UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Sander Soo

Towards Proactive Mobility-Aware Fog Computing

Master's Thesis (30 ECTS)

Supervisor: Chii Chang, PhD
Supervisor: Satish Narayana Srirama, PhD

Tartu 2017

Towards Proactive Mobility-Aware Fog Computing

Abstract:
A common approach for many Internet of Things (IoT) and business applications is to rely on distant Cloud services for the processing of data. Several of these applications collect data from a multitude of proximity-based ubiquitous resources to provide various real-time services for their users. However, this has the downside of resulting in explicit latency of the result, being especially problematic when the application requires a rapid response in the edge network. Therefore, researchers have proposed the Fog computing architecture that distributes the computational data processing tasks to the edge network nodes located in the vicinity of the data sources and end-users, to reduce the latency. Although the Fog computing architecture is promising, it still faces challenges in many areas, especially when dealing with support for mobile users. Utilizing Fog for real-time mobile applications faces the new challenge of ensuring the seamless accessibility of Fog services on the move. Further, Fog computing also faces a challenge in mobility when the tasks originate from mobile ubiquitous applications in which the data sources are moving objects. In this thesis, a proactive approach for Fog computing is proposed, which supports proactive Fog service discovery and process migration using Mobile Ad hoc Social Network in proximity, enabling Fog-assisted ubiquitous service provisioning in proximity without distant Cloud services. Moreover, a proactive approach is also applied for the Fog service provisioning itself, in order to hasten the task distribution process in Mobile Fog use cases and provide an optimization scheme based on runtime context information. In addition, a case study regarding the usage of Fog Computing for the enhancement of Mobile Mesh Social Network was presented, along with a resource-aware Cost-Performance Index scheme to assist choosing the approach to be used for the transmission of data. The proposed elements have been evaluated by utilizing a combination of real devices and simulators in order to provide proof-of-concept.

Keywords:
Fog Computing, Edge Computing, Mobile Computing, Service Discovery, Internet of Things, Distributed Processing

CERCS: Computer science, numerical analysis, systems, control (P170)

Proaktiivse mobiilsusteadliku uduandmetöötluse suunas

Lühikokkuvõte:
Paljude värkvõrk- ja ärirakenduste tavapäraseks osaks on sõltuvus kaugete pilveteenuste poolt pakutavast andmetöötlusvõimekusest. Arvestatav hulk seesugustest rakendustest koguvad andmeid mitmetelt ümbritsevatelt heterogeensetelt seadmetelt, et pakkuda reaalajal põhinevaid teenuseid oma kasutajatele. Taolise lahenduse negatiivseks küljeks on aga kõrge viiteaeg, mis muutub eriti problemaatiliseks, kui vastava rakenduse efektiivne töö on väleda vastuse saamisega otseses sõltuvuses. Taolise olukorra puhul on viiteaja vähendamiseks välja pakutud uduandmetöötlusel põhinev arhitektuur, mis kujutab endast arvutusmahukate andmetöötlusühikute jaotamist andmeallikate ja lõppkasutajatele lähedal asuvatele arvutusseadmetele. Vaatamata sellele, et uduandmetöötlusel põhinev arhitektuur on paljutõotav, toob see kaasa uusi väljakutseid seoses kvaliteetse uduandmetöötlusteenuse pakkumisega mobiilsetele kasutajatele. Käesolev magistritöö käsitleb proaktiivset lähenemist uduandmetöötlusele, kasutades selleks lähedalasuvatel kasutajatel baseeruvat mobiilset ad hoc võrgustikku, mis võimaldab uduteenusetuvastust ja juurdepääsu ilma pilveteenuse abi kasutamata. Proaktiivset lähenemist kasutatakse nii teenusetuvastuse ja arvutuse migratsiooni kui ka otsese uduteenuse pakkumise käigus, kiirendades arvutusühikute jaotusprotsessi ning parendadades arvutuste jaotust vastavalt käitusaegsele kontekstiinfole (nt. arvutusseadmete hetkevõimekus). Lisaks uuriti uduarvutuse rakendusviisi mobiilses sotsiaal–silmusvõrgustikus, tehes andmeedastuseks optimaalseima valiku vastavalt kuluefektiivsuse indeksile. Lähtudes katsetest nii päris seadmete kui simulaatoritega, viidi läbi käesoleva magistritöö komponentide kontseptuaalsete prototüüpide testhindamine.

Võtmesõnad:
Uduandmetöötlus, Servandmetöötlus, Mobiilandmetöötlus, Teenusavastus, Värkvõrk, Hajustöötlus

CERCS: Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria) (P170)

List of Publications

The content of this thesis has been published in the following conference proceedings and also in an upcoming journal article:

1. S. Soo, C. Chang and S. N. Srirama, "Proactive Service Discovery in Fog Computing Using Mobile Ad Hoc Social Network in Proximity," 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 2016, pp. 561-566. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.126

2. C. Chang, M. Liyanage, S. Soo and S. N. Srirama, "Fog Computing as a Resource-Aware Enhancement for Vicinal Mobile Mesh Social Networking," 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), Taipei, Taiwan, 2017, pp. 894-901. doi: 10.1109/AINA.2017.18

3. S. Soo, C. Chang, S.W. Loke, S.N. Srirama, "Proactive Mobile Fog Computing using Work Stealing: Data Processing at the Edge," International Journal of Mobile Computing and Multimedia Communications (IJMCMC 2017), 8(4), *(In Press)*.

Acknowledgments

First and foremost, I would like to thank my supervisors Dr. Chang and Dr. Srirama for their invaluable support and assistance. I am eternally grateful to have had the opportunity to work on my thesis under their supervision and for an experience that has truly been enriching.

Additionally, I would like to thank all the people from the University of Tartu and especially from Mobile & Cloud Lab for various insightful discussions and assistance during my studies.

I am thankful for my friends and family, who have supported me during this thesis and provided various inspiring distractions.

In addition, this research has been partially supported by Study IT in Estonia.

Contents

List of Figures

1 Introduction

1.1 Fog Computing

The advent of Cloud computing has enabled a new paradigm of ubiquitous and on-demand access to computing resources [BBG10]. However, as the new wave of Internet of Things (IoT) [GBMP13] is emerging, the Cloud may not be a feasible choice to meet the requirements of the highly distributed applications due to the issue of latency. The information systems designed for integrating the Internet of Things [GBMP13] are usually applying the global centralized model, in which the IoT devices rely on distant management systems. Such a model is considered to be a drawback in terms of agility [BMZA12].

In many real-time ubiquitous applications such as augmented reality, environmental analytics, ambient assisted living etc., mobile device users require rapid responses. However, the latency caused by the distant centralized model is quite high, even though the mobile Internet speed has improved during the last few years. To address this problem, Fog Computing (Fog) [BMZA12] introduces data pre-processing with the computers in the vicinity of the data source and end-user applications located in the edge network of IoT systems.

In general, Fog computing resources, which are known as Fog nodes, are mediating devices that connect the edge network with the Internet. Some typical examples are industrial integrated routers (e.g., Cisco 829 Industrial Integrated Services Routers), home hubs or set-top boxes that are employed as wireless Internet access points together with embedded virtualization technologies (e.g., Virtual Machines) or containerization technologies (e.g., Docker containers[1]), which allow clients to deploy software onto them. Compared to the traditional distant Cloud computing model, which requires sending the data to the Distant Data Center (DDC) for processing, Fog can provide much better agility.

Figure 1 illustrates the generic distributed Fog computing infrastructure of a Cloud-centric IoT system. In general, the embedded systems (i.e. sensors, actuators, heterogeneous networked objects etc.) connect to the Cloud via the mediate gateways, which also act as Fog nodes that provide their computational mechanism as services. Moreover, the Fog nodes are capable of interconnecting to one another. Therefore, the IoT system can utilize the Fog nodes to form a hybrid Cloud computing environment that is capable of distributing processes among different locations between the core and edge network of IoT.

[1] `https://www.docker.com`

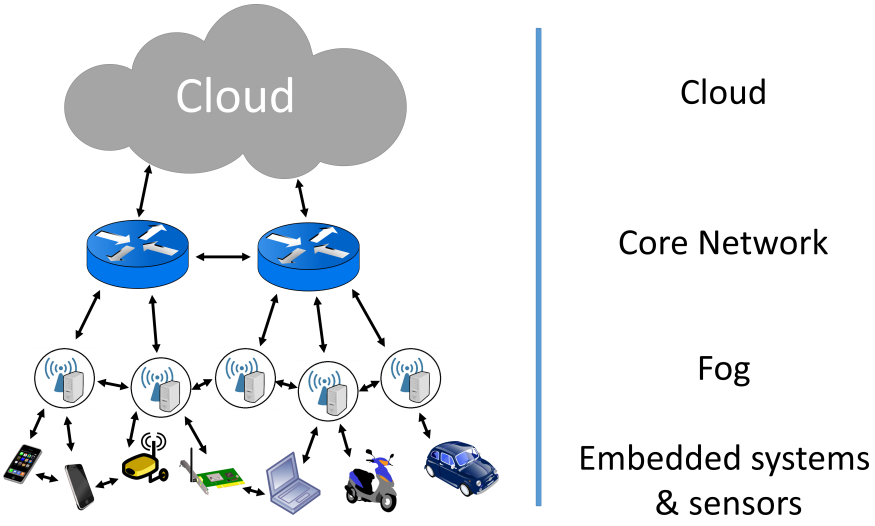


Figure 1. Generic Distributed Fog Computing Infrastructure.

1.2 Problem Statements

When a Fog-driven IoT system involves mobile nodes, it faces the following challenges.

Service Discovery. In contrast to the classic Cloud services, which commonly are centralized systems, offer seemingly infinite resources and are stably accessible as long as the Internet connection is available, the accessibility of a Fog node is constrained by its wireless network signal range.

Since mobile application users are moving objects, supporting the seamless accessibility of Fog for mobile applications becomes a critical challenge, especially in an environment which does not rely on the distant Cloud-based central management system to support the interaction between the mobile applications and Fog nodes.

In order to support the seamless accessibility of Fog, mobile applications need an adaptive service discovery and process migration strategy. In the past, several research projects (e.g., [TV04, LH05]) have proposed solutions to the similar problem. However, the assumptions of the past works, such as stable Internet connectivity, the availability of a data registry or the pre-established network topology, may not always be applicable in the mobile Fog environments.

Mobility-Aware Task Distribution. Imagine a mobile ubiquitous application needs to provide real-time environmental information to its user by continuously collecting and processing data collected from the surrounding environment while its user is moving in outdoor areas. For the purpose of improving the efficiency, the mobile

device (i.e. delegator) is distributing its computational tasks to vicinal Fog servers (i.e. workers).

The issue lies in the fact that the delegator may need to repeatedly re-send the tasks to different Fog nodes, due to the dynamic nature of mobile environment, where the limited wireless signal coverage of the Fog nodes could cause the delivery of results to fail. Moreover, this can result in a very poor quality of experience to the user.

The main research aspects are thus as follows:

- *Service discovery* — How can the system support Fog node discovery and process migration in a fully decentralized manner, without relying on the Cloud resources?
- *Mobility* — How can the system avoid the situation that requires the delegator (i.e., mobile application node) to re-send tasks to the other workers (i.e., Fog nodes) due to the failed process result delivery?

1.3 Research Objectives and Contributions

The research objectives and contribution of this thesis are as follows:

- Investigate, develop and validate a solution to support Fog service discovery and process migration in a fully decentralized manner, without relying on the Cloud resources.
- Investigate, develop and validate a solution to avoid the situation that requires the delegator to re-send tasks to the other workers due to the failed process result delivery.
- Case Study: Apply Fog computing in the Mobile Mesh Social Network (MMSN) context in order to enhance reliability and cost efficiency.

In order to accomplish the objectives, this thesis proposes the follows:

- A mechanism to enable Fog service discovery in proximity, by utilizing Mobile Ad Hoc Social Network (MASN) in proximity. The proposed method supports the mobility by continuously gathering information about nearby Fog nodes and utilizing this information to maintain seamless connectivity towards providing automatic handover to next available Fog node. Moreover, the solution does not require distant Cloud resource because the proposed system utilizes Distributed Hash Tables (DHT) to persist the information in the proximity of the Fog node.

- A mechanism to support mobility-aware Fog service provisioning, by utilizing an enhanced version of the work-stealing approach, which takes into account the characteristics of workers and tasks. Particularly, the proposed method supports user mobility by taking into account the user's route for worker selection and utilizes a results delivery process that identifies the best approach in a given context.
- A framework to enhance the reliability and cost efficiency of Mobile Mesh Social Network by applying Fog computing and adaptive deployment of Fog Social Network capabilities where necessary. Specifically, the proposed method utilizes Cost-Performance Index to improve the system's ability to adapt to the dynamic nature of mobile network environment, making use of different data transmission options based on the efficiency calculations.

1.4 Structure of the Thesis

The rest of this thesis is structured as follows. Chapter 2 provides an overview of related works. Chapter 3 describes the design and implementation of Fog service discovery. Chapter 4 focuses on the aspect of supporting mobility and context-aware task distribution. Chapter 5 discusses the application of Fog Computing in vicinal social network. Chapter 6 provides analysis for the experimental evaluation. The thesis is concluded in Chapter 7 along with future research directions.

2 Literature Review

This Chapter consists of the literature review in three domains that correspond to the three research contributions of this thesis—Fog service discovery, mobility-aware task distribution in edge network, and the decentralized mobile social networking.

2.1 Fog Service Discovery

P2P-Bluetree [LH05] describes a location aware peer-to-peer (P2P) information sharing system over Bluetooth, by organizing peers into a tree-based overlay network.

Datta et al. [DDB16] proposed a resource discovery framework for smart and legacy devices. In their framework, the search engine component ranks the resources and returns a Uniform Resource Identifier (URI) for accessing each resource and also introduces a lifetime attribute, which denotes the resource's discoverable period. The functionality is offered through a RESTful web service. The core discovery mechanism depends on a resource configuration registry, which is a component for querying and requesting data.

Tchakarov and Vaidya [TV04] also address the issue of resources and content location in a location-aware network. In their approach, the nodes periodically send update messages through the network into the geographical directions (North, South, East or West). Nodes along these trajectories cache the information. Hence, when a client sends a request in the geographical directions in a similar manner, it will receive the results that were previously cached by the intersection nodes. It is based on the assumption that in a dense enough network, the trajectories of the previous advertisement and the current request are likely to intersect in at least one trajectory.

Ma and Jamalipour [MJ09] propose an epidemic routing mechanism in Mobile Ad Hoc Network using a buffer management policy to improve the efficiency of the finite space restriction. The approach is based on the evaluation of the requests. For example, by using mobility statistics, the requests with lower evaluation results are dropped when the buffer gets full.

Existing works have several limitations, such as the assumption of a tree topology, which may not be the case in real life, because there may be links between peers in the network that collectively form the routing cycles. Additionally, the notion of parent and child node in an actual peer-to-peer graph may not be applicable because fundamentally all the peers are equals. On the other hand, the proposed approach of this thesis takes the physical location into account and does not need to rely on a special logical address based on the node's position in the tree. Further, it is designed for a fully decentralized P2P environment in which the user may not have access to the Internet at all and is unable to query any registry for data. Periodically advertising content through a network is not applicable, due to the amount of unnecessary update traffic it would generate. In addition, the inherent assumption of a high density of nodes may not be applicable in

the real world. Instead, the proposed approach of this thesis is to employ a Distributed Hash Table (DHT)-based approach for the Fog node information (location, SSID of co-located Wi-Fi router, etc.).

Although the approach in this thesis utilizes the flooding-based scheme, it provides a more adaptive routing protocol that sends the message in the last known direction of the intended recipient. Further, the response message received by the requester contains the information for the requester to directly establish a connection with a Fog node.

2.2 Mobility-aware Task Distribution in Edge Networks

2.2.1 Computation Offloading

Computation offloading is a common strategy to reduce the resource consumption and to improve the overall performance of mobile ubiquitous applications. Specifically, earlier works such as MAUI [CBC+10] or Cuckoo [KPKB12] have introduced the schemes that assist the system in offloading the process from mobile devices to central surrogates such as the Cloud.

Considering the latency caused by the centralized offloading schemes, recent strategies have introduced the utilization of vicinal computational resources such as Virtual Machine (VM)-based Cloudlet [SBCD09]. In general, Cloudlet represents the VM-enabled server machines located in the same network as the mobile application nodes. For example, a local business may provide Cloudlet machines to their customers to improve the Quality of Experience of the mobile applications used by the customers.

