

# **Shim6 Assisted Mobility Scheme (SAMS)**



**By**

**Muhammad Mudassir Feroz  
2009-NUST-MS-CCS-05**

**Supervisor  
Dr. Adnan Khalid Kiani**

This thesis is submitted in partial fulfillment of the requirements  
for the degree of Masters of Science

School of Electrical Engineering and Computer Science (SEECS)  
National University of Sciences and Technology (NUST)  
Islamabad, Pakistan.

December 2012

## APPROVAL

It is certified that the contents of thesis document titled, “Shim6 Assisted Mobility Scheme” submitted by Mr. Muhammad Mudassir Feroz have been found satisfactory for the requirement of degree.

Advisor: Dr. Adnan Khalid Kiani

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee member 1: Dr. Zawwar Hussain

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee member 1: Mr. Farooq Cheema

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

Committee member 1: Mr. Ammar Karim

Signature: \_\_\_\_\_

Date: \_\_\_\_\_

TO

MY LOVING FAMILY

## **CERTIFICATE OF ORIGINALITY**

I declare that the research work titled “**Shim6 Assisted Mobility Scheme**” is my own work and to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project’s design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Muhammad Mudassir Feroz

Signature: \_\_\_\_\_

## **ACKNOWLEDGMENTS**

I am grateful to Almighty ALLAH who gave me strength, courage and determination to complete this thesis. I owe my deepest gratitude to my thesis advisor, Dr. Adnan Khalid Kiani, for his kind attention and guidance throughout my research work. I am, also thankful to my worthy Committee members, Dr. Zawwar Hussain, Mr. Farooq Cheema and Mr. Ammar Karim for their support and becoming a part this work. I am, also, thankful to professional members of CERN lab and researchers of LinShim6 for their technical guidance and support.

# TABLE OF CONTENTS

<b>LIST OF TABLES</b> .....	<b>viii</b>
<b>LIST OF FIGURES</b> .....	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b> .....	<b>x</b>
<b>ABSTRACT</b> .....	<b>xi</b>
<b>CHAPTER 1</b> .....	<b>1</b>
<b>INTRODUCTION</b> .....	<b>1</b>
1.1. Motivation.....	1
1.2. Thesis Contributions.....	2
1.3. Thesis Organization .....	3
<b>CHAPTER 2</b> .....	<b>4</b>
<b>LITERATURE SURVEY</b> .....	<b>4</b>
2.1. Mobile Internet Protocol version 6 (MIPv6) .....	4
2.2. Site Multihoming by IPv6 Intermediation (Shim6).....	6
2.3. Related work .....	8
<b>CHAPTER 3</b> .....	<b>12</b>
<b>SHIM6 ASSISTED MOBILITY SCHEME (SAMS)</b> .....	<b>12</b>
3.1. Proposed Architecture.....	12
3.2. Algorithm .....	15
<b>CHAPTER 4</b> .....	<b>19</b>
<b>IMPLEMENTATION SETUP</b> .....	<b>19</b>
4.1. Kernel configuration.....	19
4.1.1 Patching an additional module.....	20
4.1.2 Compiling the kernel source.....	21
4.2. User space configuration.....	23
4.3. Test bed Setup .....	24
4.4. Running the Shim6 enabled hosts .....	26
4.5. Shim6 Context establishment .....	29
4.5.1 Shim6 context before switch .....	31
4.5.2 Shim6 context after switch.....	32

<b>CHAPTER 5</b> .....	<b>36</b>
<b>RESULTS AND DISCUSSION</b> .....	<b>36</b>
5.1. Single switching and multiple switching .....	36
5.2. Jitter.....	38
5.3. Bandwidth .....	39
5.4. Data Transferred.....	41
5.5. Packet loss .....	42
5.6. Switching delay .....	44
<b>CHAPTER 6</b> .....	<b>48</b>
<b>CONCLUSION &amp; FUTURE WORK</b> .....	<b>48</b>

## LIST OF TABLES

Table I: SAMS Testbed specifications .....	25
Table II: Shim6 console commands .....	28
Table III: Traffic used for experiments.....	37
Table IV: Switching delay comparison with existing techniques.....	45
Table V: Implementation results (traffic 1) .....	46
Table VI: Implementation results (traffic 2) .....	46



## LIST OF FIGURES

Figure 1: MIPv6 Signaling architecture .....	5
Figure 2: Shim6 context establishment.....	7
Figure 3: Shim6 layer architecture .....	7
Figure 4: Shim6 assisted intelligent switching mechanism.....	14
Figure 5: Signaling in REAP .....	17
Figure 6: Signaling in proposed mechanism.....	18
Figure 7: Experimental topology for SAMS.....	26
Figure 8: jperf-2.0.2, Network traffic generator and measurement tool.....	30
Figure 9: Shim6 console showing Shim6 context before switching.....	34
Figure 10: Shim6 console showing Shim6 context after switching .....	35
Figure 11: Jitter comparison of single switch between REAP and SAMS .....	38
Figure 12: Jitter comparison of multiple switches between REAP and SAMS .....	39
Figure 13: Throughput comparison in single switch between REAP and SAMS .....	40
Figure 14: Throughput comparison in multiple switches between REAP and SAMS .....	40
Figure 15: Data transferred comparison in single switch between REAP and SAMS.....	41
Figure 16: Data transferred comparison in multiple switches between REAP and SAMS....	42
Figure 17: Packet loss comparison in single switch between REAP and SAMS .....	43
Figure 18: Packet loss comparison in multiple switches between REAP and SAMS.....	44
Figure 19: switching delay comparison .....	45

## LIST OF ABBREVIATIONS

SHIM6	Site Multihoming by IPv6 intermediation
MIPv6	Mobile Internet Protocol version 6
FMIPv6	Fast Mobile Internet Protocol version 6
SEMO6	Seamless Mobility using Shim6
SAMS	Shim6 Assisted Mobility Scheme
MN	Mobile Node
IETF	Internet Engineering Task Force
CGA	Cryptographically Generated Addresses
DHCP	Dynamic Host Configuration Protocol
HBA	Hash Based Addresses
CN	Correspondent Node
HoA	Home Address
NAT	Network Address Translation
REAP	Reachability Protocol
SEND	Secure Neighbor Discovery
ULID	Upper-Level Identifier
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
RTT	Round Trip Time
NAT	Network Address Translation
RFC	Request For Comments

## ABSTRACT

Devices with multiple interfaces are the future of mobile internet. Site Multi-homing by IPv6 Intermediation-Shim6 is a proposal presented in IETF to provide multi-homing support in IPv6 based networks. Although initially it was intended for static networks but recently it has been tested to provide end host mobility. By default, Failure detection and recovery in Shim6 is performed through REACHability (REAP) Protocol. When failures occur, REAP is activated and the communication is switched from the failed path to a different active path. Existing protocol for mobility management in IPv6 networks is the MIPv6. Due to the inherent flaws in MIPv6, some new protocols/techniques have been tested which improve the overall mobility performance in IPv6 networks. In this research work, we presented a mechanism for providing a “make before break” switching service. We call our proposed mechanism “Shim6 Assisted Mobility Scheme (SAMS)” as it provides both mobility and multi-homing support. We implemented LinShim6 with our proposed intelligent switching in wireless environment and presented the results. The proposed intelligent module anticipates the switch by continuously monitoring the QoS parameters for specific period of time and helps in decision making. Further we made the performance comparison between REAP based switching and the proposed switching mechanism. The comparison was made in terms of switching delay, packet loss, jitter, throughput and data transferred. Through experiments we have shown that the switching delay reduced by 20% to 20.286 ms and packet loss during switching reduces from 24.71% to 1.72%.

**INTRODUCTION**

**1.1. Motivation**

With the growth of internet the end users are rapidly increasing. This leads to a tremendous increase in internet traffic over the deployed networks. There are more constraints for real time traffic which should reach its destination without any unnecessary delay or interruption. End users carry mobile devices which are installed with heavy and advanced software applications. Those applications exchange data with other users continuously while on the move. Devices used for such applications include laptops, palm devices, smart phones and other gadgets which directly connect to internet through some interface. This connectivity is continued even when the user is traveling and keeps changing its location as well as the network to which it is attached. These devices usually are embedded with multiple interfaces. Each interface can be utilized by a single connection.

As a mobile user travels to different locations, the connection to the network experiences pauses and interruptions. In real time applications, these issues matter a lot. This is due to the signal strength or in extreme cases, due to the shifting onto other networks. If this shifting is seamless, the user does not feel any interruption in the service. But if this is long enough to be noticed by the user then this is very unhealthy for the service provider. This phenomenon of shifting onto other available networks is known as “*Mobility*”.

Mobility is the enabling of the services for a mobile device while moving from one network to another. This process is known as handover. There are two types of handovers, *Horizontal* and *Vertical*. Horizontal handover occurs when end user shifts to another subnet of

the same network access technology. Vertical handover occurs when end user shifts to another network access technology. To tackle this issue of handovers in IP based networks, MIPv6 [1] is the most popular protocol. However, MIPv6 still faces delays and the handover phase experiences noticeable interruptions in the services. Recent research has shown that in MIPv6 a single handover takes around 5 seconds to complete. This is too long if communication is happening in real time. Main reason behind this is the availability of just one point of attachment to the network. It follows the “*Break before make*” approach. Connection to the original network is disconnected and then end user is connected to new network. To overcome this issue many proposals such as FMIPv6 [4], HIP [5] etc are presented. But these protocols have inherent flaws. For instance HIP requires an additional user space to be deployed and FMIPv6 does not provide better results than MIPv6. Its handover delay is almost equivalent to that of MIPv6. So considering the flaws inherited in the above mentioned solutions we need to overcome the highlighted flaws and have more efficient and reliable solution in order to satisfy the intended internet users.

## **1.2. Thesis Contributions**

In this contribution, we have shown that efficient mobility in heterogeneous IPv6 environment can be provided by employing multi-homing techniques such as shim6. Furthermore, default failure detection and recovery mechanism in shim6 known as REAP is too slow to provide support for real time mobile applications. In this work, we have proposed a quick failure detection and recovery mechanism in mobile multi-homed environment. It is called Shim6 Assisted Mobility Scheme (SAMS). We have implemented a testbed in order to validate the working of Shim6 and above that we have performed the implementation of our proposed intelligent switching mechanism. SAMS significantly reduces the overall switching time through

use of triggers. In this contribution we have performed the comparison of both (REAP and SAMS) techniques considering the QoS parameters.

### **1.3. Thesis Organization**

There are five chapters in this thesis which are organized as follows:

In Chapter 2, literature survey based on the introduction of different solutions presented in the field of wireless environment is presented. Apart from history, existing technologies are also discussed, including the most important protocols under research these days, Shim6.

In Chapter 3, a better solution for Shim6 based mobility scheme is presented.

In Chapter 4, methodology to carry out this work is presented along with detailed and step by step implementation of test bed.

In chapter 5, experimental results are presented and explained in detail. Also, comparison with default failure detection and recovery mechanism in Shim6 is also discussed.

In Chapter 6, conclusion of our thesis is given with proposals of some possible extensions to this work. In the end, references are given.

**LITERATURE SURVEY**

In every research work, strong literature survey plays a key role to identify the real problem and also to identify the tips and trick to tackle it. Different topologies present the results that are specific to the implemented topology. We have focused on the topologies that are relevant to our intended protocol and which can be implemented using our proposed mechanism. The following is the literature survey which goes around wireless environments and protocols. Starting with the official standards to recently proposed mechanisms are discussed. This approach is adopted in order to follow the research as it has been done by researchers.

**2.1. Mobile Internet Protocol version 6 (MIPv6)**

Early deployments in internet infrastructure were based on fixed IP addresses. There was no compatibility for devices which keep changing their locations. IPv6 had full support for fixed users. Later when mobile devices and advanced mobile phones were produced in market, there was a need of a protocol which could support the connectivity of users even when they are traveling. Hence, it was Mobile Internet Protocol version-6 (MIPv6) which was deployed to support such feature. The specifications are described in (RFC 3775).

In traditional IP routing, IP addresses represent a topology. Routing mechanisms rely on the assumption that each network node will always have the same point of attachment to the Internet, and that each node's IP address identifies the network link where it is connected. In this routing scheme, if you disconnect a mobile device from the Internet and want to reconnect through a different network, you have to configure the device with a new IP address, and the appropriate net mask and default router. In this scheme, routing protocols have no clue for

delivering packets, because the device's network address doesn't contain the necessary information about the node's current network point of attachment to the Internet. MIPv6 allows a mobile node to transparently maintain connections while moving from one subnet to another. Each device is identified by its home address although it may be connecting to through another network. When connecting through a foreign network, a mobile device sends its location information to a home agent, which intercepts packets, intended for the device and tunnels them to the current location.

