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)

Proactive Service Discovery in Fog Computing using Mobile Ad Hoc Social Network in Proximity

Sander Soo[1], Chii Chang[2], Satish Narayana Srirama[3]

Mobile and Cloud Computing Laboratory, Institute of Computer Science, University of Tartu, Estonia
[1]sander.soo@ut.ee, [2]chii.chang@acm.org, [3]srirama@ut.ee

***Abstract*—Emerging Internet of Things system utilizes heterogeneous proximity-based ubiquitous resources to provide various real-time multimedia services to mobile application users. In general, such a system relies on distant Cloud services to perform all the data processing tasks, which results in explicit latency. Consequently, Fog computing, which utilizes the proximal computational and networking resources, has arisen. However, utilizing Fog for real-time mobile applications faces the new challenge of ensuring the seamless accessibility of Fog services on the move. This paper proposes a framework for proactive Fog service discovery and process migration using Mobile Ad hoc Social Network in proximity. The proposed framework enables Fog-assisted ubiquitous multimedia service provisioning in proximity without distant Cloud services. A proof-of-concept prototype has been implemented and tested on real devices. Additionally, the proposed Fog service discovery and process migration algorithm have been tested on the ONE simulator.**

***Index Terms*—Fog computing; service discovery; Internet of Things; Mobile Ad-Hoc Social Network in Proximity**

I. Introduction

The advent of Cloud computing has enabled a new paradigm of ubiquitous and on-demand access to computing resources [1]. However, as the new wave of Internet of Things (IoT) 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. Compared to the distant Cloud computing model, Fog computing model proposes utilizing the computational resources in the close proximity of the requesters. Such resource can be the computers co-located with the cellular base stations, computers co-located with Wi-Fi access points, or they can be the wireless network routers themselves. These computers can host Virtual Machines (VM) to support the similar mechanisms as a Cloud does. The basic idea of Fog is to bring the Cloud to the ground and to be as close to the users as possible.

Different to the classic Cloud services, which commonly are centralized systems and are stably accessible as long as the Internet connection is available, the accessibility of Fog computing service providers (or Fog nodes) 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., [2], [3]) have proposed solutions to the similar problem. However, the assumptions they made are not applicable in the Fog computing model.

In this paper, we propose a framework for Fog node discovery and process migration in a fully decentralized manner. The proposed framework does not require a distant Cloud-based management service, neither the need for the Internet connection. The underlying strategy is to utilize Mobile Ad hoc Social Network (MASN) in proximity [4] for the information sharing. MASN represents a virtual social network environment in which the mobile applications behave as the delegates of their users to perform social activities autonomously with one another when they meet opportunistically within the wireless network range. By utilizing MASN, while the mobile application user is moving, the application is continuously gathering information about nearby Fog nodes and uses this information to maintain the accessibility of Fog.

The proposed solution allows the trusted *private social network* [5] users who are in close relationships such as colleagues, close friends or family members to share the information of Fog service provider nodes. Hence, the proposed solution does not involve the general security concerns in public social networks. Further, the information sharing approach also decouples the Fog nodes from the information regarding them, enabling the fact that information about the Fog node can be relayed further than what the nodes capabilities would perhaps allow, and also the future opportunity for extensibility with user provided additional information, e.g., quality assessment and feedback for Fog nodes that could not be simply dropped in the case of negative assessment by some users.

This paper is organized as follows. Section II provides an overview and comparison of related works. Section III describes the proposed system design and architecture. Section IV provides the analysis of the experimental results. This paper is concluded in Section V together with future work.