In order to optimize the efficiency of the Cloudlet-based computation offloading, a number of researchers have proposed the use of machine learning algorithms to help the mobile applications' decision in whether or not to offload the tasks [ZNW15, THTN14].

Similarly, the offloading optimization is also an imperative research question in terms of balancing the workload between Cloud and Fog [LS16] and optimizing the task distribution in mobile ad hoc Clouds [SCD15, YCGN16].

Although existing works described above have proposed numerous strategies for distributing the computational tasks from mobile devices to external resources, most of them have assumed the communication between the delegator and the workers is fairly stable in which they did not fully address the challenge raised in this thesis.

2.2.2 Result Routing

An important aspect in mobile Fog is how to route the process result back to the delegator. In particular, the delegator may have moved out from the wireless network coverage of the Fog node, which has taken the computational tasks.

Instead of assuming the connectivity between the delegator and workers can last long enough, a number of related research projects have proposed corresponding strate-

gies. For example, authors in [ZSM+16, SLZL15] propose collaborative caching with proximal peers opportunistically. Further, authors in [FLR13, SLAZ12, FLR16] utilize the Time-To-Live (TTL) policy, which defines the work expiring time, i.e. time before the delegator restarts its delegation process. However, these approaches can potentially cause extra latency. Hence, authors in [RP14] have proposed the interconnected Cloudlet scheme in which the delegator can establish a data routing network among multiple Cloudlet machines on the move.

2.2.3 Load Balancing and Efficient Deployment

Several works [LS16, GLL+16, CPS17, HLR+13] have introduced approaches for optimizing the workload or discussed the efficiency of deployment for applications that could be improved via Fog. Specifically, authors in [CPS17] have proposed a scheme by optimized placement of Virtual Machines (VM) to provide improved computation support; authors in [HLR+13] proposed a process placement algorithm based on utilizing the customized scaling policy acquired from the users.

Existing works [HCL10, Mar09] did not take into account the heterogeneous device capabilities of the other worker nodes. Consequently, this raises the issue of assuming the devices are homogeneous. However, in the case of uniformly distributing the works, some nodes may be overloaded and cannot accept more works. Further, a weaker node may not be able to effectively complete the tasks it received in time and will thereby result in the bottleneck issue.

2.2.4 Work Stealing Task Distribution

Contrary to having tasks assigned to workers, in the work-stealing approach [BL99], the workers are actively taking the work items from the delegators.

Loke et. al. [LNA+15] introduce an extension of the work-stealing scheme for mobile ad hoc Cloud computing environment. Particularly, in addition to the workers and delegators, the scheme introduces intermediate worker nodes situated between the delegators and the regular workers. Such an approach extends the resource availability by enabling tasks to be distributed to nodes that may otherwise be unavailable (e.g., out of range).

Fernando et. al [FLR16] describe further extensions to the work-stealing approach. The extensions include the expiration of tasks, in order to prevent more capable nodes waiting endlessly for weaker nodes to complete computationally intensive tasks, task steal limit to prevent infinite back-and-forth stealing of tasks between workers, etc. The approach proposed in Chapter 4 is the extension of the multi-layered work-stealing [LNA+15] approach.

2.3 Decentralized Mobile Social Networking

Classic centralized distant data center-based social network services (CSNS; e.g. Facebook[2], Twitter[3], Foursquare[4] etc.) have grown to be one of the most popular mobile applications [Kha15, Smi15, com16]. On the other hand, Internetless Mobile Social Network (MSN) applications do not require an Internet connection with the distant data center (DDC) for performing social activities, but instead utilize the built-in wireless communication mechanisms of mobile devices (e.g., Wi-Fi and Bluetooth) to enable the data routing among the participating devices.

The aim of this section is to outline the multitude of technologies and approaches available to support decentralized Mobile Social Network interactions.

2.3.1 Mobile Mesh Social Network

Mobile Mesh Social Network (MMSN) represents a MSN environment established by Wireless Mesh Network (WMN) [AW05]. Specifically, the environment enables the spatio-temporal social activities among the participants, and potentially brings opportunities for the MSN users to participate in the vicinal real time events.

MMSN, such as Open Garden's FireChat[5] or Serval Project's Serval Chat[6], are useful in the situation when the regular cellular network is unstable, unavailable (e.g. in disaster recovery scenarios) or simply when the MSN users want to save their mobile Internet bandwidth in the area where the free access points are not available. To enumerate, existing open frameworks that can realize MMSN include but are not limited to Apple's Multipeer Connectivity[7], AllSeen Alliance's Alljoyn[8], Open Connectivity Foundation (OCF)'s IoTivity[9], etc.

In MMSN, mobile devices can act as both regular peers and routers. As a router node, the mobile device will try to deliver the data to the destination node for its nearby peers based on the routing information shared among the peers. Ideally, the WMN topology used by MMSN is flexible and scalable since it does not rely on particular peers to be the cluster heads (super peers). However, it may face a number of limitations as illustrated in Figure 2:

- *Unstable connectivity*. For example in Figure 2, suppose Node-B intends to send a message to Node-F. Initially, the only connected node to Node-F was Node-D.

[2] `https://www.facebook.com`
[3] `https://www.twitter.com`
[4] `https://www.foursquare.com`
[5] `https://www.opengarden.com/firechat.html`
[6] `http://www.servalproject.org`
[7] `https://developer.apple.com/reference/multipeerconnectivity`
[8] `https://www.allseenalliance.org/framework`
[9] `https://www.iotivity.org`

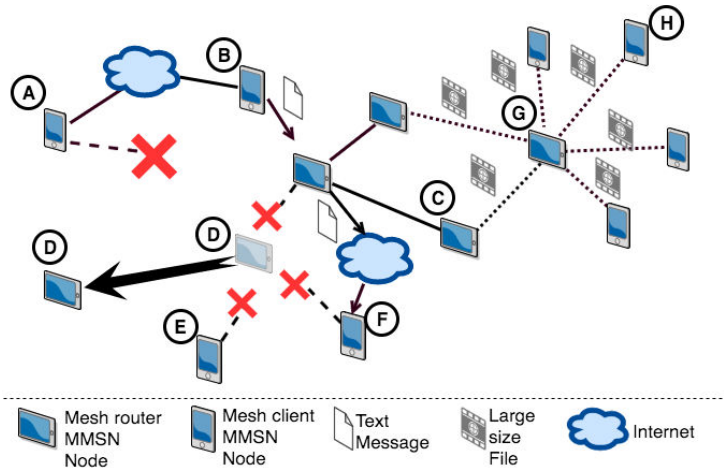


Figure 2. MMSN example.

Since Node-D has moved out from the network, Node-B's message cannot reach Node-F unless the intermediated node utilizes Internet connection (if Node-F is connected to the Internet). Alternatively, the intermediate node has to wait until it encounters another node that can route the message to Node-F.

- *Bottleneck issue*. In MMSN, there is a chance that a node becomes the only router node for a group of participants. For example in Figure 1, Node-G has faced such a situation. Consequently, due to the heavy traffic occurred at Node-G, the transmission at that edge has become extremely slow. Further, it may discourage Node-G from maintaining its connection and eventually it may disconnect from the network because it may have consumed too much of its own hardware resources and energy (battery).

- *Lack of routing path*. For example, Node-A does not have any connected path. Hence, the only way for Node-A to participate in the network is to utilize the Internet. Similarly, Node-E is facing the same issue, as Node-D has moved out from the network.

2.3.2 Edge Computing for Mobile Social Network

Cloudlet [SBCD09] is an open source edge computing solution[10] that has a similar concept as Fog. In contrast, the initial concept of Cloudlet is based on providing Virtual Machine (VM)-enabled service from the wireless network communicable computing machines of local business (e.g. cafeteria) to the users in the vicinity. In general, it is expected that the Cloudlet machines are fairly powerful in terms of computing and

[10] `http://www.openedgecomputing.org`

networking. Hence, they can provide on-demand computational process offloading services (e.g., real-time face recognition [SMF+12]).

A number of works utilized Cloudlet to enhance multimedia content delivering or streaming [XYCD13, WKC+13]. Further, similar strategies have also been applied in Fog for online game streaming [CLH14, LS16]. These works are focusing on streaming the content between the distant server and the edge network mobile devices.

2.3.3 Opportunistic Mobile Social Network

Opportunistic mobile social network (OppMSN) [WW14] represents the vicinal MSN established by utilizing Delay-Tolerant Networks (DTNs) among the participative mobile devices. The social activities among the participants rely on the mobile ad hoc routing mechanisms. Initially, OppMSN is triggered by academic research projects such as [WW14]. The discontinued Haggle[11] mobile app was a representative OppMSN enabler.

OppMSN-based research projects [POL+09, KFA+10, ZLFZ15] usually focus on how to enable the efficient data routing in the DTNs. Expressly, works such as MobiClique [POL+09], Peoplerank [KFA+10] were more focused on improving the opportunity of the social encounter among the participants. Further, they did not consider utilizing the infrastructures of Cloudlet or Fog to improve the efficiency. Different from their works, the proposed FSN tends to enhance the efficiency of MMSN towards improving the overall QoE.

2.3.4 Mobile Social Network in Proximity

> *"Mobile Social Network in Proximity (MSNP) is a wireless software application environment in which participants in the environment can use heterogeneous mobile applications to share information and perform computational social network application activities without necessarily receiving assistance from central management services."* [CSL15]

Mobile Social Network in Proximity (MSNP)-based approaches [CSL14, CSL15] focus on open standard-based interoperability among participants. Similar to OppMSN, MSNP frameworks are also based on ad hoc topology. Instead of using vicinal resources (e.g. Cloudlet), MSNP utilizes distant Cloud to extend its computational resources, which is the major difference between MSNP and the proposed FSN in this thesis.

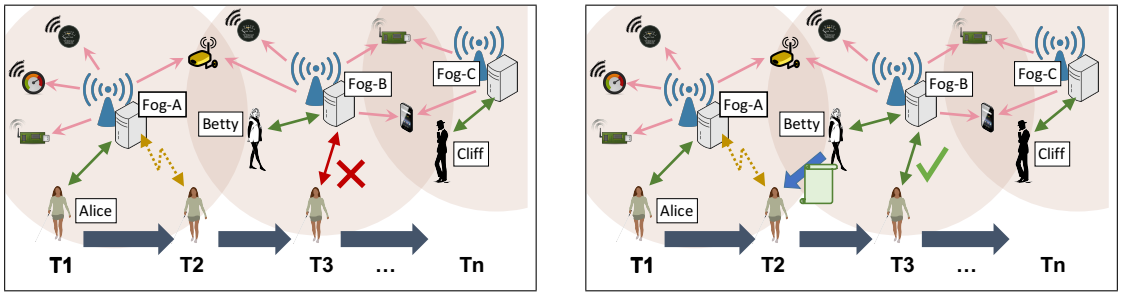
[11] `https://play.google.com/store/apps/details?id=org.haggle.kernel`

3 Proactive Fog Service Discovery

Emerging Internet of Things systems utilize heterogeneous proximity-based ubiquitous resources to provide various real-time services to mobile application users. Fog computing reduces the response latency caused by data processing on distant Cloud servers, by utilizing the proximal computational and networking resources. However, utilizing Fog for real-time mobile applications faces the new challenge of ensuring the seamless accessibility of Fog services on the move.

This chapter discusses the design of the mechanism to support Fog service discovery and process migration in a decentralized manner. The proposed framework enables Fog-assisted ubiquitous service provisioning in proximity without relying on the distant Cloud services. The underlying strategy is to utilize Mobile Ad hoc Social Network (MASN) in proximity. As the mobile user is moving, the platform continuously gathers information about nearby Fog nodes and uses this information to maintain the accessibility of Fog.

3.1 Overview of Environment



(a) (b)

Figure 3. Fog node discovery scenarios.

Figure 3 illustrates a scenario in which a disabled person Alice is using her Ambient Assisted Living (AAL) mobile application to assist her while she is in the outdoor area. The AAL application provides the information of Alice's surroundings rapidly in order to help her avoid crowded areas. Since such a mechanism requires collecting and processing a large quantity of data from the surrounding IoT devices continuously, it is an ideal opportunity to utilize Fog to reduce the burden of the mobile device and also enhance the overall computational and networking performance.

In general, a mobile application can access the Fog only when the user is within the wireless network range of the Fog. Imagining Alice's AAL application currently is using Fog node — Fog-A (timestamp T1 in Figure 3a). While Alice is moving, the

connectivity between her mobile device and Fog-A is getting weaker (T2 in Figure 3a). When Alice moves out from Fog-A's range, her AAL application loses the access with Fog entirely even though she has reached Fog-B's range. The application doesn't switch to Fog-B due to the lack of knowledge regarding its existence. Ideally, such information should have been pre-fetched by the AAL application beforehand. However, there is a chance that the AAL application has missing information about the physical area or there may be a gap area between two already known Fog areas. In such a situation, the application needs to perform all the tasks by itself or utilize Cloud services. Either way, the AAL application will suffer from the latency issue of mobile Internet or the resource constraint issue of the mobile device.

On the other hand, if the AAL application utilizes MASN to retrieve the up-to-date information of the Fog in its proximity, it can autonomously establish the connection with Fog-B and migrate its current tasks from Fog-A to Fog-B (see Figure 3b). Suppose Alice's AAL application is associating with MASN, at T2, Alice's mobile device encounters Betty's mobile device, which shares the information about Fog-B to Alice's mobile device. With such information, Alice can establish the connection with Fog-B and migrate the tasks to it autonomously.

3.2 Overview of the Proposed Fog Discovery Scheme

In the proposed framework, when the user is moving, his/her mobile device is continuously gathering information about nearby Fog nodes and using this information to keep the device connected to the Fog, autonomous of the Internet connection or the distant Cloud.

It is realized by employing MASN in proximity or transitive proximity. Firstly, Alice's device uses the built-in communication mechanisms (e.g., Bluetooth, Wi-Fi Direct service, etc.) to retrieve the information of MASN peers in proximity that may carry the information about the Fog nodes in the vicinity. Moreover, the MASN peers can provide information about other peers that are further away and thus unreachable to Alice's device directly. Consequently, Alice's device can interact with them via the intermediary nodes and retrieve information regarding the Fog nodes in further destinations from them.

The distributed knowledge between the devices collectively forms a DHT. When Alice's AAL application needs to connect to a Fog node and yet it does not have knowledge of any nearby Fog, it can query the DHT. The process is as follows:

1. First, Alice's device sends a request message to the closest MASN peers.

2. The MASN peers may reply immediately with the information of the available Fog nodes in proximity, or they may forward the request to their connected peers in order to collect more information about the Fog nodes that have higher quality, better characteristics, etc.

3. As soon as an MASN peer retrieves the corresponding information, it routes the information through any intermediary peers back to Alice's device.

4. Afterwards, the device can automatically configure its connection with the nearby Fog node and migrate its tasks to the newly connected Fog.

Since the networks that enable the connectivity with the Fog may be password protected, the passwords are shared in the same DHT fashion.

The passwords are encrypted for the initial requester (i.e., Alice's device) with the public key provided by Alice's device. In general, this approach may reduce the efficiency of the DHT because the credential information cannot be simply forwarded and reused further from any arbitrary intermediary. However, it ensures that the network is only accessible for the users in the authorized group. The location and other general information can still be shared freely in DHT. Further, the request for the public key of another user is executed in the same manner as the one for the Fog node information.

The proposed solution allows the trusted *private social network* [PKFS09] users who are in close relationships such as colleagues, close friends or family members to share the information of protected Fog service provider nodes. Hence, the proposed solution does not involve the general security concerns in public social networks.

3.2.1 Transport

The proposed framework is not bound by a specific low-level transport protocol. Ideally, MASN peers can communicate using common standard protocols such as Wi-Fi.

The proposed system is transport agnostic. Specifically, when MASN nodes are sending messages to one another, the framework handles the underlying transport seamlessly, which decouples the application code from the heterogeneous network protocols. Further, any messages can be sent via intermediary nodes, thus enabling the transmission of messages between devices with different available transport capabilities, by using the intermediary node as a translator.

3.2.2 Routing

Each MASN peer holds entries in its routing table that contains the information about how a message can reach the respective devices, including any directly connectable peers and the peers within two hop range, as per the example of [LH05]. Further, peers can gain more information about the environment by simply listening and forwarding requests that have been made by others and thus growing their routing table larger and giving them more information about their surrounding peers.

A single entry in the routing table describes the information on the destination peer and the gateway, which is the peer that enables the connectivity to the final destination.

Peers attempt to send the response back to the initial requester using the same path. For instance, if a message was sent via peers A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D then the attempted response message path will be D $\rightarrow$ C $\rightarrow$ B $\rightarrow$ A.

