

A Tutorial for HMIPv6 Modelling and Simulation in IPv6Suite

Johnny M. Lai, Eric Wu, Ahmet Şekercioğlu
School of Electrical and
Computer Systems Engineering
Monash University
Melbourne, Australia
Email: johnny.lai@eng.monash.edu.au

Abstract—MIPv6 has been ratified recently. It has the potential to offer seamless mobility between heterogeneous access networks. The only problem is that the handling of real time traffic arising from multimedia rich applications is delayed by the lengthy handover process at the IP layer.

HMIPv6 has been proposed as a possible solution to circumvent some of the delays associated with MIPv6. However HMIPv6 requires manual configuration in order to be used effectively. By simulating HMIPv6 in a close replica of the ISP's access network, it can help in making an informed decision of MAP placement. IPv6Suite is aimed at this type of application and we are making steady progress towards this. This paper is a simple tutorial to introduce users to HMIPv6 simulation using IPv6Suite. It also serves as a basic introduction to the use of R with OMNeT++ in general.

I. INTRODUCTION

Internet adoption has surpassed all other communication technologies before it including telephone, radio, television and personal computers [1]. The landscape of today's telecommunications portrays an amazing patchwork of heterogeneous networks, with very few and complex bridges between them. In this context, IP technology has emerged as a natural means of initiating network convergence as seen in the adoption of IP infrastructure 3rd Generation (3G) mobile networks.

Mobile IPv6 has been ratified recently [2]. It has the potential to offer seamless mobility between heterogeneous access networks. The only compromise is that the handling of real time traffic arising from multimedia rich applications is delayed by the lengthy handover process at the IP layer. There are other issues besides handover latency when running real time applications in a mobile environment such as resource unavailability that are out of the scope of this paper, and perhaps the network layer too depending on your view of what services the network layer should provide.

Hierarchical Mobile IPv6 [3] has been proposed as a possible local mobility management protocol to circumvent some of the delays associated with binding with the home agent (HA). However Hierarchical Mobile IPv6 (HMIPv6) requires manual configuration in order to be used effectively. By simulating HMIPv6 in a close replica of the ISP's access network topology, it can help in making an

informed decision of MAP placement and MAP selection algorithms for use at the MN.

IPv6Suite is aimed at this type of application as it has support for the Mobile IPv6 (MIPv6) specification [2] and also the Localised Mobility Management (LMM) extension HMIPv6. We are making steady progress to enable that application. In this paper we shall investigate a simple network scenario to test the MIPv6 and HMIPv6 capabilities available in IPv6Suite and give an example of how easy it is to collate and analyse the results using an Open Source statistics package called R [4].

II. ANATOMY OF HMIPv6 SIMULATION MODEL

The HMIPv6 behaviour has been added into IPv6Suite as a set of C++ classes belonging to the HierarchicalMIPv6 namespace as shown in the Doxygen documentation available online¹. The main classes which implement HMIPv6 functionality HMIPv6MStateHost, HMIPv6MStateMAP, HMIPv6NDStateHost and HMIPv6NDStateRouter all inherit from their MIPv6 counterparts. The NDState classes reside in the Neighbour Discovery OMNeT++ module and are responsible for coordinating the HMIPv6 behaviour whilst the MState classes provide basic functionality to deal with the mobility messages like Binding Update (BU) and Binding Acknowledgement (BA) introduced by MIPv6. However not all of the HMIPv6 functionality resides in those classes, conditional compilation with the macro USE_HMIPv6 was used in various places in the base classes to quickly override MIPv6 behaviour. This was necessary and easier otherwise there would be a duplication of some functions with only a slight difference in one or two sections. It could be argued that the functions could be broken down a lot further however this would reduce their usefulness and make it a cognitive nightmare as the functions would have high coherency.

In HMIPv6 there are two types of handovers global and local. Local handovers occur when the mobile node (MN) moves from one access router (AR) and uses the same movement detection heuristics as MIPv6

¹<http://www-personal.monash.edu.au/~swoon/IPv6Suite-doc/IPv6Suite-doc/namespaceHierarchicalMIPv6.html>

(absence of previous AR's router advertisement), but instead of sending a BU to the HA it is sent to the Mobility Anchor Point (MAP). This occurs in `MIPv6NDStateHost::movementDetectedCallback()` and `HMIPv6NDStateHost::arhandover()` in the simulation. Global handovers arise due to a change in MAP options advertised (a better MAP is found or bound MAP is absent) by the Router Advertisement (RA) and so the MN binds to the new MAP and sends a BU to HA too. The corresponding functions for global handovers are `HMIPv6NDStateHost::processRtrAdv()`, `HMIPv6NDStateHost::discoverMAP()` and `HMIPv6NDStateHost::mapHandover()`.