II. Related Works

P2P-Bluetree [3] describes a location aware P2P information sharing system over Bluetooth, by organizing peers into a tree-based overlay network. Datta et al. [6] proposed a resource discovery framework for smart and legacy devices. In their

978-1-5090-5880-8/16 $31.00 © 2016 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.126

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 resources 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 [2] 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 [7] 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 following limitations: (1) The assumption of a tree topology may not be the case in real life because there may be links between peers in the network that collectively form the routing cycles. (2) 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, our approach takes the physical location into account, and does not need to rely on a special logical address based on the nodes position in the tree. Further, our work is designed for a fully decentralized P2P environment in which the user may not have access to 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 to the inherent assumption of a high density of nodes, which may not hold in real life. Instead, we have employed a Distributed Hash Table (DHT)-based approach for the Fog node information (location, SSID of co-located Wi-Fi router, etc.).

Although our approach 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.

III. System Design

A. Overview of Environment

Figure 1 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 approach to utilize Fog to reduce the burden of the mobile device and also enhance the overall computational and networking performance.

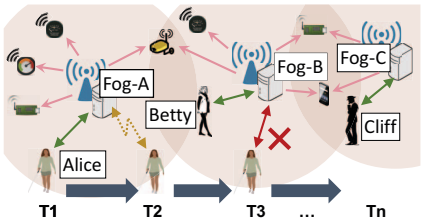

Figure 1: Fog node discovery scenarios.

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 1), while Alice is moving, the connectivity between her mobile device and Fog-A is getting weaker (T2 in Figure 1). 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. 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.

B. Overview of the Proposed Fog Discovery Scheme

In our 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, Alices 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 cannot be reached directly by Alices device. Alices 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 Alices 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. First, it sends a request message to the closest MASN peers. These MASN peers may reply immediately with the information of the available Fog 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.

As soon as an MASN peer retrieves the corresponding information, it routes the information through any intermediary peers back to Alices device. Afterward, 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., Alices device) with the public key provided by Alices device. 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. The request for the public key of another user is executed in the same manner as the one for the Fog node information.

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. 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. 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.

2) Routing: Each MASN peer holds entries in its routing table that contain 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 [3]. 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 holds 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.

Considering that there is a chance that the original route for delivering the response to the requester is no longer available (e.g. intermediary peers disconnected), 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 [8].

There is also the possibility that the last accessible node in the path is not the final destination. Since the routing device can only verify that the path exists up to two hops from the current location, then in order to improve the chances that a message will be successfully delivered to the final destination, 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 we assume 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, e.g., if the node in question was turned off or switched to fly-mode. The usage of the TTL value for the response is activated when the current transmitting node determines that the previous path no longer exists.

Algorithm 1 Core router logic

```
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
10:            if pathExists(m.viaPath) then
11:                nodes = limit( KN )
12:                updateTtl(m)
13:                send(m, nodes)
14:            else
15:                setTtl(m)
16:                send(m, KN)
17:            end if
18:        end if
19:    end if
20: end function
```

Algorithm 1 illustrates the core of the message routing algorithm. The message to be routed is denoted by the variable named “m”. The algorithm begins with a check whether the messages 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 (KN; Line 5). Then the algorithm determines if the message is a request (Line 6) and if so, the router decrements the TTL value of the message, if one exists, and 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, then, it defines a subset of KN based on the predefined limit (Line 11), decrements the

TTL value of the message, if it exists (Line 12), and forwards the message to the peers (Line 13).

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 15) and forward the message to KN (Line 16).

3) Messages: The nodes can intercommunicate in the system by passing around messages in a common data-interchange format, e.g. JSON.

All the messages, regardless of their type, contain common information such as: the message UUID (128bit Universally Unique Identifier) to differentiate two messages; routing information, which denotes who are the sender and receiver of the message; and the possible intermediary nodes sender and receiver IDs that can be used in the routing. If the message is a response to a previously sent message, then this information is also present in the message.

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 request all the MASN peers of another peer. This message can be used e.g. 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 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; and the specific users 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 users 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. 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 something goes wrong or 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.

C. DHT design and lookup

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

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, our framework assumes by default that the location where the user creates the information of the Fog node is roughly the same as the Fog nodes 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.

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 [9] 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, our 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 requesters 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 [10]. We are considering to investigate this approach in the future.