The proposed framework also tends to address the runtime failure in the following situations.

Missing path. There is a chance when the original route for delivering the response to the requester is no longer available (e.g. intermediary peers disconnected). Correspondingly, one approach is using flooding in the direction of the last known GPS location of the requester. Such an approach floods the message to the peers that are known to the current device, and which are nearest to the final destination of the message [RFH$^+$00].

Incorrect destination. There is also the possibility that the last accessible node in the path is not the final destination. Initially, the routing device can only verify the existence of the path up to two hops from the current location. In order to improve the chances of successful message delivery, the current peer will transmit the message to a number of its peers nearest to the final device, in addition to the next node in the potential return path of the message. Although similar to the usual flooding transmission of the message, in this case, the message is forwarded to fewer surrounding nodes, since the assumption is that the return path of the message still exists.

Each message also has a Time-To-Live (TTL) value (maximum number of hops a message is forwarded), which is to avoid the infinite flooding and forwarding of the message in case the destination node no longer has an existing path in the system.

Algorithm 1 illustrates the core of the message routing. The message to be routed is denoted by the variable named "m". The algorithm begins with a check whether the message's TTL value (Line 2) exceeds a threshold. If it exceeds the threshold, the router does not forward the message.

If the TTL value is not exceeded, the router finds a number of peers that are known to the current device, and which are nearest to the final destination of the message (denoted by KN; Line 5). Then the algorithm determines if the message is a request (Line 6). If so, the router decrements the TTL value of the message. If such value exists, it passes the message to the previously determined peers.

If the message is a response, the algorithm determines up to a distance of two hops, whether the original path to the destination exists (Line 10). If so, it defines a subset of KN based on the predefined limit (Line 11). If the message contains a TTL value (Line 12), the algorithm decrements the TTL value of the message (Line 13). Finally, it forwards the message to the peers (Line 14).

If the message is a response and the algorithm has determined that the original path does not exist, the router will initialize the TTL value (Line 16) and forward the message to KN (Line 17).

Algorithm 1: Core router logic.

```
   Input: Message to be routed
1  Function route (m)
2      if isTtlExceeded(m) then
3          drop(m);
4      else
5          KN = getCloseNodesTo(m.to) ;
6          if isRequest(m) then
7              updateTtl(m);
8              send(m, KN);
9          else if isResponse(m) then
10             if pathExists(m.viaPath) then
11                 nodes = limit( KN );
12                 if isTtlPresent(m) then
13                     updateTtl(m);
14                 send(m, nodes);
15             else
16                 setTtl(m);
17                 send(m, KN);
```

3.2.3 Messages

The nodes can intercommunicate in the system by passing around messages in a common data-interchange format such as JSON[12].

All the messages, regardless of their type, contain the following common information:

- The message UUID (128bit Universally Unique Identifier) to differentiate two messages.
- Routing information, which denotes who are the sender and receiver of the message.
- The possible intermediary nodes' sender and receiver IDs that can be used in the routing.

[12] see `http://www.json.org`

If the message is a response to a previously sent message, then this information is also present in the message.

The following information describes the details of the request and response messages.

Request Messages

The proposed framework uses 3 types of request messages:

- *Key Discovery message*—is for requesting the public key of any user in the system. The key is used when the peer sends and receives messages with sensitive content (e.g., passwords). This search can be restricted to a specific user with their ID or simply to any peer in proximity.
- *Fog Discovery message*—is for querying the information from the DHT about a specific Fog node or any node in the close proximity of the requester based on the GPS location.
- *Peer Discovery message*—is for requesting the information about the peers in the DHT of the system. The query may be performed with a specific location to find MASN nodes closest to that position or to request all the MASN peers of another peer. This message can be used with a time interval to get the knowledge of MASN peers in the 2-hop distance.

Response Messages

The routing messages may contain the following types of response messages:

- *Fog node message*—contains the following information.
  - The information of Fog node in physical proximity of the requester.
  - The Service Set IDentifier (SSID) of the Wi-Fi router co-located with a Fog node.
  - The password, if any, for the Wi-Fi network that is encrypted for a specific user.
  - The specific user's ID (e.g., his/her email address)

  Additional information regarding the Fog node may also be included in the form of key-value pairs, such as the characteristics of the node (e.g., signal strength, etc.).
- *Key message*—contains the user's ID and the public key linked to the user, which other MASN nodes can use to encrypt sensitive data.

- *Peer message*—contains the transport-layer specific information necessary for another MASN node to establish a connection to this peer in the future. The receiver has to update the routing table with the received peer connection information and also specify the previous forwarder as the gateway.

- *Error message*—is the response when an error occurs or when the contract of the message cannot be fulfilled. For example, when the user requests peers, but the queried peer has no nodes to return as the response. The reason for the failure is also stated in the message whenever possible.

3.3 DHT Design and Lookup

A user can save a message to the DHT in two main steps.

1. As a first step, a message is created locally and it contains either the Fog node information, public key information or peer information. Peer message information is saved into the routing table, Fog node information and public key messages are saved in their respective local storage instance.

   Currently, the proposed framework assumes by default that the location where the user creates the information of the Fog node is roughly the same as the Fog node's location. If the user has more information about the exact location of the Fog node, s/he may provide it. In this case, the system will attempt to use the underlying DHT and MASN nodes to transmit the message to the more specific location.

2. As a second step, the message is routed to a number of peers of the initiator, which perform the same process recursively, up to a depth of two hops. The full propagation of all information to all of the MASN nodes is not performed, due to the message pollution and network congestion it would most likely cause, with offering little in return.

Generally, MASN peers may join and leave and the ones who held the message before may not be available in the same place forever. Therefore, the responsibility for keeping the message alive is not on those who happened to be near the location of the Fog node at the time the Fog message was created, but rather who are at the location at the current point in time. This is similar to the way how Kademlia [MM02] peers keep track of data, with the closest nodes to the data being responsible for it. The difference is that Kademlia relies on the XOR distance of IDs, the proposed framework utilizes the physical distance of the peers to Fog nodes.

The simplest way for an MASN node to determine whether or not it is responsible for the message is by GPS distance calculation bounded by a predefined threshold.

Another way is for the message storage to limit the maximal number of locally persisted messages, ordered by the distance between the current location of the device and the Fog node location from the message. This is a simple cache replacement algorithm, with the distance as the heuristic.

There are also more complex ways, such as combining the distance and the fact of whether a requester's peers up to a number of hops already have enough copies of the message so as to satisfy a minimum replication level criteria. This algorithm is similar to that of Beehive [RS04]. This approach is being considered for investigation in future research.

The process to look up any information from the DHT is similar to the process of persisting a message. First, the node checks for any messages that fill the criteria in its own local storage. If the data is found locally, the process terminates. If the information is not found locally or the local data is insufficient, the device queries its respective peers for the data. Afterward, the requester node has to interpret the result(s) accordingly.

3.4 Summary

This chapter introduced an approach for proactive Fog service discovery using MASN, which enhances the need for seamless task migration between different Fog service providers while the user is moving.

The proposed discovery scheme utilizes MASN in proximity or transitive proximity. Specifically, the MASN peers utilize a DHT approach to distribute the knowledge of the network in a collaborative manner. The DHT provides information that enables the peers to seamlessly connect with the available Fog nodes.

Finally, the chapter concluded with a discussion on the core router logic and the core messages in the discovery process, along with the DHT design and information lookup methodology.

4 Proactive Mobile Fog Computing

A common design of the Internet of Things (IoT) system relies on distant Cloud for management and processing. Since this approach faces the challenge of latency, especially when the application requires rapid response in the edge network, Fog computing architecture could be utilized to distribute computation to nearby edge network nodes, thus reducing the latency.

Although the Fog computing architecture is promising, it still faces a challenge in mobility when the tasks come from mobile ubiquitous applications in which the data sources are moving objects.

In order to address the challenge, in this chapter, a proactive Fog service provisioning approach is proposed, which hastens the task distribution process in Mobile Fog use cases. Further, an optimization scheme is provided in task allocation based on runtime context information.

4.1 Overview

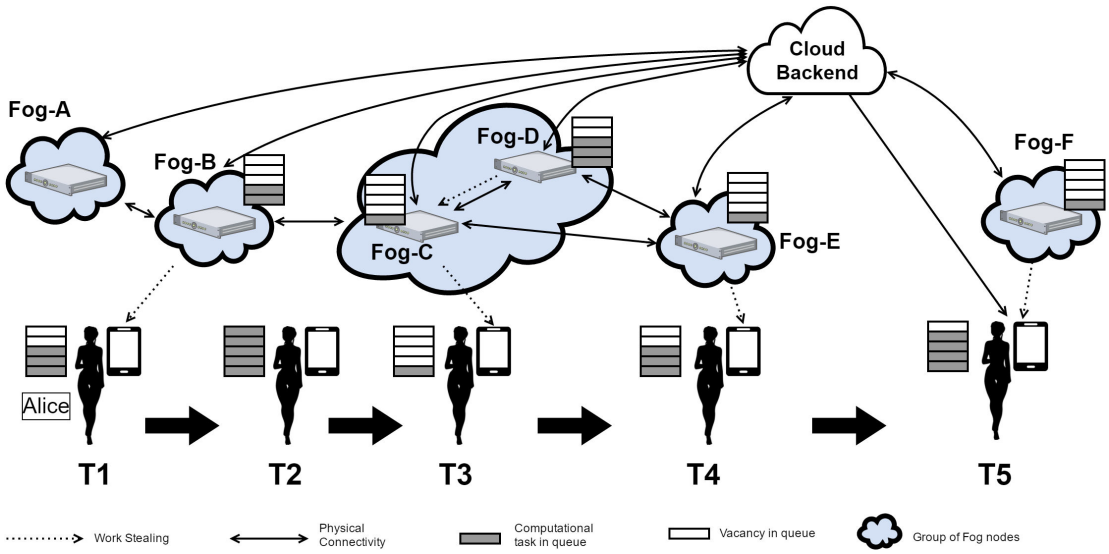


Figure 4. Overview of proactive Fog.

Figure 4 illustrates an overview of the proposed system based on an Ambient Assisted Living (AAL) scenario in which the user Alice's mobile device is operating an AAL service that is continuously collecting and processing the data derived from the surrounding IoT devices in order to provide useful information to Alice while she is walking in an urban area. Considering that the application includes a large volume of

data processing, the Cloud backend management system of the AAL application has utilized Fog services to provide rapid responses.

One assumption that has been made is that Alice's route has been pre-defined using the corresponding mechanism based on the historical records [RMHS17] or Google Maps API[13].

Another assumption is that the Cloud backend has gained access to the Fog servers before Alice starts moving. In general, the Fog servers either belong to the same provider of the AAL service or they are provided by the collaborative providers.

Cloud backend selects the candidate Fog servers based on the route of the user (see Section 4.2). Further, it also deploys the software to the Fog servers to trigger the proactive behavior. The Fog nodes are also configured into non-overlapping groups beforehand, to enable the more efficient partitioning of tasks among the members of a group (see Section 4.3).

In Figure 4, T1 to T5 represent the timestamps of the user's route. In general, while Alice is moving, the AAL application (i.e., delegator) advertises its existence to Fog nodes (i.e., workers) by sending any of them a registration message. Additionally, the message contains the information of the tasks in the delegator's queue, including what type of computational tasks they are. For example, the task can be CPU-intensive, GPU-intensive, RAM-intensive and so on. This information is updated periodically via the steal requests.

As Figure 4 shows, in T1, a chosen candidate Fog-B actively 'steals' two work items from the delegator. While Alice is moving, the AAL app has generated more tasks (T2). At T3, the delegator has encountered a new worker Fog-C. Since there are two workers in the group where Fog-C belongs, both of them will assist the AAL application.

As can be seen from the figure, Fog-C and Fog-D have acquired a different number of tasks (Fog-D has stolen some tasks from the delegator via Fog-C). In summary, the number of works they acquire is based on the runtime context (e.g., resource availability, workload and bandwidth) of the Fog nodes. The details are described in Section 4.4.

There is an inevitable situation, where the Cloud backend does not find any direct connection between two Fog nodes on the user's moving route. For example, Fog-E does not have the connection to Fog-F. In such a case, Fog-E may deliver the process result to the delegator via the Cloud backend. The Cloud backend can either directly send the results to the delegator or it can indirectly deliver the results via Fog-F. The corresponding strategy of the result delivery is described in Section 4.5.

4.2 Candidate Worker Selection

Based on the route of Alice, the Cloud backend can identify the candidate Fog nodes needed in assisting Alice's AAL application. The Cloud backend selects the Fog servers

[13]see `https://developers.google.com/maps`

based on the scheme below.

Let E = { E_1, ... , E_n} be the set of all possible encounter Fog nodes on the end user's path. E_k = {e_i , where $1 \leq i \leq \mathbb{N}$ } denotes the current encounter node(s) and E_{k+1} denotes the next encounter node(s) after E_k. E_1 are the closest encounter nodes to the starting point of the user.

Let E_x be one of the members of E. Then each of the $e_i \in E_x$ (denoted by e_i^x) would be included based on the following considerations:

- If e_i^x has a route through the network infrastructure to e_i^{x+1} without utilizing the Cloud, then e_i^x is considered as a priority candidate.
- Let e_y^x to be one of the e_i^x. If e_y^x has a direct route to a priority candidate, it is also considered as a candidate.
- An isolated Fog node, with high computational capabilities on the moving path of the user, where there are no alternatives, is also considered as a candidate.

Once the Cloud backend has selected the candidate Fog servers, it will deploy the corresponding software to the Fog servers in order to trigger the proactive behavior.

4.3 Worker Network

The Cloud backend forms a worker network for the AAL application based on configuring the candidate Fog nodes into non-overlapping subgroups beforehand, such that all the members of the group are aware of the other Fog nodes in their group. The work stealing process among Fog nodes only takes place within the boundaries of the group.

The members of the group receive updates on various characteristics of their peers, such as the current CPU usage, RAM usage, etc., delivered by a resource-efficient publish-subscribe protocol (e.g., MQTT [BG14]). MQTT also offers a Last Will and Testament feature, to notify peers when a node unexpectedly goes offline. Such information is useful in the optimization process.

Upon receiving information from the delegator, the Fog node also notifies the group members about the number of work items available per type (e.g., CPU intensive, GPU intensive or RAM intensive etc.). This notification has the added benefit of stopping infinite steal attempts for a time when there are no users and no work (e.g. early morning) and continuing when work arrives (e.g. morning rush-hour).

Once the Fog node steals the task(s) from the delegator, the Fog node will also register to a topic of the delegator's events (e.g., update of the current location). Thus, the Fog nodes would have the knowledge of the user's currently connected Fog node and could transmit the results of the computation accordingly.

Transmitting the results back to the user via the network of Fog nodes could still cover as many Fog nodes as needed.

The grading value to partition the tasks (explained in more detail in Section 4.4) yields the expected number of tasks a Fog node will handle and can be used as an estimate for the initially stolen tasks. Any Fog node will effectively make the final decision on whether or not to take some tasks for processing.

4.4 Context-Aware Work Stealing Scheme

The context-aware extended scheme for work-stealing would allow for the multi-layered distribution of several work items within a group of workers, taking into account the current capabilities of its members. The work items or tasks to be done contain information about the primary hardware resource to be utilized, thereby allowing the members to steal works based on resource availability. For example, they will steal CPU-intensive tasks if they have available CPU resources. Fog nodes would also query for runnables to process the work, either directly from the delegator (e.g., jar files or offline Docker images) or from the Internet (e.g., Docker Hub).

To extend the basic possibility of stealing one task for each resource type, a rating is included for each of the primary resources of a Fog node in this version of the work stealing approach.

The ratings will be fetched by each of the Fog nodes (e.g., at the start of every day from an external service) and would show the capability of the Fog node in the given context of the resource (e.g., CPU), thus enabling different Fog nodes to be mutually comparable based on these values. An example of such external service could be `www.cpubenchmark.net` for the ratings of CPUs.

It is beneficial to allow the Fog node to steal work until its resources are properly utilized. However, it is not sensible to steal much more work that can be currently processed by the adjacent Fog nodes, due to the overhead of routing the computed results back to the continuously moving user.

A key aspect is thus to consider the estimation of the number of work items that should be stolen by a specific Fog node.

A set of context parameters considered in the performance measurement of Fog node can include e.g., CPU capability, RAM capability, network speed capability, etc. As a basis, a normalized value is required for each context element of each Fog node in a given group. The calculation for the case when a higher raw value is better is illustrated in Equation 1.

$$v_l^x = \frac{raw_l^x \times uw_l^x}{\sum_{i=0}^{|O|} raw_l^i \times uw_l^i} \tag{1}$$

where:

– v_l^x denotes the normalized value of context element l of Fog node x.

