SCALABLE DEVICE MOBILITY

by

Shabir Ahmad

2011-NUST-MS PhD-CSE(E)-32

MS-11(CSE)

Submitted to Department of Computer Engineering

in fulfilment of the requirements for the degree of

Masters of Science

in

Computer Software Engineering

Thesis Supervisor

Dr. Shoab Ahmed Khan

College of Electrical and Mechanical Engineering

National University of Science and Technology

December 2013

This page is intentionally left blank

ACKNOWLEDGEMENTS

"*Those who do not thank people, they do not thank Allah*." *(Al-Tirmidhi 1878).*

There is an old saying ‘It takes a village to raise a child’ and I am not an exception. It is with the grace of Almighty that I was led into the company of the following people, whose generosity, enthusiasm, and good shepherding sustained me in producing this work: My friends always backed me up very strongly. Therefore, I thank every single one of them for their support.

To my Parents, Advisors and colleagues.

ABSTRACT

The continuously increasing Internet coverage and its availability gave rise to an issue that once has been considered not important to take into consideration. Today a number of applications use the Internet to deliver time critical messages. The usage of wireless Access Points takes considerable percentage to connect mobile devices to the Internet provider. However, these relatively cheaper Internet Access Points have their own disadvantages as compare to the GSM and ADSL. The access points cover a very limited area and thus in order to cover wide area multiple access points needs to be installed. In other words, as the user moves he is supposed to switch between access points. Nevertheless, the basic problems in such cases are packet loss during handover. In today's technological advancements these issues, though very small, are no more insignificant but needs to be handled properly. So protocols such as MobileIP , LISP, HOST have been proposed. Furthermore, in this thesis a mechanism to reduce such packet losses have been studied and proposed in relation to the SensibleThings Internet-of-things platform. A work around solution known as Mobile DCXP has been proposed and implemented and comparison with the existing system have been carried out. In addition, a generic solution has been discussed in detail and compared with the Mobile DCXP. However, the implementation of the generic solution has been deferred to the future. Finally, the idea of Mobile DCXP has been illustrated with proof-of-concept apps and implementation of a simple Android Application known as IChat has been done. The IChat is a simple app that helps to find out the packet losses during handover and carry out comparison.

Keywords – *Internet of things, MediaSense, SensibleThings, LISP, MobileIP, DCXP, HIP*

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Illustration
IoT	Internet of things
IP	Internet Protol
HMIPv6	Hierarchal Mobile IPv6
PMIPv6	Proxy Mobile IPv6
DCXP	Distributed context exchange protocol
IEEE	Institute of Electrical and Electronic Engineers
MDP	Mobile DCXP Proxy
MD	Mobile DXCP
POC	Proof of concept
HIP	Host Identity protocol
LISP	Location identifier separation protocol
UCI	Universal context identifier
CUA	Context User agent
W3C	World Wide Web Consortium
XML	Extensible Markup Language
IETF	Internet Engineering Task Force

Chapter 1
INTRODUCTION

Human beings possess amazingly voracious appetite for a better and easier life. In the end of the last century as well as in the previous decade we have witnessed a huge move with regard to the innovations in the area of Information Technology, electronics and electro-mechanical technologies among others. In the area of information technology, the Internet is one of the revolutionary kinds of innovation though the Internet came into its currently socially valuable technology through time. Moreover, in the current advancement of human beings' imagination and available supporting technologies, there is a buzzword known as Internet-of-things all around the Internet and in the research labs. In fact some companies have tried to take advantage by running into the business right from the time the technology is more of conceptual than actually feasible and profitable when observed from the business point of view.

It has been a few years since the Internet-of-things overlay known as the MediaSense has been proposed from Mid Sweden University in cooperation with European Union and other parties. Moreover, a number of projects related to the MediaSense have been carried out. The MediaSense project has produced components for the accumulation of context information from sensors and wireless sensor networks. The context information originates from numerous different sources such as sensors attached to mobile phones or home gateways. These devices are communicating via IP addresses. Using mobile devices with changing Internet access, these IP addresses may change without notice during a session. This becomes an issue when both the producer and the consumer change their point of attachment simultaneously.

In this thesis the concern is how to deal with context-aware applications in a situation where a user switches between networks. In addition, the thesis tries to find a place for the developments, improvements and proposals within the SensibleThings platform.

1.1 Background and Motivation

Applications that can change their behaviour based on the context of users are known as context-aware applications. These applications have a good market penetration with the introduction of smart phones and others similar related devices, which come with a multitude of embedded sensors and built in actuators. This thesis is part of an effort to come up with a next generation Internet-of-things architecture and its supporting protocols.

It has been observed that recently there are developments with regard to the utilization of the context-aware applications for example in an area where tourists go frequently, mobile workers, and adverts. The use of the technology within the modern society is in its infant stage. Moreover, common architectures and design principles is being developed and studied by different parties. This thesis has an aim to work on the implementation of the context-awareness with a focus on the mechanisms on how to deal with a situation where such an apps running on different devices manage switching a network from one to the other smoothly without affecting its usage.

1.2 Overall aim

In this thesis the aim is to survey current mobility solutions and identify their shortcomings based on future requirements on mobility as exemplified from a scenario where nodes change their point of attachment arbitrarily in a distributed environment. Additionally, part of the overall aim is to propose a solution based on the SensibleThings architecture. Therefore, the thesis finds out a solution on how to incorporate the mobility solutions in the existing SensibleThings Architecture. Finally, part of the aim has been to implement the proposed solution as an extension to the modules of the SensibleThings platform.

1.3 Concrete and verifiable goals

The goal of the project is to propose and implement an extension to the SensibleThings architecture that could accommodate mobile nodes that disconnect and connect in different networks.

The concrete and verifiable goals of the thesis have been divided into the following three goals:

GOAL ONE: Study the existing mobility solutions and propose scalable mobility solution for the SensibleThings platform.

GOAL TWO: Implement the proposed mobility solution as an extension for the SensibleThings platform.

GOAL THREE: Evaluate the performance of the proposed mobility solution.

1.4 Outline:

Chapter one introduces the backgrounds of wireless access point handover issues. In this chapter, the problem, the aim, the concrete and verifiable goals and the scope of the project are briefly presented.

Chapter two discusses about related works. In this section existing mechanisms to handle packet loss due to wireless access point handover have been discussed. Here protocols, solutions and optimization task have thoroughly covered.

Chapter three presents the overall methodology followed to arrive at the right result and conclusion. In this section the task have been divided into smaller goal which could link up together and come to the intended results.

Chapter four discusses about the design of implementations carried out. Moreover, it also explains in detail the proposed workaround solution (or the add-in layer extension) and discusses some other possible solutions. In here proof-of-concepts, diagrams and tables have been used to illustrate the concepts.

Chapter five presents the implementation of sections. In this chapter implementation of Mobile DCXP, IChat and proof-of-concept app have been presented in detail.

Chapter Six presents the results sections. The results sections present the evaluation and comparison of the proposed Mobile DCXP with the existing Mobile DCXP proxy. Here graphs, tables and diagrams have been used for illustration purposes

Chapter Seven discusses the conclusion from the thesis work. In this chapter, future work and contribution of the thesis has been discussed.

Chapter 2

LITERATURE REVIEW

The need for optimization and efficiency comes not at the beginning but at the later stages of development or any progress due to the very reason related to the nature of human learning instinct. The usage of radio technology for data communication and eventually as a networking device was such undeniable success. However, in the beginning hardly anybody could imagine what might possibly come as research ambition related to the wireless-access-point. Thanks to the kindness of time, the question we are asking today is not how to connect devices wirelessly rather it is about how to avoid packet loss during roaming and thus perform smooth handover when devices switch from one access point to another.

Therefore, this chapter presents the theoretical backgrounds of the protocols, techniques and mechanisms proposed and being utilized currently in order to perform smooth handover in wireless-access-point. In addition, literature overview on the backgrounds of the concepts of the SensibleThings and Dissemination Context Exchange Protocol has been presented.

2.1 Core Issues and Problems

Currently, the usage of wireless-access-point is getting more and more prevalent as well as valuable in the community. This is totally attributed to human deep rooted seek for freedom and freedom of movement in particular, which is for granted in this case. Of course, Wireless-access-points are not recommended for all kinds scenario where communication needs to be built. Thus, wireless-access-points have their cons and also pros1However, the concern of the thesis is not to study about these technologies but to delve deep and work on the efficient handover during switching from one wireless access point to another access point.

In the existing IP address utilization, handover is carried out successfully however the session will not be kept. In other words, smooth handover where there is no packet loss can't be carried out. The most pressing problem in the existing packet based networking protocols are Loss of trust , surge of unwanted traffic, choking routing systems poor support of mobility and multi-homing , lack of privacy and accountability2. In order to carry out smooth handover in wireless access-point switch a number of mechanisms have been proposed and different protocols have been ratified. In this section the most commonly accepted mechanisms are presented.

2.2 Proposed Solutions and Currently Utilized Mechanisms

Protocols proposed with regard to the issue of smooth handover in the wireless access-point serve different purposes however all of solutions have one cause and that is to carry out smooth handover or help to carry out smooth handover. Protocols for example HIP,LISP, Mobile IPv6 proposes their own means to tackle the issue. However, Hierarchical MIPv6 , PMIPv6 have a focus on the optimization of the existing Mobile IPv6.

2.2.1 Host Identity Protocol

The Host Identity Protocol (HIP) is an internetworking architecture and an associated set of protocols, developed at the IETF since 1999 and reaching their first stable version 2007 3. HIP is an additional name space besides the two name spaces (IP and DNS) used in the Internet architecture. HIP is a cryptographic in its nature ; it is a public key of asymmetric public key pair**Error! Reference source not found.** HIP integrates IP-layer mobility, security, multi-homing and multi-access, NAT traversal and IPv4/v6 interoperability. Technically, the basic idea behind HIP is to add a new name space to the TCP/IP stack. In HIP layer Hosts given identifications, Host identifications. Each host identity represents a unique host.

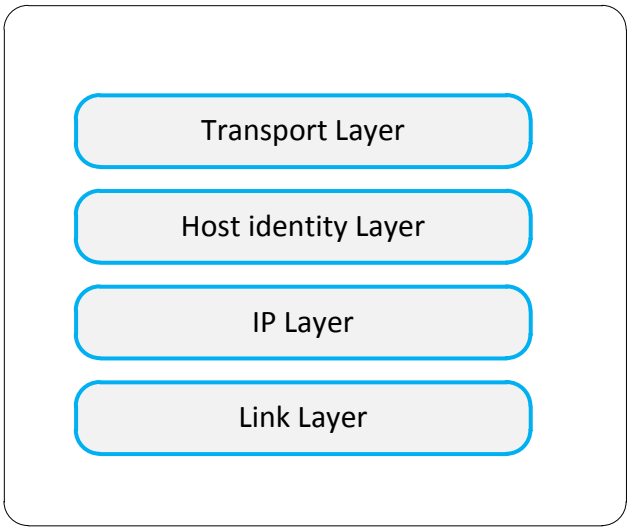


Figure 1 Location of HIP layer in the TCP/IP stack.

Benefits of HIP4

- **Non-mutable:** The address sent is the address received.
- **Non-mobile:** The address does not change during the course of an "association".
- **Reversible**: A return header can always be formed by reversing the source and destination addresses.
- **Omniscient**: Each host knows what address a partner host can use to send packets to it.

Mobility

When a host moves to another network notification will be send to its peers. The notification is done through HIP update packet containing LOCATOR parameter. Acknowledgment will be send back to the moving host. The update is retransmitted again in order to have more reliable communication incase of packet loss.

2.2.2 LISP

LISP (aka Locator Identifier Separation Protocol) provides a set of functions for routers to exchange information used to map from non globally routable End Point Identifiers (EIDs) to routable Routing Locators (RLOCs)8. LISP has been proposed based on observation made from a different angle. The Basic observation is using a single address for both identifying the device and locating the device needs compromise of topology based identifier assignment and none explicitly based identifier assignment. In order to carry out efficient routing there should be topology based identifier assignment whereas in order to manage efficiently when a number of devices exist and to handle situations where devices needs renumbering it is advisable not to have explicit attachment of identifier with the topology.

Moreover, due to the fact that LISP is map-and-encap protocol, there is no need to change the host stack. See figure 2 Map-and-encap protocols appends header to the existing header.

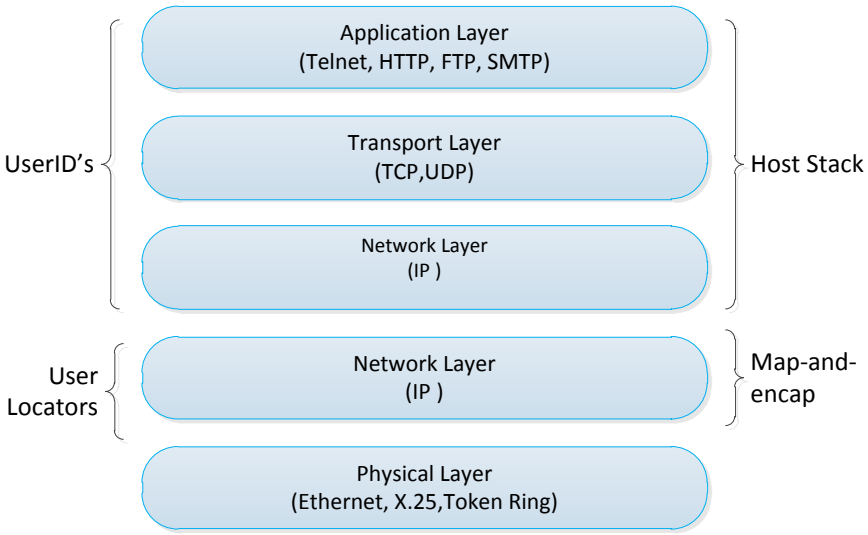


Figure 2 LISP

Routing Scalability issue has been solved by assigning two types of numbers for each device's IP address: RLOCs - Routing Locators and EIDs - Endpoint Identifiers. RLOCs are assigned topology based; RLOCs are used for data forwarding and routing in the network. However, EIDs are assigned independently of the topology; EIDs are used for numbering.

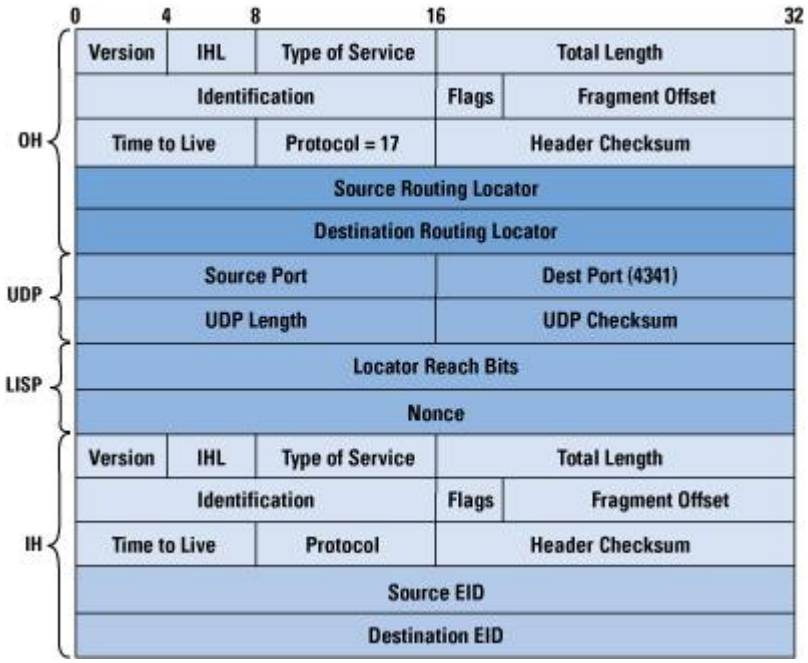


Figure 3 LISP Header Format 8

Benefits of LISP 8

- Improved routing system scalability by using topologically-aggregated RLOCs
- Provider-independence for devices numbered out of the EID space (IP portability)
- Low-OPEX multi-homing of end-sites with improved traffic engineering
- IPv6 transition functionality
- IP mobility (EIDs can move without changing - only the RLOC changes!)

2.2.3 Mobile IPv6

Mobile IPv6 targets to offer smooth hand over of mobile nodes during switching between access-points. Mobile IPv6 provides unbroken connectivity for mobile nodes when roaming between wireless access points in a different subnet in an operation known as L3 (Layer 3) handover 9 Handover might be carried out on layer two or layer three depending on the wireless access points involved. For wireless access points on the same subnet, the handover occurs on the layer 2 however two switch from one access point to the other which belongs in a different subnet the handover is carried out layer 3.

The Mobile node is identified by its two addresses in its entire process. The first address is known as home address and the other is known as care-of-address. The home address represents the IP address of the mobile node when it is attached to home network. However, when the mobile node wanders around and gets attached to another access point it will be assigned a care-of-address and this care-of-address will be registered to its home agent and correspondents. The association made between home address and care-of-address is known as binding 8. In this protocol until a binding update is sent to the home agent (see figure 4 below) the packets coming to the home address will be lost.

