[Lin] Linux foundation announces edgex foundry to drive standardization of edge computing. https://www.forbes.com/sites/janakirammsv/2017/04/28/linux-foundation-announces-edgex-foundry-to-drive-standardization-of-edge-computing/2/#c5ab7542a46f. (Accessed on 04/30/2017).

[LS07] Huaizhi Li and Mukesh Singhal. Trust management in distributed systems. *Computer*, 40(2):45–53, 2007.

[LWR09] M. Li, H. Wang, and D. Ross. Trust-based access control for privacy protection in collaborative environment. In *2009 IEEE International Conference on e-Business Engineering*, pages 425–430, Oct 2009.

[MG+11] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[Min04] Robert P Minch. Privacy issues in location-aware mobile devices. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2004.

[MK15] Carolina V L Mendoza and João H. Kleinschmidt. Mitigating on-off attacks in the internet of things using a distributed trust management scheme. *International Journal of Distributed Sensor Networks*, 2015, 2015.

[ML15] Li Ma and Guangjie Liu. A hierarchical trust model for cluster-based wireless sensor network. In *2015 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 337–342, Oct 2015.

[NGA14] Michele Nitti, Roberto Girau, and Luigi Atzori. Trustworthiness management in the social internet of things. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1253–1266, 2014.

[NGMU15] S. Namal, H. Gamaarachchi, G. MyoungLee, and T. W. Um. Autonomic trust management in cloud-based and highly dynamic iot applications. In *2015 ITU Kaleidoscope: Trust in the Information Society (K-2015)*, pages 1–8, Dec 2015.

[RS17] Ammar Rayes and Salam Samer. *Internet of Things From Hype to Reality*. 2017.

[Sat16] Mahadev Satyanarayanan Satya. Edge Computing : Vision and Challenges. 3(5):637–646, 2016.

[SBCD09] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres, and Nigel Davies. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[SF99] Vipin Swarup and J. Fábrega. Trust: Benefits, models, and mechanisms. *Secure Internet Programming*, 1999.

[SHL08] Yan Sun, Zhu Han, and K J Liu. Defense of trust management vulnerabilities in distributed networks. *IEEE Communications Magazine*, 46(2):112–119, 2008.

[SK11] Radhika Saini and Manju Khari. Article: Defining malicious behavior of a node and its defensive methods in ad hoc network. *International Journal of Computer Applications*, 20(4):18–21, April 2011. Full text available.

[Sta10] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[SW14] Ivan Stojmenovic and Sheng Wen. The Fog Computing Paradigm: Scenarios and Security Issues. *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, 2:1–8, 2014.

[Thi] This is smart city - smart innovation norway. http://en.ncesmart.com/smart-cities-and-communities/this-is-smart-city/. (Accessed on 05/22/2017).

[Tru] Trust | define trust at dictionary.com. http://www.dictionary.com/browse/trust. (Accessed on 04/05/2017).

[VRM14] Luis M. Vaquero and Luis Rodero-Merino. Finding your Way in the Fog. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, 2014.

[WLC+15] Yating Wang, Yen-Cheng Lu, Ing-Ray Chen, Jin-Hee Cho, Ananthram Swami, and Chang-Tien Lu. LogitTrust: A Logit Regression-based Trust Model for Mobile Ad Hoc Networks. 2015.

[YMSG+14a] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky. Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing. *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD 2014*, pages 325–329, 2014.

[YMSG+14b] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky. Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing. In *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 325–329, Dec 2014.

[YQL15a] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and Privacy Issues of Fog Computing: A Survey. pages 685–695, 2015.

[YQL15b] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 685–695. Springer, 2015.

[YVJ+17] M. Yannuzzi, F. Van Lingen, A. Jain, O.L. Parellada, M.M. Flores, D. Carrera, J.L. Perez, D. Montero, P. Chacin, A. Corsaro, and A. Olive. A new era for cities with fog computing. *IEEE Internet Computing*, 21(2), 2017.

[YZV14] Zheng Yan, Peng Zhang, and Athanasios V. Vasilakos. A survey on trust management for Internet of Things. *Journal of Network and Computer Applications*, 42:120–134, 2014.

[ZBC+14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, Feb 2014.

[ZLTV10] Theodore Zahariadis, Helen C Leligou, Panagiotis Trakadas, and Stamatis Voliotis. Trust management in wireless sensor networks. *Transactions on Emerging Telecommunications Technologies*, 21(4):386–395, 2010.

[ZX15] Shaojie Zhou and Hui Xia. Node Trust Assessment and Prediction in Mobile Ad Hoc Networks. 9(10):361–372, 2015.