– raw_l^x denotes the raw value of context element l of Fog node x. This is the rating value for the resource of the Fog node that denotes the capability of the Fog node, usually in regards to an execution of a common algorithm or a benchmark.

– uw_l^x denotes the utilization weight of context element l of Fog node x. The weight can be used to account for the actual unutilized percentage of a resource on a Fog node. In this case, the weight would be equal to the idleness of the Fog node in the given context l (i.e. $1 - \mathcal{U}_l$, where $\mathcal{U}_l$ denotes the current load).

– O denotes the set of all Fog nodes, who belong to the same group.

For the case when a lower raw value is better (e.g., number of intermediate hops involved in delivering the result to the user), simply the formula (1 - v_l^x) is used.

Based on the data from Equation 1, the overall grade per resource context can be calculated, in order to gain a preliminary estimate on the ratio of the work items to be taken by any given Fog node. This concept is illustrated by Equation 2.

$$grade_l^x = \frac{\sum_{l=0}^{|C|} v_l^x \times cw_l}{\sum_{i=0}^{|O|} \sum_{l=0}^{|C|} v_l^i \times cw_l} \tag{2}$$

where:

– $grade_l^x$ denotes the preliminary estimate of the ratio of all work items to be taken by Fog node x that utilize the resource of context l.

– v_l^x denotes the normalized value of context element l of Fog node x.

– O denotes the set of all Fog nodes, who belong in the same group.

– C denotes the union set of the current context and common contexts. Common contexts are the ones that impact the performance of all the contexts (e.g., the network speed capability).

– cw_l denotes the context weight of context element l in the overall perspective. Not all contexts may be equally important, e.g., number of hops to the user may be considered less important than the actual CPU capability of the Fog node.

The estimate on the actual number of work items for fog_x to handle (denoted by FW_l^x) is thus deducible from the grading value and the number of all works. This concept is also illustrated in Equation 3.

$$\#FW_l^x = [grade_l^x \times |W_l|] \tag{3}$$

where:

– W_l denotes the set of all work items with the given context l as the primary resource.

The formulae have been validated experimentally and the results are reported in Section 6.3.

4.5 Results Delivery

In general, there is a possibility that the delegator node has moved out from the range of the worker node before the worker node has completed the tasks and delivered the result to the delegator. Fundamentally, there are two basic approaches for handling the situation.

1. *Worker Network Routing*. As mentioned previously, the Cloud backend has chosen the Fog nodes based on the priority of the connectivity (see Section 4.2). Hence, the workers can always attempt to route the process result to the node that is currently connected with the delegator.

2. *Cloud-assisted Routing*. In the case of missing routing path to the currently connected Fog node or due to heavy traffic among the nodes within the routing path, the worker can choose to route the process result to the delegator via the Cloud backend.

In order to identify the best approach for the process result delivery, the workers may need to continuously update the network status. Considering the status needs to be up-to-date, the workers will only keep the information in the vicinity (i.e. within the group). Every time some Fog node in the group has to transmit data to the Cloud backend, it would also keep a record of the communication speed. Hence, if any node has a choice to possibly transmit data to the cloud (or alternatively use the Fog node network), it would aggregate the communication speed data, and compute the weighted average of the times to the cloud, where the most recent communications have the highest weight. For optimization reasons, some of the calculation results may be cached, in order to improve the performance of the system.

Another crucial aspect would be the distance of hops from the original worker that is handling the computation to the current Fog node that the delegator is connected to.

The distance in hops could be statically calculated, assuming that each Fog node would know at least the Fog nodes in its vicinity (i.e. a subgraph of the vicinal network of Fog nodes) or a similar approach as for the Cloud context can be used. As the user is constantly moving, at each point in time when the user connects with a Fog node, this node would publish a message to the topic of the user's location. Only the Fog nodes that have some outstanding computation for the user would subscribe to this topic.

4.6 Summary

This chapter introduced a mobility-aware approach for proactive Fog service provisioning, in order to improve the process of task distribution and handling in Mobile Fog use cases.

In this chapter, the author proposed an algorithm for candidate worker selection based on the delegator's route. Particularly, the proposed service provisioning approach involves a discussion on the formation of the worker network to better partition the tasks between workers, including the various aspects and the benefits of the approach.

Further, the author proposed a context-aware extension to the basic multi-layered work-stealing methodology of task distribution, which allows the tasks to be better distributed based on various characteristics of the workers and tasks.

Finally, the author discussed different approaches for delivering the results, along with the methodology to identify the best approach for any given moment and scenario.

5 Fog Social Network

In Mobile Mesh Social Network (MMSN), mobile device users are capable of performing various virtual social network activities in the physical vicinity within the wireless network range.

Ideally, the Wireless Mesh Network (WMN) topology used by MMSN is flexible and scalable since it does not rely on particular peers to be the cluster heads (super peers). Moreover, the characteristics of MMSN such as the Internetless activities and WMN-based connectivity has potential including but not limited to business opportunities, scalable crowdsourcing or crowdsensing deployment, edge computing and so on. However, it may also face a number of limitations, such as unstable connectivity, bottleneck issue due to heavy traffic, lack of a routing path between peers, etc.

One promising solution to overcome the limitation of MMSN is utilizing the Fog computing model. This chapter provides a case study for applying Fog Computing in Mobile Mesh Social Network, which is termed as Fog Social Network (FSN).

5.1 Overview of Environment

Figure 5 illustrates an FSN environment. FSN is a multi-layered system that consists of Cloud and Edge network.

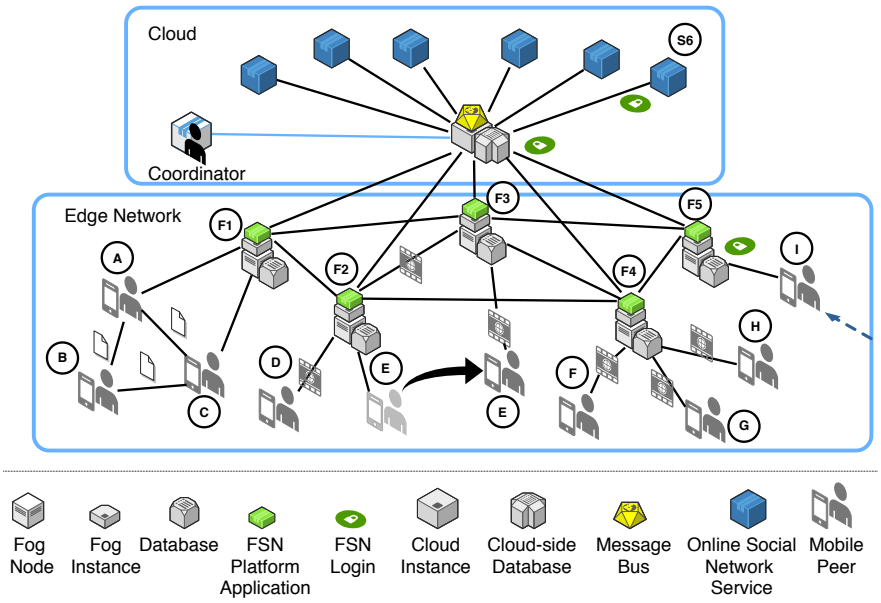


Figure 5. Fog Social Network environment.

Cloud
The upper layer Cloud consists of following main elements:

- *Online Social Network Service (OSNS)* providers (e.g. Facebook, Twitter, Foursquare) are the sponsors of FSN. They handle the end users' mobile Internet access bandwidth cost for their bootstrapping to FSN. The bootstrapping phase will be described later. This assumption is based on the incentive that an SNS provider is willing to offer free Internet for limited accessibility [BBC16].

- *Message Bus (MB)* is an elastic Cloud service for routing messages between OSNS and Fog nodes. It is managed by the Coordinator. MB only routes data or message exclusively to the corresponding OSNS servers. For example, a Facebook User-A, who signed in to FSN can share content to another Facebook User-B via FSN. Since the content will be sent through a Fog node, Facebook can subscribe for such content via MB in order to synchronize the content to User-A and User-B's Facebook accounts. On the other hand, since the content was from a Facebook user, Twitter servers will not be able to access or subscribe to it.

- *Coordinator* is a trustworthy party who has been delegated the responsibility to manage the MB for the OSNS providers. The coordinator can be an individual organization established by the OSNS providers or it can be the Cloud service provider itself. Note that although the data is routed in MB, the Coordinator does not store any user data in its database. The Coordinator only maintains the up-to-date FSN deployment package and the connection states of FSN users.

Edge Network
The lower layer Edge Network consists of Fog nodes (e.g. F1, F2, F3, F4 and F5 in Figure 5) and the participative mobile peers (users) of FSN.

The edge network of FSN consists of the following characteristics:

- *Retain social activities without the Internet*. For example, Node-A, Node-B and Node-C can still communicate in the P2P mode.

- *Rapid file sharing regardless of mobility issues*. For example, Node-D intends to share a large file to Node-E. However, Node-E has moved away from the signal range of Node-D. In this case, Node-D can forward the file to Fog node F2. Afterwards, Node-E will receive the file when it connects with another Fog node (e.g., F3). It is achieved based on utilizing the MB in Cloud that helps to maintain the peers' connection states. In other words, F2 will try to track which Fog node coverage Node- E is in by utilizing the MB, then F2 will forward the file to that node either by direct connection (e.g., they are in close proximity or connected to the Internet via the same network provider) or via the MB.

- *Large size file sharing with low latency*. For example, Node-F can share a large size media file to Node-G and Node-H via F4.
- *Secure sign-in*. A newly joined peer (e.g. Node-I) who has an OSNS account can establish its connection to FSN with secure sign-in. The details will be described in an upcoming paragraph.
- *FSN Platform Application package* is a software that can be deployed on a Fog node. It helps users to connect to the FSN. Fundamentally, it works similar to a regular MMSN router node. Additionally, it has a connection to the Cloud-side MB to assist FSN users with signing-in to the FSN without utilizing their own mobile Internet. In summary, FSN Platform App provides the following basic mechanisms:
  - Assist FSN participants with the sign-in process.
  - Provide subscription for OSNS to their users' content.
  - Assist FSN participants in the discovery of their vicinity.
  - Assist FSN participants in interacting with their surrounding devices including FSN user devices and IoT devices.
  - Temporary storage of data and delivering files via Fog Director cluster to receivers.
- *General Access*. General desktop computer users can also use FSN via Fog node or even directly subscribe to the MB.

Sign-in
In MMSN such as FireChat, users need to access the Internet in order to sign-in. In FSN, it is handled by the Fog node. For example in Figure 5, Node-I can send a sign-in request to F5. F5 will then forward the request to MB, who will forward the request to corresponding OSNS server based on the account used by the initial requester. Suppose Node-I is using the account of OSNS provider S6. Hence, MB forwards the request to S6. If the account of Node-I is valid, S6 will confirm the request is valid, then S6 will send a request to the Coordinator to help Node-I in establishing a connection to FSN via F5.

Deployment
If the FSN user moves to an environment where the FSN has not yet been deployed, the user can send an FSN deployment request to the Fog node in the vicinity. For example in Figure 6, the request message will be forwarded to the MB Coordinator. The Coordinator will confirm the user account with OSNS providers. If the account is valid, the Coordinator will send the FSN platform deployment package to the Fog node

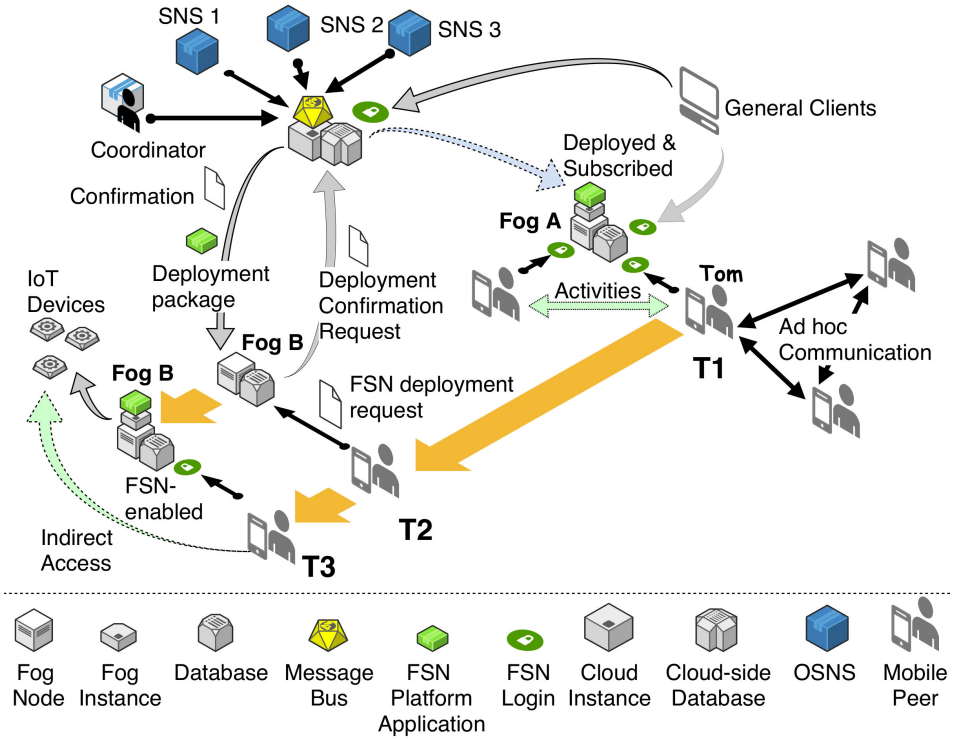


Figure 6. Fog Social Network deployment example.

and deploy the FSN. Afterward, the requester will receive an acknowledgment and it will be connected to FSN. On the other hand, if the Fog node has previously deployed FSN platform and the image is still available in its local storage, then it can deploy the FSN platform without needing the Coordinator to send the package to it again.

MMSN applications in FSN can use the Fog nodes not only for transmitting messages, but they can also use Fog nodes for further needs such as retrieving information from environmental publicly accessible Internet of Things (IoT) devices. This topic is considered as one of the future research directions.

Adaptation

Social activities will be based on both MMSN and FSN. For small activities such as text messaging or small file sharing between participants, the FSN application will calculate the efficiency. It will use direct P2P when:

- **Reliability**. Direct P2P signal strength is high (connection is good) [GB07] and the mobility of the participants will not affect the complement of their data transmission. This can be measured based on the accelerometer of their mobile devices and the latency testing based on the measurement of the data transmission.

- **Agility**. Direct P2P transmission will be faster than utilizing the Fog. It needs

to be calculated based on comparing the two approaches (the Fog-based and the direct P2P-based).

- **Cost**. Direct P2P transmission does not require more cost than utilizing Fog. The cost can be for example, the energy consumption. It involves the protocols and data size they are using. For example, Fog nodes are commonly Wi-Fi-based (based on Cisco's Fog Computing approach). If the data transmission required between two participants can be done rapidly using Bluetooth, then the cost-performance value of the direct P2P-based approach can be higher. The details of the cost-performance computing will be discussed in next section.

Constraint
FSN may not provide regular Web browsing to its users, neither the content hosts in their OSNS servers. Users who intend to use the Fog node for additional Internet access may need to handle the cost by themselves, unless either the Fog provider or the OSNS provider is willing to cover the cost.

5.2 Cost-Performance Index Scheme

If the entire system is fully relying on one approach, it may not be adaptive due to the dynamic nature of mobile network environment. For example, in some cases, utilizing the classic mobile peer-to-peer (MP2P) communication can be more efficient than utilizing Fog when the participants are sending small size messages. On the other hand, Fog can be more efficient than MP2P when the participants intend to share large size media files. Furthermore, in the case where the participant is situated in a high-density area, where Wi-Fi and Bluetooth signal interaction is occurring frequently, which can make the short range based communication extremely slow, the Cloud-based approach may become a better option. With this in mind, the proposed FSN framework also contains an adaptive resource-aware Cost-Performance Index (CPI) scheme, which can autonomously decide which is the best route to deliver the messages for MMSN nodes based on the runtime context factors. This section provides the details of the CPI scheme.

Let $O = \{o_i : 0 \leq i \leq \mathbb{N}\}$ be a set of routing options (e.g. via mobile P2P, via Fog, via Cloud etc.).