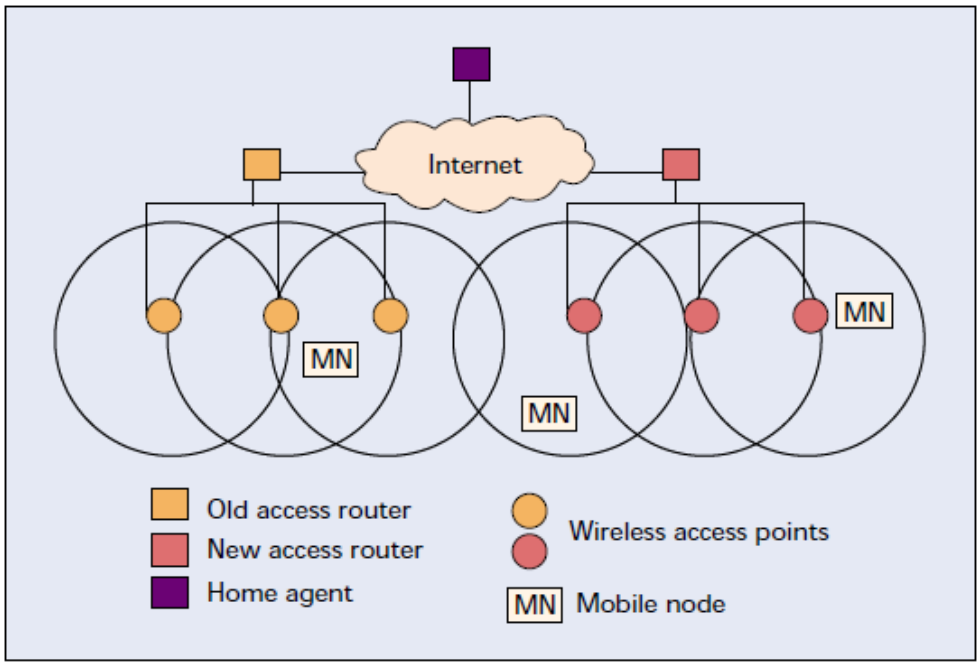


Figure 4 Mobile IPv6 Wireless Network Architecture9

Procedure

1. MN detects that it has moved to another network through a periodic advertisement coming from the router.
2. Based on the messages being advertized from the router , the Mobile node gets a new care-of-address
3. Mobile node performs duplication address detection (DAD) on its link-local address
4. Uses either Stateful or stateless address authoconfiguration
5. Mobile node performs DAD for the care –of-address.
6. Mobile node carries out bind update.

Benefits of Mobile IPv6

The major benefit of this standard is that the mobile nodes (as IPv6 nodes) change their point-of-attachment to the IPv6 Internet without changing their IP address.10

2.2.4 Hierarchical Mobile IPv6

The Hierarchical Mobile IPv6 (aka HMIPv6) follows similar concepts as in the Mobile IPv6 presented above. However, HMIPv6 introduces new functions known as Mobile Anchor Points (MAP) and minor extensions. The Mobile node might send packet to its Home agent immediately after update binding however in cases where the home agent and the Mobile nodes are far apart the home agent couldn't trace back to the Mobile node before receiving the binding update so packets will be lost. This drawback could have a significant impact on data communication where time critical handover is going on.

The introduction of the MAP provides a solution to the issues

Within the Mobile IPv6:

- The mobile node sends binding updates to the local MAP rather than the home agent (HA) (which is typically further away) and correspondent nodes (CNs)**Error! Reference source not found.**.
- Only one binding update message needs to be transmitted by the mobile node (MN) before traffic from the HA and all CNs is re-routed to its new location. This is independent of the number of CNs with which the MN is communicating12.

Benefits of Hierarchical Mobile IPv6

Handover performance improvement due to the fact that local handover is performed locally which results in faster transition time and thus less packet loss.

Reduces the mobility management signaling load on the network

2.2.5 Fast Handover

Fast Handover is not a protocol standing by its own to handle the issue of wireless access point handover however the protocol is aimed to improve the handover latency which might occur during switching of a node from one access point to another using Mobile IPv6. The handover latency is caused by movement detection, new care of address configuration and binding update. Although the handover latency for Mobile IPv6 is small, it has an impact on the normal

communication of nodes carrying out voice over IP and thus intolerable in some situations. Fast handover is concerned on the following two questions: How to allow to allow a mobile node to send packets as soon as it detects its new subnet link, and how to deliver packets to a mobile node as soon as its attachment is detected by the new access router 12.

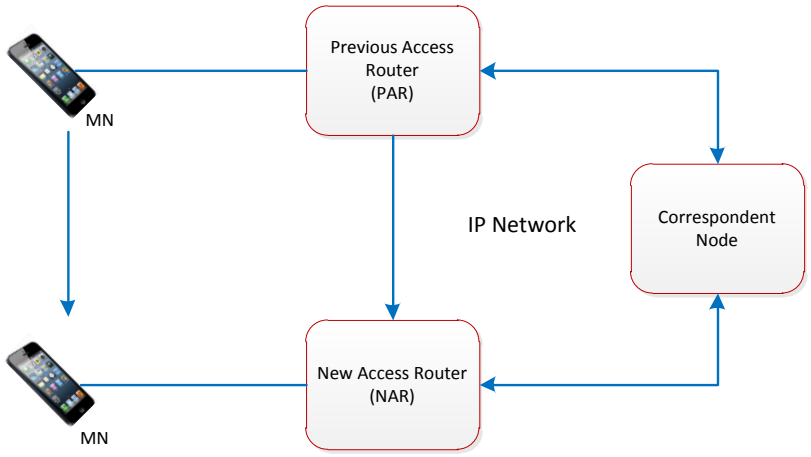


Figure 5 Handover Diagram13

Benefits of Fast Handover

The protocol reduces packet loss by combining packet tunnelling with buffering during the time the mobile node is switching between access routers20.

2.2.6 MOBIKE

MobIKE also known as IKEv2 Mobility and Multihoming Protocol, allows the IP addresses associated with IKE2 (Internet Key Exchange) and tunnel mode IPsec Security Associations to change14. IKE2 is used for performing mutual authentication, as well as establishing and maintaining IPsec Security Associations15

The main Scenario for MOBIKE is keeping the VPN user security associations without a need for re-establishing the task all over again later on.

MOBIKE also supports more complex scenarios where the VPN gateway also has several network interfaces 15.

Table 1 A Simple MOBIKE Exchange in mobile environment

INITIATOR	**RESPODER**

1) (IP_I1:500 -> IP_R1:500)

HDR, SAi1, KEi, Ni,

N(NAT_DETECTION_SOURCE_IP),

N(NAT_DETECTION_DESTINATION_IP) -->

<-- (IP_R1:500 -> IP_I1:500)

HDR, SAr1, KEr, Nr,

N(NAT_DETECTION_SOURCE_IP),

N(NAT_DETECTION_DESTINATION_IP)

2) (IP_I1:4500 -> IP_R1:4500)

HDR, SK { IDi, CERT, AUTH,

CP(CFG_REQUEST),

SAi2, TSi, TSr,

N(MOBIKE_SUPPORTED) } -->

<-- (IP_R1:4500 -> IP_I1:4500)

HDR, SK { IDr, CERT, AUTH,

CP(CFG_REPLY),

SAr2, TSi, TSr,

N(MOBIKE_SUPPORTED) }

(Initiator gets information from lower layers that its attachment point

```
and address have changed.)
```

```
3) (IP_I2:4500 -> IP_R1:4500)

      HDR, SK { N(UPDATE_SA_ADDRESSES),

                N(NAT_DETECTION_SOURCE_IP),

                N(NAT_DETECTION_DESTINATION_IP) }  -->



                              <-- (IP_R1:4500 -> IP_I2:4500)

                                   HDR, SK { N(NAT_DETECTION_SOURCE_IP),

                                        N(NAT_DETECTION_DESTINATION_IP) }


   (Responder verifies that the initiator has given it a correct IP
address.)
```

```
 4)                          <-- (IP_R1:4500 -> IP_I2:4500)

                                   HDR, SK { N(COOKIE2) }



      (IP_I2:4500 -> IP_R1:4500)

      HDR, SK { N(COOKIE2) }  -->
```

2.2.7 PMIPv6

PMIPv6 (aka Proxy Mobile IPv6) is a protocol proposed by the IETF to reduce latency during handover and thus packet loss that occurs in during handover in the MIPv6 protocol. PMIPv6 is intended for providing network-based IP mobility management support to a mobile node,

without requiring the participation of the MN in any IP mobility related signaling16. PMIPv6 offers two new functional entities the Local Mobility Anchor (LMA) and the Mobile Access Gateway (MAG). The MAG detects the mobile nodes attachment and provides IP connectivity. The LMA is an entity which assigns one or more Home Network Prefixes (HNP) to the MN and is the topological anchor to all traffic belonging to the MN.

The MN and LMA should have a local policy in place which makes sure that packets are forwarded coherently for unidirectional and bi-directional communication. The MN decides on the final IP flow mobility decisions, and then the LMA follows that decision and update its forwarding state based on the decisions made.

Benefits of Mobile PMIPv6

The delay of sending signalling to LMA is lower as compared to sending signalling to remote home agent in the case of Mobile IPv6.

Less overhead as compared to IPv6

2.3 The SensibleThings Project

The SensibleThings is a novel architecture for Internet-of-things application development. SensibleThings is a distributed architecture that enables Internet-of-things based on sensor and actuator information 17. The entire SensibleThings platform is divided into five layers as shown in figure 6

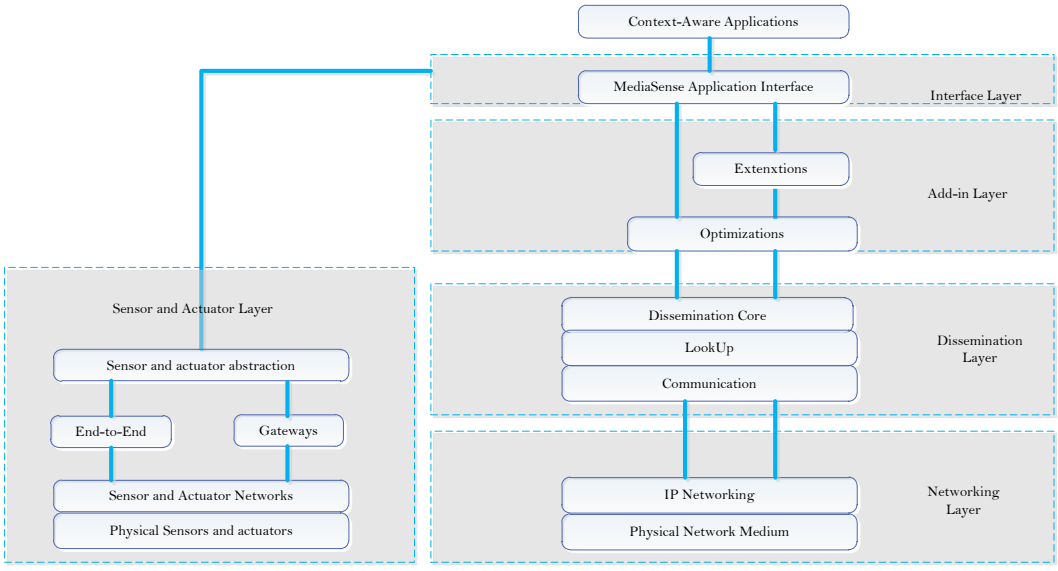


Figure 6 the SensibleThings Platform Architecture

- **The Interface Layer:**

The Interface layer is the public interface through which applications interact with the SensibleThings platform 18

- **Sensor and Actuator Layer**

The Purpose of the sensor and actuator layer is to enable a generalized method to produce information and provide it to the SensibleThings platform18.

- **Add-in Layer**

The Add-in layer is intended for developer who would like to add extensions to the platform or carryout optimization works.

- **Dissemination Layer**

The dissemination layer is involved in disseminating information among the participating entities in the Internet-of-things.

  - Dissemination Core
  - Lookup
  - Communication

- **Networking Layer**

The networking layer performs connection of IP based networking communications. The networking layer is divided into two components: IP networking and physical network medium. The SensibleThings platform is independent of a particular networking medium and it is designed to run on a heterogeneous networks.

2.4 Distributed Context Exchange Protocol – DCXP

The Distributed Context Exchange Protocol (DCXP) is a peer –to-peer protocol used within the SensibleThings framework to exchange context between users and entities 19. DCXP is a SIMPLE19-inspired protocol with five primitives (REGISTER_UCI, RESOLVE_UCI,GET,SUBSCRIBE, NOTIFY). DCXP is nothing but an XML based

application level protocol which serves reliable communication of context information among nodes that participate in the overlay network. Although the term is a bit fuzzy, the design of DCXP satisfies the real-time requirements for provisioning of context information.

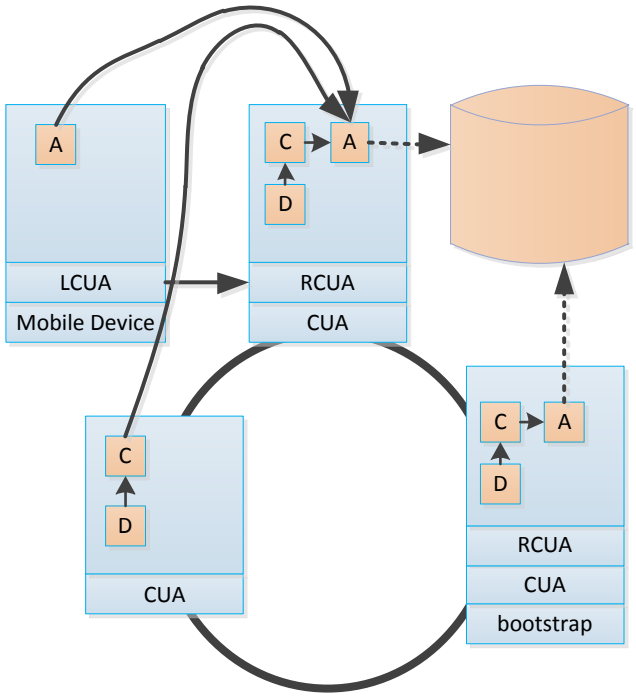


Figure 7 DCXP Architecture 19

- CUA corresponds to a node in the DHT ring
- C- a database client
- D-data miner
- A-a database Agent
- RCUA- Remote CUA
- LCUA- Limited CUA

2.5 DCXP Messages

REGISTER_UCI:

A CUA (Context User Agent) uses REGISTER to register the UCI of a CI (Context Information) with the DS.

RESOLVE_UCI:

In order to find where a CI is located, a CUA must send a RESOLVE to the CS.

GET:

Once the CUA receives the resolved location from the Context Storage, it GETs the CI from the resolved location.

SUBSCRIBE:

SUBSCRIBE enables the CUA to start a subscription

to a specified CI, only receiving new information when the CI is updated.

NOTIFY:

The source CUA provides notification about the latest information to subscribing CUAs ever time an update occurs or if asked for an immediate update with GET.

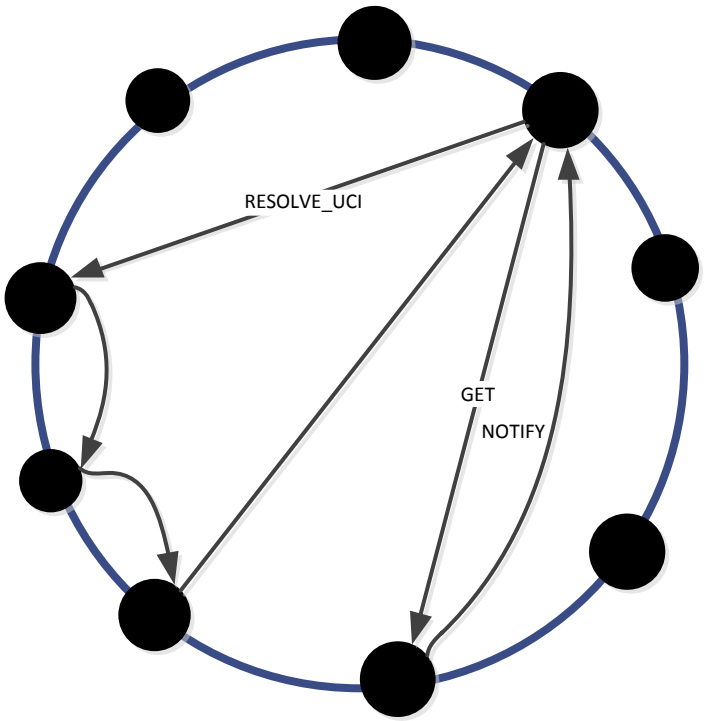


Figure 8 DCXP Signalling 19

2.6 Mobile DCXP Proxy

Mobile DCXP proxy (aka MDP) is a server providing two important functions in the DCXP network.

1. Mobile devices must register on the MDP to gain access to the peer-to-peer network In addition, it processes computation and thus reduces burden from the mobile devices.

2. Shields the peer-to-peer network from packet loss and other likely disruptions arising from the behavior of radio communication link.

2.7 Summary

In this chapter detailed literature review are carried out. Those protocols and research work are presented that address scalability and mobility issues. LISP, HIP, MobileIP and its variants are covered in great details along with limitation and pros. Furthermore , the architecture on which the thesis based on i.e. Sensiblethings is covered and the also the structure of DCXP and Mobile DCXP proxy is also discussed in detail.

Chapter 3

METHODOLOGY

The thesis approaches the tasks mentioned in the overall aim section in Chapter One by dividing the bigger task into pieces of goals. So having studied the existing mobility solutions, which helps to gain enough knowledge base about the possible solutions and helps in proposing and implementing scalable mobility solution for the SensibleThings platform, the thesis presents the design and implementation of mobility solutions for SensibleThings platform.

Accordingly, the order of approaching the challenges is first of all I will study the existing related mobility solutions and present these mechanisms. Second of all I study the SensibleThings platform and implement scalability solution as an extension to the existing implementation. Finally, I will evaluate the performance of the solution that has been proposed. The approaches has been precisely presented and explained as in the following subsections.

3.1 The Existing Scalable Mobility Solutions

Brief study of the existing scalable mobility solutions is presented and discussed. The thesis covers those solutions which consist of a better support as well as provide common and open solutions. The thesis has a target in this case to present all the relevant technologies and specifically put emphasis on those technologies designed to encompass the architecture of the future Internet-of-things. So in the piece of task , we have planned to cover study LISP, HIP, Mobile IPv6, and other commonly used protocols.