There is a variable returned by `rt->hmipSupport()` that determines support for HMIPv6 behaviour during runtime. This allows the user to simulate networks with three distinct types of nodes HMIPv6, MIPv6 and no MIPv6 support (when `rt->mobilitySupport()` returns false). The variable is turned on using the eXtensible Markup Language (XML) attribute `hierarchicalMIPv6Support`.

Besides the degree of HMIPv6 support there is another set of variables that deal with the role of a node. In HMIPv6 a new type of HA is introduced called a MAP. To determine if a node has MAP support you can use the following function `rt->isMap()`. This variable is turned on using the XML attribute `map`. For a full list of XML attributes available in IPv6Suite please look at the incomplete documentation for the XML schema².

The MAP options are specified on a per interface basis for MAPs only in the `AdvMAPList` element as shown in Listing 1. Note that the attribute `AdvSendAdvertisements` needs to be on otherwise no MAP options are propagated. All routers will forward received MAP options and increment the distance by 1 regardless of their MIPv6 and HMIPv6 support or lack thereof.

Listing 1: XML fragment for a MAP

```
<local node="map" routePackets="on" mobileIPv6Support="on"
mobileIPv6Role="HomeAgent" hierarchicalMIPv6Support="on"
map="on"
mapReverseTunnel="on">
...
<interface name="ppp2" AdvSendAdvertisements="on"
AdvHomeAgent="on">
<inet_addr>3018:AAAA:0:2:0450:90ff:fe5d:f971</inet_addr>
<AdvMAPList>
<AdvMAPEnt>3018:AAAA:0:2:0450:90ff:fe5d:f971</
AdvMAPEnt>
</AdvMAPList>
</interface>
...
</local>
```

III. GETTING STARTED

It is assumed that the reader has some knowledge of HMIPv6 and MIPv6 to understand the terminology associated with this tutorial. For a thorough introduction in these topics please refer to [5]. Working knowledge in

OMNeT++ [6] is also assumed. This tutorial is written from the perspective of a user on a GNU/Linux based system.

You will need to install IPv6Suite and its dependencies according to the first set of instructions listed at the IPv6Suite Installation web page³. The following options should be turned on in `ccmake BUILD_MOBILITY, BUILD_HMIP` and `USE_XMLWRAPP`. The last option is only available when the advanced mode is active (press 't' to toggle advanced mode). If the `libcwd` library is installed then it is advisable to enable `LIBCWD_DEBUG` option too as it gives detailed information on what IPv6Suite is doing. To follow the analysis in R you will need to download and install R (I used 1.8.2).

A. Running the simulation

The files from this tutorial are already included in IPv6Suite in the subdirectory `IPv6Suite-0.92.20040805/Examples/HMIPv6Network`.

To run the tutorial network go into the directory above and execute the command `./HMIPv6Network -f HMIPv6Sait.ini -r 1` and a graphical representation of the HMIPv6 scenario should resemble that of Figure 1. This is a network with two MAPs with four access point (AP)s connected to each MAP. The MN (the laptop in Figure 1) moves according to the following sequence: $H_{AP}, AP_1, AP_a, AP_b, AP_2, AP_3, AP_c, AP_d, AP_4$. The $AP_1 \rightarrow AP_a$ transition is a global handover and the $AP_a \rightarrow AP_b$ transition is a local handover.