MIPv6 adds another entity to the existing network, which is the corresponding node (CN). This node provides mobility operations and keeps record of the current location of mobile node if it is away from its home network. A simple mobile scenario is shown in the following diagram.

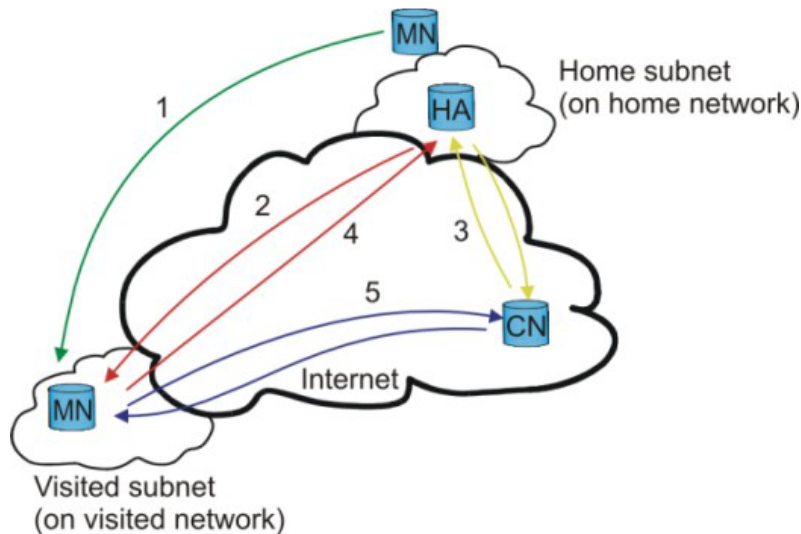


Figure 1: MIPv6 Signaling architecture



## **2.2. Site Multihoming by IPv6 Intermediation (Shim6)**

IETF initiated an effort for providing Multihoming support to IPv6 network known as shim6 protocol. It specifies a layer 3 shim approach and protocol for providing locator agility below the transport protocols, so that Multihoming can be provided for IPv6. (RFC 5533) describes all specifications in detail. Hosts in a site which has multiple provider allocated IPv6 address prefixes, will use the shim6 protocol to setup state with peer hosts, so that the state can later be used to failover to a different locator pair, and should the original one stop working. This approach was basically intended to facilitate the subnet operations. Originally this meant for static networks. However, subsequently due to its efficient nature researchers have used it for the mobility of mobile users. This protocol can be used for multi-homed sites (providing Multihoming) as well as multiple interfaced end hosts (providing mobility).

In this approach, a mobile user initiates communication in a normal way. Then according to some heuristics he can activate shim6 signaling. This heuristics can be a number of packets or time stamp. To activate Shim6 context, both hosts should be shim6 enabled. This requires implementation of shim6 on both hosts. Basically, shim6 context establishment is based on four control messages, I1, R1, I2 and R2 respectively. All the information is exchanged in these messages. If both hosts are not shim6 enabled, communication through shim6 context will not take place. In this way initiator realizes that other end is not shim6 enabled. Shim6 allows a host to change locator during an active communication. It is used to ensure redundancy when the current path fails. To achieve this, two functions phases are defined: “Failure Detection” and “Locator Pair exploration”. The “Failure Detection” function is used to detect disconnections in the current path, while the “Locator Pair Exploration” is used to define new valid locator pairs that could be used when the current one fails. The following diagram shows normal context

establishment of two Shim6 enabled hosts.

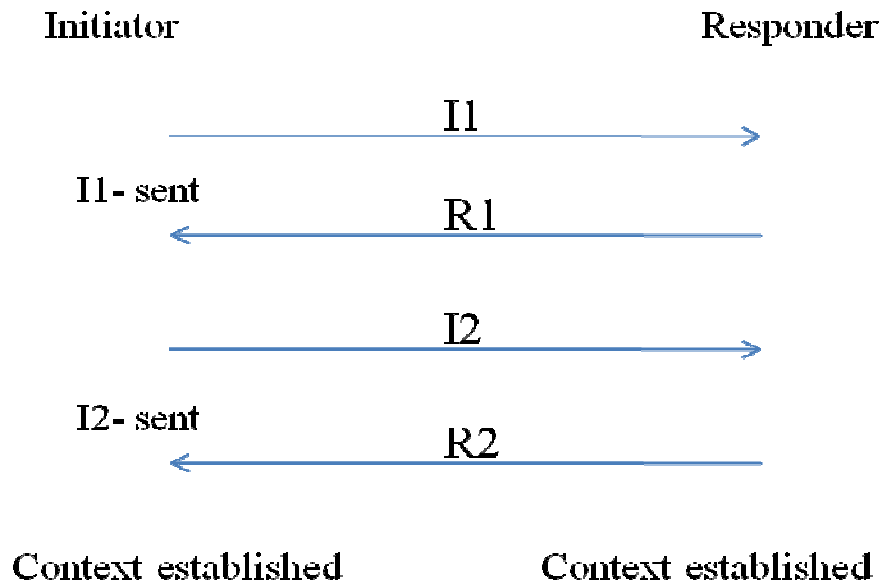


Figure 2: Shim6 context establishment

Following figure shows the location of shim6 sub-layer within internet layer architecture.

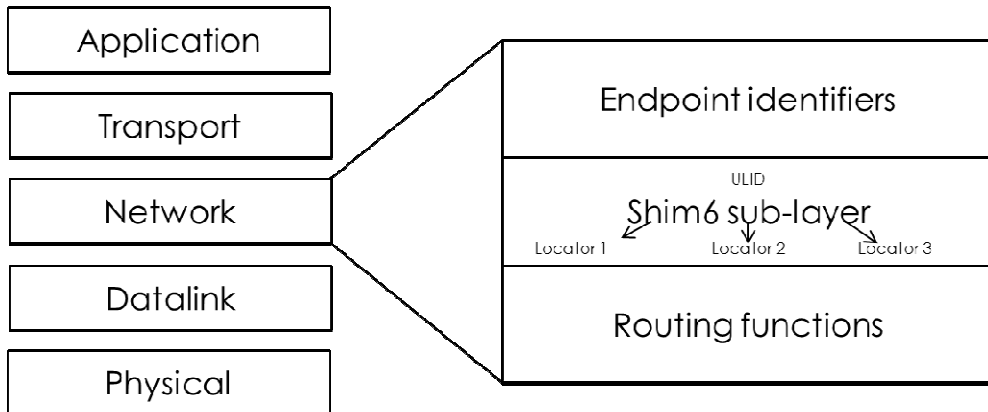


Figure 3: Shim6 layer architecture

Shim6 sub layer within the network layer and differentiates the characteristics of identifiers and locators. For the upper layers it is known by the identifier. While the layers down identify it by locators. Locators are basically used to locate the current existence of mobile user. The

identifier is used to identify the point of attachment to the upper layers. Having all these features, deployment of shim6 does not require any additional things. It is based on IPv6 internet infrastructure.

As packets are passed from the IP Endpoint sub-layer to the IP Routing sub-layer, the endpoint identities are mapped to a current pair of locators. The reverse mapping is applied to incoming packets, where the incoming locator pair is stripped off the packet, and the associated endpoint identity pair is associated with the packet which is then passed to the IP Endpoint sub-layer. De-multiplexing the IP packet to the appropriate transport session is based on the endpoint identities. In this Shim6 approach the endpoint identities and the locators are both IP addresses. The endpoint identities are the initial addresses used between the two hosts. The locators are the set of IP addresses that are associated with endpoint. The intention of this approach is to minimize the amount of change required to support dynamic locator agility in the protocol stack.

### **2.3. Related work**

Mobile users like to stay connected on the go. Internet mobility has thus become a focal point of research over the years. To provide internet mobility support, different protocols have been designed. These protocols support handover between wireless networks. MIPv6 is the standard protocol to provide mobility services in IPv6 networks. There are however some problems when MIPv6 is adopted to support real-time communication. These include high handover latency and packet loss. During handover, there is a period in which the mobile node is unable to send or receive packets due to link-layer switching and IPv6 protocol layer operations. This overall handover latency resulting from baseline MIPv6 procedures, namely movement detection, new care-of address configuration, and binding updates with peer entities, is often unacceptable for real-time application services e.g. video-conferencing, voice-over-IP etc.

MIPv6 movement detection is based on neighbour un-reachability detection and Router discovery. Neighbor un-reachability detection observes the default router and upon receiving no response, it decides to switch to another available router. This availability is based on Router Discovery mechanism. These phases increase the overall switching delay. In addition to high handover latency, MIPv6 doesn't provide multi-homing support to end users, i.e. a user cannot be simultaneously configured with multiple active interfaces and switch between them seamlessly.

In view of the above-mentioned shortcomings associated with MIPv6, some solutions were proposed to provide seamless mobility support. A new fast handover approach, based on Fast Handovers for Mobile IPv6, was proposed in [4]. The work supported seamless movement in between IPv6 domains using IEEE 802.11 network infrastructure. A new low latency handoff method for IEEE 802.11 is proposed where access point beacons are utilized for carrying IPv6 prefix information without altering the Mobile IP or IEEE 802.11 specifications. A WLAN service continuously monitors the radio signal quality of the attached access point and, if necessary, switches to another access point in range. This feature and the elimination of firmware-based active scanning during link-layer handovers reduce the overall link-layer handover delay to about 10% of the previous value. Experiments showed that average handover delay is between 2-4 seconds which is still not acceptable for many real-time applications.

In [12], MIPv6 handover evaluation is presented. Authors have checked each phase of the process. Analysis has been carried out using OMNeT++ simulator. In this contribution the authors have concluded that major reason large handover delay is the excess time spent during movement detection. The authors have shown that movement detection phase makes up 87% of the total handover time. The authors have proposed a Fast Detection Movement Layer 3

(FDML3) algorithm. With this new algorithm, the overall delay is improved by around 25% which is again not acceptable for real-time communications.

In [12] authors have presented a survey on mobility and multi-homing in IPv6 based networks. Traditionally the two have been considered as disjoint concepts. Authors have argued that this has led to the development of two protocol families separately. However, many new IPv6 terminals are mobile and are equipped with multiple interfaces. Thus, there is a need to adopt a new unified framework providing both multi-homing and mobility. In the paper, authors have focused on shim6 protocol and have investigated its feasibility to be used as a mobility solution. Shim6 is a multi-homing protocol proposed by the IETF which provides support for end-host to manage multiple addresses but does not provide native support for mobility. Experiments were carried out to gauge the impact of mobility before, during and after shim6 context establishment. The results showed that shim6 can manage mobility on its own without the assistance of any other protocol during and after context establishment. In this Amine et al proposed a mobility solution using SHIM6. Their approach is based on shim6 for a single interfaced Mobile Host (MH). They have shown that handover latency is less than that of MIPv6 but still more than 2 seconds.

Shim6 is considered by many as an alternate solution to provide layer 3 mobility support in IPv6 based networks. Shim6 also enables a mobile user to be multi-homed. Failure detection and recovery of addresses in shim6 protocol is carried out through REAP sub-protocol. REAP thus forms a backbone of mobility service through shim6. Shim6 was first introduced by Bagnulo et al in IETF [3] [15]. It was purely a multi-homing solution. However, since then, Bagnulo et al have extended their work to include mobility support through Shim6 protocol. Rehman S. et al [10] presented a proposal SEamless MObility using SHIM6 (SEMO6), a multi-

homing based mobility protocol framework for host mobility. Tests were performed comparing MIPv6 with SEMO6. Although SEMO6 gave a better handover performance than MIPv6, the handover delays experienced by applications were still high (in seconds). They used shim6 protocol to perform mobility operations and have shown that SEMO6 can improve the performance of applications in IP-based mobile networks.

In [14], authors have compared the performance of MIPv6 and a mobility scheme based on shim6. They have performed the simulations of both protocols and have shown that the handover delay is significantly reduced if shim6 is used. REAP is one of the essential parts of shim6 mobility operations. It is used to detect and recover from failures in active communication links.

An implemented of shim6 was carried out by Sebastian Bare et al. They have described the implementation in detail in [17]. We have used their implementation for our analysis namely linShim6. Authors have given detailed account of the implementation of shim6 kernel based implementation in [18].

### SHIM6 ASSISTED MOBILITY SCHEME (SAMS)

#### 3.1. Proposed Architecture

In Shim6, a mobile user initiates communication in a normal way. Then according to some heuristics he can activate shim6 signaling. This heuristics can be a number of packets or time stamp. According to LinShim6 authors the heuristic to activate Shim6 context can be customized. To activate Shim6 context, both hosts should be shim6 enabled. This requires implementation of shim6 on both hosts. Basically, shim6 context establishment is based on four control messages, I1, R1, I2 and R2 respectively. All the information is exchanged in these messages. If both hosts are not shim6 enabled, communication through shim6 context will not take place. In this way initiator realizes that other end is not shim6 enabled. Shim6 allows a host to change locator during an active communication. It is used to ensure redundancy when the current path fails. The default locator switching mechanism in shim6 is defined as REAchability Protocol (REAP). REAP is based on two parts: “failure detection” mechanism which detects failures between two communicating hosts, and “recovery mechanism” which activates address-pair exploration mechanism and finds another operational address pair to resume the communication incase of failure. The failure detection mechanism uses two timers (*Keepalive Timer, Send Timer*) and *Keepalive message* to detect the failures between two communicating hosts. The concrete process in REAP is described as follows:

- a) When host A sends data packets to host B, a “*send timer*” is initiated at the same time.