3.2 Scalable Mobility Solution for the SensibleThings platform

As explained in the first chapter, one of the goals in this thesis has been to design and implement a mobility solution for the SensibleThings platform. So to this end the thesis studies the existing solutions and proposes a new solution for the SensibleThings platform. Besides , we will study the DCXP protocol which has been used to disseminate context information between nodes. More specifically, the focus is on the Mobile DCXP proxy server (MDP) and the intention is turn this part of DCXP onto scalable solution and as well as keep the smooth mobility feature. To

make the idea and the proposal clear , the thesis presents designs, proof-of-concepts and diagrams

3.3 Implementation of Mobility Solution

After a thorough study, for example using proof-of-concepts, and close examination of the concepts used in the proposed solution, we design the solution. Based on the proposed design, we implement the solution using Java programming language. The implementation which is to be done in this section ensures mobile and scalable solution. In architecture of the SensibleThings extension work needs to done on the Add-in layer. Nevertheless, the mobility and scalability features more related to the Dissemination layer and the actual task to be done in this section is modification of DCXP. Therefore , implementation of our proposed solution will be carried out on the dissemination layer

3.4 Evaluate the Performance of the Proposed Mobility Solution

The last task has been to evaluate and prove that the proposed mobility solution could be applied in a real scenario. To this end an android app has been developed and run on a different scenarios as well as hardware. In addition, proof-of-concept application has been developed which could show how the proposed solution could actually works. In part of the task, we will count the packet loss as compared to the existing system. The results will then be presented using diagrams and figures. In this case we might need networking tools such as WireShark™.

3.5 Software tools and important equipments used in the Performance Evaluation

During the development of the Android apps we used Eclipse development environment. To draw diagrams and figures , Microsoft Visio 2010 has been used. In addition, networking tools, for example WireShark, has been utilized in order to measure the timing that might occur during switching a network. Finally, Smartphone running on the Android Mobile platform has been used. The actual devices we used in this project are Samung S3 and Samsung Nexus 2.

3.5 Evaluate the output of the research

The research carried out in this thesis work aims to propose and implement scalable mobility solution on the existing SensibleThings platform. Accordingly, two alternatives have been proposed one is within the scope while the other is out of the scope. The actual outputs from this task have a new mechanism to increase the scalable mobility on the SensibleThings platform known as Mobile DCXP which is an improvement to the previous Mobile DCXP proxy. We

have done the implementation of Mobile DCXP and performed the evaluation of the our solution. In addition, comparison of the existing system and the proposed solution have been carried out. Mobile application have been developed to study the scalable mobility and furthermore proof-of-concept application have been created to show the concept of Mobile DCXP (MD).

Chapter 4

DESIGN

The SensibleThings platform is an all-in-all module for the Internet-of-things. The ultimate goal of the platform is to serve as fully distributed overlay network. The dissemination layer of the platform is responsible for the look up, join/leave and resolving the address. In order to carry out these tasks as intended the Distributed Context eXchange Protocol (DCXP) has been utilized. Therefore, our target in these sections have been to design a suitable mechanism on how to smoothly perform context information dissemination in a cases when a nodes detaches itself from wireless access point and joins another access-point.

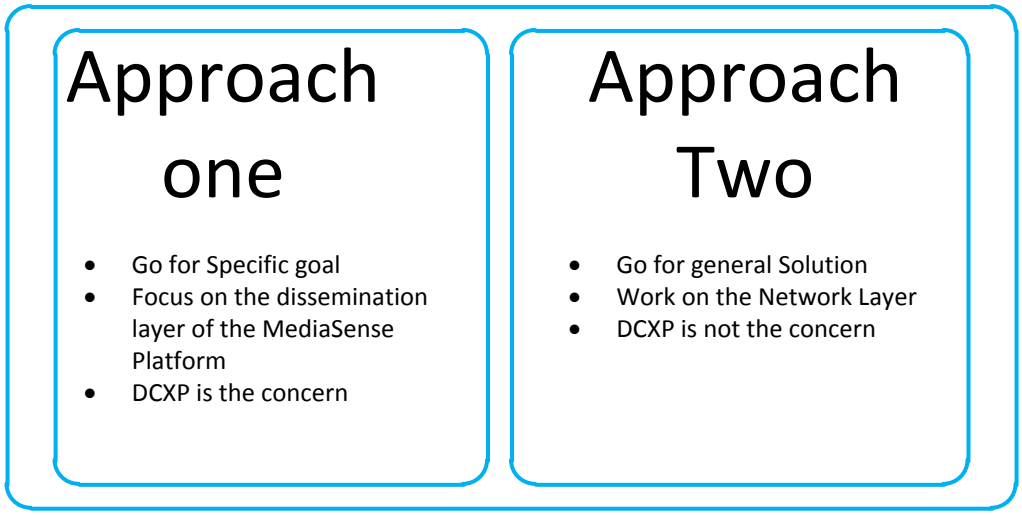


Figure 9 the mechanism

4.1 Proposed Mobility Solutions and Important Points to Examine

One of the root causes for problematic issues related to handover within access points has been the fact that devices address are topology dependant. It is not a mere random choice but the topology dependence nature of the device identifier offers efficient routing of devices within a network. Moreover, if the assignment of topology dependant identification couldn't help smooth handover then it is convincing to add another address which is not topology dependant. So using the former address provides efficient routing within the network. However, using the latter address offers smooth handover. Therefore, the second approach in this thesis is to deal with the real root cause of mobility issues. The second approach works on the network layer of the devices. The implementation has been deferred for future work and only the proposal has been

discussed. Furthermore, the general approach takes quite similar assumptions with the existing LISP protocol. However, the ultimate concern in the approach two is to find solution for the smooth messaging on the SensibleThings platform rather than to work for a general solution.

The first approach is to circumvent data loss during handover using the DCXP protocol. So this approach takes into consideration that data loss occurs in the lower layers. However, if a node acknowledges on receiving data, then it could be possible to keep the node updated as far as the device stays connected to the overlay network. This approach in effect doesn't keep the node connected during handover rather it is concerned on the update of information and thus prevent data loss.

4.1.1 Extended Comparison of the two approaches

	Approach One	**Approach Two**
Features	Implemented on the dissemination layer of the SensibleThings platform. In this case additional primitives are proposed. Moreover, approach one has nothing to do with the lower layer protocols.	Implemented on the network layer of the five layer model. This approaches attacks the problem from the root cause and this could be extended for other platforms.
Scalability	Approach one fully accomplishes its intended real task only if the latency taken by Mobile IP is too small to outdate the relevancy of a message intended to be transmitted to disconnected node. So if the message being transmitted could be retransmitted within the time slot allocated then Approach one is fully Scalable.	Approach two is a kind of solution similar two LISP but follows its own implementations and ideas. Therefore, in this case the concern is not just to reach the messages to the intended node but also to avoid any packet drop that may occur when handover occurs
Packet drop	Yes. packet drop occurs	No. Packet drop occurs
Advantages	Easy to implement and could be	Better Solution as compared to the

	used as work around solution. In this case, a little modification on the existing SensibleThings platform could suffice.	other approach. It could be used in other similar platforms.
Disadvantages	During an extended delay, the relevancy of a message to be retransmitted gets outdated so that means even if it is possible to retransmit the packet the messages are old enough to be considered irrelevant. So eventually, a lot of packet drop occurs.	Not easy to implement and takes longer time from design phase to testing phase as compared to the other approach. So this approach can't be considered for implementation in this thesis
Related works and Expectations	We expect using this approach to increase the scalability of SensibleThings a little bit further if not to 100%. The achievements are presented in the Results section.	Related works for example LISP ,HIP. The expectations in this case would be quite similar protocols as in these existing but our case would work on optimization.

4.2 Approach one – Focus on Dissemination of Context Information

In this approach the target has only been to find out a solution for smooth dissemination of messages when handover in wireless network happens and the belief is that DCXP is the only source of problems. So the problem has to be treated on the dissemination layer. The fact that the dissemination of information occurs in the real time carries a meaning that only within a certain period of time difference that information is considered valid for dissemination. Based on this fact arise the question "*Is it not possible to retry dissemination of packets that have been lost dues to handover if the context information is valid?"*Well if that retransmission is possible it means that there is a chance that in the retrial the peers shows up in another wireless access-point within threshold of validity of the information and able to confirm its legitimacy. The exact answer for question entirely depends on the structures of DCXP.

DCXP

DCXP is a protocol for a peer-to-peer real-time context data exchange, in other words, real-time context exchange refers to the communication delay which is insignificantly very minimal and

extended latency is not tolerable. Besides, radio link disruptions and issues related to packet loss are handled through the MDP (Mobile DCXP Proxy). Furthermore, the delays for exchange of messages on DCXP are expected to be a few seconds.

However, in real environment mobile devices don't get continuous and smooth access to the peer-to-peer network. In real environments, handover takes place during mobility and which means in most cases packet drop occurs.

MDP

MDP acts as a server to shield the DCXP network from packet loss due to radio link nature. So MDP is part of DCXP network where the thesis is actually concerned. It has been explained in section 2.6 that MDP is a server where each of the nodes need to register when they join the peer-to-peer network. MDP could be considered as a node by itself but having a much better computing capability as compared to the real nodes. So here we have a server which could possibly prevent packet loss somehow. However, such idea of employing a server as one option to work on the packet loss due to radio link nature is not scalable but only insures mobility. Therefore, the secondary function of MDP needs be addressed by distributed manner not by a centralized severer if the need is to achieve scalable mobility. The thesis detaches the task of mobility from MDP server and proposes a distributed solution to achieve mobility as well as scalability at the same time. However, the primary task which is to register nodes and act as main computation center is left unchanged in proposed modifications though this would mean there is still scalability friction because of the presence of central server.

Mobile DCXP (MD) – Our Solution

MD is a distributed approach to handle the problem of packet loss arising due to the nature of radio communication. Each of the nodes is responsible to keep mobility of nodes. Besides, such a system is scalable as compared to the previous solution. The mechanism in this section is to

introduce additional primitives that help to trace out the fate of a message disseminated. Accordingly, if a packet ends up nowhere due to handover, then retransmission need to follow or if the packet successfully reached the destination then no retransmission. See the pseudo code below.

```
Node node;
Message message1, message 2, acknowledge;
// Function 1
function Disseminate (Message m , Node[] nodes)
{
// send messages to all the nodes in the member list.
}
// Function 2
function Redisseminate (Message m, Node[] nodes)
{
// Nodes=  unacknowledged nodes
if (message is not obsolete)
// send messages to all unacknowledged nodes
retry Disseminate(m,nodes);
else
//if the message is obsolete due to extended handover delay or due to the node itself , don't retransmit
return;
}
```

4.2.1 Proof-of-Concept

In the proposed solution the real aim is to carry out retransmission of messages in a proper time to those nodes going through handover and thus experiencing packet loss. The real headache is that the messages gates obsolete and irrelevant through period of short time. Besides, as already mentioned, keeping smooth follow of data is unthinkable as long the device gets disconnected during the handover. In other words, this would mean if the latency of handover is greater than the time for the message relevancy, then our proposed solution do nothing more than prevent packets loss and make it available at improper time. However, since the time taken to switch from one access-point to another is very minimal , approach one could be an alternative work around solution. As illustrated in figure 10 four active nodes and two access points have been shown.In addition , the dotted red line demarks the point where a possible packet drop out occurs. Nodes 1,1',2' are static and node 2 is on move. As can be seen in figure 10 c node 2 jumps on the red line and gets disconnected from the networks. However, as it moves further to the left (figure 10 d) it get connected to the network again. The time we need to redeem in this example is the period of time t between d and b. If , for example message m destined to node 2, had been dropped in the time t, then the source node gets not acknowledgment so it needs to retransmit as far as the message is relevant. The relevancy of a message matters for protocols such as DCXP where the real target is to disseminate context information which is continuously evolving in most cases.

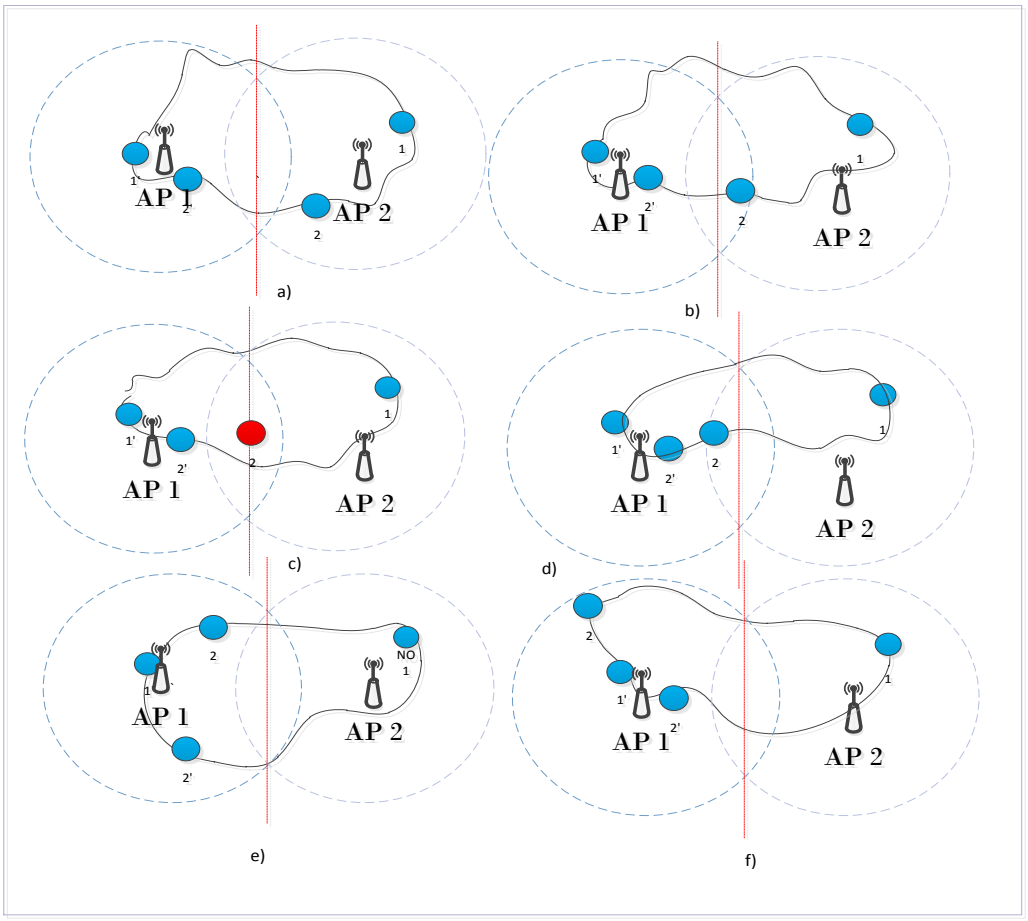


Figure 10 Handover

In figure 4b, node 2 is on the verge of disconnection from access point 2 . Let's assume right after that node two is no more connected and the time is T1. Furthermore, node 2 get connected with another IP adress possibility but the same ID through access point 1 at time T2. Therefore, the time the proper handover took in this case is T= (T2-T1). If node 1, for example, was trying to reach node2 in figure 4b, the packet loss will inevitability occurs. However, the solution in this case is that node 1 keeps sending to node 2 until the nodes shows up as shown in figure 4d. The limitation on the number of retrial could be based on the maximum time the handover might take or the deadline of the message relevancy.

4.2.2 Mobility Extension on the SensibleThings Platform

The SensibleThings platform is a fork project from the noble MediaSense platform. The MediaSense is an end-to-end platform for the internet-of- things. DCXP takes the core task for context data dissemination. On the SensibleThings platform presented in section 2.5, extension and optimization work should be carried out on the Add-in layer and so that keeping the entire

architecture unaltered. Since the intention in the proposed solution is to add one more primitive into the existing list of primitives, then the modification could have been done conveniently on the dissemination layer and more specifically on the dissemination core.

The existing system (as shown in figure 11a) throws destination unreachable error in the case when destination node is not accessible because of various reasons including a very short disruption due to access point handover. Therefore, in this case the packets get dropped and the destination node is not able to get the message even if the switch takes a few seconds.

However, in the proposed modification the source nodes send the message again and again until the current messages get outdated or until the node receives the message. Once the message arrives at the destination node, acknowledge message is sent back to the source node (See figure 11b).

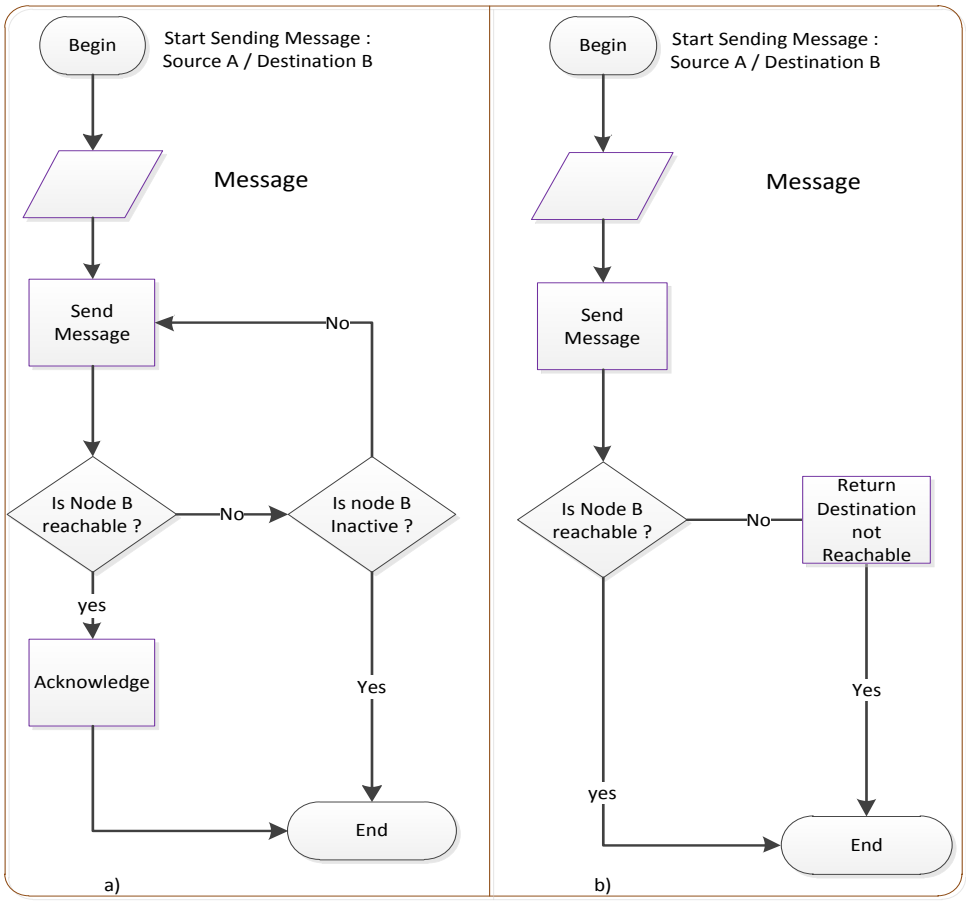


Figure 11 The SensibleThings Message Forward Flowchart a) the Proposed modification to enhance scalable Mobility b)the Existing Message Forwarding as shown using flowchart.