Cost of an option $o_x \in O$ (denoted by $cost_x$) is computed by:

$$cost_x = \sum_{j=0}^{|C|} \left(\frac{v_j^x}{\sum_{i=0}^{|O|} v_j^i} \times \left(1 - \sqrt{1 - (w_j)^2} \right) \right) \tag{4}$$

where:

– $C = \{c_j : 0 \leq j \leq \mathbb{N}\}$ is set of cost elements considered by the sender.

– v_j^x denotes the value of the cost element c_j of option o_x.

– v_j^i denotes the value of the cost element c_j of option o_i.

– w_j is the weight of the cost element c_j.

Figure 7 illustrates the generated value from $1 - \sqrt{1-(w_j)^2}$ where $\gamma = 1 - \sqrt{1-(w_j)^2}$, which defines the stimulus of weight to the cost.

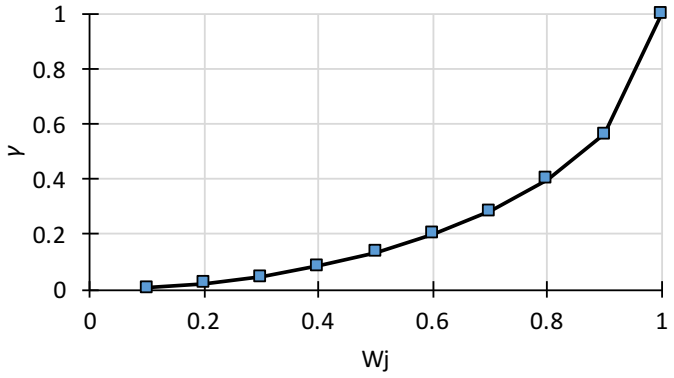


Figure 7. The weight calculation.

The weight w is a dynamic value influenced by the hardware usage. The higher the current hardware usage, the higher the weight can be.

The weight of a cost element $c_y \in C$ (denoted by w_y) is computed as below:

$$w_y = \frac{(\mathcal{U}_y)^{con}}{\sum_{j=0}^{|C|}(\mathcal{U}_j)^{con}} \tag{5}$$

where:

– $\mathcal{U}$ is the current average resource usage value. It is between 1 to 100 (representing 1% to 100% of the resource usage). For example, “CPU = 67" denotes CPU currently has 67% usage of its available threshold; “battery = 30" denotes battery has been used up to 30% (i.e. 70% left) of its available threshold.

– con is the value that can influence the constraint of the weight and $con \geq 1$. By default, con is 2, in which the variety of the weight increases explicitly high when it is closer to the threshold.

Note that the available threshold does not necessarily equal to the total availability of the hardware resource. The user can decide for example that the total available threshold of the energy is 70% of the hardware battery capacity. The details of how the user can

define the threshold will involve the user interface design, which is out of the scope of this thesis.

The performance of each option is measured based on their timespan. Let $\mathscr{T} = \{\tau_i : 0 \leq i \leq \mathbb{N}\}$ be a set of timespan values of the options, where τ_i denotes the timespan of o_i. Let τ_x be the normalized timespan value of option $o_x \in O$, which is computed by:

$$\tau_x = \frac{\sum_{l=0}^{|E^x|} \frac{S}{\mathcal{M}_l - ED_l}}{\sum_{i=0}^{|\mathscr{T}|} \tau_i} \tag{6}$$

where:

- $E^x = \{e_l : 0 \leq l \leq \mathbb{N}\}$is a set of node pairs for delivering a message from sender to the receiver involved in option o_x.
- S is the size of the data that needs to be sent from the sender to the receiver(s). It is formatted in Megabit (Mb).
- $\mathcal{M}_l$ is the default transmission speed of pair e_l in Megabit per second (Mbps), which depends on the protocol used by the involved nodes.
- ED is the sum of the delays of all the environmental delay factors $\mathfrak{D}$. $\mathfrak{D} = \{D_m : 0 \leq m \leq \mathbb{N}\}$, which are dynamic delay value influenced by the environmental network conditions. For example, in a high-density environment, a large number of wireless network transmissions are happening, the ED can be high.

Let τ' be the normalized value of τ. For option $o_x \in O$, its normalized τ value (denoted by τ'_x) is computed by:

$$\tau'_x = \frac{\tau_x}{\sum_{i \in |O|} \tau_i} \tag{7}$$

Afterward, the performance rank of an option $o_x \in O$ (denoted by $perf_x$) is computed by:

$$perf_x = 1 - \tau'_x \tag{8}$$

where the lower timespan the option involves, the higher performance rank it has.

Finally, the CPI score of option $o_x \in O$ (denoted by CPI_x) is computed by:

$$CPI_x = \frac{perf_x}{cost_x} \tag{9}$$

Higher CPI values imply a more cost efficient approach, while lower values indicate that the cost is higher than the gained performance value.

5.3 Summary

This chapter introduced Fog Social Network, a framework to enhance the reliability and cost efficiency of Mobile Mesh Social Network.

Initially, the overview of the environment described various aspects of the multi-layered Fog Social Network approach.

These aspects included the elements of the Cloud layer, the Edge network and its characteristics, sign-in process, adaptation of the system in relation to various characteristics (e.g., the agility of a given routing option, etc.).

In addition, the author introduced the Cost-Performance Index (CPI) scheme, in order to improve the system's ability to adapt to different scenarios caused by the dynamic nature of the mobile network environment. The decision is made by weighing the performance of the possible routing approach to the cost, by calculating the CPI value of the approaches.

6 Evaluation

This chapter consists of the experimental evaluation of the previously introduced chapters and concepts of the thesis, corresponding to the Fog service discovery, mobile Fog computing and the proposed Fog Social Network.

6.1 Prototype System Architecture

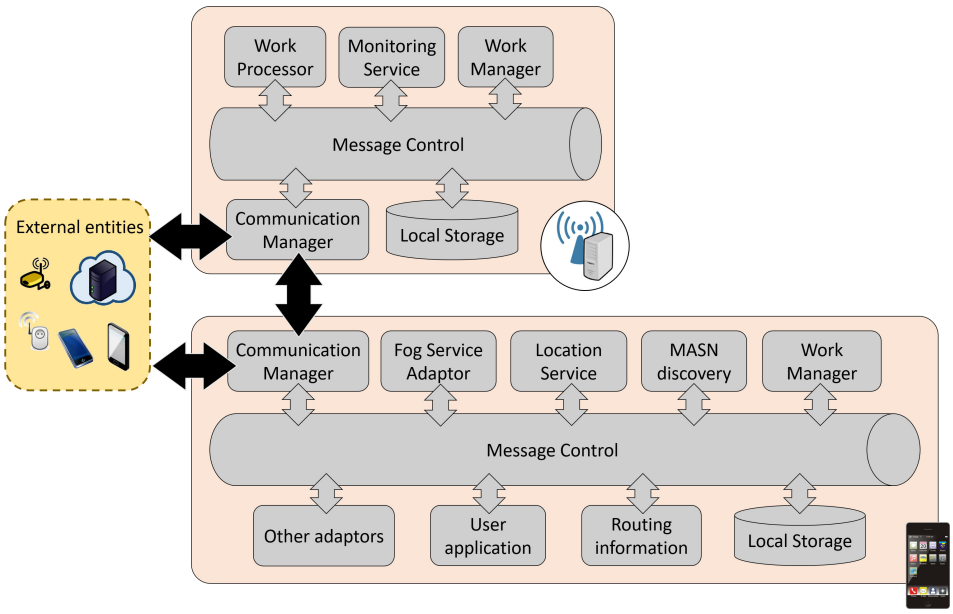


Figure 8. Components of the prototype system architecture.

Figure 8 illustrates the main components of the proposed prototype architecture, which consists of the components of Fog server node on the top and the components of the end-user node (i.e., mobile host) on the bottom. Correspondingly, the descriptions of the components are as follows.

Fog server

Work Processor is responsible for the actual processing of the work, by running the runnable to process the data.

Monitoring service is responsible for handling the information regarding the current hardware usage of the Fog node. This is the component that will have the most influence in the context of whether or not work will be stolen or executed.

Work Manager handles the work items and runnables. On the Fog node it is responsible for requesting and distributing of work runnables, work items and results. It would

also be responsible for the downloading of any runnables, such as the jar file or Docker image.

Communication manager enables the interaction between the Fog server and the MASN peers, IoT sensors, and other external entities.

Local Storage component is responsible for the local persistence of the work items and runnables. This is also the parent component for the work item queue.

Message Control leverages the interaction among the components of the system.

Mobile host

Communication manager receives or transmits the requests and responses between the the user's device and the Fog node or any other possible external entities, such as interaction with the MASN peers, IoT sensors, etc.

Fog Service Adaptor is a component responsible for keeping the user's device connected to a Fog node, including the autonomous seeking and reconnecting with a new Fog node (with the assistance from the other components) in the case of a disconnection.

Location Service consists of map capabilities and GPS mechanism. It provides a current location for the other components when necessary.

MASN Discovery component handles the peer discovery in MASN.

Work Manager handles the work items and runnables. Its responsibilities would include the distribution of the work runnables, work items and also for receiving the results of the processing.

Routing Information component handles the knowledge of which peers the device should send the message to in order to let the message reach the destination node. It records the intermediate peers that previously have provided a connection to a certain node, based on the peer-to-peer traffic and peer requests.

User Application represents the end-user application which requires the assistance from Fog.

Other Adaptors denotes the other components the system uses to interact with IoT services and other external entities. Generally, these are the components that utilize RESTful APIs to interact with external service providers.

Local Storage component is responsible for the local persistence of the work items, runnables and the messages regarding discovery, e.g., Fog Discovery messages.

Message Control leverages the interaction among the components of the system.

6.2 Proactive Fog Service Discovery

The aim of this section is to evaluate the process migration and the routing algorithm involved in the Fog service discovery, as described in Chapter 3.

The evaluation consists of two parts.

1. The first part validates the proposed routing algorithm, which has been tested by using an HP Elitebook 840 (Intel i5-4200U@1.60GHz, 12GB RAM) with the ONE simulator (v1.6) [Ker09].

2. The second part evaluates the performance of the task migration between Fog nodes. It involves the following devices: HP Elitebook 840 as *Fog node 1*, Lenovo V570 (Intel i7-2670QM@2.2GHz, 16GB RAM) as *Fog node 2*, LG G4C as *MASN peer 1*, ZTE Maven as *MASN peer 2* and Nexus 5 as *MASN peer 3*.

6.2.1 Evaluation Scenario

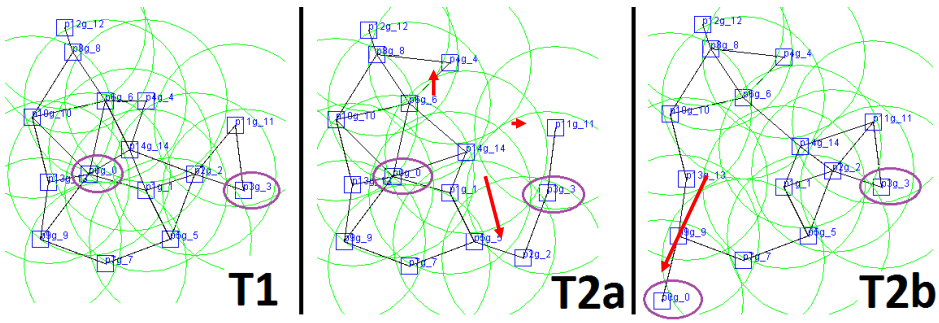


Figure 9. ONE simulator node movement scenario at timestamps T1, T2a, T2b.

The use case scenario of the evaluation follows three timestamped steps: T1, T2a and T2b (Figure 9). T2a and T2b are both successors to T1. The thin lines between the small squares denote the connections between MASN peers, thin circles represent the Bluetooth signal coverage of the peer when it broadcasts messages. The peers denoted with bold ovals are p0g_0 on the left and p3g_3 on the right. Arrows denote peer movement in-between timestamps.

In the scenario, node p0g_0 sends a message to node p3g_3 (T1), to which node p3g_3 responds. However, in the time the peers route the message from node p0g_0 to node p3g_3 and back again, some of the intermediary nodes (T2a) or even the terminal node (T2b) may have moved. The routing scheme, as described in a previous section, has different ways of handling these situations. It depends on whether the current node can assume that the path taken by the initial message still exists or not. Assume the initial message followed the path of p0g0_0 $\rightarrow \cdots \rightarrow$ p2g_2 $\rightarrow$ p3g_3. In the case of

T2a and T2b, node p3g_3 tries to send the response message back through the path of the initial message.

In the case of T2a, node p2g_2 has moved and is no longer connected to the next link. The peers closest to the last known location of node p0g_0 to which node p3g_3 can connect to, are node p11g_11 and node p2g_2. Since node p11g_11 is a dead end, it leaves the path p3g_3 $\rightarrow$ p2g_2 $\rightarrow$ p5g_5 $\rightarrow$ p1g_1 $\rightarrow$ p0g_0, which requires 4 hops.

In the case of T2b, node p0g_0 has moved away from its previously known path, which is known by node p3g_3. At first, node p3g_3 would assume the backward routing path exists, because the 2-hop distance would include all the nodes necessary. However, as soon as the message is forwarded to node p2g_2, it determines that the path in fact no longer exists and opts for the flooding approach as described in the routing section previously. This eventually forwards the new message to node p0g_0. Consequently, the delivery time and the number of messages sent among the peers have increased.

6.2.2 Experimental Results

Message Routing

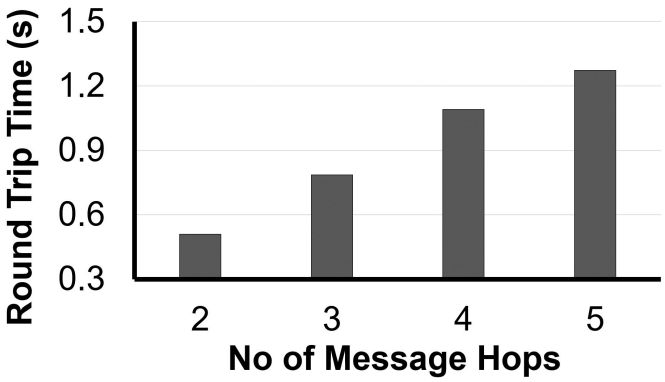


Figure 10. Round trip time per number of message hops.

Figure 10 illustrates the simulation results on the relation of the number of hops that the message travels. It starts from the initiator peer, who sends the request message, to the destination recipient peer who holds the information of the Fog node in proximity. The results show a near-linear correspondence of the variables. Since the ONE simulator's radio link is abstracted to a communication range and bit-rate (no signal attenuation, etc.), these results may differ from the real world use cases.

The real world results may also differ due to user settings. Although sending to more nodes may improve the message delivery rate, it also drains more battery, and therefore the user may decide to decrease this setting.

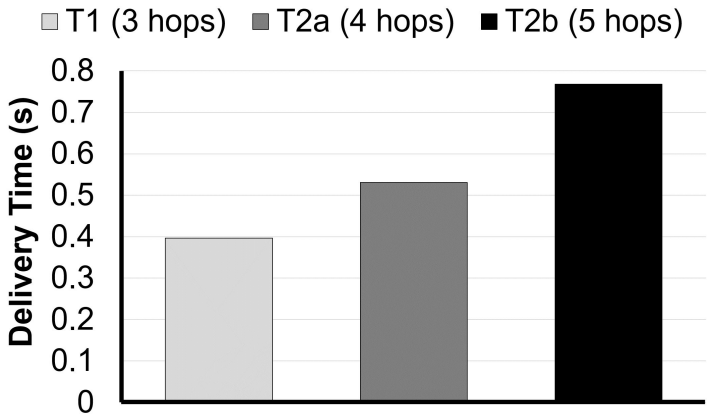


Figure 11. Node movement effect on message routing.

Figure 11 attempts to show the overhead induced by the node movement changes illustrated in Figure 9. The results show that the delivery time and the number of hops required to send a message to the destination peer have increased. This is due to the node movement and consequent differences in routing.