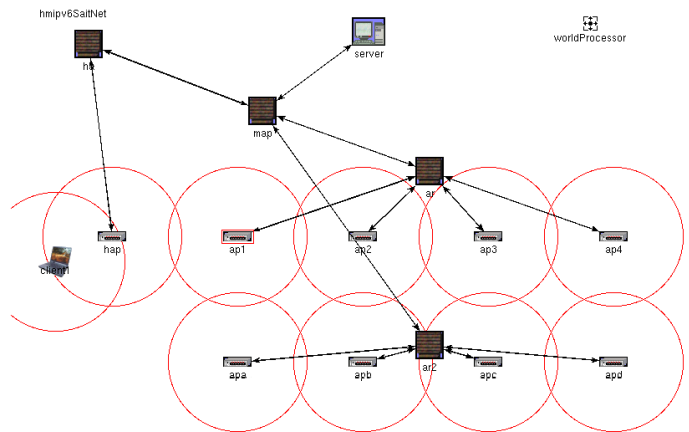


Fig. 1: HMIPv6 scenario under observation

The default settings for the first run do not use Route Optimisation (RO) and so a reverse tunnel to the MAP from the MN exists and another from the MAP to the HA.

B. Troubleshooting

When creating your own HMIPv6 scenarios using IPv6Suite keep the following things in mind if it does not appear to work.

³<http://ctiware.eng.monash.edu.au/twiki/bin/view/Simulation/Installation>

²<http://www-personal.monash.edu.au/~jlai/xsddoc/>

- Is HMIPv6 support compiled in (BUILD_HMIP in cc-make)?
- Have you assigned the correct XML file in the omnetpp.ini file or used -f something.ini to run the simulation if the ini file is called something.ini?
- Is HMIPv6 support turned on for the MAPs in the XML file?
- Are there MAP options defined for the MAP at an *advertising* interface?
- Are the routers advertising at the right interfaces so that the MAP options can propagate from the MAPs all the way down to the ARs?

This is by no means an exhaustive list feel free to add more and forward it to me. In order to answer these questions you will have to install the libcwd⁴ (also available from IPv6Suite’s installation page), which outputs what the simulation is doing almost on a step by step basis.

C. Automating Simulation Execution

I have used a rustic script **graph-simple.sh** which resides in **IPv6Suite/Etc/scripts** subdirectory to automate twenty runs of two variations to the network above using the following command line **sh /scripts/graph-simple.sh p 54 10 20**. The first variation disables HMIPv6 support and the other variation disables Fast Solicited Router Advertisement (FSRA)[7]. The script generates the required number of seeds for the number of runs (20). It then runs each configuration for the configured number of runs, feeding in the seed values into the ini file and upon completion or timeout (54*10 seconds) moves the output files to a subdirectory of the results directory. The name of the subdirectory is formed from the configuration name. This directory contains many numbered subdirectories indicating each run. This way as long as each configuration is named differently the output files for each configuration are contained in a common subdirectory.

We can now run a script to walk through the directories to read out the values from the output vector files and feed them into R. It is possible to do the same from R but not as easily or as convenient, since R was not designed as a general scripting language. In this tutorial I have only included the basics of how to read data from the output vector files and left the automation as an exercise for the reader. To read the values from an output vector assuming the name of **omnetpp.vec** generated from the tutorial network into a data frame in R one can follow the lines 1,6 and 8 in Listing 2. Note that it is better to run R from GNU Emacs using Emacs Speaks Statistics [8] than the command line because it has the following features: command completion; command history; the help page (e.g. **help("plot")**) appears in a separate window; syntax highlighting; and many other benefits.