4.3 Approach two

The second approach solves the problem from the root. The cause of the problem has been the well known issue of latency and packet loss in the existing wireless network handover mechanisms. Therefore, in this section we propose a general approach for the SensibleThings platform. The Implementation of the second approach is deferred for the future task but we have explained all the necessary techniques in this section.

Mobility Solution

We believe that the observation made by the Location Identifier Separation Protocol proposers is the suitable beginning to start with the scalability problems in the existing SensibleThings platform. As shown in figure 12 The device could be in one of the positions as shown on the position 1 , AP1 could only provides the service however on position 2 and 3 both AP1 and AP2 provides the service. Moreover, in position 4 only AP2 could provide a service. So in this diagram on the positions 2 and 3, there is a session disconnection problem as the devices move to the other end. If we give each device two address i.e. one topology dependant and another topology independent address, it is possible to have smooth handover in positions 2 and 3.

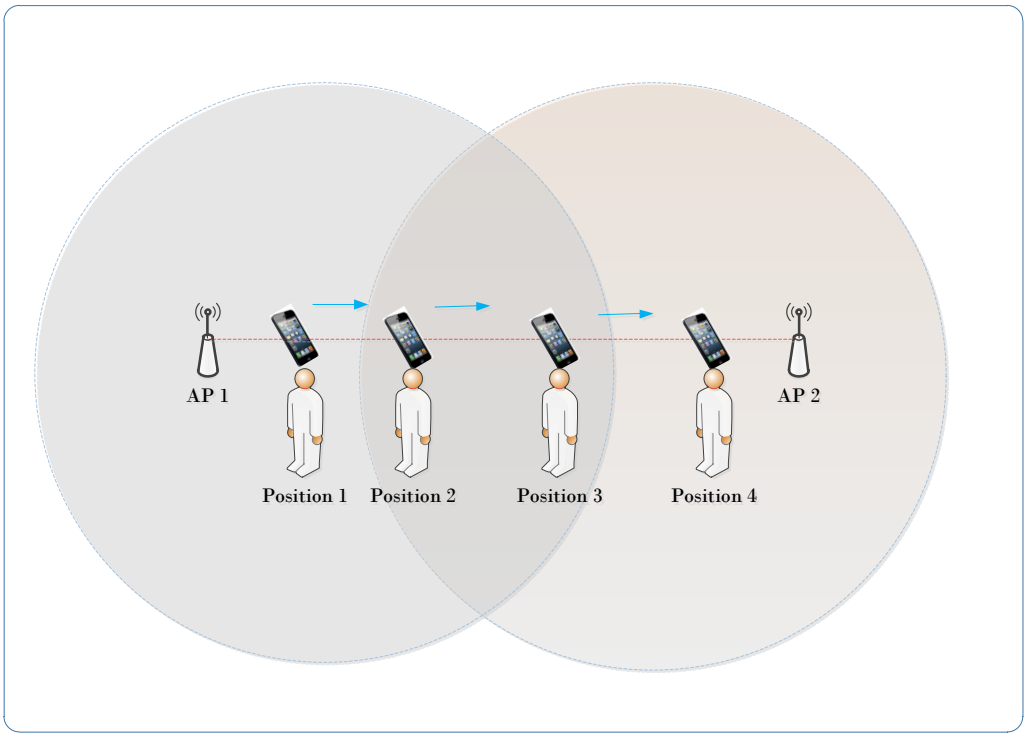


Figure 12 Handover

4.4 Summarization

In this section the proposed solution, alternative solution and the existing system has been discussed briefly. Moreover, comparison of approach one/ proposed solution and approach two / deferred generic solution has been explained. Furthermore, the proposed solution named Mobile DCXP (MD) has been presented and proof-of-concept for MD has been shown in detail.

Chapter 5

IMPLEMENTATION

In this section of the report the implementations of what has been named the Mobile DCXP or MD has been briefly presented. Figure 4 below shows the implementation done in this section on the top of the existing SensibleThings platform core. The IChat Android application has been developed to evaluate the Mobile DCXP proposed. Therefore, IChat runs on the top of SensibleThings platform. Furthermore, the Extensions added on the Add-in Layer of the SensibleThings have been used in the application.

In addition, the proof-of-concept explained in detail in the Chapter four has been implemented in this section. Therefore, the proof-of-concept app has been intended to show the concept of Mobile DCXP more clearly.

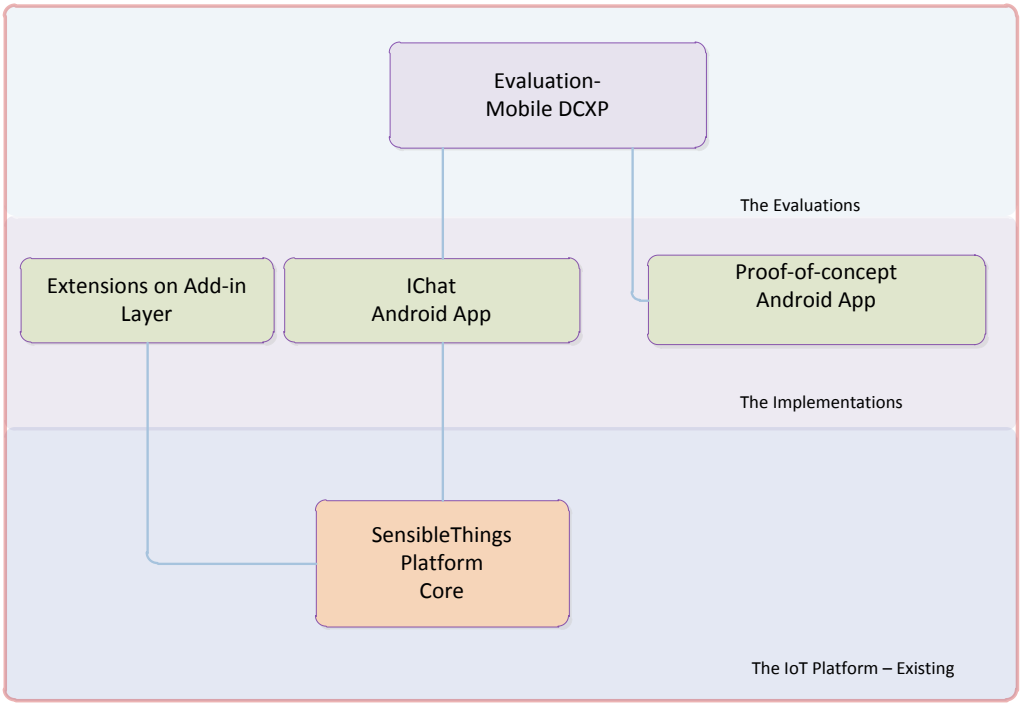


Figure 13 Overall diagram of separation of tasks and the implementations carried out

All subsection of the implementations shown in figure 4 has been explained in detail in this Chapter. However, the evaluations section has been presented in Chapter 6 and Chapter 7 .

5.1 Extensions on Add-in Layer

In the add-in layer we have added functions that could enable '*DestinationNotReachable'* error to be thrown not just after the first transmission failure but after going through continues retrial within the given time period.

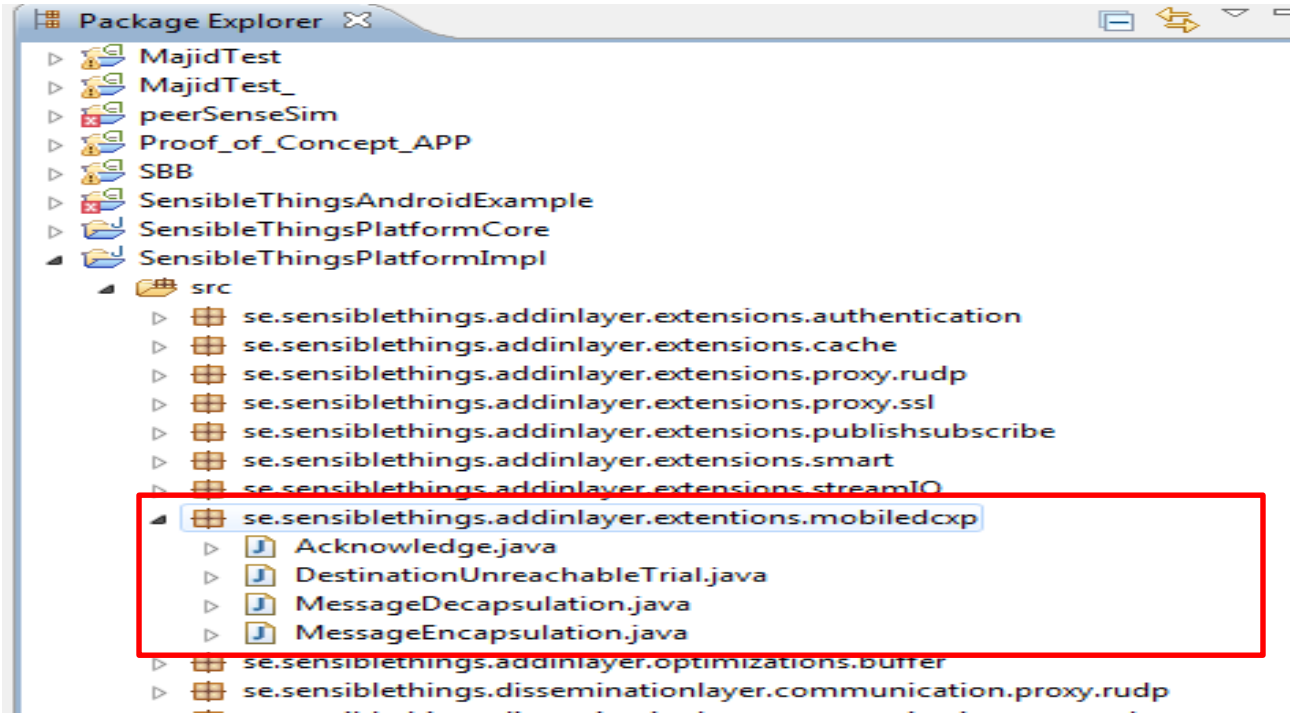


Figure 14 Extensions on Add-in Layer

Functions and Codes Added

Acknowledge:

Acknowledge is the sixth primitive we have proposed to have a better scalable mobility. This function does exactly similar task as in the Message.java but the difference is *Acknowledge* is intended to confirm whether the packets have arrived.

DestinationUnreachableTrial:

In the existing system when a node is disconnected all of a sudden an error "*DestinationNotReachable*" is thrown and after that even if the disconnection is just for an instance as in the case it happened during the handover, the destination node doesn't have a second chance to receive a message. In the TCP socket when there is a problem with the

destination of packet then an error is caught and thus handling this error has been properly done in the existing implementation however it is not efficient. The work we have done in this part could be categorized under optimization as the real need is to handle the *DestinationNotReachable* error in manner that would optimize the efficiency of message dissemination in time where there is high mobility.

MessageEncapsulation:

The code developed in this section is to send messages having another format which is the message to be transmitted in the existing SensibleThings Platform. So the task in this section is to add additional sections to the Message and then retrieve the extra information on the end with out change anything on the existing System.

MessageRetrival:

In this part of of the code we do the "Decapsulation" at other end. So the extra information attached to the message is retrieved and dealt on accordingly.

5.2 IChat Android Application

IChat is simple Android application developed in order to study and find out the advantages of having mobile DCXP on the existing SensibleThings platform. So IChat will be connected to the existing SensibleThings Platform and the packet loss is examined as well as the same procedure will be followed with mobile DCXP included on the SensibleThings platform. The Very IChat application doesn't come up with advanced idea but it is just an ordinary chat application having in addition to that auto generated message exchange (or chat). Therefore, the application allows studying the amount of packet loss with and without mobile DCXP.

5.3 Proof-of-Concept App

The proof-of-concept application illustrates the concept of the Mobile DCXP shown in the diagrams in section 4.2.1 in a clearer manner. The application in this section simulates the events happening during handover between two wireless access points and four connected nodes. Therefore, the messages exchanged and the retransmission trial can be observed more clearly.

Chapter 6

RESULTS

The results section presents in detail the experiment results using graphs and diagrams, the developed applications (IChat and proof-of- concept applications). In this part of the thesis comparison of existing system with the proposed system have been presented in detail. The improvements achieved using Mobile DCXP has been presented in figures. In addition, the extensions developed to achieve Mobile DCXP (MD) have been explained in detail.

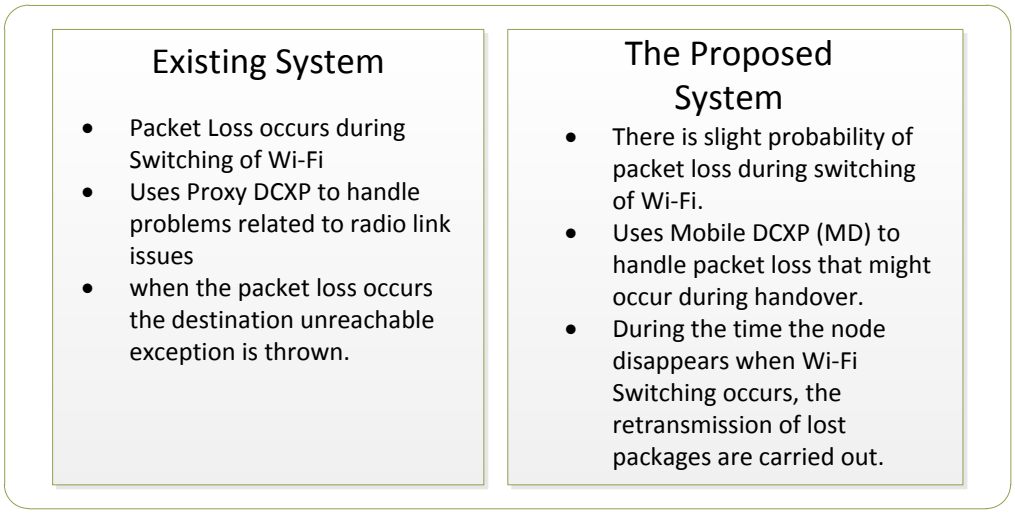


Figure 15 The proposed System and the modifications

6.1 Comparison and Evaluation

To examine the performance gain of the proposed system over the existing system with regard to the packet loss, we have carried out an experiment. In the experiment we assign unique numbers (Identification) for each of the messages exchanged. The identification could help to array the incoming messages as well as tells us which packet is missing. Therefore, the two system has been tested by switching the Wi-Fi. In this case , there is one factor which is the time of relevancy of the message. When the point of discussion is the context information coming from under laying sensors and sensory networks, the experiment should consider the context information relevancy factor. Therefore, in this case since the SensibleThingsPlatform is

intended for real time context-exchange , after a couple of extended trial we need to stop sending the message. So in this case if the node is taking longer than the maximum time limit for real time communication delay then there happens the inevitable packet loss. In fact that was the slight chance a packet loss might occur in the proposed system. However, based on the kind of applications , the retransmission of the lost packet for further extended period of time could be useful than just leave the lost packet and just turn to the new packets.

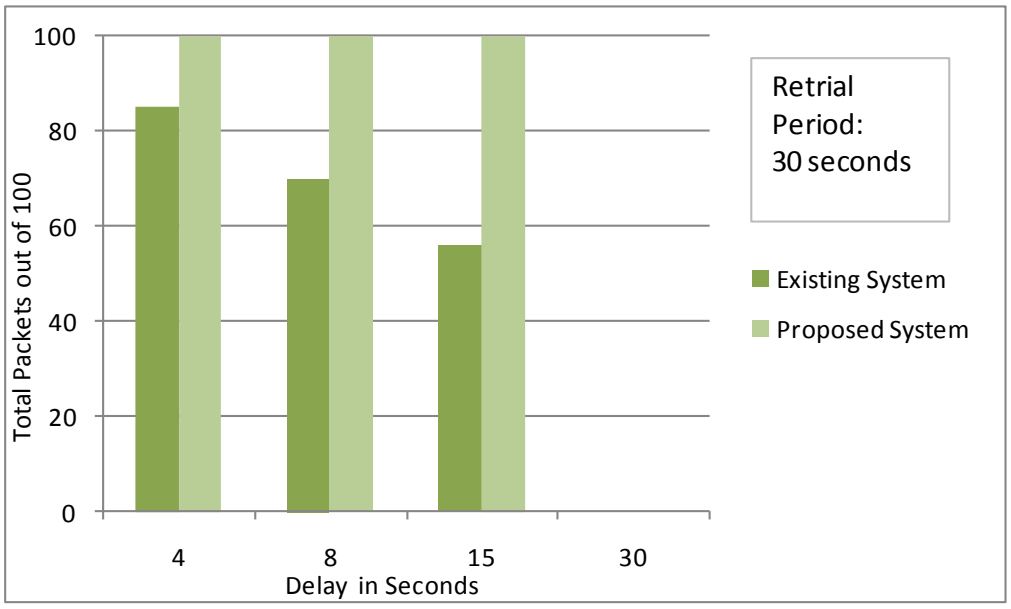
In this experiment we used the local network (the SensibleThingPlatform with DistributedLookup intended for the connections without NAT transversal). Furthermore, the network (see Figure 16) we set up for the experiment consist of two tablets (Samsung galaxy tab 7.0 Model number : GT-P3110) and (XTouch Model number: X704) as well as a laptop to examine the packets coming in and going out.