The process to look up any information from the DHT is similar. 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 then simply has to interpret the result(s) accordingly.

IV. Evaluation

The evaluation consists of two parts. The first part validates the proposed routing algorithm, which has been tested by using

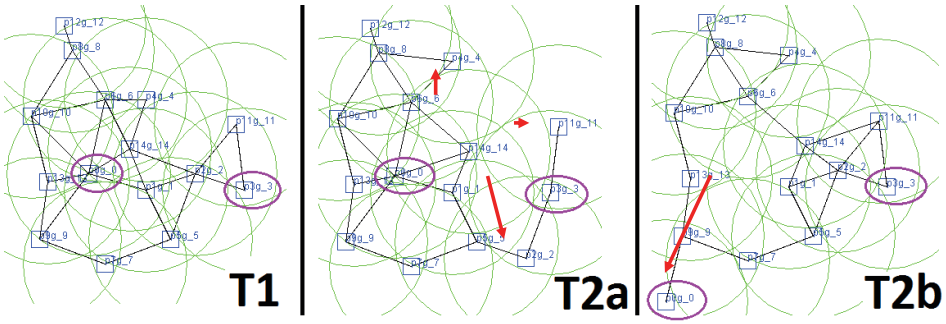

Figure 2: ONE simulator node movement scenario at timestamps T1, T2a, T2b

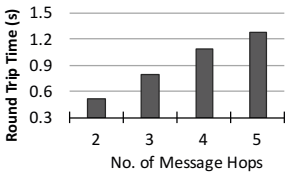

Figure 3: Round trip time per number of message hops.

an HP Elitebook 840 (Intel i5-4200U@1.60GHz, 12GB RAM) with the ONE simulator (v1.6) [11]. 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*.

A. Evaluation scenario

The use case scenario of the evaluation follows three timestamped steps: T1, T2a and T2b (Figure 2). 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, known to 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, but as soon as the message is forwarded to node p2g_2, it sees 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.

B. Experimental results

1) Message Routing: Figure 3 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 simulators radio link is abstracted to a communication range and bit-rate (no signal attenuation, etc.), these results may differ a bit in the real world.

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 4 attempts to show the overhead induced by the node movement changes illustrated in Figure 2. 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, explained in more detail in the previous section.

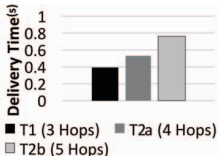

Figure 4: Node movement effect on message routing.

Table I: Effect of density on message routing.

	20 nodes	25 nodes	100 nodes	200 nodes
Latency	0.413	0.39	0.378	0.384
Hop Count	3	3	3	3
Round Trip Time (s)	0.826	0.779	0.755	0.768

Table I illustrates the effect of the number of nodes in proximity to the overall efficiency of the message routing.

Since the closest nodes in proximity that are actually used is a finite subset chosen by the sender node, and even if the number of nodes in the vicinity of the user gets very high, the routing logic would still only pick a small subset. Therefore it limits the number of messages sent. The messages Time-To-Live is also a limiting factor. Based on these simulations, we

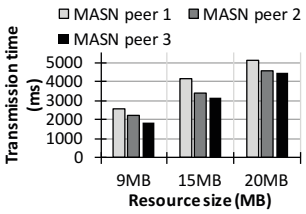

Figure 5: Web Application Resource transmission time from MASN peers to Fog node 1.

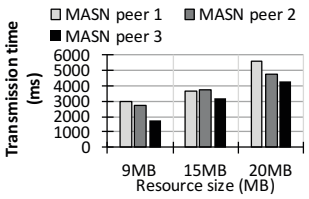

Figure 6: Web Application Resource transmission time from MASN peers to Fog node 2.

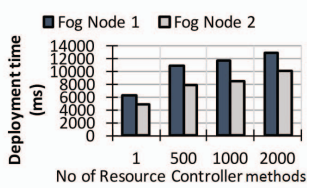

Figure 7: Web Application Resource deployment time.