The suggested timeout value for this timer is 10s.

- b) When host B receives same data packets sent from host A, a “*Keepalive timer*” is started simultaneously. The suggested timeout value for this timer is 3s. If due to some reason host B sends no packets to host A within the *keepalive* interval (less than *Keepalive timeout* value) after receiving data packets from host A, host B sends a *Keepalive message* to host A instead.
- c) On the other hand, if host A receives no packets (including data packets and *Keepalive message* packets) from host B before the send timer at host B is expired, it can be determined that there is a failure of current locator path. At this stage the address exploration mechanism is activated and finds another operational locator.

The proposed scheme SAMS is based on packets which include the detail of new locator addresses. In real time applications user needs efficient and reliable communication source. Based on the literature survey and the research work carried out so far, we believe that there is a need for a better locator switching mechanism to provide support to real-time applications. This solution should also be able to use existing infra structure. According to requirements it should not include any additional namespace or additional network element to be deployed. Any such solution must be easily deployable.

Hence, in this context we propose an intelligent trigger based switching mechanism named as “Shim6 Assisted Mobility Scheme (SAMS)”. This name is given to differentiate this mechanism with others (such as SEMO6). This mechanism is similar in working as REAP but does not take a long time to re-establish the connection. It is not based on timers as in REAP. The intention behind REAP was to facilitate the static networks. Here we intend to facilitate the dynamic wireless networks. Subnets usually can tolerate some delay but when it comes to end hosts (Mobile nodes) efficiency and reliability is a major concern. This mechanism significantly



reduces the handover delay such that on-going application session does not experience a break during switching phase. It is a shim6 based switching mechanism which observes the QoS parameters after context has been established between two communicating hosts. The intelligent module decides whether to divert the traffic to other path or not, based on the status of active path. The intelligent module acts like a control module for switching mechanism. The flow chart of our proposed algorithm is shown in figure 4.

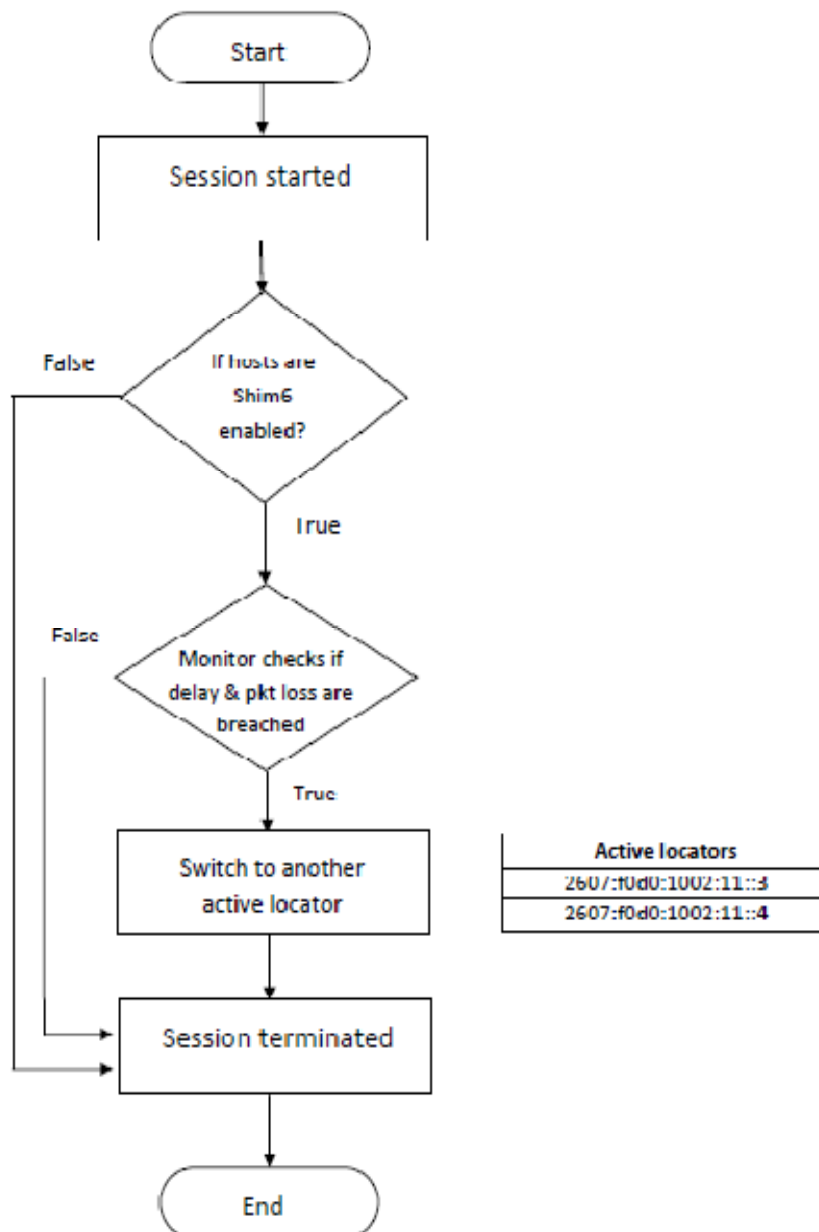


Figure 4: Shim6 assisted intelligent switching mechanism

### 3.2. Algorithm

The proposed algorithm is presented here. Two hosts have decided to contact each other. The normal communication is started between both hosts. After the heuristic condition has met, shim6 context establishment is activated. Our monitor module is also activated when two hosts are communicating over shim6 context. After some time during session our monitor module determines whether we already have Shim6 context established or not. This verifies that both communicating hosts are Shim6 enabled. The next step is to observe the path conditions for some period of time. At this step it is compared with the predefined threshold values of packet delay and loss. If those values remain within limits then nothing is activated. If they are breached our module triggers the *switch\_locator ()* function. This function further triggers *update\_shim6\_context ()* function which basically updates the kernel space and user space. The purpose of I2 and R2 packets in REAP is the same as the purpose of this function. Figure 5 show the REAP signaling in failure detection and recovery mechanism. As we can see that if any of the timer *send\_timer* or *keep-alive timer* expires on client or server side respectively, the other host does not receive any kind of packet. With these two conditions the client side or server side decides that there is a problem with the active link. In result multiple probe packets are sent to peer on each locator. This determines that which locators are available. If any hosts receives probe acknowledgement then it is considered to be the best available locator. At this point receiver changes the current locator with the best available locator and updates its peer. After a while communication resumes on the new locator. This methodology works on network layer. In SAMS, application layer mechanism is implemented. The intelligent module monitors the QoS parameters for a while and determines the deteriorating condition of current locator. If it remains within the threshold values then nothing happens. Communication keeps going on in normal

way. If those values are breeched, monitor module signals the host to change the current locator with the new one. All the locators are already saved and assigned to interfaces. Newer locator is then assigned to current client link and activated. This procedure works transparent to network layer. Another advantage of this approach is that the switching is performed transparent to application running at that time. Since applications are only concerned with ULIDs, which in this case remains same and locator to which it was mapped before switching is now mapped to different locator. Figure 6 shows the signaling in proposed mechanism. Here we can see that there is no timer implemented, thus reducing our switching delay. Following is the pseudo code of proposed algorithm.

```
While (1)  
{  
If (shim6 context )  
    then  
        For (t= specific duration of time)  
            if (Pkt loss & delay => threshold value)  
                switch_locator{  
                    update shim6 context(new_locator);  
                }  
            else  
                continue  
            //end if  
        //end for
```

```

else
continue
//end if
}