Figure 16 the Peer-to-peer network.

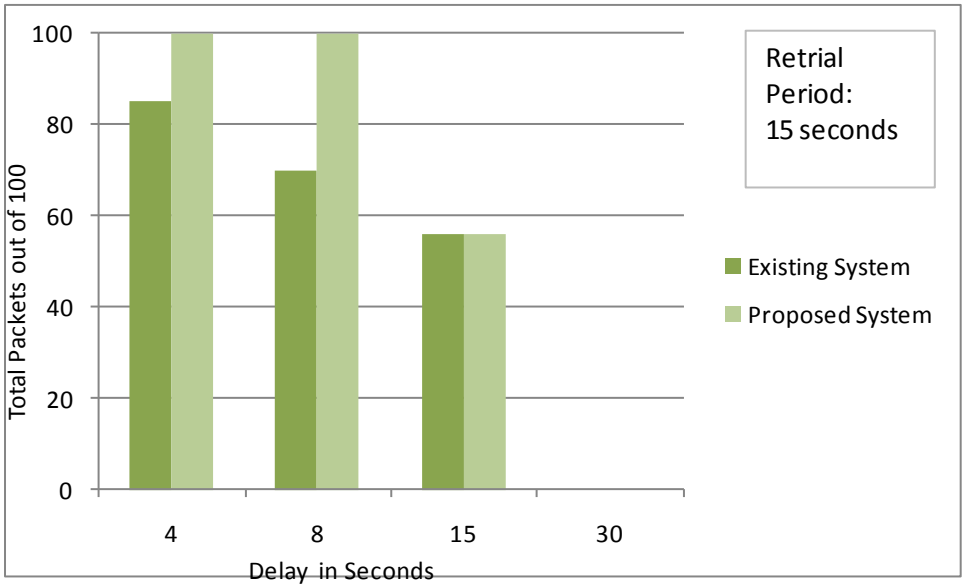
The procedure:

In order to find the number of packet loss in proposed and existing systems , the Access Point have been switched on and off to simulate the handover. This method have been preferred because of convenience and due to the fact that it is possible to manage delay that could possibly occur during handover.

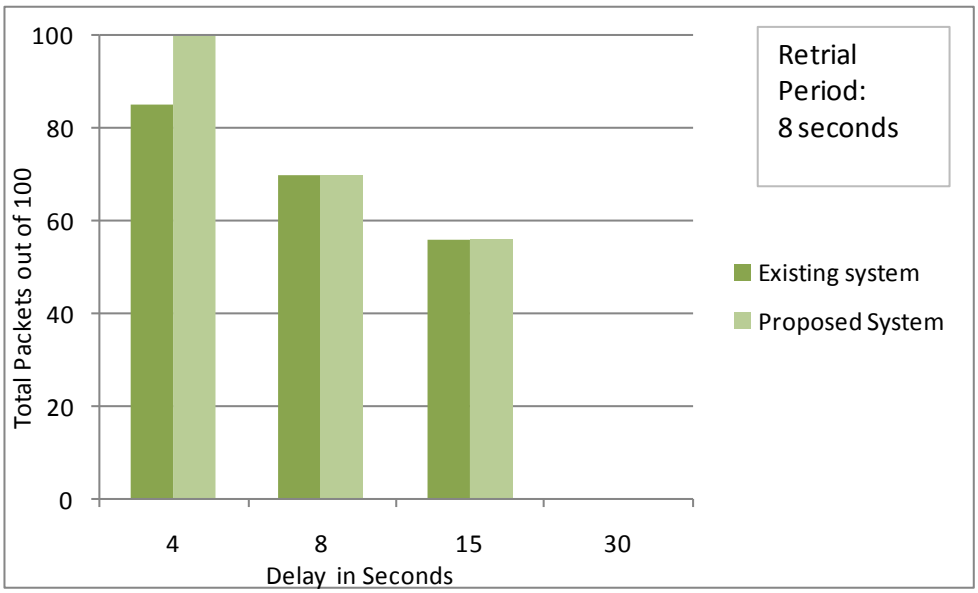
Therefore, the actual time Access Point takes to reboot is around 4 seconds in effect which means peer-to-peer communicating devices will encounter ***DestinationNotReachable*** error. In the existing system packet loss happens and no way to stop that. However, in the proposed system each peer retries the transmission lost packets and we have used varied retrial periods. In this case to measure the probability of packet loss as it happens in the extended delay has been shown. So we have collected the percentage of packet loss for delay times of 4sec, 8 sec, 15sec, 30sec, 45second and 60second. A total number of 100 messages have used in the study and each message is assigned a unique serial number.



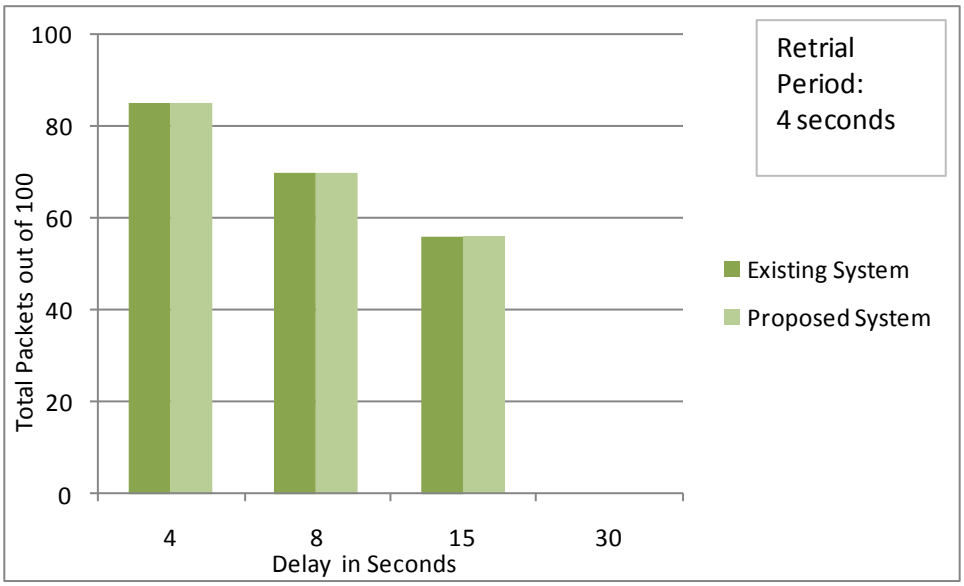
a)


100
80
60
40
20
0
Total Packets out of 100
4
8
15
30
Delay in Seconds
Retrial Period: 15 seconds
Existing System
Proposed System

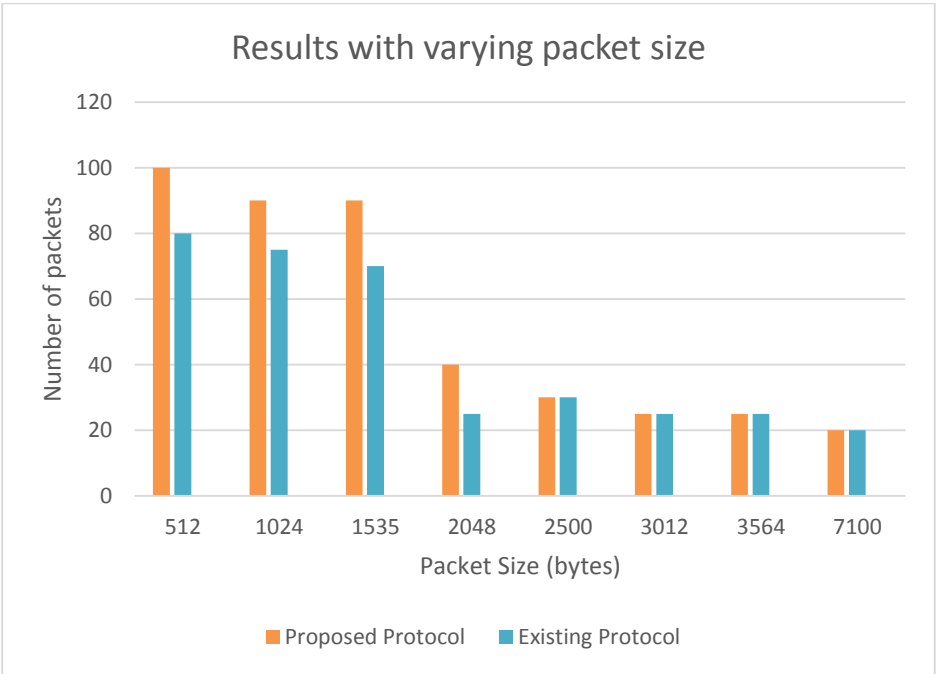
b)


100
80
60
40
20
0
Total Packets out of 100
4
8
15
30
Delay in Seconds
Retrial Period: 8 seconds
Existing system
Proposed System

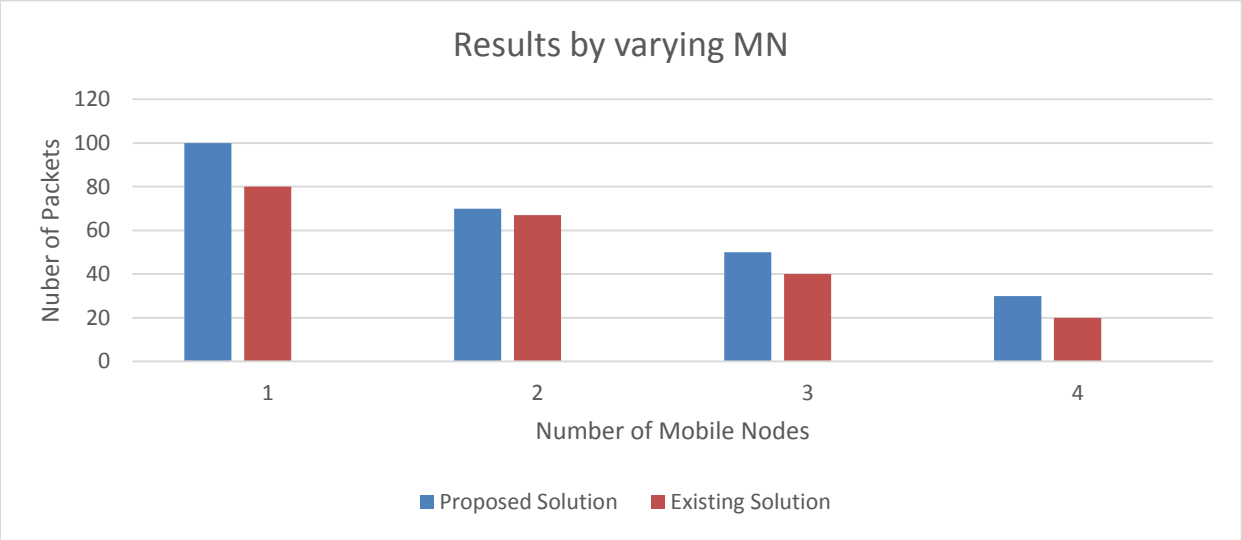
c)


100
80
60
40
20
0
Total Packets out of 100
4
8
15
30
Delay in Seconds
Retrial Period: 4 seconds
Existing System
Proposed System

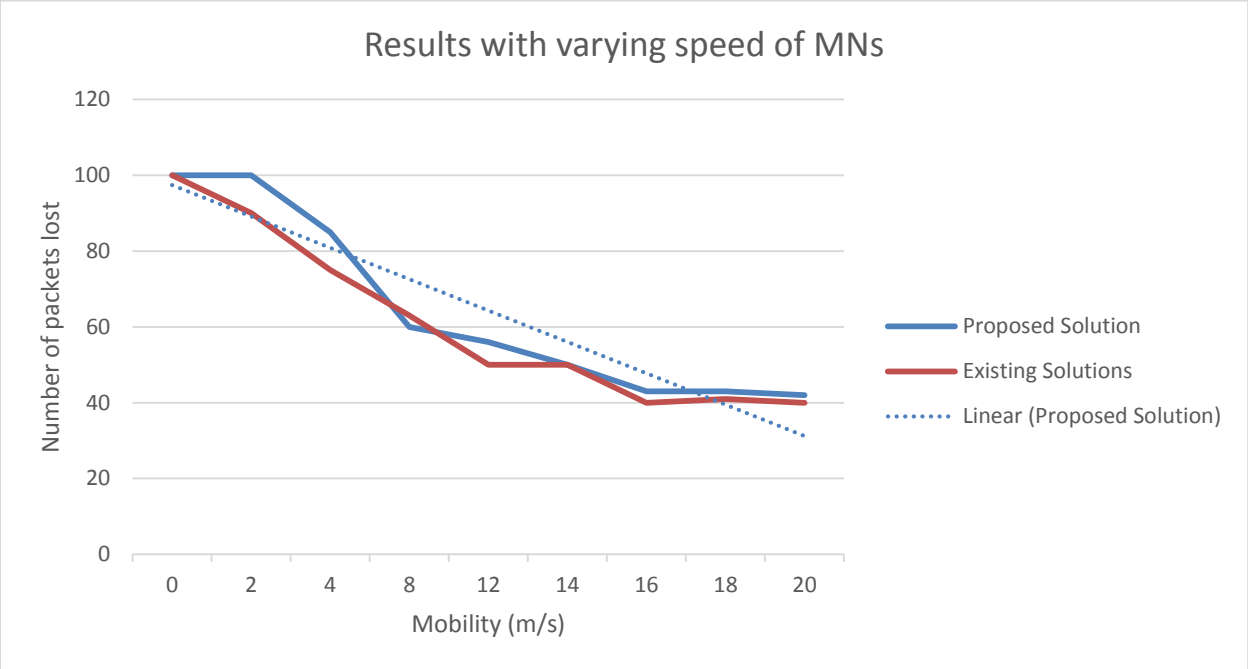
d)


Results with varying packet size
120
100
80
60
40
20
0
Number of packets
512
1024
1535
2048
2500
3012
3564
7100
Packet Size (bytes)
Proposed Protocol
Existing Protocol

E



F



G

Figure 17 Bar Chart illustrating the packet losses in the existing system and proposed system. A) Result with retrial period = 30s b) Retrial period = 15s c) Retrial period = 8s d) Retrial period = 4s e) Results with varying packet size f) Results with increasing number of Mobile Nodes G) Results with changing speed of mobile nodes

The charts in the figure 17 shows the gains in the performance of the proposed system over the existing SensibleThingsPlatform. The result clearly shows that as delay of Wi-Fi handover increases the gain of the existing system decreases. Though it could have been a possible solution to consider further extended retrial period but that doesn't fulfil the requirement of the Sensiblethingsplatform. Therefore, the probability of packet loss has been decrease but still there exists a breach to loss packets in the cases of extended time taken by handover. However, under normal circumstances it doesn't take longer period than mentioned in this research.

Several scenarios are considered like varying packet size, varying number of MNs and varying speed on MN but in all cases it clearly shows the performance gain of the proposed syste.\ms

The other crucial issue to consider has been the challenge that could result from the fact that IP address could change and the exact node would be difficult to trance out. In this experiment during turning on and off the wireless access points the IP addresses doesn't change so it has not been a challenge in here. Nevertheless, in the experiment involving two access points and where there are many connections and disconnections this could be a problem to handle. One of the options to handle this could be to find out the nodes by their UCI. The resolve function could only results in either currently working IP address or port or it just throws an error.

6.2 Application Developed for testing purpose: IChat Android app

The IChat sends and receives messages coming out and coming in. It is such a simple application which allows the user to trigger the communicaiton , the intended purpose of IChat is just to help us test the Mobile DCXP and compare it with the existing SensibleThingsPlatform's Proxy Server. Therfore, the basic features of a chat application has been added to the IChat as shown in the figure below. The area labeled by the "Active nodes" takes the existing two peers (a bootstrap which is device1- Samsung galaxy and the normal node which is XTouch tab) we are working on for testing.