Listing 2: Simple R session

```
bash> grep vector omnetpp.vec
```

⁴<http://libcwd.sf.net/>

```

2 vector 3 "hmipv6SaitNet.client1.ping6App" "pingRTT" 1
vector 4 "hmipv6SaitNet.client1.ping6App" "pingDrop" 1
4 vector 5 "hmipv6SaitNet.client1.ping6App" "handoverLatency" 1
bash>R
6 R> rttscan<-scan(pipe(paste("grep ^3", "omnetpp.vec")),
list(dummy=0,time=0,rtt=0)),
8 R> rtt <- data.frame(run=1,rtt=rttscan$rtt,
time=rttscan$time)
10 R> rttscan<-scan(p <- pipe(paste("grep ^3", "omnetpp-1.vec"
)),
list(dummy=0,time=0,rtt=0))
12 R> rtt2 <- data.frame(run=2,rtt=rttscan$rtt, time=rttscan$
time)
R> rtt <- rbind(rtt, rtt2)
14 R> close(p);rm(rtt2, rttscan,p)
R> summary(rtt)
16 run rtt time
Min. :1.000 Min. :0.04043 Min. : 40.1
18 1st Qu.:1.000 1st Qu.:0.10115 1st Qu.:168.9
Median :1.000 Median :0.10134 Median :297.8
20 Mean :1.499 Mean :0.09966 Mean :296.6
3rd Qu.:2.000 3rd Qu.:0.10153 3rd Qu.:425.9
22 Max. :2.000 Max. :0.10273 Max. :550.0
R> plot(rtt$time[rtt$run == 1], rtt$rtt[rtt$run == 1], type="S"
)
24 R> plot(rtt$time[rttt$run == 1][100:8000] ,
rttt$rtt[rttt$run == 1][100:8000], type="S",
26 sub="7900 samples of the first run's RTT",
main="Round Trip Time for HMIPv6 Run 1", xlab="Time (s)"
,
ylab="RTT (s)")
> source(path.expand("~/src/IPv6Suite/Etc/scripts/functions.
R"))
30 > jl.ci(rtt$rtt, rtt$run)
No. of Handovers Mean Lower CI limit Upper CI limit
32 [1,] 9952 0.0996647 0.09946919 0.09986023
[2,] 9905 0.0996532 0.09945673 0.09984966

```

While importing many runs into R may appear tedious, it can be automated (interested readers can look at the **RImportOmnet.rb** script in IPv6Suite distribution). Once the values are stored as data frames in R, one can issue **save.image("mySession.Rdata")** to store any variables (to see which variables exactly do **ls()**) that have not been deleted via **rm**. I have also shown how to combine another run into the same data frame as shown in lines 10,12 and 13. The ease of R shines when one combines all the runs from a scenario with many different variations as described in Section III-C. This allows graphical comparisons to be made as shown in Figure 2.

Figure 3 shows the output from line 24 (this should show all eight handovers if maximised to 1280x1024 resolution) but with the samples reduced to show the first three handovers as gaps. The **plve** utility in OMNeT++ can do the same and is often quicker to use for this type of graph. Note that instead of referring to samples the whole round trip time column can be converted into an approximate time series (do **help(ts)** in R). Repetitive tasks in R can be written into functions and saved for later use or shared between users. A saved file **functions.R** shown in line 29 is loaded and its confidence interval (CI) function is exercised in line 30.

IV. CONCLUSION

IPv6Suite is capable of providing credible results by allowing multiple runs to be executed easily. Together with

```

> attach(s.handoverLatency.5)
> jl.boxplotmeanscomp(handoverLatency, scheme, s.handoverLatency.5,
+   plotMeans = F, fn = "output/R-bp-ho", print = T)
> detach(s.handoverLatency.5)
> dev.off()

```

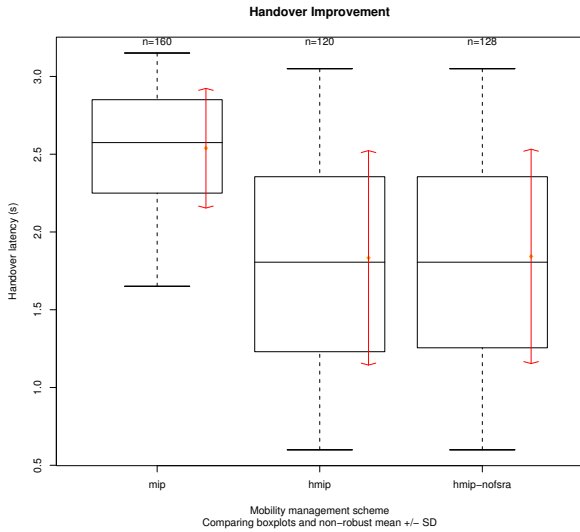


Fig. 2: Box plot of handover latency for the variations in handover scheme described in Section III-C

R it is a perfect combination in analysing the performance of any network scenario you care to come up with. As long as you do not mind spending some time crafting your particular algorithm in C++, carefully designing an experiment to prove or disprove your hypothesis and analysing the results in R with scrutiny then only the sky is the limit to how much you can achieve with these free and indispensable tools.

V. ACKNOWLEDGEMENT

This work has been partially supported by the Australian Telecommunications Cooperative Research Centre (ATCrc).