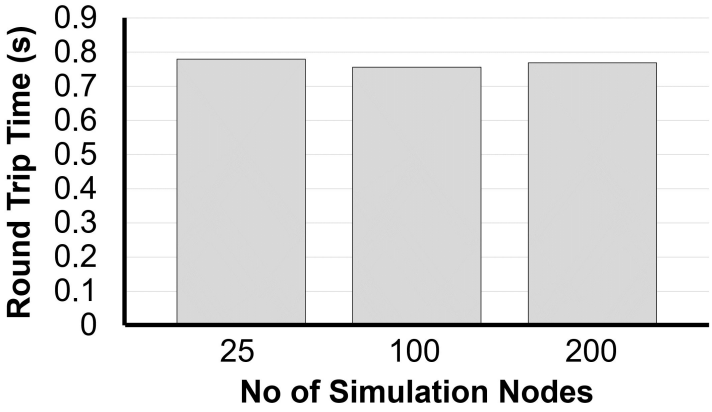
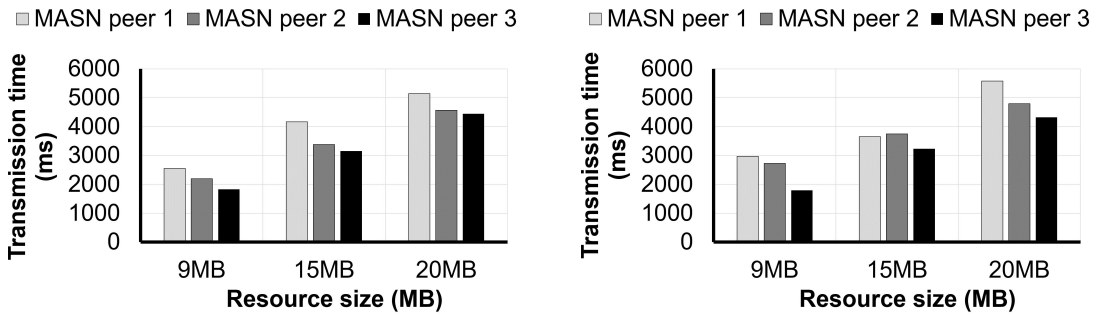


Figure 12. Effect of density on message routing.

Figure 12 illustrates the effect of the number of nodes in proximity to the overall efficiency of the message routing.

In general, the sender node only sends messages to a finite subset of the closest nodes in proximity, with the messages' Time-To-Live also being a limiting factor. Therefore, based on these simulations, it can be seen that even with a high number of proximal peers, the number of hops or the time taken remain relatively constant, even with the overhead induced by node movement.

Task Migration Performance



(a) Transmission time to Fog node 1.

(b) Transmission time to Fog node 2.

Figure 13. Web Application Resource transmission time from MASN peers to Fog node.

Figure 13a and Figure 13b illustrate the time taken by a specific mobile node to send the Web Application Resource (WAR) file to the Fog node for deployment onto the application container (e.g., Apache Tomcat[14]). The maximum time measured was approximately 5.5 seconds when data was being sent from MASN peer 1 to Fog node 2.

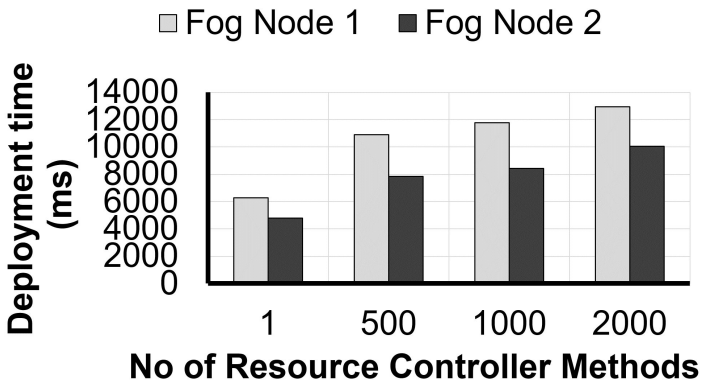


Figure 14. Web Application Resource deployment time.

Figure 14 illustrates the measurement of minimal standard controller endpoint methods for a RESTful web service written in Spring Framework[15] (v4.3.2) and bundled into a WAR file to be deployed to a Tomcat (v8.5.4) web server. An example of a single controller endpoint method would be a single handler for a single RESTful call, e.g., the handler for HTTP GET method for the URL: `http://www.example.com/`

[14] `https://tomcat.apache.org`
[15] `https://www.spring.io`

`app-1/api/ping` that returns an HTTP response consisting of the current timestamp. The measurements were conducted from starting with an application with a single controller endpoint method to 2000 such methods. The maximum time reached was approximately 13 seconds with 2000 controller methods. The deploy times between the devices are roughly comparable.

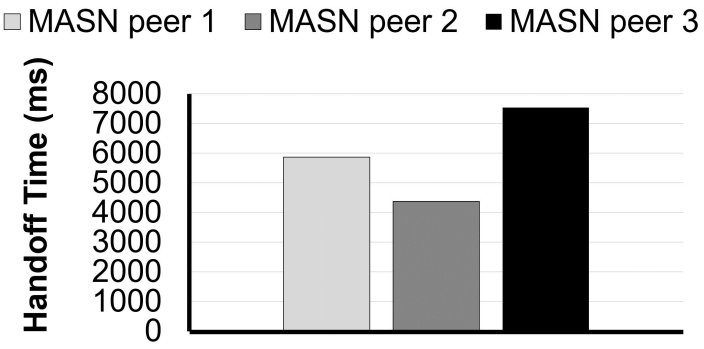


Figure 15. Smartphone experimental Fog handoff time.

Figure 15 illustrates the average time of a use case in which a mobile node migrates its process from one Fog node to another Fog node.

It was measured from the moment the mobile node was disconnected from Fog node 1 until it connected to Fog node 2. Rather interestingly, the MASN peer 3 (Nexus 5) was the slowest with the average time at 7.5 seconds. This was most likely influenced by the Wi-Fi adapter hardware. It also indicates that the result shown in Figure 9 (by the ONE simulator) can be quite different in the real world.

6.3 Proactive Mobile Fog Computing

This section aims to evaluate the performance of the proposed proactive mobile Fog computing scheme described in Chapter 4. The evaluation consists of the task handling performance, Docker image transfers and task execution.

6.3.1 Reactive and Proactive Task Handling Performance

This section aims to evaluate the performance between the proposed proactive approach and a reactive approach. In the reactive approach, the user's work items are expired upon disconnection with one Fog node and retransmitted upon connection with a new Fog node, as compared to a proactive approach, where the assumption is that the Fog nodes can transmit the results back to the user via the local network between Fog nodes.

The devices involved in this testing were as follows:

- Fog-1 and Fog-2 — HP Elitebook Folio 9470m (Intel i5-3437U, 8GB RAM).
- Delegator — Nexus 5 smartphone (LG-D821).

The experiment begins with the delegator transmitting a registration message to Fog-1. Fog-1 then steals work item(s) and also the runnable from the delegator. In the current experiment, only a single work item exists. As soon as Fog-1 begins the calculation, the delegator disconnects from Fog-1 and connects with Fog-2.

The sizes of the runnable and result are constant values of 25MB. The calculation time is a fixed value of 5 seconds on both of the Fog nodes. The work item data is a varying unit with a size of 25MB, 50MB, 75MB or 100MB.

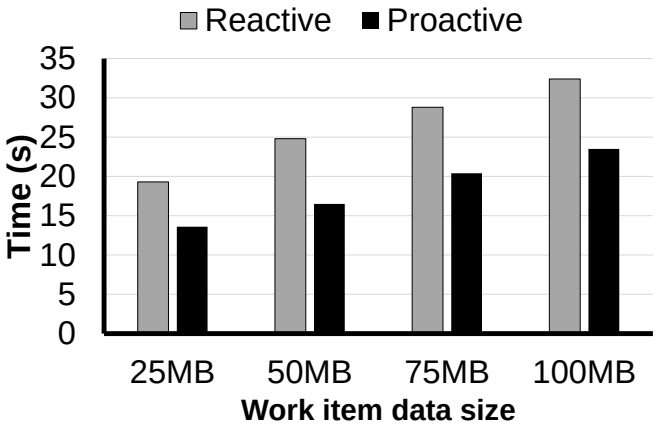


Figure 16. Comparison of reactive and proactive approach of utilizing Fog.

Figure 16 illustrates the differences in the context of utilizing the Fog with either a reactive or a proactive approach.

In the reactive case, the delegator transmits the work item and runnable initially to Fog-1 and upon delegator disconnection from Fog-1, the exact same data is transmitted again to Fog-2. No data is transmitted between the Fog nodes and the computation is simply dropped by Fog-1.

In the proactive case, the work item and runnable data is transmitted once to Fog-1. When connecting with Fog-2, there is nothing left to steal, since the only task was handed to Fog-1 and has not expired yet. When Fog-1 finishes the processing, results are transmitted back to the delegator via Fog-2 (i.e. the Fog node where the delegator is currently connected).

A lot of data needs to be retransmitted when using the reactive approach, therefore the proactive approach is shown to perform better under the circumstances.

6.3.2 Docker Image Transfer Performance

This section aims to evaluate the performance of using Docker both in the local scenario, where the user transmits the Docker image via smartphone using Wi-Fi, and also in the scenario, where the image is downloaded from Docker Hub, via image name and Docker provided API(s).

The experiments were conducted using Gigabit Ethernet connection for the Fog node to the Internet and 802.11n Wi-Fi network for smartphone communication.

The Docker images were chosen from the more popular real Docker Hub images listing, so that the file sizes (not compressed) would be near-linearly increasing (php:alpine 57.3MB, maven:alpine 115.7MB, python:slim 198.6MB).

The devices that were involved were as follows:

- Fog-1 — HP Elitebook 840 (Intel i5-4200U, 12GB RAM)
- Delegator — Nexus 5 smartphone (LG-D821).

Local Docker Image Transfer

The process starts with downloading the image file from the delegator and ends with loading the image into the Docker infrastructure running on a Fog node.

The compression used in the experiment was 7-zip normal preset with the standard deflate compression method.

Docker Image Transfer

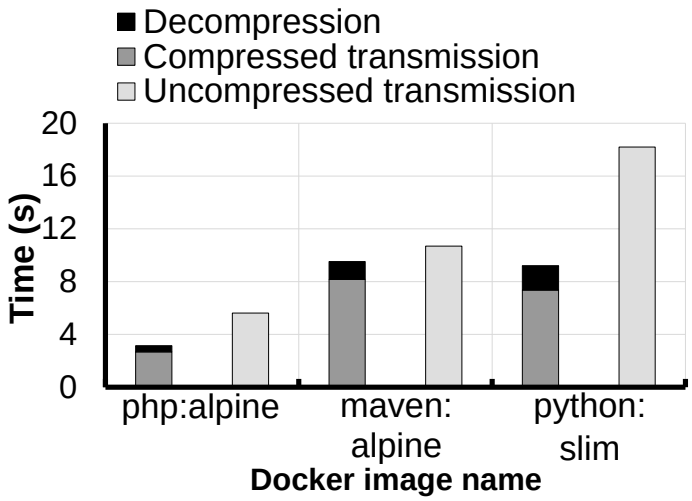


Figure 17. Docker image transmission time.

Figure 17 illustrates the Docker image transmission times for different images for both with and without the use of compression.

This shows that the transmission times can still be quite high, even in the local network. The relatively low speeds were most likely influenced by the Wi-Fi adapter hardware, especially that of the delegator.

In the case of transmission with using compressed files, the image files would have to be decompressed on the Fog node before they are used. Therefore, this experiment also includes the decompression time. It is important to note that the time improvement from compression may or may not be substantial, depending on the exact image. For example, in the specific case of this experiment, the compressed sizes of the files did not turn out to be linear. The compressed maven:alpine image was larger than the compressed python:slim image, which reduced the benefit of compression to under a second for maven:alpine. The sizes of the compressed images were approximately between 30-65 % smaller than their uncompressed counterparts (php:alpine 25.3MB, maven:alpine 77.5MB, python:slim 70.3MB).

Loading Image into the Docker Infrastructure

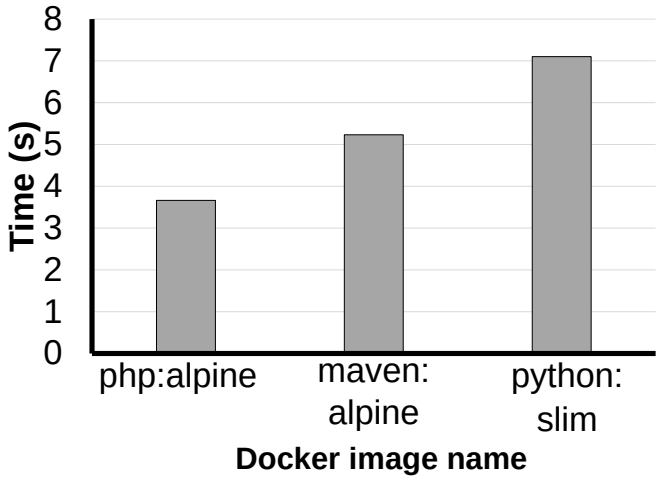


Figure 18. Docker image load time into Docker infrastructure.

Figure 18 illustrates the time taken to load the Docker image into the Docker infrastructure running on a Fog node. When this action completes, the Docker infrastructure will contain the new image that can be used to run a Docker container.

Total Local Docker Image Load into the Docker Infrastructure

Figure 19 illustrates the time for the composition of all intermediary tasks required to move the Docker image in the local network from the delegator to a loaded image ready to be deployed as a container on the Fog node, for both with and without using compression for the image files.

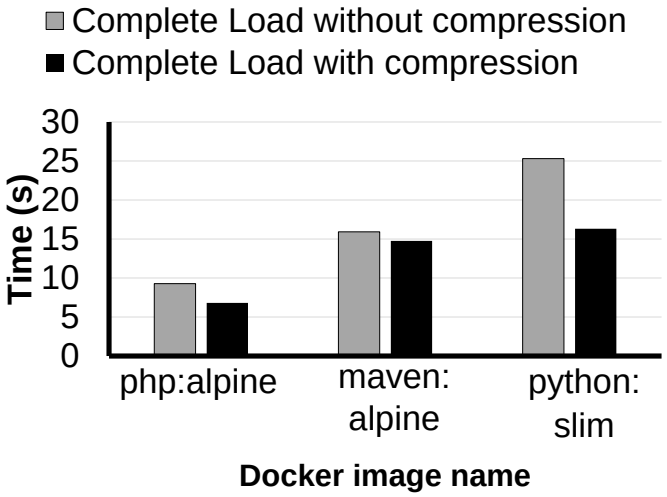


Figure 19. Total local Docker image load into Docker infrastructure.

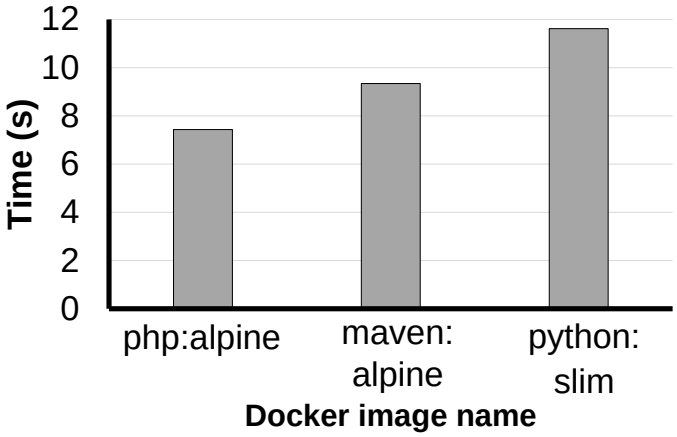


Figure 20. Total Docker image transfer and load via Docker Hub (Internet).

Docker Image Transfer via Docker Hub

Instead of using local files to directly transfer the images from delegator to Fog node, there is the option of using a link to, for example, a Docker Hub image as the runnable. In addition to the benefit of using fewer resources from the delegator to transfer the files, it may incur less overhead of the transfers altogether.

Figure 20 illustrates the complete time to download the image with all its layers from Docker Hub and loading the image into Docker infrastructure. This is the over-the-Internet equivalent of Figure 19. Even though the local network has the reduced latency due to the devices being in close physical proximity, the physical hardware limitations become very relevant when dealing with more constrained devices.

6.3.3 Task Execution Performance

This subsection aims to compare the performance of task execution by using direct node execution (tasks are not distributed, but solely calculated by directly connected Fog node), work-stealing approach or a simple round-robin task assignment (even number of tasks distributed to all nodes).

The devices that were involved were as follows:

- Fog-1 — HP Elitebook 840 (Intel i5-4200U, 12GB RAM), where approximately 50% RAM and 50% CPU were utilized before the start of the experiment, in order to account for a Fog node that is already busy with some other task beforehand.
- Fog-2 — HP Elitebook Folio 9470m (Intel i5-3437U, 8GB RAM), where approximately 10-15% of RAM was utilized by the OS by default.
- Fog-3 — Lenovo V570 (Intel i7-2670QM, 16GB RAM), where approximately 10-15% of RAM was utilized by the OS by default.
- Delegator — Nexus 5 smartphone (LG-D821).