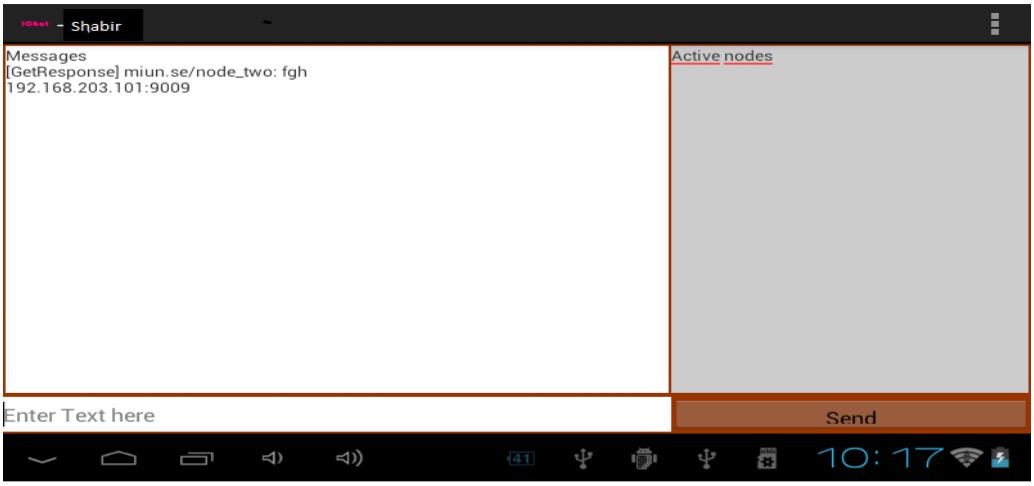
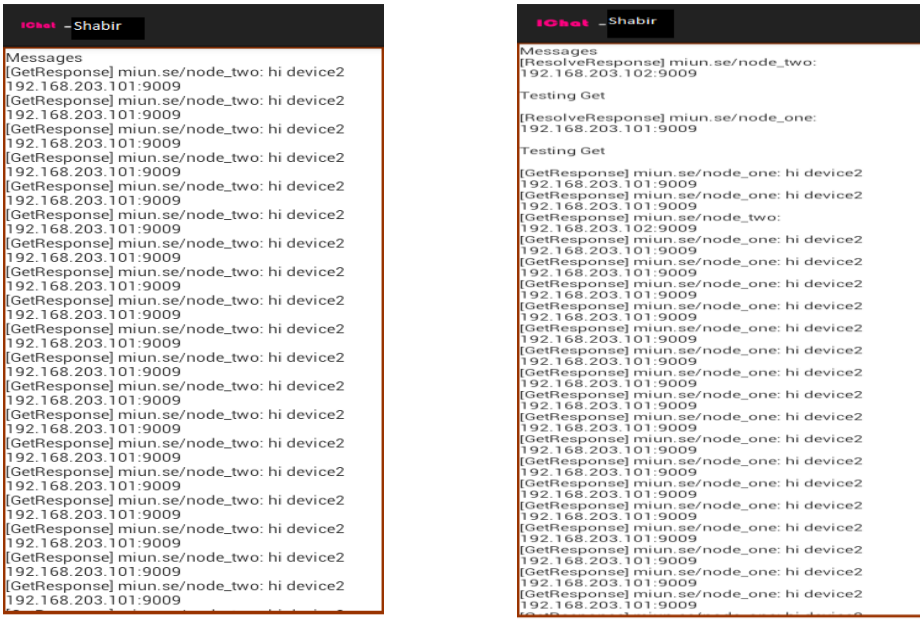


Figure 18 IChat Android based testing application

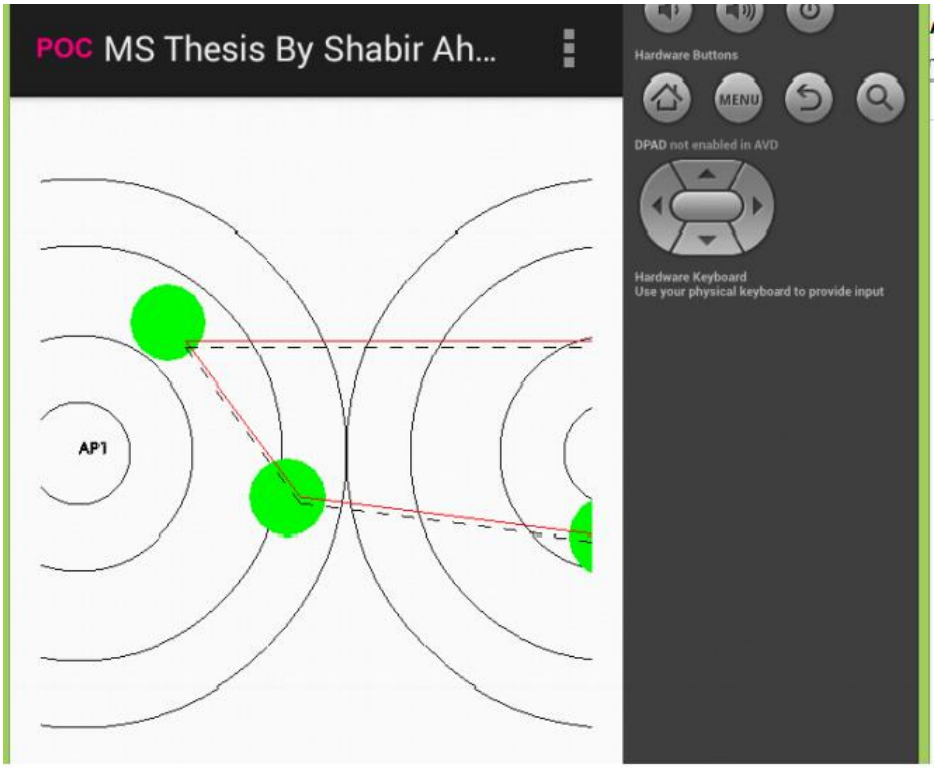


a) b)

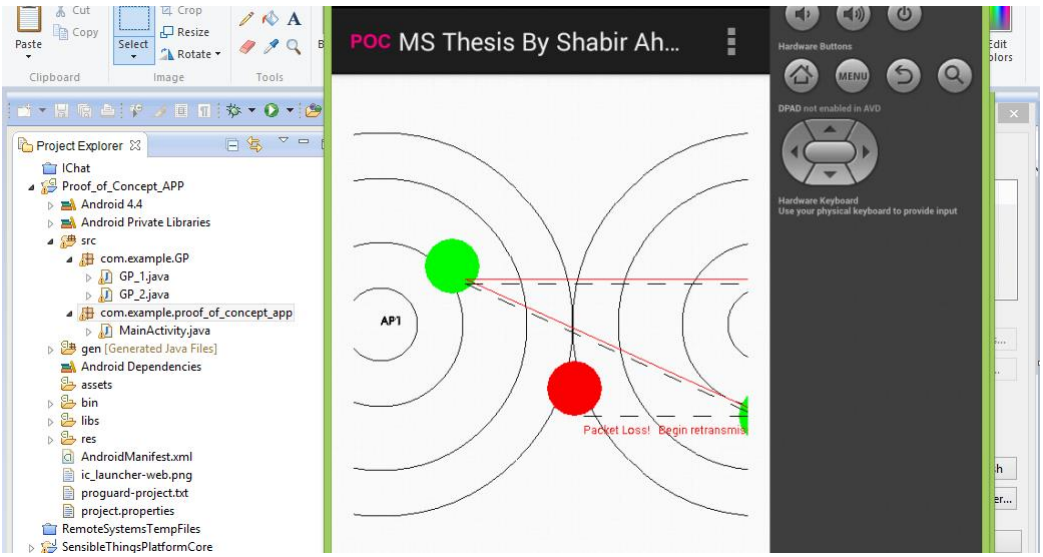
Figure 19 Two nodes showing the messages exchanged between them. a) the bootstrap node whose ip address is 192.168.203.101 b) the other node which shows the normal node whose ip address is 192.168.203.102. In this figure the bootstrap node send a message "hi device2" and both of them will receive that message.

6.3 Proof-of-concept Application

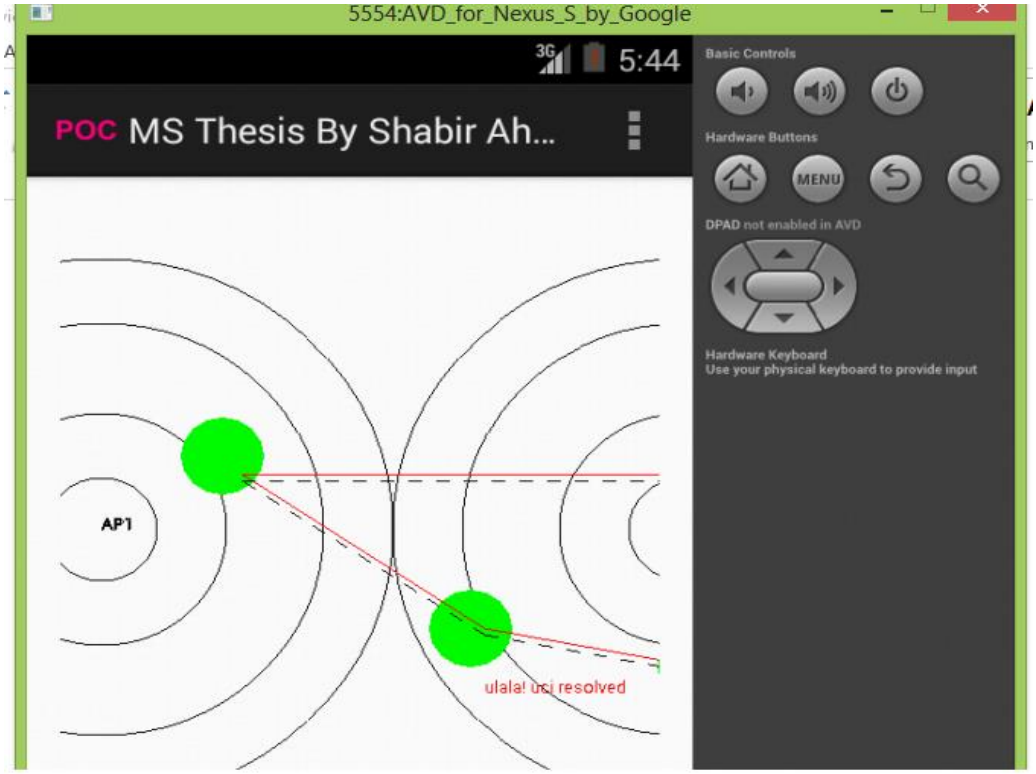
In this section the proof-of-concept application has been shown with different states. The implementation done in this part of the report is directly taken from the detailed explanation of the report presented in section 4.2.1.



a)



b)



c)

a) Initial Screen packet Is being transmited from AP2 to AP1 b) Handover occurs Node disconnected. c) Node retransmitted

Figure 20 Proof-of-concpet applications

Chapter 7

CONCLUSION AND FUTURE WORKS

7.1 Conclusion

The tasks clearly outlined in chapter one and carried out in this research wok could rather be presented as optimization work. This is due to the fact that the solution presented in this thesis work could exactly be presented as work around solution to come up with a better scalable mobility for the nodes connected over the SensibleThingsPlatform. We have proposed two solutions for the research questions mentioned. The workaround solution of Mobile DCXP (MD) as it is referred specifically has been chosen to show the performance gain over the currently existing version of the platform with regard to packet loss. However, the other solution which could handle the problem from the root cause and which could yield a better result has been kept for future work.

The tasks we have done in this research work could be summerized as in the following relative to what has been outlined for in the proposal and the introduction chapter.

In the chapter of two of this report existing mobility solutions have been presented briefly. Host Identity Protocol, LISP, Mobile IPv6, Hierarchical Mobile IPv6, Fast Handover ,MOBIKE, PMIPv6 has been studied. Moreover, the SensibleThingsPlatform and its functionalities have been presented in detail.

In chapter four of this research work solutions to handle the research question has been proposed thoroughly discussed. In this chapter two approaches haven forwarded and compared. Proof-of-concept apps and diagrams have been used to illustrate the concept of Mobile DCXP (MD). Furthermore, in chapter five the Add-in layer extensions have been implemented and the techniques have been shown.

In the results section we have carried out experiments to find out the size of packet losses in the existing system and the proposed system. In this section the IChat Android app , the proof-of-concept app has been presented in detail.

7.2 Challenges and Coding Errors

The implementation of the IChat has been a challenge. The writing of the codes doesn't take more than the what has been allocated in the time time however connecting and bring the app to a working state has been very problematic. The main reason being the size of the code for SensibleThingsPlatform and the networking we needed to set up. The documentions for users interested to communicate with platform is not detail enough. The other buggy and holding back condition has been when setting up the communication with platform within the LAN, the code does directly to the *RudpProxycommunication* rather than *Rudpcommunication* that part took longer to trace and finally we had to disable one of the conditions See the piece of code below.

The other difficult to trace out bug and time consuming challenge has been the time the platform takes to converge in cases where there is a problem with the underlying networking hardware. See the figure 21 below for problems encountered in this regard.

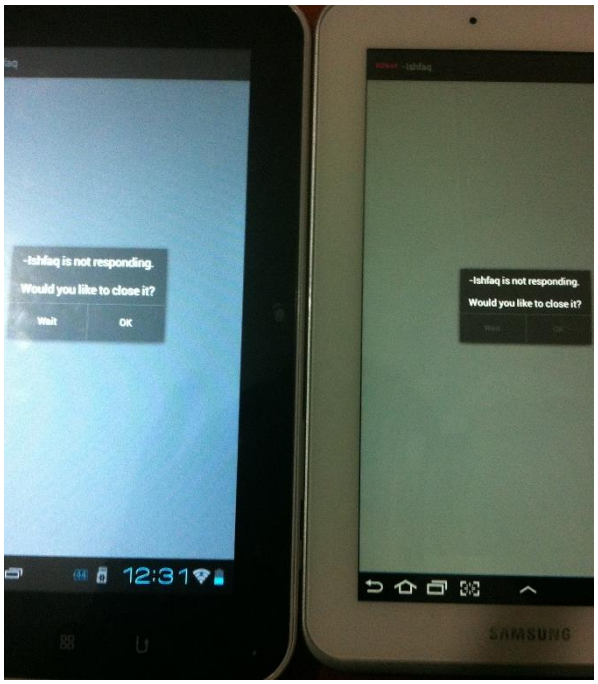


Figure 21 indefinite time for convergence. Not hint to trace out the actual error.

```
 * The SensibleThings platform itself, which exposes all functionality towards
 * the application developers.
 *
 */
public class SensibleThingsPlatform {

    private DisseminationCore disseminationCore = null;
    private AddInManager addInManager = null;
    private SensorActuatorManager sensorActuatorManager = null;

    /**
     * Initializes the SensibleThings platform. Must be called before using the
     * any other functions. This is now the suggested way to start
     * SensibleThings. It uses normal DHT lookup and RUDP. But switches to Proxy
     * automatically if you are behind NAT.
     *
     * @param listener
     *            The SensibleThingsListener, for all callbacks
     */
    public SensibleThingsPlatform(SensibleThingsListener listener) {

        //if (isBehindNat()) {
            // This forces Proxy when behind NAT
        //  initalize(LookupService.DISTRIBUTED, Communication.PROXY_RUDP);
        //} else {
            // Or else, normal DHT and RUDP
            initalize(LookupService.DISTRIBUTED, Communication.RUDP);
        //}
```

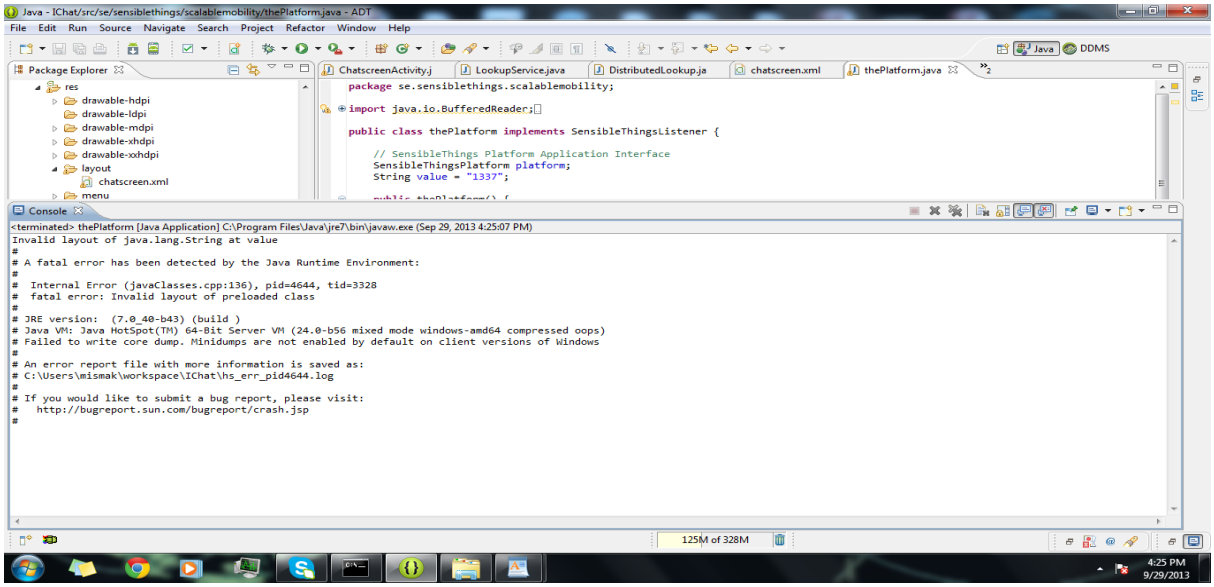


Figure 22 errors

7.3 Discussion

Mobile DCXP or MD has been proved that it offers performance gain as compared to the Proxy Server in the existing system. The fact that this app has not been tested in an environment where there is a lot of disconnections and connections happens may not change the result we come up with but in that case it could be necessary to handle the issue related with changing IP address

7.4 Contribution and impact

The actual work carried out in this research work contributes immensely both the existing system and for future research looking to pursue in the coming up with a better and generic scalable mobility solution for the SensibleThingsplatform. So for the existing system we have achieved a

better performance gain with regard to packet loss during Wi-Fi handover and other short radio link disruptions. In addition , in the long run the thesis has laid down a kind of necessary information for future researchers and students.

7.5 Future work

The ultimate future work related to this thesis has been as mentioned again and again throughout the report the implementation of approach two presented in chapter four. In this approach a kind of scalable mobility solution similar to LISP is worth considering in the future.

In addition , some other smaller tasks' which has not been tested within the scope of this work for example, testing the Mobile DCXP within environments where a number of node exist and some other factors such as could be part of the task worth considering.

Finally, during the implementation of the IChat android app we have found out that applications even the *SensibleThingsPlatformExample* App which could be downloaded for testing purpose from the www.SensibleThings.com is a little bit unstable in case of network error and keeps the resources indefinably busy unless forced to stop the process. In this case the app doesn't crash and exception is not captured either. Exception should have been caught after a couple of tolerable resource busy staff. This very problem could have come from the time the platform takes to converge as we have found out later on in the properly working system. If that is so , this problem could be expressed shortly as "indefinite time to converge" happens during network error. This could be one more task worth considering for future work.