REFERENCES

- [1] D. Awduche, "MPLS and Traffic Engineering in IP networks," *IEEE Communications Magazine*, pp. 42–47, December 1999.
- [2] D. B. Johnson, C. E. Perkins, and J. Arkko, "RFC 2473 Mobility Support in IPv6," June 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3775.txt>
- [3] H. Soliman, C. Castelluccia, K. El-Malki, and L. Bellier, "Hierarchical Mobile IPv6 mobility management (HMIPv6)," Internet Draft, June 2004, work in progress. [Online]. Available: <http://www.watersprings.org/pub/id/rfc/draft-ietf-mipshop-hmipv6-02.txt>
- [4] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2003, ISBN 3-900051-00-3. [Online]. Available: <http://www.R-project.org>
- [5] H. Soliman, *Mobile IPv6 - Mobility in a Wireless Internet*. Addison-Wesley, 2004.
- [6] "OMNeT++ object-oriented discrete event simulation system," URL reference: <http://www.hit.bme.hu/phd/vargaa/omnetpp.htm>, 1996.
- [7] J. Kempf, M. M. Khalil, and B. Pentland, "IPv6 fast router advertisement," Oct. 2003, work in progress. [Online]. Available: <http://www.watersprings.org/pub/id/draft-mkhalil-ipv6-fastra-04.txt>

Round Trip Time for HMIPv6 Run 1

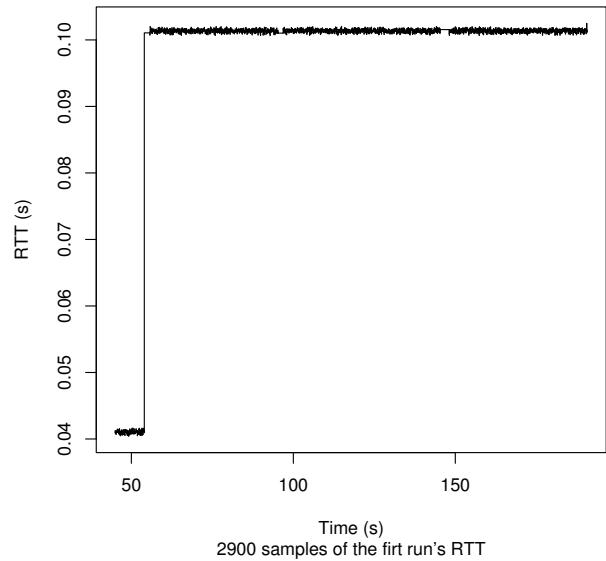


Fig. 3: Plot from line 24 of Listing 2 with a reduced number of samples

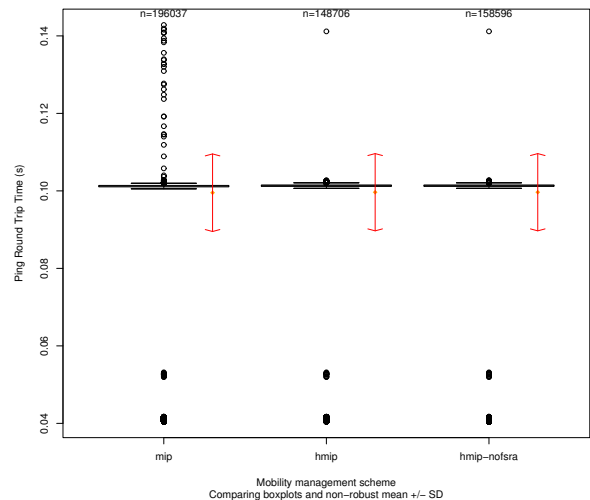


Fig. 4: Box plot of round trip time for the simulation experiment described in Section III-C

- [8] R. M. Heiberger, "Emacs speaks statistics: One interface – many programs," in *Proceedings of the 2nd International Workshop on Distributed Statistical Computing (DSC 2001)*, K. Hornik and F. Leisch, Eds. Vienna, Austria: Technische Universitat Wien, 2001. [Online]. Available: <http://www.ci.tuwien.ac.at/Conferences/DSC.html>
- [9] F. Leisch, "Sweave, part I: Mixing R and L^AT_EX," *R News*, vol. 2, no. 3, pp. 28–31, December 2002. [Online]. Available: <http://CRAN.R-project.org/doc/Rnews/>