```

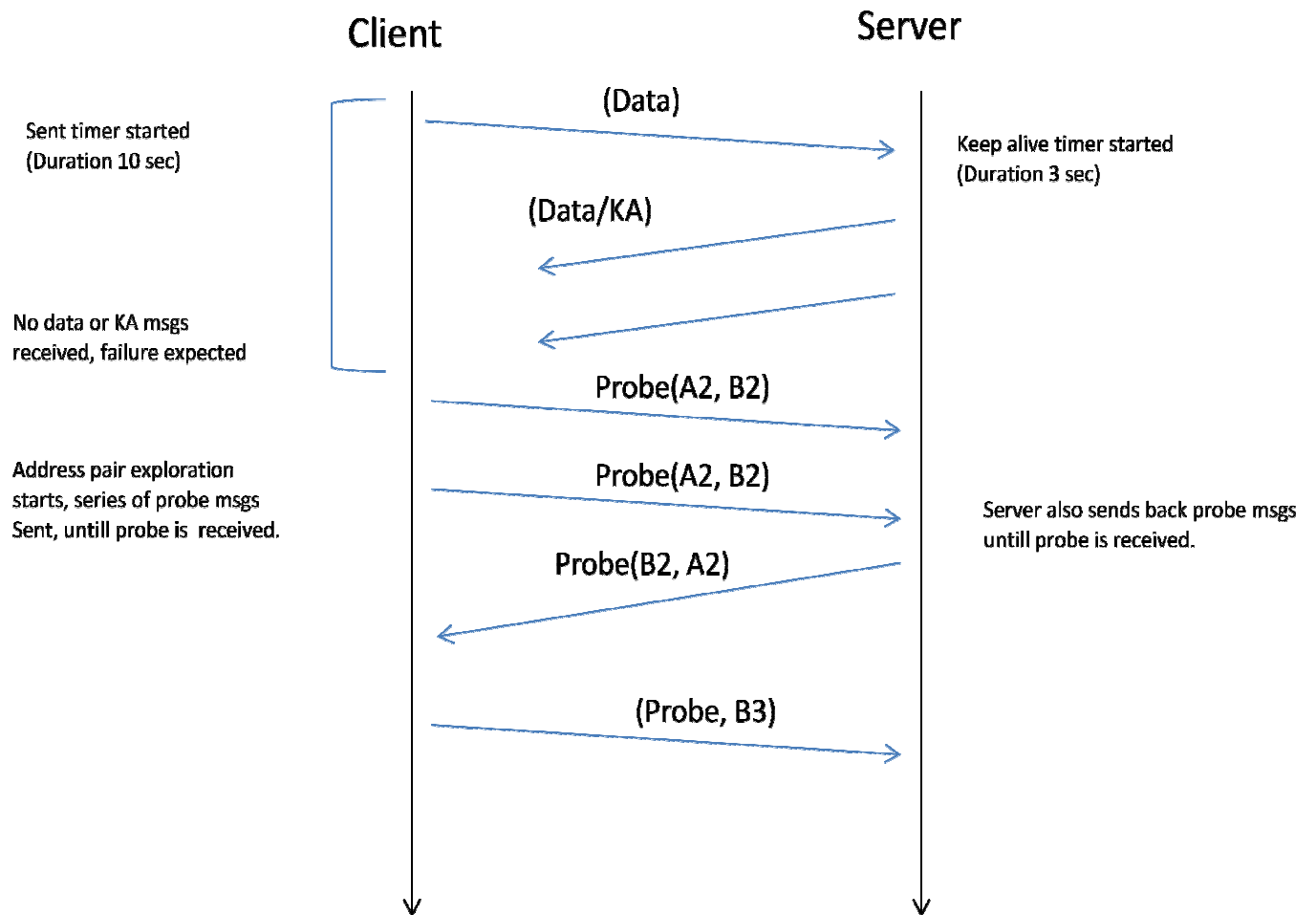


Figure 5: Signaling in REAP

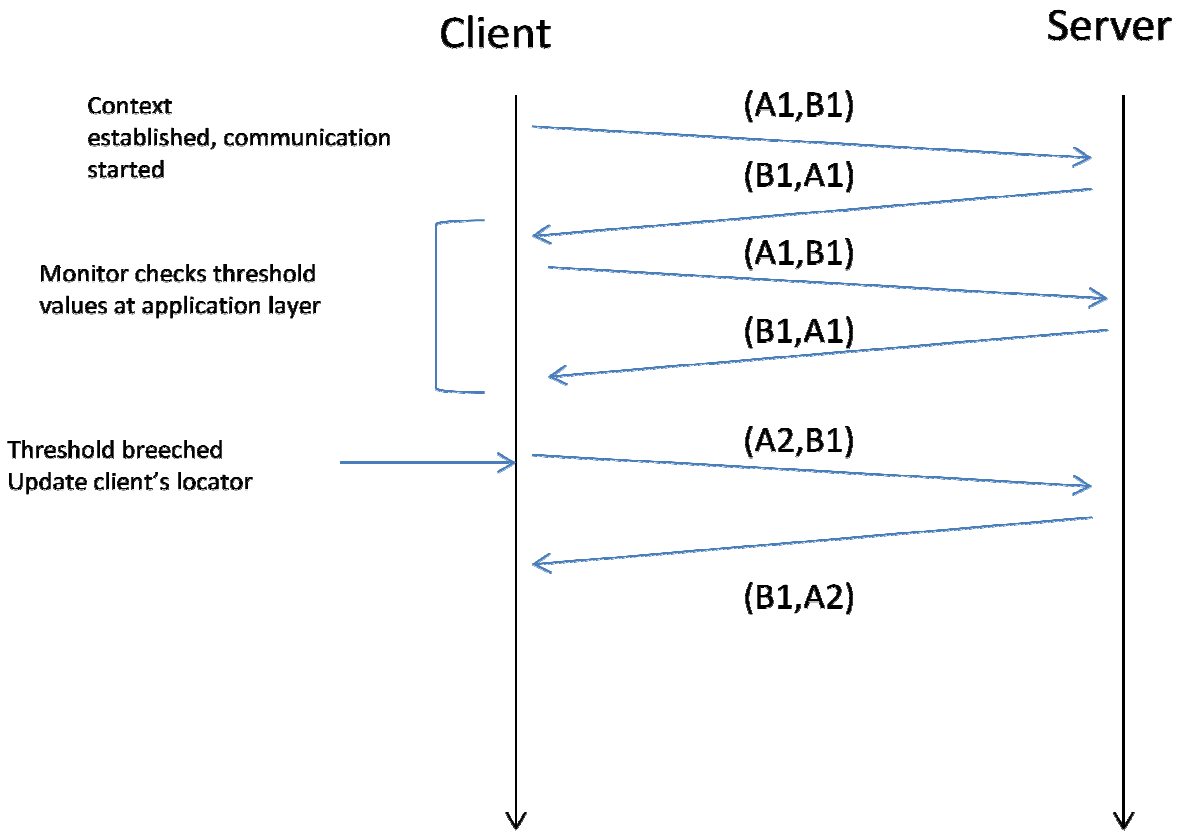


Figure 6: Signaling in proposed mechanism

### IMPLEMENTATION SETUP

In this section we explain in detail the procedures from compiling the required kernel to running the Shim6 enabled host. The chapter discusses all the steps and procedures which would also be helpful for any new user who wants to build up the proposed scenario. First we discuss the steps to configure the modules in kernel. After the modules have been loaded and enabled into kernel configuration, we discuss the compilation procedure of Linux kernel. This is the most time consuming and important phase of our implementation. All the mechanism is based on the correct configuration of kernel modules. We compiled on earlier as well as newer kernel versions (then 2.6.27) on almost all the versions of Ubuntu, Fedora and CentOS. We had errors and problems in booting the newly compiled kernel in almost all the versions of above mentioned flavors. Only CentOS-5.4 worked in our case, so we recommend using CentOS-5.4 in this implementation. After the compilation step if the required kernel is booted successfully, we move on to installing the user space code. This code is publically available using git ([git://scm.ucl.info.ac.be/LinShim6.git](https://scm.ucl.info.ac.be/LinShim6.git)). Command for using git will be given in later section. Once the user space code is installed, we implement the same on all the PCs required as per our test bed. In our case we implemented two hosts, communicating within the same network. In addition to this LinShim6 hosts require IPv6 settings and test bed devices should also be IPv6 compatible in order to successfully implement SAMS.

#### 4.1. Kernel configuration

For the implementation of LinShim6, we require kernel configuration and user space configuration. The required Linux kernel was 2.6.27. Linux kernel source can be downloaded from publically available repository at ([www.kernel.org](http://www.kernel.org)). Kernel compilation is not an easy task

to do. It requires some knowledge of Linux terminal and package installation. It needs a lot of patience since this process takes hours to build a new kernel. Depending on the machines it may take lesser time. If a system has more than one processor, the time can be minimized using a modified command which utilizes all the processors. The option is “-jx” where x is the number of processors+1 in the system. (e.g. make -j3 install). Kernel compilation consists of two phases if it requires any patch to be included.

- a) Patching an additional module
- b) Compiling the kernel source

#### 4.1.1 Patching an additional module

This phase is required when Linux kernel does not support a required functionality. Additional functionality is added to kernel source by patching a certain file. All steps can be done easily using terminal. Open the source directory of kernel source in terminal. This step can be done by the following command

- `cd <space> <source directory path>`

Before patching and going further, few dependencies are required to be installed in Linux host. This depends on the operating system. As per our implementation we required following libraries to be installed.

- gcc
- ncurses-devel
- glibc

The command format for installing each of the above packages is

- `yum install <package name>`

Now we patched the Shim6 module using the following command

- `patch -p1 < (full path of file to be patched)`

Note: After this command if you get any error saying that “Hulk failed at line etc”, then you must remove that. Either the kernel version is not compatible or you must remove this error by manually editing the source files.

#### 4.1.2 Compiling the kernel source

Change the current location of control to the main directory of kernel source. Now all the commands are implemented from this location. Using the following few commands will build the required kernel.

- `make menuconfig`

This command shows the configuration menu of kernel. Using this menu we can select the modules which are required for our experiments, including the Shim6 module. Make sure that following options are checked.

➤ **General Options --> Networking Support --> Networking options -->**

**IP protocol -->**

- Shim6 support

- Shim6 debug messages

➤ **General Options --> Networking Support --> Networking options -->**

- Uncheck DCCP protocol (unless really required)



➤ **General Options -- >Device drivers**

- Serial ATA and Parallel ATA

- Via PATA support

➤ **General options -- > File system -- >**

- EX4dev/EX3 extended fs support development

➤ **General options -- > File system -- > DOS/FAT/NT File System-- >**

- NTFS File system support

Now close the configuration menu and run the following commands one by one.

- make bzImage

This command creates the image of kernel based on the selected modules. This step is important to be executed correctly. If this step is executed with no errors, terminal shows up with a message of “bzImage ready!” and prompts us to move forward. Here we can jump to “make install” command but it is preferred to go step by step in order to fix any error if occurred.

- make modules

Use this command to make the modules, existing as well as the patched ones.

- make modules\_install

This command installs the modules which are successfully built.

- make install

This command installs all the modules and prepares the kernel for boot up.

- make

This command is not required in CentOS and Fedora. The previous command does it all. CentOS and Fedora automatically update the grub boot loader file otherwise it should be manually updated. Make sure it is updated at the location “/boot/grub/menu.lst” (in CentOS-5.4). You can also change the boot order and boot screen duration in this file. Save changes and reboot, you will see the new compiled Linux kernel in the booting list.

#### **4.2. User space configuration**

To install the user space code we also need some dependencies. Again depends on the operating system. In CentOS-5.4 we installed the following development package.

- readline-devel
- openssl.i386
- openssl-devel
- libtool

Now go to folder “LinShim6” which was downloaded using git. Using the following commands install the code.

- automake
- libtoolize --force
- aclocal
- ./configure CPPFLAGS="-isystem /home/mudassir/Desktop/linux-2.6.27/include"

(Remember not to give spaces before and after the equal sign (=), otherwise it will give error.)

- make
- sudo make install

During the installation process if the system is missing any dependencies for LinShim6 code, it will prompt the user and user can manually install them using internet.

### 4.3. Test bed Setup

To analyze and compare the performance of REAP and our proposed SAMS algorithm we established a live linShim6 test-bed. A communication link was manually broken and the two protocols were analyzed in terms of packet loss, throughput and jitter. The proposed SAMS algorithm intelligently detects that communication through the currently used path or locator is suffering and that a switch should be made to another locator. More specifically when threshold values (delay > 100 ms and packet loss > 1%) are breached for short interval of time (5 sec in our case), our module detects and triggers the *switch\_locator( )* function. This function updates the current path with new locator available. Shim6 session was broken down at t=100 by the client. It was a controlled script driven operation so that switch could take place between multiple locators at defined time intervals (t=20 sec, 40 sec and 60 sec). Multiple locators are sequentially selected from list. The test-bed setup is shown in figure 6. Virtual machines are implemented on two physical systems (client & server). It is worth mentioning here that each virtual machine was configured to have 3 network interfaces (eth0, eth1 and eth2), each assigned with a separate IPv6 address. The rest of the test-bed is based on real network entities. To be able to communicate directly on the internet, these addresses should be global. Both systems are connected through wireless interfaces to the router, and router is then connected to a DSL

modem. The specifications of two systems are identical and are given in Table I. Both machines are based on the same kernel and LinShim6 user-space code. This way both machines are configured as shim6 enabled. They communicate through real network entities within the same subnet. The IPv6 addresses are statistically assigned to interfaces. During the communication they are picked automatically provided they are available. Shim6 based switching can be analyzed using different types of topologies. Switching can be triggered in two cases. First when the mobile node is moving from one network to another. Second when mobile node is not moving and switching is triggered at end host due to some failure in network conditions, making use of its multiple interfaces. This topology was chosen to specifically analyze the locator switching mechanism at end host.

Table I: SAMS Testbed specifications

<b>Host Processor</b>	Intel(R) Core(TM) i5 2410M @ 2.30 GHZ
<b>Host RAM</b>	2 GB
<b>Host OS</b>	Windows 7 Professional
<b>Virtualization software</b>	VMware Workstation
<b>Guest OS</b>	CentOS 5.4 (compiled with 2.6.27 kernel)
<b>Router</b>	Linksys Wireless Broadband Router-G
<b>Traffic generator</b>	jperf-2.0.2

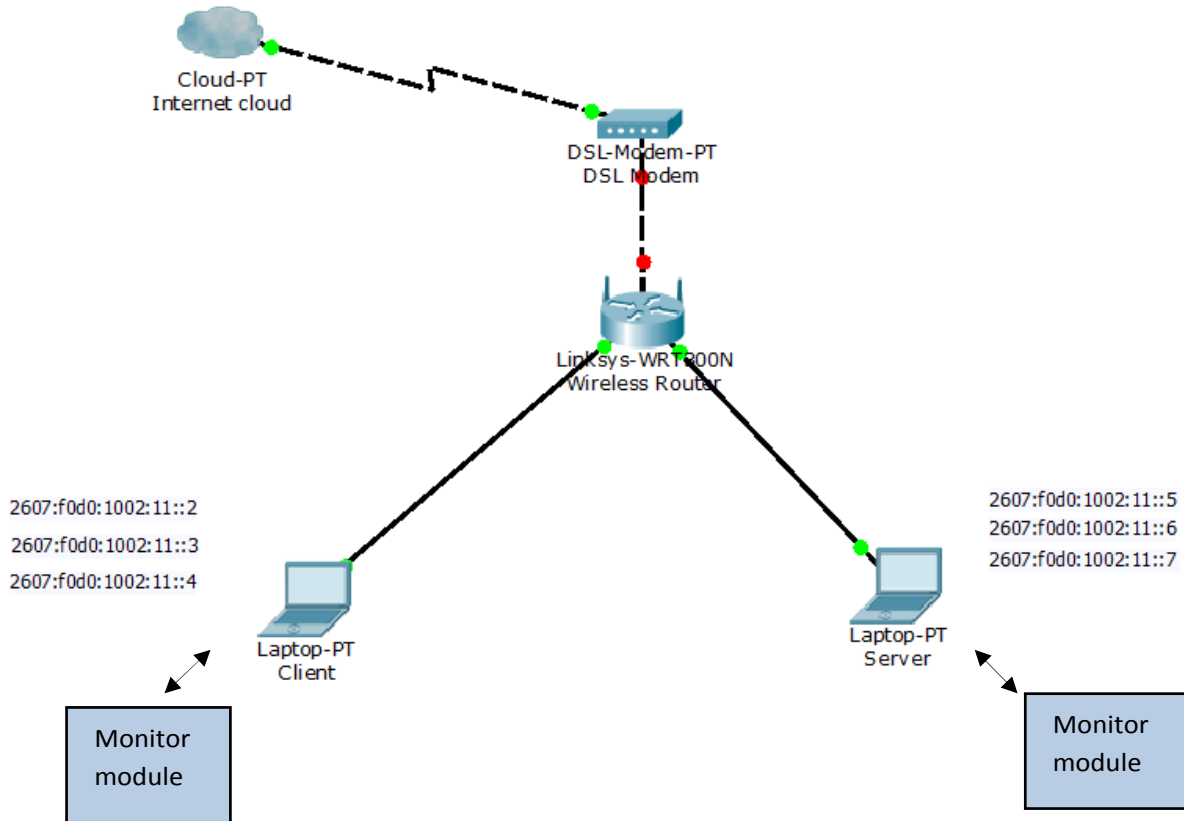


Figure 7: Experimental topology for SAMS

#### 4.4. Running the Shim6 enabled hosts

When both the kernel and user space configuration has been made, we are able to use the Shim6 protocol on both hosts. In order to get started we need to familiarize ourselves with the commands used in user space console. The following commands are used to start the Shim6 user space console. These commands are specific to LinShim6 implementation by Sebastian. Barea. More functionalities or features can be added to this console based on what user wants to analyze. These commands basically invoke the corresponding functions written inside the code.