APPENDICES

Appendix A.1: Code Snippet for Proof of concept App

GP_1.java

```
package com.example.GP;

import java.util.ArrayList;
import java.util.Random;

import android.animation.Animator;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.BlurMaskFilter;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.DashPathEffect;
import android.graphics.Paint;
import android.os.Build;
import android.os.Handler;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Scroller;

@SuppressLint("NewApi")
public class GP_1 extends ViewGroup {

	private boolean ploss = false;
	int tempX = 0, tempY = 0, X1 = 50, Y1 = 250, X_ = 0, Y_ = 150, X2 = 50,
			Y2 = 300, X__ = 0, Y__ = 500, X3 = 50, Y3 = 300, temp_rndX_1 = 100,
			temp_rndY_1 = 100, temp_rndX_2, temp_rndY_2, temp_rndX_3,
			temp_rndY_3, temp_rndX_4, temp_rndY_4;
	Paint mPaintone = new Paint();
	Paint mPainttwo = new Paint();
	Paint mPaintthree = new Paint();
	Random randomInt, randomInt_;// = new Random();

	Random randomBool = new Random();
	int rndX = 0, rndY = 0;
	int arr1[] = new int[30];
	int arr1_[] = new int[30];
	int arr2[] = new int[30];
	int arr2_[] = new int[30];
	int arr3[] = new int[30];
	int arr3_[] = new int[30];
	int index = 1;
	int index_ = 1;
```

```
int index__ = 1;
boolean flag = false, flag_ = false;
Handler uiThread = new Handler();
int count = 0;

public GP_1(Context context) {
	super(context);
	initialize();

}

public GP_1(Context context, AttributeSet attrs) {
	super(context, attrs);
	initialize();
}

@Override
protected void onLayout(boolean arg0, int arg1, int arg2, int arg3, int arg4) {

}

@SuppressLint("DrawAllocation")
@Override
protected void onDraw(final Canvas canvas) {

	super.onDraw(canvas);

	float startX = 0, startY = 0, stopX = 0, stopY = 0;
	// mPaintone.setColor(Color.RED);
	// canvas.drawCircle(10, 250, 302, mPaintone);

	mPainttwo.setColor(Color.BLACK);
	mPainttwo.setStyle(Paint.Style.STROKE);
	mPaintthree
			.setPathEffect(new DashPathEffect(new float[] { 10, 10 }, 0));
	canvas.drawCircle(30, 250, 210, mPainttwo);
	canvas.drawCircle(30, 250, 160, mPainttwo);
	canvas.drawCircle(30, 250, 90, mPainttwo);
	canvas.drawCircle(30, 250, 40, mPainttwo);
	canvas.drawText("AP1", 30, 250, mPainttwo);

	// mPaintone.setColor(Color.RED);
	// canvas.drawCircle(500, 250, 302, mPaintone);

	canvas.drawCircle(450, 250, 210, mPainttwo);
	canvas.drawCircle(450, 250, 160, mPainttwo);
	canvas.drawCircle(450, 250, 90, mPainttwo);
	canvas.drawCircle(450, 250, 40, mPainttwo);
	canvas.drawText("AP2", 450, 250, mPainttwo);

	mPaintone.setColor(Color.BLACK);
	tempX = 0;
	tempY = 0;
	X1 = 50;
	Y1 = 250;
	X_ = 0;
	Y_ = 150;
	X2 = 50;
	Y2 = 300;
	X__ = 0;
	Y__ = 500;
	X3 = 50;
```

```
Y3 = 300;
index = index_ = index__ = 1;
rndX = 10;
rndY = 300;
startY = 300;
startX = 10;
randomInt = new Random();
rndX = 20 + randomInt.nextInt(50);
rndY = 40 + randomInt.nextInt(70);
mPaintone.setColor(Color.GREEN);

// for (int i = 100; i < 400; i = i + 100) {
// this.invalidate(0,0,20,20);
canvas.drawCircle(70 + rndX, 100 + rndY, 30, mPaintone);
canvas.translate(14, 14);
canvas.drawCircle(400 + 2 * rndX, 100 + rndY, 30, mPaintone);
canvas.drawCircle(400 + rndX, 250 + rndY, 30, mPaintone);
if (6 * rndX < 275 && 6 * rndX > 200) {
	ploss = true;
	mPaintone.setColor(Color.RED);
	//
	canvas.drawCircle(6 * rndX, 220 + rndY, 30, mPaintone);

	try {

		synchronized (randomInt) {
			randomInt.wait(1000);
			randomInt_ = randomInt;
			invalidate();
		}
	} catch (InterruptedException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	canvas.drawLine(70 + rndX, 100 + rndY, 400 + rndX, 250 + rndY,
			mPaintone);

	canvas.drawLine(70 + rndX, 100 + rndY, 400 + 2 * rndX, 100 + rndY,
			mPaintone);
	canvas.drawLine(400 + 2 * rndX, 100 + rndY, 400 + rndX, 250 + rndY,
			mPaintone);

	mPaintthree.setPathEffect(new DashPathEffect(
			new float[] { 20, 20 }, 0));

	try {

		synchronized (randomInt) {
			randomInt.wait(500);
			randomInt_ = randomInt;
			invalidate();
		}
	} catch (InterruptedException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}

	canvas.drawLine(70 + rndX, 105 + rndY, 400 + rndX, 255 + rndY,
			mPaintthree);

	try {
```

```
                        synchronized (randomInt) {
                                randomInt.wait(500);
                                randomInt_ = randomInt;
                                invalidate();
                        }
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                canvas.drawLine(70 + rndX, 105 + rndY, 400 + 2 * rndX, 105 + rndY,
                                mPaintthree);
                try {

                        synchronized (randomInt) {
                                randomInt.wait(1000);
                                randomInt_ = randomInt;
                                invalidate();
                        }
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                canvas.drawLine(400 + 2 * rndX, 105 + rndY, 400 + rndX, 255 + rndY,
                                mPaintthree);

                try {

                        synchronized (randomInt) {
                                randomInt.wait(2500);
                                randomInt_ = randomInt;
                                invalidate();
                        }
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                canvas.drawText(
                                "Packet Loss! \n Begin retransmission for 1.5 seconds
to the same uci",
                                10 + 6 * rndX, 270 + rndY, mPaintone);

                canvas.drawLine(10 + 6 * rndX, 250 + rndY, 500 + rndX, 250 + rndY,
                                mPaintthree);

        } else {

                canvas.drawCircle(6 * rndX, 220 + rndY, 30, mPaintone);
                mPaintone.setColor(Color.RED);
                if (ploss == true)
                        canvas.drawText("ulala! uci resolved", 10 + 6 * rndX,
                                        270 + rndY, mPaintone);

                // canvas.drawCircle(10 + rndX + i, 350 + rndY, 30, mPaintone);
                // canvas.drawLine(10 + 6 * rndX, 220 + rndY, 70 + rndX, 100 + rndY,
                // mPaintone);
                try {

                        synchronized (randomInt) {
                                randomInt.wait(1000);
                                randomInt_ = randomInt;
                                invalidate();
```

```
        }
} catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
canvas.drawLine(10 + 6 * rndX, 220 + rndY, 70 + rndX, 100 + rndY,
                mPaintone);
canvas.drawLine(10 + 6 * rndX, 220 + rndY, 400 + rndX, 250 + rndY,
                mPaintone);
canvas.drawLine(70 + rndX, 100 + rndY, 400 + 2 * rndX, 100 + rndY,
                mPaintone);
canvas.drawLine(400 + 2 * rndX, 100 + rndY, 400 + rndX, 250 + rndY,
                mPaintone);

// //

try {

        synchronized (randomInt) {
                randomInt.wait(500);
                randomInt_ = randomInt;
                invalidate();
        }
} catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}

canvas.drawLine(10 + 6 * rndX, 225 + rndY, 70 + rndX, 105 + rndY,
                mPaintthree);

try {

        synchronized (randomInt) {
                randomInt.wait(500);
                randomInt_ = randomInt;
                invalidate();
        }
} catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
canvas.drawLine(10 + 6 * rndX, 225 + rndY, 400 + rndX, 255 + rndY,
                mPaintthree);

try {

        synchronized (randomInt) {
                randomInt.wait(500);
                randomInt_ = randomInt;
                invalidate();
        }
} catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
canvas.drawLine(70 + rndX, 105 + rndY, 400 + 2 * rndX, 105 + rndY,
                mPaintthree);
try {

        synchronized (randomInt) {
                randomInt.wait(1000);
```

```
                                randomInt_ = randomInt;
                                invalidate();
                        }
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                canvas.drawLine(400 + 2 * rndX, 105 + rndY, 400 + rndX, 255 + rndY,
                                mPaintthree);

                ploss = false;

        }

    }

    public boolean getSimStatus() {

        return this.ploss;
    }

    public void setSim(boolean status) {
        this.ploss = status;
    }

    private void setLayerToSW(View v) {
        if (!v.isInEditMode() && Build.VERSION.SDK_INT >= 11) {
            setLayerType(View.LAYER_TYPE_SOFTWARE, null);
        }
    }

    private void setLayerToHW(View v) {
        if (!v.isInEditMode() && Build.VERSION.SDK_INT >= 11) {
            setLayerType(View.LAYER_TYPE_HARDWARE, null);
        }
    }

    private void initialize() {
        setLayerToSW(this);
    }

}
```

GP_2.java

```
package com.example.GP;


import java.util.ArrayList;
import java.util.Random;

import android.animation.Animator;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.BlurMaskFilter;
```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Build;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Scroller;

@SuppressLint("NewApi")
public class GP_2 extends ViewGroup {

	public GP_2(Context context) {
		super(context);
		initialize();
		// TODO Auto-generated constructor stub
	}



	public GP_2(Context context, AttributeSet attrs) {
		super(context, attrs);
		initialize();
	}

	@Override
	protected void onLayout(boolean arg0, int arg1, int arg2, int arg3, int arg4) {

	}

	@Override
	protected void onDraw(Canvas canvas) {
		super.onDraw(canvas);
		int tempX = 0, tempY = 0, X1 = 50, Y1 = 250, X_ = 0, Y_ = 150, X2 = 50, Y2
= 300, X__ = 0, Y__ = 500, X3 = 50, Y3 = 300;
		Paint mPaintone = new Paint();
		mPaintone.setColor(Color.BLACK);
		Random randomInt = new Random();
		int rndX = 0, rndY = 0;
		int arr1[] = new int[30];
		int arr1_[] = new int[30];
		int arr2[] = new int[30];
		int arr2_[] = new int[30];
		int arr3[] = new int[30];
		int arr3_[] = new int[30];
		int index = 1;
		int index_ = 1;
		int index__ = 1;
		for (int i = 1; i < 10 * 50; i = i + 50) {
			rndX = 20 + randomInt.nextInt(85);
			rndY = i + randomInt.nextInt(100);

			if (rndY < 200) {
				tempX = rndX;
				tempY = rndY;

				if (tempY > Y_) {
					X_ = rndX;
					Y_ = rndY;
				}
				arr2[index_] = rndX;
```

```
                        arr2_[index_] = rndY;
                        index_++;
                        mPaintone.setColor(Color.DKGRAY);
                } else if (rndY > 200 && rndY < 400) {

                        tempX = rndX;
                        tempY = rndY;

                        if (tempX > X1) {
                                X1 = rndX;
                                Y1 = rndY;
                        }
                        if (tempY < Y2) {
                                X2 = rndX;
                                Y2 = rndY;
                        }
                        if (tempY > Y3) {
                                X3 = rndX;
                                Y3 = rndY;
                        }
                        arr1[index] = rndX;
                        arr1_[index] = rndY;
                        index++;
                        mPaintone.setColor(Color.GRAY);
                }

                else {
                        tempX = rndX;
                        tempY = rndY;

                        if (tempY < Y__) {
                                X__ = rndX;
                                Y__ = rndY;
                        }
                        arr3[index__] = rndX;
                        arr3_[index__] = rndY;
                        index__++;

                        mPaintone.setColor(Color.BLACK);
                }
                canvas.drawCircle(rndX, 20 + rndY, 15, mPaintone);
                canvas.drawCircle(rndX + 60, 45 + rndY, 15, mPaintone);
                canvas.drawCircle(rndX + 160, 30 + rndY, 15, mPaintone);

                mPaintone.setColor(Color.RED);
                canvas.drawCircle(400, 250, 20, mPaintone);
        }

/*
        mPaintone.setColor(Color.BLUE);
        canvas.drawLine(X1 + 160, 30 + Y1, 400, 250, mPaintone);
        canvas.drawLine(X_ + 160, 30 + Y_, X2 + 160, 30 + Y2, mPaintone);
        canvas.drawLine(X__ + 160, 30 + Y__, X3 + 160, 30 + Y3, mPaintone);
        for (int i = 1; i < index; i++) {
                canvas.drawLine(arr1[i], arr1_[i] + 20, X1 + 160, 30 + Y1,
                                mPaintone);
                canvas.drawLine(arr1[i] + 60, 45 + arr1_[i], X1 + 160, 30 + Y1,
                                mPaintone);
                canvas.drawLine(arr1[i] + 160, 30 + arr1_[i], X1 + 160, 30 + Y1,
                                mPaintone);
        }
        for (int i = 1; i < index; i++) {
```

```
                    for (int j = 1; j < index; j++) {
                    canvas.drawLine(arr1[i], arr1_[i] + 20, arr1[j] + 160, 30 +
arr1_[j],
                                  mPaintone);
                    canvas.drawLine(arr1[i] + 60, 45 + arr1_[i], arr1[j] + 160, 30 +
arr1_[j],
                                  mPaintone);
                    canvas.drawLine(arr1[i] + 160, 30 + arr1_[i], arr1[j] + 160, 30 +
arr1_[j],
                                  mPaintone);
                    }
             }
             for (int i = 1; i < index_; i++) {
                    canvas.drawLine(arr2[i], arr2_[i] + 20, X2 + 160, 30 + Y2,
                                  mPaintone);
                    canvas.drawLine(arr2[i] + 60, 45 + arr2_[i], X2 + 160, 30 + Y2,
                                  mPaintone);
                    canvas.drawLine(arr2[i] + 160, 30 + arr2_[i], X2 + 160, 30 + Y2,
                                  mPaintone);
             }
             for (int i = 1; i < index_; i++) {
                    for (int j = 1; j < index_; j++) {
                           canvas.drawLine(arr2[i], arr2_[i] + 20, arr2[j] + 160, 30 +
arr2_[j],
                                        mPaintone);
                           canvas.drawLine(arr2[i] + 60, 45 + arr2_[i], arr2[j]  + 160,
30 + arr2_[j],
                                        mPaintone);
                           canvas.drawLine(arr2[i] + 160, 30 + arr2_[i], arr2[j]  + 160,
                                        30 + arr2_[j], mPaintone);
                    }
             }
             for (int i = 1; i < index__; i++) {
                    canvas.drawLine(arr3[i], arr3_[i] + 20, X3 + 160, 30 + Y3,
                                  mPaintone);
                    canvas.drawLine(arr3[i] + 60, 45 + arr3_[i], X3 + 160, 30 + Y3,
                                  mPaintone);
                    canvas.drawLine(arr3[i] + 160, 30 + arr3_[i], X3 + 160, 30 + Y3,
                                  mPaintone);
             }

             for (int i = 1; i < index__ ; i++) {
                    for (int j = 1; j < index__; j++) {
                           canvas.drawLine(arr3[i], arr3_[i] + 20, arr3[j] + 160,
                                        30 + arr3_[j], mPaintone);
                           canvas.drawLine(arr3[i] + 60, 45 + arr3_[i], arr3[j] + 160,
                                        30 + arr3_[j], mPaintone);
                           canvas.drawLine(arr3[i] + 160, 30 + arr3_[i], arr3[j] + 160,
                                        30 + arr3_[j], mPaintone);
                    }
             }
             */

      }

      private void setLayerToSW(View v) {
             if (!v.isInEditMode() && Build.VERSION.SDK_INT >= 11) {
                    setLayerType(View.LAYER_TYPE_SOFTWARE, null);
             }
      }
```

```
	private void setLayerToHW(View v) {
		if (!v.isInEditMode() && Build.VERSION.SDK_INT >= 11) {
			setLayerType(View.LAYER_TYPE_HARDWARE, null);
		}
	}

	private void initialize() {
		setLayerToSW(this);
	}
}
```

Appendix A.2: Script for IChat Application

chartScreenActivity.java

```
package se.sensiblethings.scalablemobility;

import java.util.ArrayList;
import android.os.Bundle;
import android.app.Activity;
import android.app.Application;
import android.content.Intent;
import android.text.method.ScrollingMovementMethod;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import se.sensiblethings.disseminationlayer.communication.Communication;
import se.sensiblethings.disseminationlayer.communication.Message;
import se.sensiblethings.disseminationlayer.lookupservice.LookupService;
import
se.sensiblethings.disseminationlayer.lookupservice.distributed.messages.unica
st.JoinMessage;
import se.sensiblethings.interfacelayer.SensibleThingsListener;
import se.sensiblethings.interfacelayer.SensibleThingsNode;
import se.sensiblethings.interfacelayer.SensibleThingsPlatform;

public class ChatscreenActivity extends Activity implements
            SensibleThingsListener {

      // The Platform API

      TextView tv, textView, tv_;
      Button btn;
      long l;
      String temp;
      ArrayList activenodes;
      String[] a;
      TcpCommunication tcpcom;
      Message message;
      String nodes = "%";
      SensibleThingsPlatform sensibleThingsPlatform = new
SensibleThingsPlatform(
                  this);
      Communication communication =
sensibleThingsPlatform.getDisseminationCore()
                  .getCommunication();
      SensibleThingsNode sensiblethingsnode;

      // thePlatform application = new thePlatform();

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.chatscreen);
            tv = (TextView) findViewById(R.id.editText1);
```

```
            tv.setMovementMethod(ScrollingMovementMethod.getInstance());
            tv_ = (TextView) findViewById(R.id.editText2);
            tv_.setMovementMethod(ScrollingMovementMethod.getInstance());
            textView = (TextView) findViewById(R.id.editText3);
            btn = (Button) findViewById(R.id.button1);
            l = System.currentTimeMillis();

            activenodes = new ArrayList();
            // tcpcom = new TcpCommunication();

            sensibleThingsPlatform.register("miun.se/node_two");

            // sensibleThingsPlatform.register("miun.se/node_two");
            // sensibleThingsPlatform.run();
            // this.runOnUiThread((Runnable) this);
            // tcpcom.createLocalNode();
            // tcpcom.createSensibleThingsNode("172.20.10.2", 4040);
            // message = new Message(tcpcom.getLocalSensibleThingsNode(),
            // tcpcom.createSensibleThingsNode("172.20.10.3", 4040));

            btn.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View arg0) {
                        if (textView.getText().toString().length() != 0) {
                              temp = System.currentTimeMillis() + "|
[Unresolved] |"
                                          + textView.getText();
                              // tcpcom.sendMessage(temp);
                              // tv.append("\n" + temp);