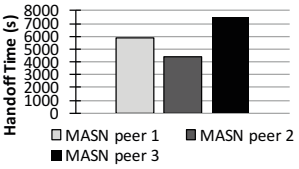

Figure 8: Smartphone experimental Fog handoff time.

could see 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.

2) Task Migration Performance: Figure 5 and Figure 6 illustrate the time taken by a specific mobile node to send the Web Application Resource (WAR) file to the Fog node for deployment to the application container (e.g., Apache Tomcat). The maximum time measured was approximately 5.5 seconds when data was being sent from MASN peer 1 to Fog node 2.

Figure 7 illustrates the measurement of minimal standard controller endpoint methods for a RESTful web service written in Spring Framework (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/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 8 illustrates results of a more Fog-like setting, showing how long on average it took for the mobile node, running our proof of concept Android application, to be handed off from one Fog node to another. 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, averaging at about 7.5 seconds for the handoff. This was most likely influenced by the Wi-Fi adapter hardware. It also indicates that the result shown in Figure 2 (by the ONE simulator) can be quite different in the real world.

V. Conclusion and Future Work

In this paper, we introduced a framework for proactive Fog service discovery using MASN that enhances the need of seamless task migration between different Fog service providers while the user is moving. The proposed mobile device-embedded system is continuously monitoring and gathering information from MASN peers in the DHT. Hence, it is able to discover new Fog connection opportunities for when it loses the access with its Fog service while the user is moving. The message persistence is handled using a DHT approach, where the MASN peers in geographical proximity are helping to route the 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 users device to continue where it left off before the disconnect event occurred.

In the future, we plan to address the following aspects: (1) The detection of too many peers in the vicinity to reduce the number of peer messages sent between nodes. (2) We plan to apply a qualitative assessment and a trust model [12] to the Fog node data to enhance the Quality of Experience. (3) Message ownership responsibility and Fog node feedback extensions as mentioned in a previous section are also of interest.

Acknowledgment

This research is supported by Estonian Science Foundation grant PUT360 and Study IT in Estonia.

References

[1] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud computing: Principles and paradigms*. John Wiley & Sons, 2010, vol. 87.
[2] J. B. Tchakarov and N. H. Vaidya, "Efficient content location in wireless ad hoc networks," in *MDM'04*, 2004, pp. 74–85.
[3] C.-F. Li and R.-H. Hwang, "A location-aware P2P information sharing system in Bluetooth-based mobile ad hoc network," in *WIRELESSCOM'05*, vol. 2, 2005, pp. 1011–1016.
[4] C. Chang, S. N. Srirama, and S. Ling, "An adaptive mediation framework for mobile p2p social content sharing," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 374–388.
[5] K. Pussep and et al., "A peer-to-peer recommender system with privacy constraints," in *CISIS'09*. IEEE, 2009, pp. 409–414.
[6] S. K. Datta, R. P. F. Da Costa, and C. Bonnet, "Resource discovery in Internet of Things: Current trends and future standardization aspects," in *IEEE World Forum on Internet of Things*, 2016, pp. 542–547.
[7] Y. Ma and A. Jamalipour, "An epidemic P2P content search mechanism for intermittently connected mobile ad hoc networks," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2009.
[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," *Proc of ACM SIGCOMM*, vol. TR-00-010, pp. 161–172, 2000.
[9] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *1st Int. Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.
[10] V. Ramasubramanian and E. G. Sirer, "Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays," *System*, vol. 1, no. 1, p. 8, 2004.
[11] A. Keränen, "The ONE Simulator for DTN Protocol Evaluation," *2nd Int. ICST Conference on Simulation Tools and Techniques*, p. 55, 2009.
[12] C. Chang, S. Ling, and S. Srirama, "Trustworthy service discovery for mobile social network in proximity," in *PerCOM '14 Workshops*. IEEE, 2014, pp. 478–483.