- *modprobe shim6*

This command activates the shim6 module which must be compiled and built when the required kernel is compiled.

- *modprobe shim6\_pkt\_listener*

This command invokes a module called shim6 packet listener, which basically monitors the shim6 packets exchange.

- *cgad*

Cryptographically generated addresses are generated by this module. If user wants to analyze the security features then this module must be invoked.

- *shim6d*

Shim6 daemon is started by this command. It runs in the background and runs unless it is stopped by host.

- *shim6c localhost*

Shim6 localhost basically connects the system with localhost.

After executing the last command, user should be able to see the user space control panel of LinShim6 as shown below.

***LinShim6-0.9.1>***

Now from this point user can check the status of Shim6 context and current communication with a peer host. In this console user provides the predefined commands and manipulates the communication. This console also verifies that the host is configured properly and is now ready to establish Shim6 context with other Shim6 enabled host. The following commands are useful in user space.

Table II: Shim6 console commands

quit	Quit the Shim6 console, Shim6 session ends after this command
exit	Quit the Shim6 console
help	Shows help for any command
dkc	Dumps kernel contexts
dkp	Dumps kernel policies
ls	It show the list of current available context, by their context tags
cat	Displays details about the context, detail includes active locators, context tag, peer locators and all the available locators for failure scenarios. When REAP is used, this command also shows the no of probes sent or received during failure recovery phase.
rm	Manually deletes the context with the given local context tag from kernel and daemon
dcp	Dumps all CGA parameters stored in the daemon
sla	This command shows all the available local addresses
nbc	Shows the total number of contexts which exist at current time
Set tsend	This command sets Tsend timer value, one of the counter used in REAP failure detection phase

#### **4.5. Shim6 Context establishment**

We communicated both Shim6 enabled hosts using two types of traffic. Traffic generator used in our experiments was jperf-2.0.2 which is the graphical version of iperf. It can generate both UDP and TCP traffic with desired data rates and bandwidth settings. In addition to that this tool provides the freedom to choose multiple parallel data streams between hosts. Based on our options chosen for experiments it also shows the equivalent terminal command in the control panel. There are different options available for generating required traffic, such as application layer, transport layer options and IP layer operations. It requires an IP address of server if user is connecting to other hosts, requesting something. All it needs is an IP address, port number and number of parallel streams to generate. On the server side, user just needs to mention the port number on which server is listening. When the session starts, server maintains a report and present after the communication is stopped. This report is CSV format and corresponding graph is also drawn. Figure 8 shows the screen shot of traffic generator jperf-2.0.2. This is the widely used generator and also recommended by researchers in this field.



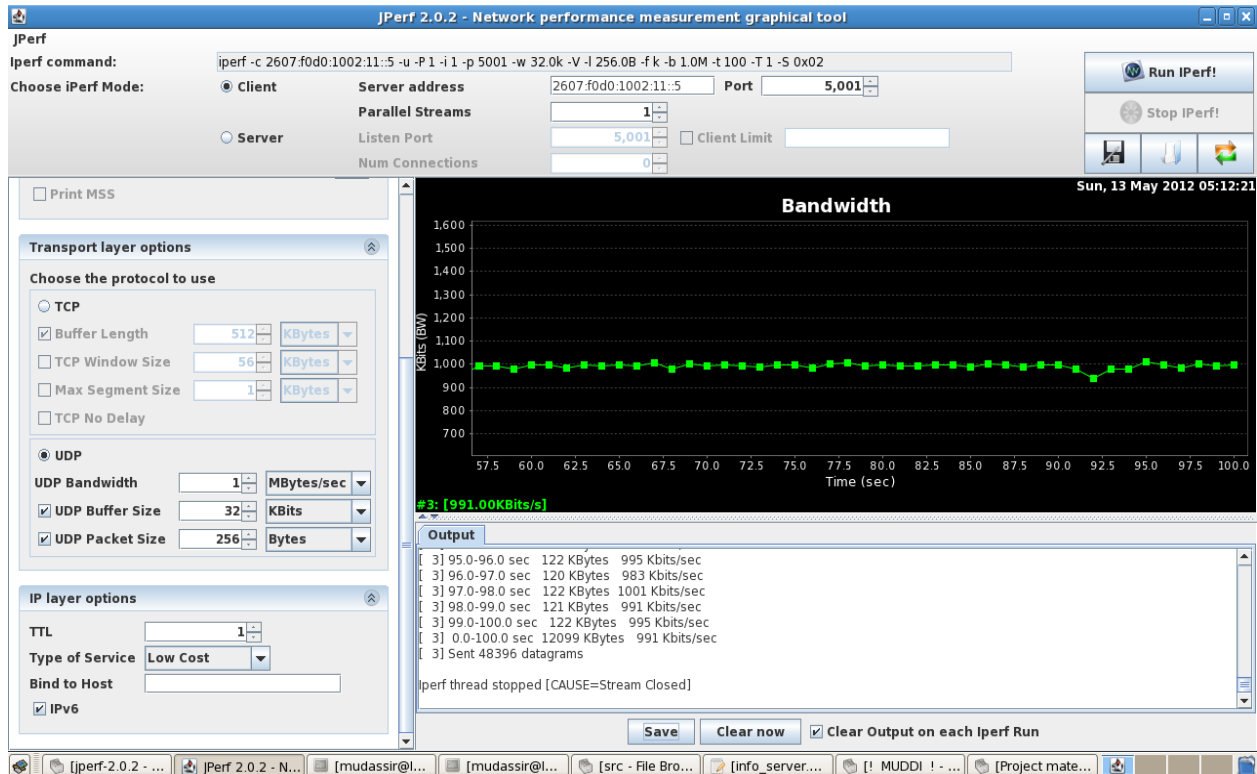


Figure 8: jperf-2.0.2, Network traffic generator and measurement tool

The following figure 9 shows the Shim6 context when client connects to server and both decide to have the Shim6 context. It is clearly seen in the figure that before the switching takes place, communication session is established between the locator 2706:f0d0:1002:11::2 and 2706:f0d0:1002:11::5. The following is the output of *cat\** command after the context has been established, but before the switch has taken place. It shows the state of “established”, which means the Shim6 context is established. Next we can see that there are two context tags, local context tag and peer context tag. This is because each pair of locator (one from client and other from server) is assigned two tags. One is assigned to the path from client to server and other from server to client. The next fields are the locator lists available at client and peer side. The list of available locators at client side is the list that is used for switching mechanisms here. The next two lines in this output show the current locators which are used in communication. REAP state

is then shown in the next field. Although REAP still works in parallel to SAMS but in our mechanism we did not deal with REAP, instead we used our approach. REAP is also tested in the same test bed against SAMS. Next few lines show the values of timers that are set. We did not change any value in these fields. Last few lines show the paths available for communication. At the start of communication path1 is chosen. As long as the switching is performed, paths change accordingly.

#### **4.5.1 Shim6 context before switch**

```
LinShim6-0.9.1>cat *+++++
```

Information from user space daemon

-----

Global state : established

local context tag : 35416c39321b

peer context tag : 55d1682feec7

Peer locator list :

2607:f0d0:1002:11::5

Local locator list :

2607:f0d0:1002:11::2 (User defined addresses, Not HBA nor CGA)

2607:f0d0:1002:11::3 (User defined addresses, Not HBA nor CGA)

2607:f0d0:1002:11::4 (User defined addresses, Not HBA nor CGA)

Current local locator : 2607:f0d0:1002:11::2

Current peer locator : 2607:f0d0:1002:11::5

REAP state : operational

Send timeout: 15.000000 seconds

Keepalive timeout: 15 seconds

nb probes sent : 0

nb probes recvd : 0

Path array :

src : 2607:f0d0:1002:11::2

dest : 2607:f0d0:1002:11::5

src : 2607:f0d0:1002:11::3

dest : 2607:f0d0:1002:11::5

src : 2607:f0d0:1002:11::4

dest : 2607:f0d0:1002:11::5

Now after the switch has been made. We can clearly see that our current locator at client side is changed from 2607:f0d0:1002:11::2 to 2607:f0d0:1002:11::3. It is also clear that we did not use any of the timers. If we did then no. of probes should be changed.

## 4.5.2 Shim6 context after switch

```
LinShim6-0.9.1>cat *+++++
```

Information from user space daemon

-----

Global state : established

local context tag : 35416c39321b

peer context tag : 55d1682feec7

Peer locator list :

2607:f0d0:1002:11::5

Local locator list :

2607:f0d0:1002:11::2 (User defined addresses, Not HBA nor CGA)

2607:f0d0:1002:11::3 (User defined addresses, Not HBA nor CGA)

2607:f0d0:1002:11::4 (User defined addresses, Not HBA nor CGA)

Current local locator : 2607:f0d0:1002:11::3

Current peer locator : 2607:f0d0:1002:11::5

REAP state : operational

Send timeout: 15.000000 seconds

Keepalive timeout: 15 seconds

nb probes sent : 0

nb probes recvd : 0

Path array :

src : 2607:f0d0:1002:11::3

dest : 2607:f0d0:1002:11::5

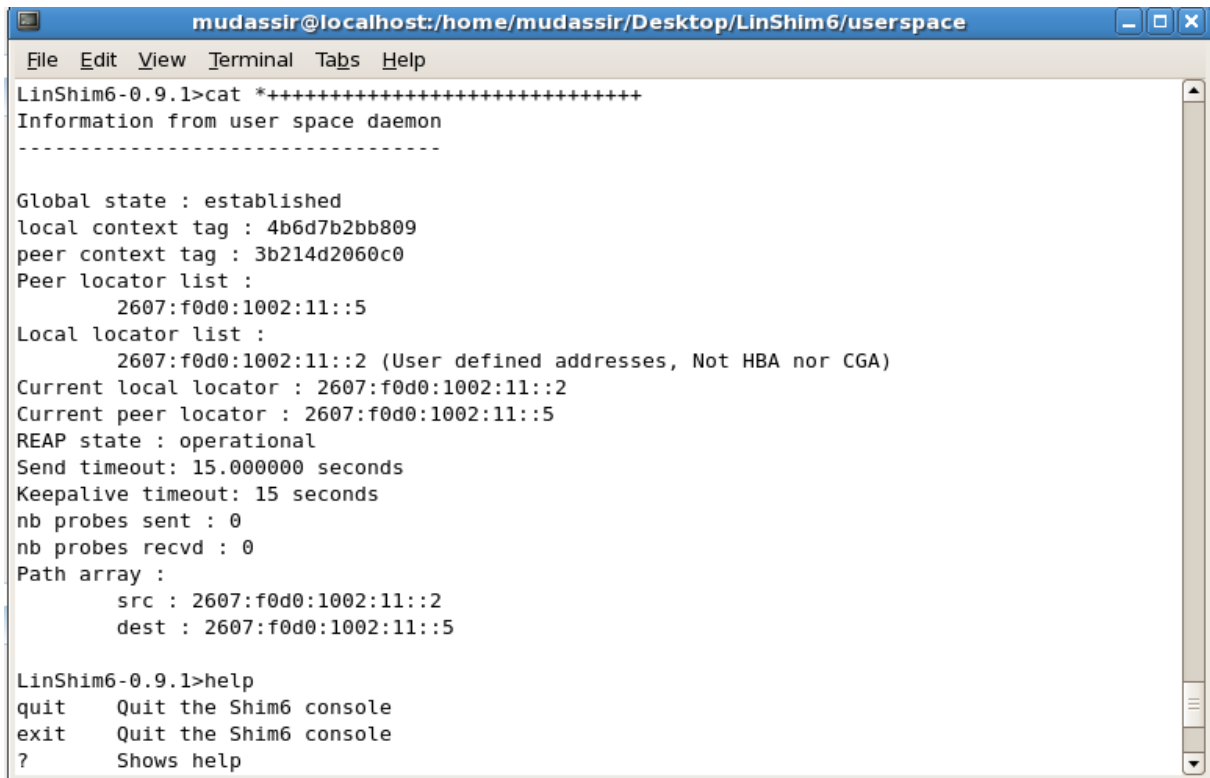
src : 2607:f0d0:1002:11::2

dest : 2607:f0d0:1002:11::5

src : 2607:f0d0:1002:11::4

dest : 2607:f0d0:1002:11::5

The following screen shots also show the same phases during shim6 context communication. The user console is implemented in c language which is controlled by terminal.



```
mudassir@localhost:/home/mudassir/Desktop/LinShim6/userspace
File Edit View Terminal Tabs Help
LinShim6-0.9.1>cat *+++++
Information from user space daemon
-----

Global state : established
local context tag : 4b6d7b2bb809
peer context tag : 3b214d2060c0
Peer locator list :
    2607:f0d0:1002:11::5
Local locator list :
    2607:f0d0:1002:11::2 (User defined addresses, Not HBA nor CGA)
Current local locator : 2607:f0d0:1002:11::2
Current peer locator : 2607:f0d0:1002:11::5
REAP state : operational
Send timeout: 15.000000 seconds
Keepalive timeout: 15 seconds
nb probes sent : 0
nb probes recvd : 0
Path array :
    src : 2607:f0d0:1002:11::2
    dest : 2607:f0d0:1002:11::5

LinShim6-0.9.1>help
quit    Quit the Shim6 console
exit    Quit the Shim6 console
?       Shows help
```