                              // a = temp.split("|");
                              // tv_.append("\n" + a[2]);


      sensibleThingsPlatform.resolve("miun.se/node_two");

      sensibleThingsPlatform.resolve("miun.se/node_one");

                              // tcpcom.getLocalSensibleThingsNode();

                              // tcpcom.sendMessage(message);

                        } else
                              Toast.makeText(ChatscreenActivity.this,
                                          "You need to Enter Test Text!",
Toast.LENGTH_LONG)
                                          .show();

                  }
            });

      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is
present.
            getMenuInflater().inflate(R.menu.chatscreen, menu);
            return true;
      }
```

```
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		Intent intent;
		switch (item.getItemId()) {
		case R.id.action_state:

			intent = new Intent(ChatscreenActivity.this,
					ChatscreenActivity.class);
			startActivity(intent);
			return true;
		default:
			return super.onOptionsItemSelected(item);
		}
	}

	@Override
	protected void onResume() {
		super.onResume();

		// Create the platform itself
		if (sensibleThingsPlatform == null) {
			sensibleThingsPlatform = new SensibleThingsPlatform(this);
		}
	}

	// @Override
	public void onClick(View arg0) {

	}

	@Override
	public void getResponse(final String uci, final String value,
			final SensibleThingsNode source) {
		runOnUiThread(new Runnable() {
			@Override
			public void run() {
				tv.append("\n[GetResponse] " + uci + ": " + value +
"\n"
						+ source);
				tv_.append(uci + "\n");

			}
		});

	}

	@Override
	public void resolveResponse(final String uci, final SensibleThingsNode
node) {
		runOnUiThread(new Runnable() {
			@Override
			public void run() {
				tv.append("\n[ResolveResponse] " + uci + ": " +
node.toString()
						+ "\n");
				tv.append("\nTesting Get\n");
				sensibleThingsPlatform.get(uci, node);

			}
		});
```

```
	}

	@Override
	public void getEvent(SensibleThingsNode source, String uci) {
		// TODO Auto-generated method stub

		// Log.i("GET EVENT- IChat", uci);
		sensibleThingsPlatform.get(uci, source);
		// sensibleThingsPlatform.notify(source, uci, "Here is message
from ");
		sensibleThingsPlatform.notify(source, uci, textView.getText()
				.toString());

	}

	@Override
	public void setEvent(SensibleThingsNode fromNode, String uci, String
value) {
		// Log.i("SET EVENT- IChat", value);
		sensibleThingsPlatform.set(uci, value, fromNode);
		// sensibleThingsPlatform.notifyAll();

	}


}
```

fingerTable.java

```
package se.sensiblethings.scalablemobility;

public class FingerTable implements Runnable {

	String rawText = "%";
	String uci;
	String str;

	FingerTable(String uci, String ip_port) {
		rawText = rawText + uci + ip_port + "%";
	}

	@Override
	public void run() {

	}

}
```

tcpCommunication.java

```
/*
 * Copyright 2013 The SensibleThings Consortium
 * This file is part of The SensibleThings Platform.
 *
```

```
 * The SensibleThings Platform is free software: you can redistribute it
and/or modify
 * it under the terms of the GNU Lesser General Public License as published
by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * The SensibleThings Platform is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with The SensibleThings Platform.  If not, see
<http://www.gnu.org/licenses/>.
 */

package se.sensiblethings.scalablemobility;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Inet6Address;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Enumeration;

import android.util.Log;

import se.sensiblethings.disseminationlayer.communication.Communication;
import
se.sensiblethings.disseminationlayer.communication.DestinationNotReachableExc
eption;
import se.sensiblethings.disseminationlayer.communication.Message;
import se.sensiblethings.disseminationlayer.communication.MessageSerializer;
import
se.sensiblethings.disseminationlayer.communication.serializer.ObjectSerialize
r;
import se.sensiblethings.interfacelayer.SensibleThingsNode;

public class TcpCommunication extends Communication implements Runnable {

	private MessageSerializer messageSerializer = new ObjectSerializer();

	private ServerSocket ss;
	private int communicationPort = 0;
	public static int initCommunicationPort = 4040;

	private SensibleThingsNode localSensibleThingsNode = null;

	private boolean runCommunication = true;

	public TcpCommunication() {
		try {


			this.ss = new ServerSocket(initCommunicationPort);
			communicationPort = ss.getLocalPort();
```

```
                this.localSensibleThingsNode = createLocalNode();

                //Start the Listener!
                Thread t = new Thread(this);
                t.start();

            } catch (Exception e){
                e.printStackTrace();
            }
        }

        @Override
        public void shutdown() {
            try {
                runCommunication = false;
                ss.close();
            } catch (Exception e) {
                //e.printStackTrace();
            }
        }


        @Override
        public void sendMessage(Message message) throws
DestinationNotReachableException {
            try {

                String toHost = message.getToNode().toString();
                String[] split = toHost.split(":");
                String toIp = split[0];
                int toPort = Integer.parseInt(split[1]);

                //System.out.println("ToHost: " + toHost);
                Log.i("ToHostttttttttt", toHost);
                Socket s = new Socket(toIp, toPort);

                byte[] data = messageSerializer.serializeMessage(message);

                OutputStream os = s.getOutputStream();
                os.write(data);

                os.flush();
                os.close();
                s.close();

            } catch (IOException e) {
                e.printStackTrace();
                throw new DestinationNotReachableException(e.getMessage());
            }
        }

        public void run() {
            while (runCommunication) {
            try {

                 final Socket s = ss.accept();


                //Thread t = new Thread(new Runnable() {
                       //    public void run() {
```

```
                                    handleConnection(s);

                        //    }

                        //});
                        //t.start();

                } catch (IOException e) {

                        e.printStackTrace();
                }
          }
        }


        private void handleConnection(Socket s) {
                try {

                        byte[] buffer = new byte[1048576];

                        InputStream is = s.getInputStream();
                        is.read(buffer);

                        String stringRepresentation = new String(buffer);

                      Message message =
messageSerializer.deserializeMessage(buffer);

                        //Send the message to the "PostOffice"
                        dispatchMessageToPostOffice(message);

                        is.close();

                        s.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        @Override
        public SensibleThingsNode getLocalSensibleThingsNode() {
                return localSensibleThingsNode;
        }

        //BIG workaround because Linux is stupid...
        private static InetAddress localAddress = null;
        SensibleThingsNode createLocalNode() {
                try {
                        Runnable r = new Runnable() {
                                public void run() {
                                        try {
                                                //Workaround because Linux is stupid...
                                          for (Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces(); en.hasMoreElements();) {
                                            NetworkInterface intf = en.nextElement();

                                            for (Enumeration<InetAddress> enumIpAddr
= intf.getInetAddresses(); enumIpAddr.hasMoreElements();) {
                                                InetAddress inetAddress =
enumIpAddr.nextElement();
```

```
                                        if (!inetAddress.isLoopbackAddress()
&& !inetAddress.isLinkLocalAddress()) {
                                          //if(!(inetAddress instanceof
Inet6Address)){ //Remove this line for IPV6 compatability
                                                localAddress = inetAddress;
                                          //}
                                        }
                                    }
                                }
                            catch (Exception e) {
                                e.printStackTrace();
                                }
                        }
                    };
                    Thread t = new Thread(r);
                t.start();

                //while(localAddress == null){
                     // Thread.sleep(200);
                //}
                Log.i("LOCAL ADDRESS",
localAddress.getHostAddress().toString());
                return new SensibleThingsNode(localAddress.getHostAddress(),
communicationPort);

            } catch (Exception e) {
                    //e.printStackTrace();
                    return new SensibleThingsNode("172.20.10.2",
communicationPort);
            }
      }
}
```

theplateForm.java

```
package se.sensiblethings.scalablemobility;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import se.sensiblethings.interfacelayer.SensibleThingsListener;
import se.sensiblethings.interfacelayer.SensibleThingsNode;
import se.sensiblethings.interfacelayer.SensibleThingsPlatform;

public class thePlatform implements SensibleThingsListener {

      // SensibleThings Platform Application Interface
      SensibleThingsPlatform platform;
      String value = "1337";

      public thePlatform() {

            // Create the platform itself with a SensibleThingsListener
            platform = new SensibleThingsPlatform(this);

      }

      public void run() {
            try {
                  // platform.notify();
```

```
                  System.out.println("SensibleThings running");

                  // System.out.println("\nPress any key to shutdown");
                  // BufferedReader in = new BufferedReader(new
InputStreamReader(
                  // System.in));
                  // in.readLine();

                  // Shutdown all background tasks
                  // platform.shutdown();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      @Override
      public void resolveResponse(String uci, SensibleThingsNode node) {

            System.out.println("[ResolveResponse] " + uci + ": " + node);

            platform.get(uci, node);

            // SET, will try to set the value at the source (set an actuator)
            // It will trigger a SetEvent on the source node.

            platform.set(uci, value, node);

      }

      @Override
      public void getResponse(String uci, String value,
                  SensibleThingsNode fromNode) {
            // This is called as a response the get function call
            System.out.println("[GetResponse] " + uci + " : " + value + " : "
                        + fromNode);

      }

      @Override
      public void getEvent(SensibleThingsNode source, String uci) {
            System.out.println("[GetEvent] " + source + " : " + uci);

            // We send back our simulated sensor value
            platform.notify(source, uci, value);
      }

      @Override
      public void setEvent(SensibleThingsNode source, String uci, String
value) {
            System.out
                        .println("[SetEvent] " + source + " : " + uci + " : "
+ value);

            // In this example we set out simulated sensor value
            this.value = value;

      }

}
```

Appendix A.3: Script for Mobile DXCP

Aknowledge.java

```
package se.sensiblethings.addinlayer.extentions.mobiledcxp;

import se.sensiblethings.disseminationlayer.communication.Message;
import se.sensiblethings.interfacelayer.SensibleThingsNode;

public class Acknowledge extends Message {

	/**
	 *
	 */
	private static final long serialVersionUID = -7587921529782114502L;
	public SensibleThingsNode fromNode, toNode;
	public float messageID;

	public Acknowledge(SensibleThingsNode fromNode, SensibleThingsNode
toNode) {
		super(fromNode, toNode);
		this.fromNode = fromNode;
		this.toNode = toNode;
		// this.messageID = id;
	}

	public SensibleThingsNode getFromNode() {
		return fromNode;
	}

	public SensibleThingsNode getToNode() {
		return toNode;
	}

	public String getType() {
		return getClass().getName();
	}

	public String toString() {
		return getType() + "from: (" + fromNode + ") - to: (" + toNode +
")"
				+ "#" + messageID;
	}

	// Used for statistics in the simulator
	public int getDataAmount() {
		// 2 x ip-address + 1 type
		return 4 + 4 + 1;
	}

}
```

DestinationUnreachable.java

```
package se.sensiblethings.addinlayer.extentions.mobiledcxp;

import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import java.net.Inet6Address;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Enumeration;

import se.sensiblethings.disseminationlayer.communication.Communication;
import
se.sensiblethings.disseminationlayer.communication.DestinationNotReachableExc
eption;
import se.sensiblethings.disseminationlayer.communication.Message;
import se.sensiblethings.disseminationlayer.communication.MessageSerializer;
import
se.sensiblethings.disseminationlayer.communication.serializer.ObjectSerialize
r;
import se.sensiblethings.interfacelayer.SensibleThingsNode;

public class DestinationUnreachableTrial extends Communication implements
            Runnable {

      // Retransmission period in seconds
      public int retryPeriod = 4;
      public boolean Active = false;// A node could be disconnected for a
very
                                                      // short period of time
(active nodes) or
                                                      // could stay
disconnected for a long period
                                                      // of time.

      private MessageSerializer messageSerializer = new ObjectSerializer();

      private ServerSocket ss;
      private int communicationPort = 0;
      public static int initCommunicationPort = 0;

      private SensibleThingsNode localSensibleThingsNode = null;

      private boolean runCommunication = true;

      public DestinationUnreachableTrial() {
            try {

                  this.ss = new ServerSocket(initCommunicationPort);
                  communicationPort = ss.getLocalPort();

                  this.localSensibleThingsNode = createLocalNode();

                  // Start the Listener!
                  Thread t = new Thread(this);
                  t.start();

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      @Override
      public void shutdown() {
```

```
		try {
			runCommunication = false;
			ss.close();
		} catch (Exception e) {
			// e.printStackTrace();
		}
	}

	@Override
	public void sendMessage(Message message)
			throws DestinationNotReachableException {

		while (retryPeriod > 0) {
			try {

				String toHost = message.getToNode().toString();
				String[] split = toHost.split(":");
				String toIp = split[0];
				int toPort = Integer.parseInt(split[1]);

				// System.out.println("ToHost: " + toHost);
				Socket s = new Socket(toIp, toPort);

				byte[] data =
messageSerializer.serializeMessage(message);
				OutputStream os = s.getOutputStream();
				os.write(data);

				os.flush();
				os.close();
				s.close();

			} catch (IOException e) {
				e.printStackTrace();
				throw new
DestinationNotReachableException(e.getMessage());
			}
			try {
				this.wait(1000);
				retryPeriod--;
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		}
	}

	public int getretryPeriod() {
		return this.retryPeriod;
	}

	public void setretryPeriod(int time) {
		this.retryPeriod = time;
	}

	public void run() {
		while (runCommunication) {
			try {
				final Socket s = ss.accept();
```

```
                        Thread t = new Thread(new Runnable() {
                              public void run() {
                                    handleConnection(s);
                              }

                        });
                        t.start();

                  } catch (IOException e) {
                        // throw new
DestinationNotReachableException(e.getMessage());
                        // e.printStackTrace();
                  }
            }
      }

      private void handleConnection(Socket s) {
            try {

                  byte[] buffer = new byte[1048576];

                  InputStream is = s.getInputStream();
                  is.read(buffer);

                  // String stringRepresentation = new String(buffer);

                  Message message =
messageSerializer.deserializeMessage(buffer);

                  // Send the message to the "PostOffice"
                  dispatchMessageToPostOffice(message);

                  is.close();
                  s.close();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      @Override
      public SensibleThingsNode getLocalSensibleThingsNode() {
            return localSensibleThingsNode;
      }

      // BIG workaround because Linux is stupid...
      private static InetAddress localAddress = null;

      private SensibleThingsNode createLocalNode() {
            try {
                  Runnable r = new Runnable() {
                        public void run() {
                              try {
                                    // Workaround because Linux is stupid...
                                    for (Enumeration<NetworkInterface> en =
NetworkInterface
                                                .getNetworkInterfaces();
en.hasMoreElements();) {
                                          NetworkInterface intf =
en.nextElement();
                                          for (Enumeration<InetAddress>
enumIpAddr = intf
```

```
                                                      .getInetAddresses();
enumIpAddr
                                                      .hasMoreElements();) {
                                                InetAddress inetAddress =
enumIpAddr
                                                            .nextElement();
                                                if
(!inetAddress.isLoopbackAddress()
                                                            &&
!inetAddress.isLinkLocalAddress()) {
                                                      if (!(inetAddress
instanceof Inet6Address)) { // Remove

                                                // this

                                                // line

                                                // for

                                                // IPV6

                                                // compatability
                                                            localAddress =
inetAddress;
                                                      }
                                                }
                                          }
                                    }
                              } catch (Exception e) {
                                    // e.printStackTrace();
                              }
                        }
                  };
                  Thread t = new Thread(r);
                  t.start();

                  while (localAddress == null) {
                        Thread.sleep(200);
                  }

                  return new
SensibleThingsNode(localAddress.getHostAddress(),
                              communicationPort);
            } catch (Exception e) {
                  // e.printStackTrace();
                  return new SensibleThingsNode("127.0.0.1",
communicationPort);
            }
      }
}
```

BIBLIOGRAPHY

1. NetGear Inc. http://documentation.netgear.com/reference/ita/wireless/WirelessNetworkingBasics-2-1.html published 2005

2. Dr. Pekka Nikander Ericsson Research, Finland Evolution of Networking: Current Problems and Future Directions
3. Host Identity Protocol (HIP): Connectivity, Mobility, Multi-Homing, Security, and Privacy over IPv4 and IPv6 Networks Pekka Nikander, Andrei Gurtov, and Thomas R. Henderson
4. http://www.ietf.org/rfc/rfc4423.txt page 1
5. http://www.ietf.org/rfc/rfc4423.txt , page 16.
6. http://tools.ietf.org/id/draft-ietf-lisp-24.txt , Page 5

7. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_11-1/111_lisp.html
8. http://lisp4.cisco.com/lisp_over.html
9. Handover Management for Mobile Nodes in IPv6 Networks Page 6
10. http://www.ipv6.com/articles/mobile/Mobile-IPv6.htm
11. http://www.ietf.org/rfc/rfc5380.txt http://tools.ietf.org/id/draft-ietf-lisp-24.txt , Page 5
12. http://www.ietf.org/rfc/rfc5380.txt
13. http://www.ietf.org/rfc/rfc4068.txt
14. http://www.ietf.org/rfc/rfc4068.txt page 4
15. IKE2 Mobility and Multihoming Protocol (MOBIKE) http://tools.ietf.org/html/rfc4555

16. A Scheme to Reduce Packet Loss during PMIPv6 Handover considering Authentication∗Seonggeun Ryu, Gye-Young Kim, Byunggi Kim, and Youngsong Mun School of Computing
17. Soongsil University, Korea

18. SensibleThings – an Internet of Things Platformfor Scalable and Decentralized Context Sharing and Control Page 2
19. Supporting global context Dissemination
20. J. Rosenberg, “SIMPLE made simple: An overview of the IETF specifications for instant messaging and presence using the session initiation protocol (SIP),” IETF, Internet-Draft, 2008.
21. http://www.tml.tkk.fi/Studies/T-110.551/2003/papers/8.pdf , page 9