The work items were either CPU-intensive or RAM-intensive tasks, where each category utilized mainly the CPU or RAM resources respectively.

CPU-intensive tasks were further subcategorized as small-cpu and large-cpu tasks, where the time to process small-cpu task was approximately equivalent to half of the large-cpu task. The tasks were designed such that the CPU would be kept utilized at about 70-80% usage level by one task on average.

RAM-intensive tasks were further subcategorized as small-ram and large-ram tasks, where the time to process small-ram task was approximately equivalent to half of the large-ram task. The tasks were designed such that approximately 3-3.5GB of RAM would be utilized by one task on average.

The size of the inputs for the tasks were 5MB for the small tasks and 10MB for the large-tasks. The size of the result data was 1MB.

The evaluation was conducted in the scenario of completing 30 tasks and 60 tasks. In the case of having 30 tasks in total, 20 tasks were CPU-intensive (10 small-cpu and 10 large-cpu tasks) and 10 were RAM-intensive (5 small-ram and 5 large-ram tasks). The case of 60 tasks in total followed the same overview as the 30 task variant, except there were exactly twice the number of each type of task. The delegator of the work was connected to Fog-1. Since there were tasks with CPU type and RAM type, then the formula used for estimated distribution of works also contained these main context values.

Task Execution Time

Figure 21 illustrates the differences of task execution times for the three approaches. Direct node execution performs the worst, because the most utilized node was handling

all the tasks. Round-robin statically assigns tasks to workers without taking into account the current load or other characteristics (e.g., computational power), and thus the work-stealing approach has shown better results in the context of execution times.

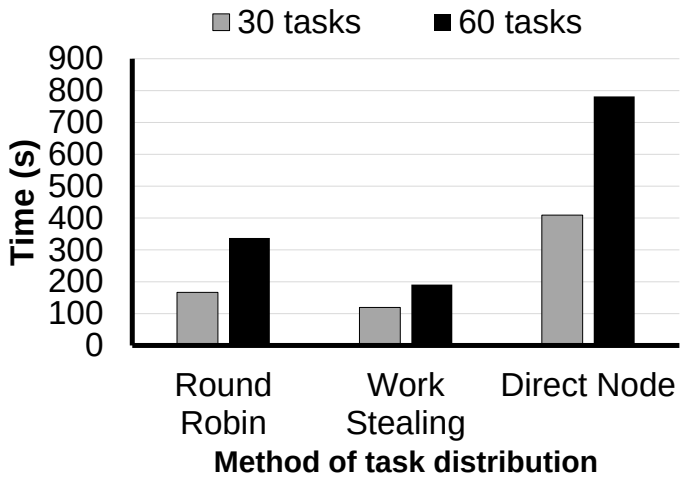


Figure 21. Task execution time.

Partition of Tasks Allocated to Fog Nodes

Figure 22 shows the partition of all tasks that were executed at any Fog Node, for all of the task distribution methods.

Direct Node Execution approach, by definition only executes on Fog-1 and does not distribute the works further.

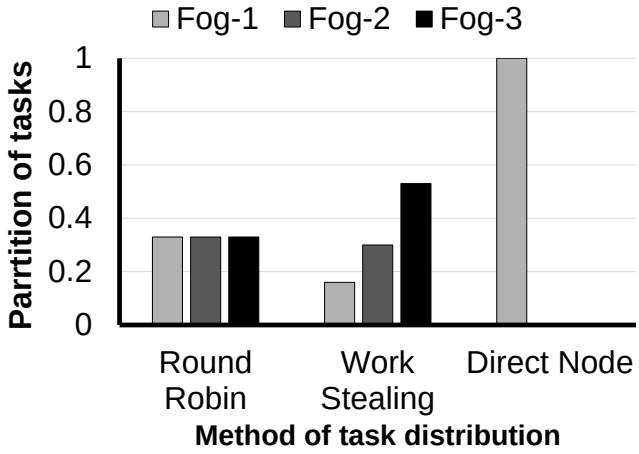


Figure 22. Partition of tasks distributed to Fog nodes.

Since round-robin does not take into account the different capabilities of the Fog nodes, the works are distributed uniformly over all the Fog nodes.

The work-stealing approach, on the other hand, considers the heterogeneous capabilities of the workers, therefore the Fog node with the highest characteristics is assigned the most work and the one with the lowest current capabilities has the least work items.

Task Distribution

Figure 23 illustrates the partitioning of types of work between Fog nodes, e.g., how were the small-cpu type of tasks partitioned between Fog nodes.

Since Fog-3 has a much more capable CPU than the other Fog nodes, it takes the majority of both the small-cpu and large-cpu typed tasks. Since Fog-1 is already considerably utilized both in the CPU and RAM categories, it will take the least amount of work items.

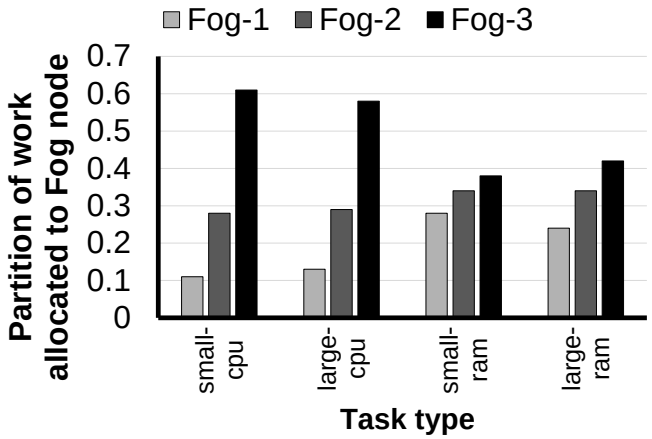


Figure 23. Work-stealing task distribution by task type.

The context of the RAM follows a similar approach. The reason for the partitions to not differ as greatly in this context is most likely because CPU-intensive tasks also use up a part of the RAM, therefore if a Fog node is busy with many CPU-intensive work items, it directly affects the amount of available RAM and thus the amount of RAM tasks to be processed. The CPU usage by the RAM-oriented work items is comparably smaller, thus not yielding the opposite effect in the other context. This is yet another aspect that cannot be easily taken into account by the static task distribution methods.

A similar figure regarding round-robin was omitted, due to the fact that it would show a uniform distribution of all types of work items between all Fog nodes. Similarly, Direct Node Execution would show everything executing on Fog-1.

6.4 Fog Social Network

The aim of this section is to evaluate the characteristics of the proposed Fog Social Network described in Chapter 5.

The goal of the evaluation is twofold.

1. Firstly, the aim is to evaluate the cost efficiency of utilizing Fog-based MMSN when compared to the classic MMSN and the Cloud-based social content delivery.

2. Secondly, the aim is to evaluate the proposed adaptive resource-aware CPI scheme with the collected data from real world experiments as the input values of the scheme.

The prototype is based on simulating an FSN environment where the communication of the participative entities is based on MQ Telemetry Transport (MQTT; ISO/IEC PRF 20922), which is a publish/subscribe-based protocol.

Publish/subscribe protocol is the core enabler of many social network applications such as Facebook Messenger, Jabber, Skype etc. Although Extensible Messaging and Presence Protocol (XMPP; IETF RFC 3920) may be more popular than MQTT since it has existed for many years, in this thesis, MQTT was chosen because it is lightweight and more feasible to operate in the resource-constrained environments.

6.4.1 Implementation

The main elements in the prototype have been implemented as below:

- MMSN participant—were simulated in LG G4C and LG Spring smartphones, which are operated in Android OS 6.0. For the classic MMSN use case, one smartphone was operating Moquette[16] MQTT broker/server and client at the same time. The rest of the smartphones were MQTT clients.

- FSN node—was simulated by a HP laptop computer, which had installed an Apache Tomcat server with MQTT broker/server Web application.

- OSNS server—was simulated by a Google cloud instance.

- Networking—The communication between MMSN participants and FSN node was based on Wi-Fi (802.11n). In the case of mobile to cloud communication, which simulated the traditional OSN activities, the communication is performed using Estonian Tele2's 4G LTE mobile Internet connection, which has on average 50 to 70 Mbps for download speed and on average 30 to 40 Mbps for upload speed.

[16] `https://www.github.com/andsel/moquette`

6.4.2 Use Cases

The evaluation aims to compare the cost and performance among the three models: OSNS, classic MMSN and FSN. The following main use cases were utilized in order to compare the models:

1. User-1, who has subscribed to the social network topic-A, intends to publish a multimedia content to the topic. Meanwhile, 3 other users have subscribed to the topic.

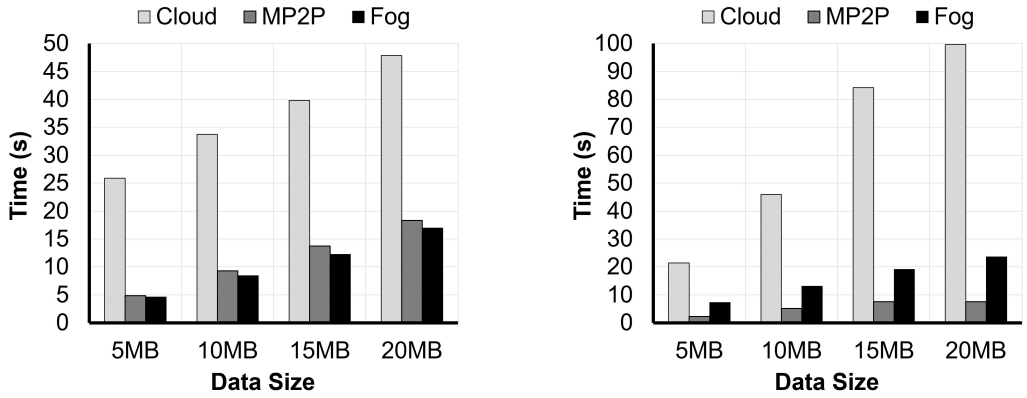
2. User-1, who has subscribed to three social network topics, is receiving three multimedia content published by the other 3 users to the three topics.

For each use case, firstly the cost and performance are compared when each of them is performed by OSNS, classic MMSN or FSN model. Afterward, the data collected from the real world testing is applied to the proposed CPI scheme to validate how well the system can autonomously adjust the decision making among the three options (i.e. publish the content via OSNS, classic MMSN or FSN).

6.4.3 Evaluation Results

Performance Comparison

Performance is based on the timespan measurement of each of the options, where different sizes of data were used as the published content.



(a) Timespan comparison for Case 1.

(b) Timespan comparison for Case 2.

Figure 24. Timespan comparison.

Figure 24a shows the measurement results in comparison of the three options in Case 1, known as publishing content of different data sizes to three topics. The timespan

represents the time from publishing the content to the time that all the subscribers have received the content and have acknowledged the MQTT broker node. As the result shows, Fog delivered the best performance, followed by MP2P. Cloud has the lowest performance.

Figure 24b illustrates the Case 2 in which a participant has subscribed to three topics and it has received the three new contents published from the other three participants. In this case, MP2P approach results in having better performance than Fog. Cloud still has the lowest performance.

Cost Comparison

In this evaluation, CPU usage and energy consumption are considered as the cost elements. The CPU usage measurement is based on recording the CPU usage while performing the activities. An example of the raw data of the CPU usage is shown in Figure 25.

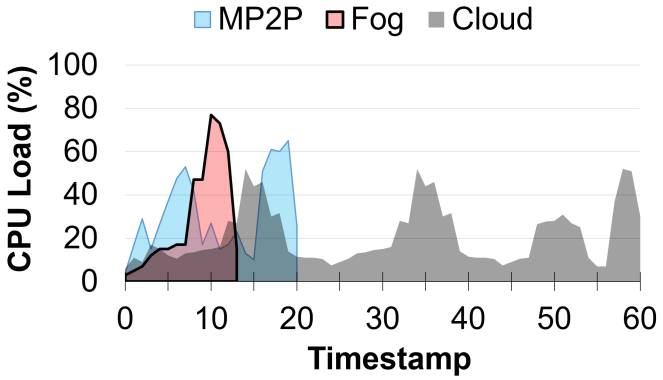
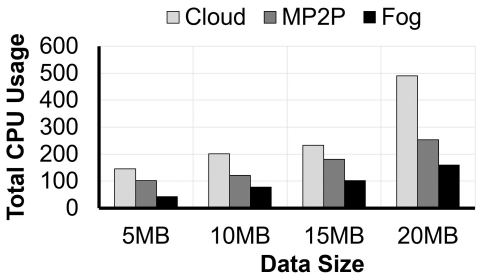
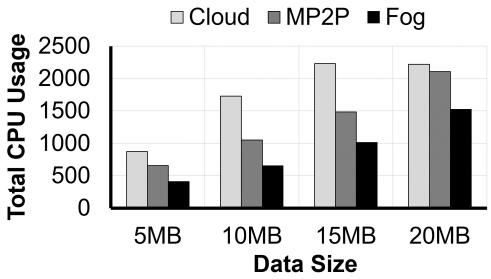


Figure 25. Raw CPU usage log example.



(a) Total CPU consumption comparison for Case 1.



(b) Total CPU consumption comparison for Case 2.

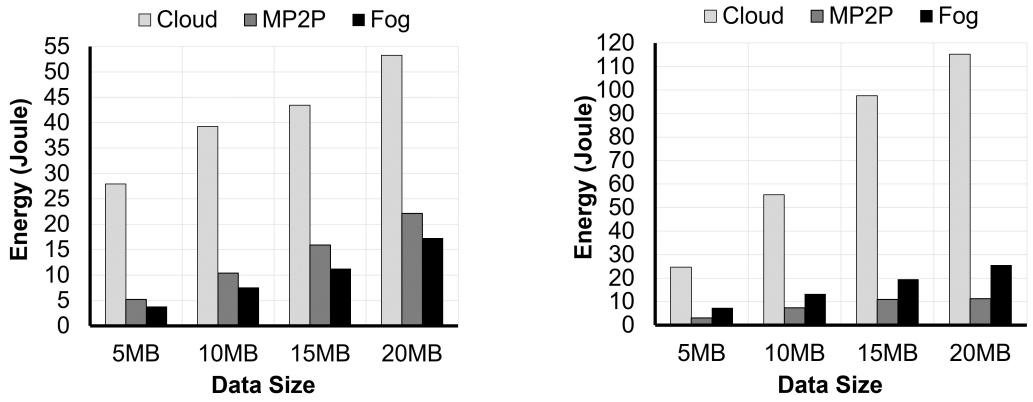
Figure 26. CPU consumption comparison

The total CPU usage is based on the usage value times the total number of the timespan. For example, while performing a content publishing task, if CPU load is 60% and it remains 60% for 5 timestamps, the total cost of the CPU will be considered as $60 \times 5 = 300$ raw cost value. Figure 26a illustrates the testing result for the CPU usage in Case 1 (publishing content to three topics).

As the result shows, Cloud-based approach consumes the highest CPU usage. Fog-based approach has the lowest CPU consumption.

Figure 26b illustrates the CPU consumption measurement and comparison of the three options in Case 2 (receiving content from three subscribed topics). A similar result was received as for the previous testing in which the Cloud-based approach is still the one which consumes the highest CPU usage, followed by MP2P. Fog-based approach still results as the approach that requires the lowest CPU usage.

Next, the energy consumption was tested using the same setting. Figure 27a shows the result of the energy consumption of the Case 1 (publishing content). As the result shows, Fog-based approach consumes the lowest energy. Cloud has the highest energy usage.



(a) Energy consumption comparison for Case 1.

(b) Energy consumption comparison for Case 2.

Figure 27. Energy consumption comparison

Figure 27b illustrates the energy consumption testing result for Case 2 (receiving content from three subscribed topics). Different to the previous testing result, in Case 2, MP2P-based approach consumed the least energy, followed by Fog. Cloud-based approach still consumed the most energy.

This result indicates that Fog-based approach is not always the best option in some cases. Therefore, utilizing CPI scheme can improve the cost-performance efficiency when considering different situations.

CPI Scheme

The aim of the evaluation of CPI scheme is to validate that the scheme is capable of ranking the options in order to identify which option is more efficient to perform the process.

This testing is based on the setting of both Case 1 and 2 with 10MB as the data size. CPU and energy consumption were considered as the cost elements of the options. Further, the timespan was considered as the performance. Based on the proposed scheme, the results shown in Figure 28 were received.