Figure 9: Shim6 console showing Shim6 context before switching

For the specific duration of time, the monitor module observes the QoS parameters such as packet loss and delay. If the communication experiences delay and packet loss above the threshold values defined, switching takes place between the old locator and new locator. To make the switch more sensitive to QoS parameters, we can take into account more QoS parameters such as jitter and throughput. Each locator is assigned to each wireless interface in the Shim6 enabled host. The following figure shows the context after the switch has taken place. It is clearly seen that current local locator is changed from 2706:f0d0:1002:11::2 to 2706:f0d0:1002:11::3.

```
mudassir@localhost:/home/mudassir/Desktop/LinShim6/userspace
File Edit View Terminal Tabs Help
Send timeout: 15.000000 seconds
Keepalive timeout: 15 seconds
nb probes sent : 0
nb probes recvd : 0
Path array :
    src : 2607:f0d0:1002:11::2
    dest : 2607:f0d0:1002:11::5

LinShim6-0.9.1>1
LinShim6-0.9.1>cat *+++++
Information from user space daemon
-----

Global state : established
local context tag : 79534956e5d2
peer context tag : 55a7074d49eb
Peer locator list :
    2607:f0d0:1002:11::5
Local locator list :
    2607:f0d0:1002:11::2 (User defined addresses, Not HBA nor CGA)
Current local locator : 2607:f0d0:1002:11::3
Current peer locator : 2607:f0d0:1002:11::5
REAP state : operational
Send timeout: 15.000000 seconds
Keepalive timeout: 15 seconds
nb probes sent : 0
nb probes recvd : 0
Path array :
    src : 2607:f0d0:1002:11::3
    dest : 2607:f0d0:1002:11::5
```

Figure 10: Shim6 console showing Shim6 context after switching

**RESULTS AND DISCUSSION**

**5.1. Single switching and multiple switching**

This research work is based on two stages of experiments. One is based on single switch performed during the whole shim6 session. Script controlled experiments were performed to trigger the switch at t=50. Other is based on multiple switches performed during the whole shim6 session. In the multiple switching stages a complete sequential cycle was performed starting with the first locator to last and back to first again. Interval of 20 seconds between each switch was considered. Both stages were performed for 100 seconds. In order to gain maximum accuracy, 50 iterations were performed for both stages. Every time threshold values (delay>100 ms and packet loss >1%) were breached, switch took place. In both stages shell script were written to carry out automatic operations.

Communication was established between the two machines. When using REAP, after some time the link was broken by manually turning off the interface. QoS factors i.e. bandwidth, packet loss, throughput and jitter are observed using jperf-2.0. In the case of SAMS algorithm, a locator change request is triggered just as delay and packet loss cross the threshold values (delay>100 ms and packet loss >1%). Furthermore it was compared with “link failure” case when REAP was used. Initially the communications link was established between eth0 (2607:f0d0:1002:11::2) at client end and eth0 (2607:f0d0:1002:11::5) at server end as shown in figure 9. As the path switching conditions were met (at t=20), communication shifted from eth0 (2607:f0d0:1002:11::2) to eth1 (2607:f0d0:1002:11::3). Thus, the new path becomes (2607:f0d0:1002:11::3) to (2607:f0d0:1002:11::5). This is again shown in figure 10. Similarly at t=40 and t=60, script controlled automatic switching was performed. At t=40 communication

shifted from interface eth1 (2607:f0d0:1002:11::3) to eth2 (2607:f0d0:1002:11::4). In result of this, the new path became (2607:f0d0:1002:11::4) to (2607:f0d0:1002:11::5). At t=60 communication shifted from interface eth2 (2607:f0d0:1002:11::4) to eth0 (2607:f0d0:1002:11::2). Thus, the new path became (2607:f0d0:1002:11::2) to (2607:f0d0:1002:11::5), which was the initial path selected. We have used jperf-2.0 for the required traffic generation and monitoring the QoS parameters (such as packet loss, jitter and throughput). We have observed the behavior of both TCP and UDP using jperf-2.0. Our main emphasis was on real-time applications so the analysis was performed on UDP traffic and presented here. We configured jperf-2.0 to generate two types of UDP traffic. One with datagram of 512Bytes, bandwidth 2 Mbps and buffer size of 64Kbits and other with datagram 256Bytes, bandwidth 1 Mbps and buffer size of 32Kbits. This bandwidth is normally used in home environments.

Both the traffics used in our experiments are suitable for VoIP and online gaming respectively. Following is the discussion on different QoS parameters observed in our experiments. Single and multiple switches figures are shown in each of the sections of QoS parameters. Table II shows the types of traffics used in our experiments.

Table III: Traffic used for experiments

<b>Parameter</b>	<b>Traffic 1</b>	<b>Traffic 2</b>
Bandwidth (Mbps)	2	1
Datagram (Bytes)	512	256
Buffer size (Kbits)	64	32



## 5.2. Jitter

Jitter is the sudden spikes observed in the channel when communication starts. The average jitter experienced in single switch by the generated traffic is shown in the figure 11. Figure 12 shows the jitter experienced in multiple switches. Path conditions play a major role in determining the jitter value. It is clear from the figure that communication does not experience large jitter when there is no switch. However, there is a steep rise when communication is switched to a different locator using REAP. On the other hand, it can be seen that when SAMS is used to switch from one locator to another, jitter experienced by the traffic is very low as compared to REAP. This behavior is even shown in multiple switches. Incase when there is a large jitter value, data is corrupted and lost.

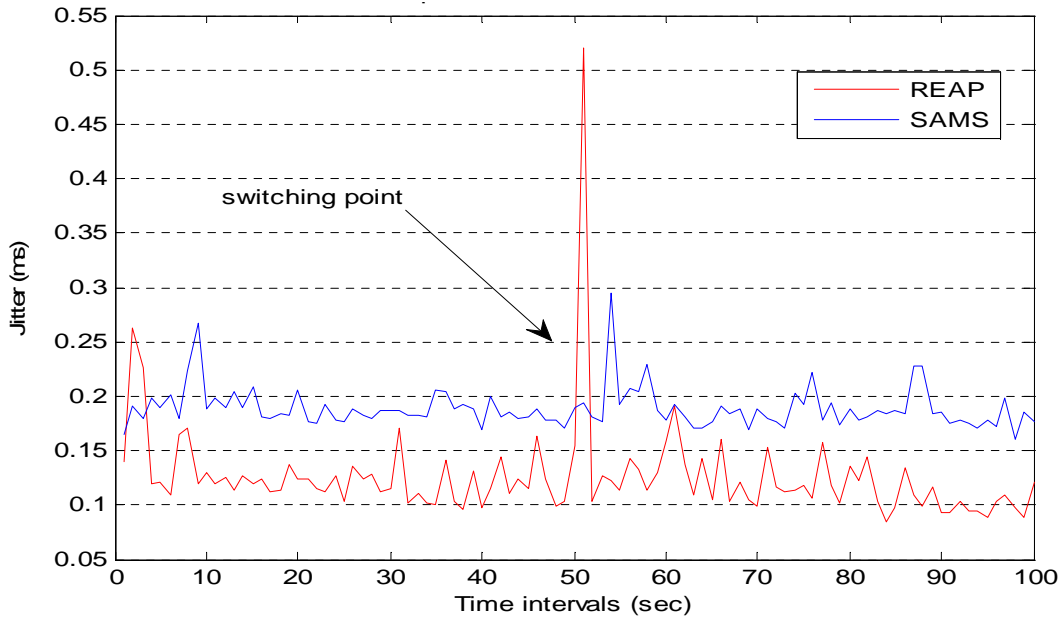


Figure 11: Jitter comparison of single switch between REAP and SAMS

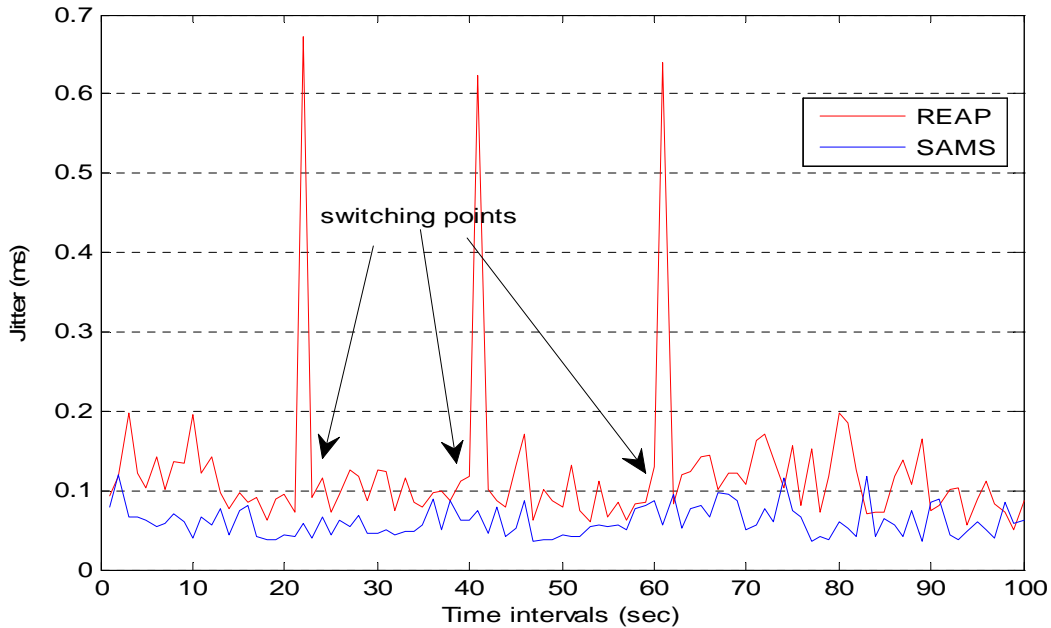


Figure 12: Jitter comparison of multiple switches between REAP and SAMS

### 5.3. Bandwidth

Bandwidth performance during single switch is given in figure 13. While figure 14 shows the throughput dropped during multiple switches. It can be seen that when the active locator fails in case of REAP, failure detection and recovery mechanism is triggered and a new working locator pair is found. It is clear from the figure that until new locator is found, the bandwidth keeps dropping. The curve shows significant drop which was due to the time taken by REAP to detect the failure, explore new locator, and update the shim6 context. The main reason behind this is the timers used in REAP. In SAMS, packet loss and delay experienced by the traffic are continuously monitored for a small period of time. As the values of delay and packet loss breach the threshold values (delay > 100 ms and packet loss > 1%), update request is triggered. Unless the

conditions are met, the link is not broken. Second curve on graph shows the bandwidth performance when SAMS is used. We can clearly see that there is significant difference in throughput between the two cases. This is mainly due to the fact that in our approach a new connection is made before the old connection is broken. In this way switching is done transparently from upper layers.

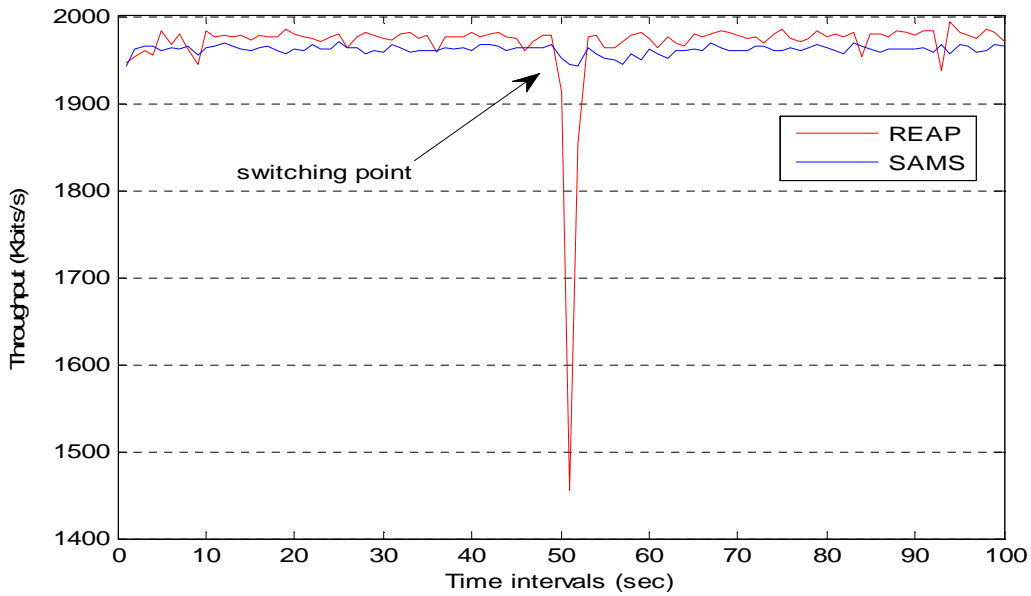


Figure 13: Throughput comparison in single switch between REAP and SAMS

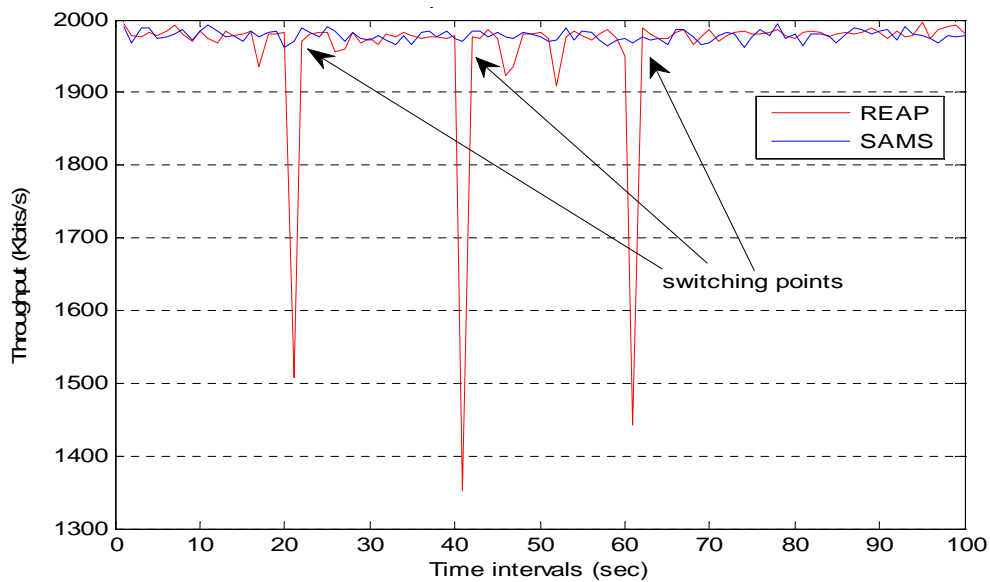


Figure 14: Throughput comparison in multiple switches between REAP and SAMS

#### 5.4. Data Transferred

Figure 15 shows data transferred during a single switch experiment (the REAP case), while figure 16 shows data transfer comparison during multiple switches (SAMS case). It can be seen that there is a big drop during all switches performed when communication link is broken down due to the link failure (which results in REAP activation). The drop is almost 65 Kbytes for the duration of link failure. On the other hand when there is no break up, or when SAMS algorithm is adopted, we do not experience large amount of drop. Only a small drop is witnessed. Similar behavior was experienced during even in multiple switches cases.

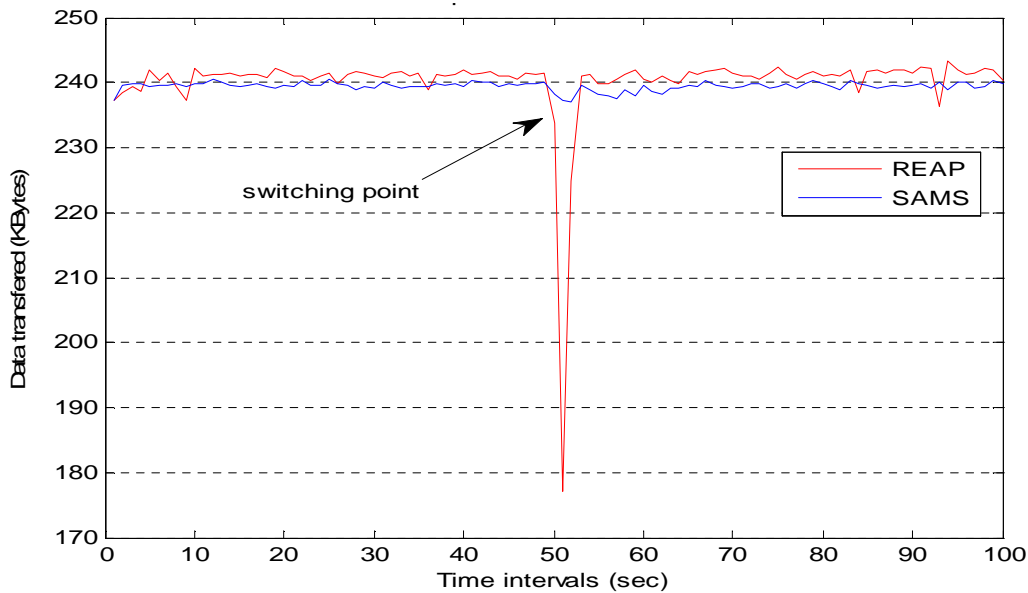


Figure 15: Data transferred comparison in single switch between REAP and SAMS

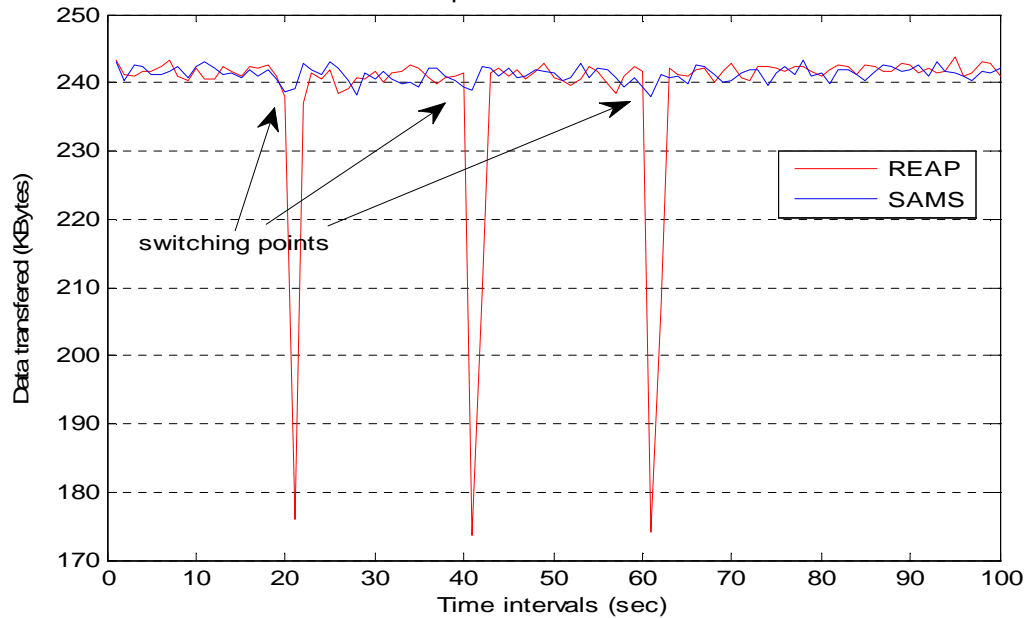


Figure 16: Data transferred comparison in multiple switches between REAP and SAMS

### 5.5. Packet loss

Figure 17 shows the comparison between REAP and SAMS in terms of packet loss. It can be observed from the results that when there is link failure in REAP significant amount of packets are lost. A total of 488 packets are sent on the average. Using REAP, during the switching phase 140 packets are lost on the average. This is a considerable amount when it comes to real-time applications. All the packets that are transferred during the phases of failure detection and recovery are lost. In figure 18, this phase is triggered at  $t=20$ ,  $t=40$  and  $t=60$  seconds. When we look at SAMS results during the same time interval i.e. at  $t=20$ ,  $t=40$  and  $t=60$  the difference is clear. There is a small amount of packet loss which is tolerable for most real time applications. If we compare the QoS parameters, SAMS gives more favorable results than REAP. We analyzed

percentage packet loss throughout the experiment. For the REAP based experiments the percentage stayed below 2%, except for the time interval when switching took place where it increased up to 24.71%. While in case of SAMS, it reaches 1.721% during switching. The packet loss percentage over whole communication (100 seconds) was found to be 1.025% in case of SAMS and 1.352% in case of REAP. Figure 18 shows the results produced during multiple switches.

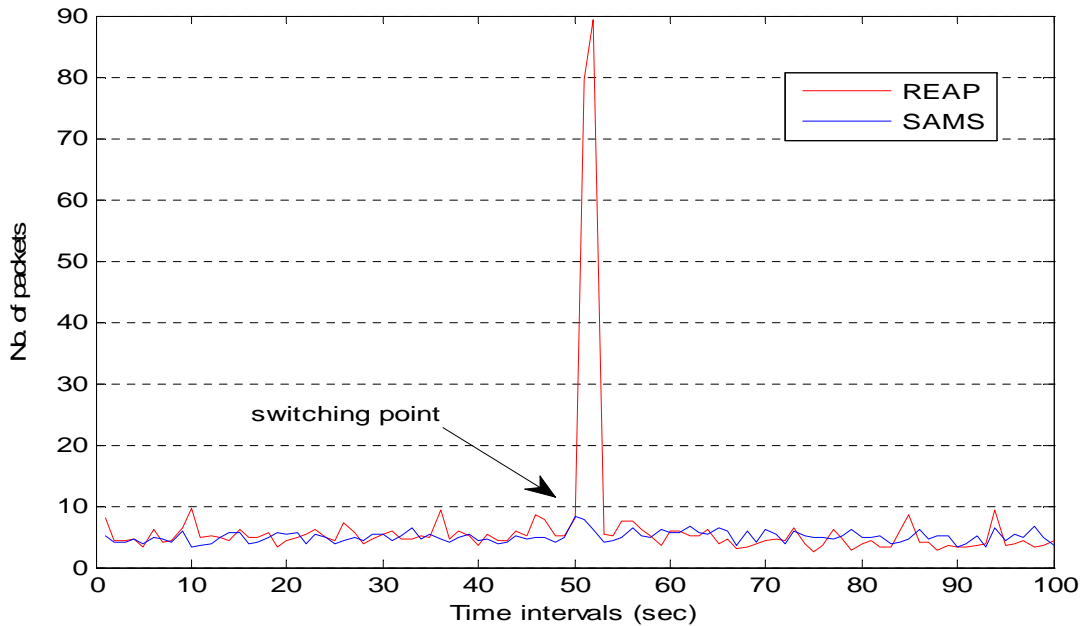


Figure 17: Packet loss comparison in single switch between REAP and SAMS

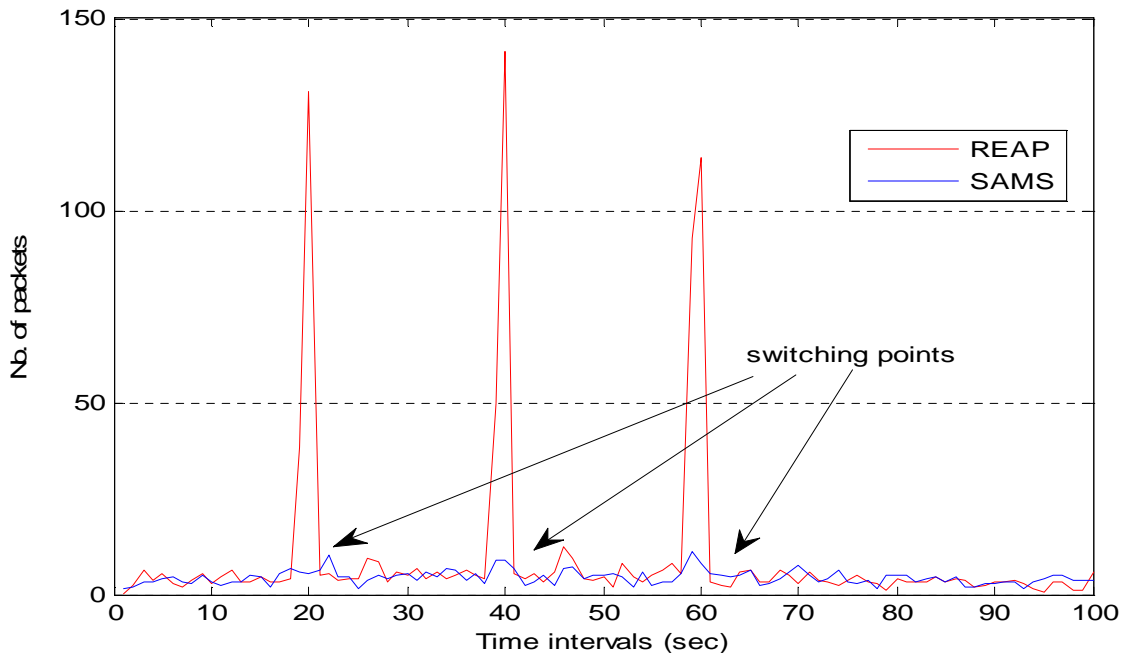


Figure 18: Packet loss comparison in multiple switches between REAP and SAMS

## 5.6. Switching delay

Switching delay is the time duration when the host is shifting from one interface to another and not receiving any kind of packets. It is the time interval which a mobile host takes to shift from one network to another network. This duration of time is different in every mobility protocol. MIPv6 takes 11.27 seconds to recover from failure and resume the communication. FMIPv6 is better approach, taking a lot less time but still goes around 343.53 ms. Another approach was SEMO6 by [10], which takes 25 ms for the handover delay. This was based on Shim6 protocol. The recent switching delay found out by [10] is 25 msec. In our implementation which is again shim6 based, we found out that if intelligent switching mechanism is adopted, we can reduce the delay to 20.286 ms. The comparison of above mentioned techniques is given in the following table.