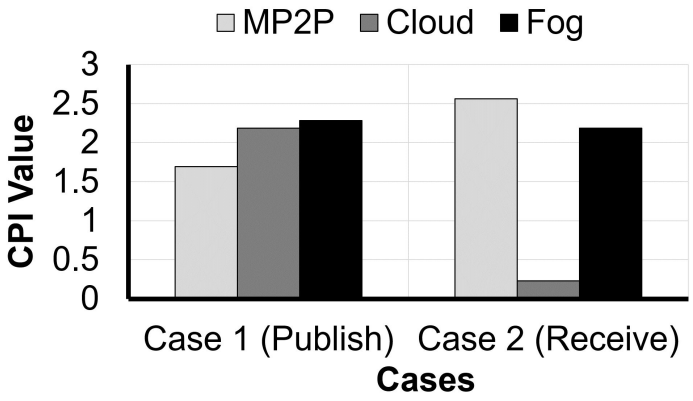


Figure 28. CPI comparison between Case 1 and Case 2 with 10MB content data size.

As the result shows, when the participant intends to publish a content to a topic which has been subscribed by three other participants, the Fog-based approach can provide the best CPI value. Hence, the content publisher node has chosen to publish the content via Fog. On the other hand, in the case of a participant who has subscribed to three topics and intends to receive the new content from the topics, the MP2P approach provides the best CPI value. Hence, the participant utilized the MP2P-based connection to subscribe to the topics.

Note that, these CPI computation measurements can be influenced by other factors such as the signal strength or RAM usage. More factors are considered to be included in future work. At this stage, the result has provided a proof-of-concept that the proposed CPI scheme is capable of computing the efficiency of different options based on the cost and performance parameters.

6.5 Discussion

The author has performed the evaluation of the proposed solutions in three segments.

Firstly, the author evaluated the Fog node discovery and process migration, showing the adaptive routing performance under various conditions. Peer movement has a negative effect on routing performance, as expected. However, the experiments showed the

ability of the adaptive routing mechanism to cope with the movement of both intermediaries and the target node. The results also support the system's ability to scale even with a high number of proximal peers. In addition, the author evaluated the task migration performance, comparing resource transmission time from mobile to Fog node, application resource deployment time and concluded with the summarizing measurement of smartphone handoff time from one Fog node to another Fog node.

Secondly, the author performed the evaluation on the proactive mobile Fog computing approach, comparing the reactive and proactive task handling approaches and showing the advantage of the proactive variant. Additionally, the author conducted the performance evaluation of Docker image transfers, showing that even though the local network has reduced latency due to the devices being in close physical proximity, the physical hardware limitations become very relevant when dealing with more constrained devices, which led to the better performance of the over-the-Internet variant of transfer. Furthermore, the chapter included the performance evaluations of the context-aware task distribution and execution as compared with direct node execution and round-robin task assignment. The experiments showed the proposed platform of this thesis as performing better in regards to execution time, partitioning the tasks in a more efficient way, and distributing the tasks more suitably among the workers, based on the characteristics of both workers and tasks.

Lastly, the author evaluated the Fog Social Network approach in the context of the cost efficiency of utilizing Fog-based MMSN when compared to the classic MMSN and the Cloud-based social content delivery, where the better performance of Fog is shown in the case of publishing data. The experiments show the cost of Fog to be the lowest in most of the evaluation cases. Based on the performance and cost measurements, the author validated the CPI scheme, showing the capability of the scheme to rank the data transmission options to identify the more efficient option for communication.

7 Conclusions and Future Work

This thesis introduces methods towards supporting mobility-awareness for Fog computing. Specifically, the proposed methods aim to address the Fog service discovery and the adaptive work distribution in Fog. Moreover, this thesis also includes a case study that composes Fog computing with Mobile Mesh Social Network (MMSN). In summary, this thesis has achieved the goals of the mobility-aware Fog computing research in the following contexts.

1. The author has developed and validated an approach that enables the discovery of new Fog service connection opportunities for the device, for when it loses connection with its Fog service while the user is moving. The proposed mobile device-embedded system continuously monitors and gathers information from peers in the DHT, where mobile peers in geographical proximity assist in routing messages. The approach aims to provide a seamless handover from one Fog node to another, by automatically detecting the connectivity change, e.g., moving out of the range of one Fog node and automatically connecting to the next available Fog node. This enables any application running on the user's device to continue where it left off before the disconnect event occurred.

2. The author has developed and validated a mobility-aware extension for proactive Fog service provisioning with an extended work-stealing paradigm with worker groups, which considers heterogeneous capabilities of Fog nodes and the heterogeneous nature of incoming tasks to be distributed amongst these Fog nodes.

3. The author has performed a case study regarding Fog Social Network, an approach to enhance the reliability and cost efficiency of Mobile Mesh Social Network. Further, the approach considers that if the entire MMSN activities fully rely on Fog, it may not be adaptive to different situations due to the dynamic nature of mobile networks. Hence, the proposed approach includes a resource-aware cost-performance index (CPI) scheme, which takes into account runtime environmental context factors to dynamically make a decision about how the interaction should be formed (i.e. using mobile P2P, Fog or Cloud). The prototype experimental results show that utilizing Fog can improve the cost efficiency of MMSN. Further, the adaptive CPI scheme can optimize the decision making by choosing the best approach for the data transmission between MMSN participants.

7.1 Future Research Directions

The following research directions are being considered for future work:

Qualitative Assessment and Trust Model in Fog Service Discovery. Assessment of peers and Fog nodes in the proposed Fog service discovery approach would encourage good manners in the system and discourage the transmission of bogus data. Therefore, the aim is to apply a qualitative assessment and a trust model [CLS14] to the Fog node data to improve mobile device handover decisions between Fog nodes and enhance the Quality of Experience.

Fog Node Reservation and Pre-Scheduling. The intent is to research the pre-scheduling or reservation of Fog nodes in a given area, where a higher priority of execution and a more aggressive variant of freeing up resources (or executing fewer tasks) for the reservation would be used.

Management of Fog Social Network Activities. The aim of this research would be to integrate Fog-centric Business Process Management System (BPMS) [KS14] with FSN to improve the management of the overall FSN activities. Further, the integrated system would interact with the IoT devices in the surrounding area in order to improve the context-awareness of the FSN.

References

[AW05] Ian F Akyildiz and Xudong Wang. A survey on wireless mesh networks. *IEEE Communications magazine*, 43(9):S23–S30, 2005.

[BBC16] BBC NEWS. India blocks zuckerberg's free net app, February 2016. [Last access] 15 Sep. 2016.

[BBG10] Rajkumar Buyya, James Broberg, and Andrzej M Goscinski. *Cloud computing: Principles and paradigms*, volume 87. John Wiley & Sons, 2010.

[BG14] Andrew Banks and Rahul Gupta. MQTT Version 3.1.1. *OASIS standard*, 2014.

[BL99] Robert D. Blumofe and Charles E. Leiserson. Scheduling multithreaded computations by work stealing. *J. ACM*, 46(5):720–748, September 1999.

[BMZA12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, New York, NY, USA, 2012. ACM.

[CBC+10] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[CLH14] Wei Cai, Victor CM Leung, and Long Hu. A cloudlet-assisted multiplayer cloud gaming system. *Mobile Networks and Applications*, 19(2):144–152, 2014.

[CLS14] Chii Chang, Sea Ling, and Satish Srirama. Trustworthy service discovery for mobile social network in proximity. In *PerCOM '14 Workshops*, pages 478–483. IEEE, 2014.

[com16] comScore. comScore Reports January 2016 U.S. Smartphone Subscriber Market Share, March 2016. [Last access] 14 Sep. 2016. [Online]. Available: `http://www.comscore.com/Insights/Rankings/comScore-Reports-January-2016-US-Smartphone-Subscriber-Market-Share`.

[CPS17] A. Ceselli, M. Premoli, and S. Secci. Mobile edge cloud network design optimization. *IEEE/ACM Transactions on Networking*, PP(99):1–14, 2017.

[CSL14] Chii Chang, Satish Narayana Srirama, and Sea Ling. Towards an adaptive mediation framework for mobile social network in proximity. *Pervasive and Mobile Computing*, 12:179–196, 2014.

[CSL15] Chii Chang, Satish Narayana Srirama, and Sea Ling. Mobile social network in proximity: taxonomy, approaches and open challenges. *International Journal of Pervasive Computing and Communications*, 11(1):77–101, 2015.

[DDB16] Soumya Kanti Datta, Rui Pedro Ferreira Da Costa, and Christian Bonnet. Resource discovery in Internet of Things: Current trends and future standardization aspects. In *IEEE World Forum on Internet of Things*, pages 542–547, 2016.

[FLR13] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. *Honeybee: A Programming Framework for Mobile Crowd Computing*, pages 224–236. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[FLR16] N. Fernando, S. W. Loke, and W. Rahayu. Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2016.

[GB07] Stuart A Golden and Steve S Bateman. Sensor measurements for wi-fi location with emphasis on time-of-arrival ranging. *IEEE Transactions on Mobile Computing*, 6(10):1185–1198, 2007.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.

[GLL+16] P. Guo, B. Lin, X. Li, R. He, and S. Li. Optimal deployment and dimensioning of fog computing supported vehicular network. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 2058–2062, Aug 2016.

[HCL10] Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing Services: Social Networks and Beyond*, MCS '10, pages 6:1–6:5, New York, NY, USA, 2010. ACM.

[HLR+13] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second*

ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC '13, pages 15–20, New York, NY, USA, 2013. ACM.

[Ker09] Ari Keränen. The ONE Simulator for DTN Protocol Evaluation. *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*, page 55, 2009.

[KFA+10] Nicolas Kourtellis, Joshua Finnis, Paul Anderson, Jeremy Blackburn, Cristian Borcea, and Adriana Iamnitchi. Prometheus: User-controlled p2p social data management for socially-aware applications. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, pages 212–231. Springer-Verlag, 2010.

[Kha15] Simon Khalaf. Seven years into the mobile revolution: Content is king... again, August 2015. [Last access] 14 sep. 2016.

[KPKB12] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal. *Cuckoo: A Computation Offloading Framework for Smartphones*, pages 59–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[KS14] Anil Kalbag and Gerald Silverman. Enriching Business Processes through Internet of Everything. Technical report, Cisco, 2014.

[LH05] Chia-Feng Li and Ren-Hung Hwang. A location-aware P2P information sharing system in Bluetooth-based mobile ad hoc network. In *International Conference on Wireless Networks, Communications and Mobile Computing*, volume 2, pages 1011–1016, 2005.

[LNA+15] Seng W. Loke, Keegan Napier, Abdulaziz Alali, Niroshinie Fernando, and Wenny Rahayu. Mobile computations with surrounding devices: Proximity sensing and multilayered work stealing. *ACM Trans. Embed. Comput. Syst.*, 14(2):22:1–22:25, February 2015.

[LS16] Y. Lin and H. Shen. Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of experience. *IEEE Transactions on Parallel and Distributed Systems*, PP(99):1–1, 2016.

[Mar09] Eugene E Marinelli. Hyrax : Cloud Computing on Mobile Devices using MapReduce. *Science*, 0389(September):1–123, 2009.

[MJ09] Yaozhou Ma and Abbas Jamalipour. An epidemic P2P content search mechanism for intermittently connected mobile ad hoc networks. In *GLOBECOM - IEEE Global Telecommunications Conference*, 2009.

[MM02] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer-Verlag, 2002.

[PKFS09] Konstantin Pussep, Sebastian Kaune, Jonas Flick, and Ralf Steinmetz. A peer-to-peer recommender system with privacy constraints. In *Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09. International Conference on*, pages 409–414. IEEE, 2009.

[POL+09] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. Mobiclique: middleware for mobile social networking. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 49–54. ACM, 2009.

[RFH+00] S Ratnasamy, P Francis, M Handley, R Karp, and S Shenker. A Scalable Content Addressable Network. *Proc of ACM SIGCOMM*, TR-00-010:161–172, 2000.

[RMHS17] Mohammad Saiedur Rahaman, Yi Mei, Margaret Hamilton, and Flora D. Salim. Capra: A contour-based accessible path routing algorithm. *Information Sciences*, 385–386:157 – 173, 2017.

[RP14] A. Ravi and S. K. Peddoju. Mobility managed energy efficient android mobile devices using cloudlet. In *Students' Technology Symposium (TechSym), 2014 IEEE*, pages 402–407, Feb 2014.

[RS04] Venugopalan Ramasubramanian and Emin Gun Sirer. Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays. *System*, 1(1):8, 2004.

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

[SCD15] H. Shi, N. Chen, and R. Deters. Combining mobile and fog computing: Using coap to link mobile device clouds with fog computing. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 564–571, Dec 2015.

[SLAZ12] Cong Shi, Vasileios Lakafosis, Mostafa H. Ammar, and Ellen W. Zegura. Serendipity: Enabling remote computing among intermittently connected

mobile devices. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '12, pages 145–154, New York, NY, USA, 2012. ACM.

[SLZL15] J. Su, F. Lin, X. Zhou, and X. Lu. Steiner tree based optimal resource caching scheme in fog computing. *China Communications*, 12(8):161–168, August 2015.

[SMF+12] Tolga Soyata, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000059–000066. IEEE, 2012.

[Smi15] Alex Smith. Mobile mania! australians spend on average more than an hour a day on their smartphones, October 2015. [Last access] 14 Sep. 2016. [Online]. Available: `http://www.nielsen.com/au/en/insights/news/2015/mobile-mania-australians-spend-on-average-more-than-an-hour-a-day-on-their-smartphones.html`.

[THTN14] Tram Truong-Huu, Chen-Khong Tham, and Dusit Niyato. To offload or to wait: An opportunistic offloading algorithm for parallel tasks in a mobile cloud. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 182–189. IEEE, 2014.

[TV04] Jivodar B. Tchakarov and Nitin H. Vaidya. Efficient content location in wireless ad hoc networks. In *Proceedings - 2004 IEEE International Conference on Mobile Data Management*, pages 74–85, 2004.

[WKC+13] Xiaofei Wang, Ted Kwon, Yanghee Choi, Haiyang Wang, and Jiangchuan Liu. Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users. *IEEE Wireless Communications*, 20(3):72–79, 2013.

[WW14] Jie Wu and Yunsheng Wang. *Opportunistic Mobile Social Networks*. CRC Press, 2014.

[XYCD13] YuanJian Xing, Zhi Yang, Chi Chen, and YaFei Dai. Beehive: low-cost content subscription service using cloudlets. *Science China Information Sciences*, 56(7):1–16, 2013.

[YCGN16] Abdullah Yousafzai, Victor Chang, Abdullah Gani, and Rafidah Md. Noor. Directory-based incentive management services for ad-hoc mobile clouds.

International Journal of Information Management, 36(6, Part A):900 – 906, 2016.

[ZLFZ15] Konglin Zhu, Wenzhong Li, Xiaoming Fu, and Lin Zhang. Data routing strategies in opportunistic mobile social networks: Taxonomy and open challenges. *Computer Networks*, 93:183–198, 2015.

[ZNW15] Y. Zhang, D. Niyato, and P. Wang. Offloading in mobile cloudlet systems with intermittent connectivity. *IEEE Transactions on Mobile Computing*, 14(12):2516–2529, Dec 2015.

[ZSM+16] C. Zhang, Y. Sun, Y. Mo, Y. Zhang, and S. Bu. Social-aware content downloading for fog radio access networks supported device-to-device communications. In *2016 IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*, pages 1–4, Oct 2016.

Appendix

I. Acronyms

AAL	Ambient Assisted Living
API	Application Programming Interface
BPMS	Business Process Management System
CPI	Cost-Performance Index
CPU	Central Processing Unit
CSNS	Centralized Social Network Services
DDC	Distant Data Center
DHT	Distributed Hash Table
DTN	Delay-Tolerant Network
FSN	Fog Social Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
ID	Identification
IoT	Internet of Things
JSON	JavaScript Object Notation
LTE	Long-Term Evolution
MASN	Mobile Ad Hoc Social Network
MMSN	Mobile Mesh Social Network
MP2P	Mobile Peer-to-Peer
MQTT	Message Queue Telemetry Transport
MSN	Mobile Social Network

MSNP	Mobile Social Network in Proximity
OCF	Open Connectivity Foundation
OppMSN	Opportunistic Mobile Social Network
OSNS	Online Social Network Service
P2P	Peer-to-Peer
QoE	Quality of Experience
RAM	Random Access Memory
REST	Representational State Transfer
SSID	Service Set Identifier
TTL	Time-To-Live
URI	Uniform Resource Identifier
VM	Virtual Machine
WAR	Web Application Resource
WMN	Wireless Mesh Network

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Sander Soo**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Towards Proactive Mobility-Aware Fog Computing

   supervised by Chii Chang and Satish Narayana Srirama

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 16.05.2017