Table IVIV: Switching delay comparison with existing techniques

Scheme	Switching delay
MIPv6	11.27 sec
FMIPv6	343.53 ms
SEMO6	25 ms
SAMS	20.286 ms

In addition to switching delay results, improvements in other QoS parameters are also observed in our experiments, such as jitter, throughput, data transferred and packet loss. These parameters are important while analyzing and comparing different switching mechanisms.

The following graph shows a comparison between similar schemes.

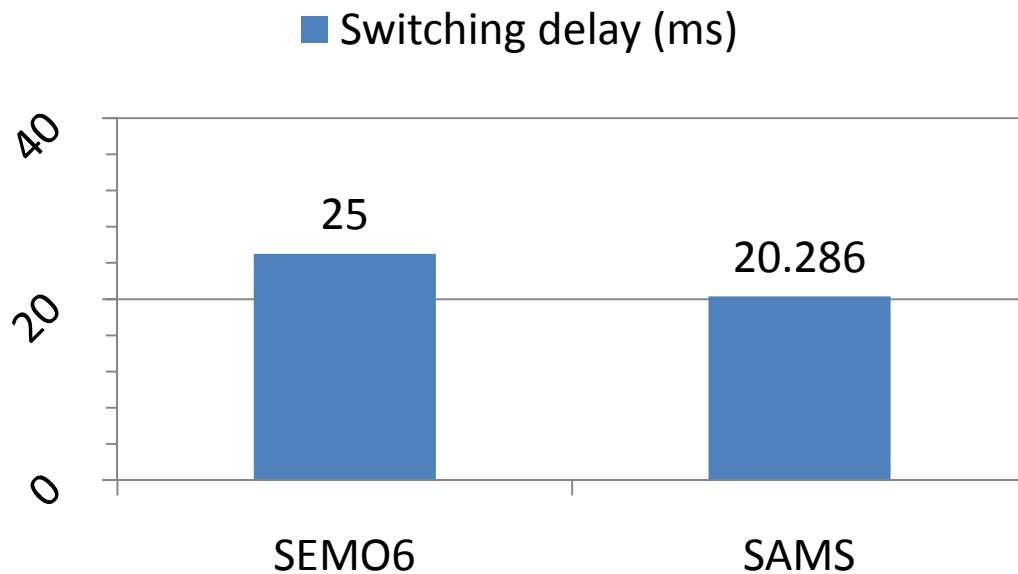


Figure 19: switching delay comparison



Tables V and VI give the comparison of both approaches (REAP and SAMS) for traffic 1 and traffic 2 respectively.

Table V: Implementation results (traffic 1)

<b>Jitter (ms)</b>			
Mechanism	Min	Mean	Max
REAP	0.0510	0.123	0.673
SAMS	0.0350	0.0608	0.119
<b>Bandwidth (Kbps)</b>			
Mechanism	Min	Mean	Max
REAP	1352	1961	1996
SAMS	1963	1978	1994
<b>Data transferred (Kbytes)</b>			
Mechanism	Min	Mean	Max
REAP	174	239	244
SAMS	238	241	243
<b>Packet loss (no. of packets)</b>			
Mechanism	Min	Mean	Max
REAP	1.60	9.707	141.5
SAMS	1.4	4.485	11.10

Table VII: Implementation results (traffic 2)

<b>Jitter (ms)</b>			
Mechanism	Min	Mean	Max
REAP	0.344	0.4114	0.6880
SAMS	0.1320	0.3966	0.6350
<b>Bandwidth (Kbps)</b>			
Mechanism	Min	Mean	Max
REAP	659	988.1	1135
SAMS	924.2	992.2	1014
<b>Data transferred (Kbytes)</b>			
Mechanism	Min	Mean	Max
REAP	80.50	120.7	138.0
SAMS	113	121.2	124

<b>Packet loss (no. of packets)</b>			
<b>Mechanism</b>	<b>Min</b>	<b>Mean</b>	<b>Max</b>
REAP	0	0.28	21
SAMS	0	0.1	9

### CONCLUSION & FUTURE WORK

Shim6 is considered by many as an alternate solution to provide layer 3 mobility support in IPv6 based networks. Failure detection and recovery of addresses in shim6 protocol is carried out through REAP sub-protocol. REAP thus forms a backbone of mobility service through shim6. We have compared the performance of REAP and SAMS on the experimental Linux based testbed. This research work is based on two stages of experiments. One is based on single switch performed during the whole shim6 session. Script controlled experiments were performed to trigger the switch at  $t=50$ . Other is based on multiple switches performed during the whole shim6 session. In the multiple switching stages a complete sequential cycle was performed starting with the first locator to last and back to first again. Interval of 20 seconds between each switch was considered. Both stages were performed for 100 seconds. In order to gain maximum accuracy, 50 iterations were performed for both stages. In this contribution we have analyzed and compared the performance of REAP and SAMS in terms of latency in failure detection and recovery in mobile environments. In using REAP, after some time the link was broken by manually turning off the interface. QoS factors i.e. bandwidth, packet loss, throughput and jitter are observed using jperf-2.0. We used LinShim6 implementation on the test-bed. Both end hosts communicate using specific type of traffic. During communication, at time  $t=20, 40$  and  $60$  we anticipate the switch by an intelligent module which continuously monitors the QoS parameters for specific period of time and helps us in decision making. When the specified threshold for parameters (delay $>100$  ms and packet loss  $>1\%$ ) are crossed, the communication switches from currently used path to another available path. We have seen significant improvement when using our proposed SAMS approach compared to REAP. If we compare switching delay of 25 ms

found in [10] with our proposed solution, it is found to be 20.286 ms, which is as efficient as [10]. Furthermore, we experimentally validated and compared that if intelligent approach is adopted, we can avoid significant packet loss during communication. This is significant especially for real time applications. The overall average packet loss of 1.025% when intelligent approach is adopted i.e. SAMS and 1.352% in the case when REAP is used. It means in our implementation if 488 packets are sent, we only lose 5 packets, which is just a tolerable amount for most real time applications. For the performance comparison between REAP and SAMS we considered packet loss, jitter, throughput and data transferred. Through this work, we have shown that efficient mobility in heterogeneous IPv6 environment can be provided by employing multi-homing techniques such as shim6. Furthermore, failure detection and recovery mechanism in shim6 called REAP is too slow to provide support for real time mobile applications. In this work, we have proposed a quick failure detection and recovery mechanism in mobile multi-homed environment. SAMS significantly reduces the overall switching time through use of triggers. For the future work, we intend to expand our work and define triggers for better mobility management support. Also we intend to build a complete real time test bed which would help us in finding more accurate results.

## REFERENCES

- [1]. Johnson, D., Perkins, C., Arkko, J., "Mobile IPv6 (MIPv6)", RFC 3775, June 2004
- [2]. Bagnulo, M., "Hash-Based Addresses (HBA)", RFC 5535, June 2009
- [3]. Nordmark, E., Bagnulo, M., "Site Multihoming by IPv6 Intermediation (Shim6)", RFC 5533, June 2009
- [4]. Koodli, G. "Fast Handovers for Mobile IPv6 (FMIPv6)", RFC 4068, July 2005
- [5]. Moskowitz, R. Ed., Jokela, P., Henderson, T., Heer, T., "Host Identity Protocol (HIP)", July 2010
- [6]. J. Arkko and I. van Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming," IETF RFC 5534, June 2009
- [7]. Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005
- [8]. S. Pack, Y. Choi, "Performance Analysis of Fast Handover in Mobile IPv6 Networks", 2003.
- [9]. E. Ivov, T. Noel, "An Experimental Performance Evaluation of the. IETF FMIPv6 Protocol over IEEE 802.11 WLANs", IEEE WCNC. 2006, April 2006
- [10]. Rehman, S. Md , Atiquzzama, M. "SEMO6 - A Multihoming-based seamless mobility management framework," IEEE MILCOM 2008, San Diego, CA, pp 1-7 November 2008
- [11]. Carmona-Murillo, J., Gonzalez-Sanchez, JL., Guerrero-Robledo, I., "Handover Performance Analysis in MIPv6, A contribution to Fast Detection Movement", International Conference on Wireless Information Networks and Systems, WINSYS 2008, pp78-81, July 2008

- [12]. Dhraief, A.; Montavont, N. "Toward Mobility and Multihoming Unification, The SHIM6 Protocol: A Case Study" Wireless Communications and Networking Conference, WCNC 2008, pp2840-2845, April 2008
- [13]. Nada, Fayza. "Performance Analysis of MIPv4 and MIPv6" International Arab Journal of Information Technology, Vol4, No. 2, pp 153-160, April 2007
- [14]. Rehman, S. Md, Atiquzzama, M., Wesley, E., William, I., "Performance comparison between MIPv6 and SEMO6," IEEE GLOBECOM 2010, Miami, FL, pp 1–5, Dec. 2010
- [15]. Nordmark, E., and Bagnulo, M., "SHIM6: Level 3 Multihoming Shim Protocol for IPv6," Internet draft, June 2009
- [16]. García-Martínez, A. Bagnulo, M. Van Beijnum, I. "The Shim6 architecture for IPv6 Multihoming," Communication Magazine, IEEE, Issue 9, pp 152-157, September 2010
- [17]. UCL implementation of LinShim6  
[Online available]: <http://inl.info.ucl.ac.be/LinShim6>
- [18]. Barre, S., "LinShim6-Implementation of the Shim6 Protocols," Technical report, February 2008
- [19]. VOIP specifications [Online available] <http://www.voip-info.org/wiki/view/QoS>
- [20]. QoS for recommendations for VOIP
- [21]. Wireshark, open source packet analyzer [Online available]: <http://www.wireshark.org/>
- [22]. jperf-2.0.2, Network Traffic Generator and Measuring tool  
[Online available]: <http://iperf.sourceforge.net/>
- [23]. Shim6 [Online available]: <http://www.shim6.org>
- [24]. S. J. Vaughan-Nichols, "Mobile IPv6 and the future of Wireless Internet Access," IEEE Comp., vol. 36, no 2, Feb. 2003, pp. 18–20.

- [25]. C. Launois and M. Bagnulo, "The paths toward IPv6 multihoming," IEEE Communications Surveys & Tutorials, vol. 8, no. 2, pp. 38–51, Second Quarter 2006.
- [26]. J. Ronan, S. Balasubramaniam, A. Kiani, and W. Yao, "On the use of SHIM6 for mobility support in IMS networks," in 4th International Conference on Testbeds and research infrastructures for the development of networks and communities, Innsbruck, Austria, March 18-20, 2008, pp. 1–6.
- [27]. A. Kiani, S. Khan, and Y. Wenbing, "A novel mechanism to support session survivability in heterogeneous MIPv6 environment," in International Conference on Emerging Technologies, Peshawar, Pakistan, November 13-14, 2006, pp. 38–43.
- [28]. V. Vaassiliou, Z. Zinonos, "An Analysis of the Handover Latency Components in Mobile IPv6", Journal of Internet Engineering, December 2009, vol. 3, no.1.
- [29]. Americas Headquarters: "Implementing Mobile IPv6", Cisco Systems Inc, March 2005.
- [30]. J. Arkko, "Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents", IETF RFC 3776, 2004.
- [31]. Dhraief, A., Belghith, A. "Mobility impact on session survivability under the SHIM6 protocol and enhancement of its re-homing procedure", Journal of Networks, November 2011, vol. 6, no. 11.
- [32]. Draves, R. "Default Address Selection for Internet Protocol version 6 (IPv6)", February 2003.
- [33]. Ahrenholz, J., Henderson, T. OpenHIP
- [Online available] <http://www.openhip.org/docs/shim6.pdf>