ENABLING NETWORK MOBILITY SUPPORT

Devan Rehunathan

A Thesis Submitted for the Degree of PhD
at the
University of St. Andrews

2012

Full metadata for this item is available in Research@StAndrews:FullText at:
http://research-repository.st-andrews.ac.uk/

Please use this identifier to cite or link to this item:
http://hdl.handle.net/10023/3205

This item is protected by original copyright
Enabling Network Mobility Support

2011

A thesis to be submitted to the UNIVERSITY OF ST ANDREWS for the degree of DOCTOR OF PHILOSOPHY

Authored by Devan Rehunathan
Declaration

I, Devan Rehunathan, hereby certify that this thesis, which is approximately 30,000 words in length, has been written by me, that it is the record of work carried out by me, and that it has not been submitted in any previous application for a higher degree.

date ___________________________ signature of candidate ___________________________

I was admitted as a research student in October 2007 and as a candidate for the degree of Doctor of Philosophy in October 2007; the higher study of which this is a record was carried out in the University of St Andrews between 2007 and 2011.

date ___________________________ signature of candidate ___________________________

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

date ___________________________ signature of supervisor ___________________________

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. We have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis:

Access to Printed copy and electronic publication of thesis through the University of St Andrews.

signature of candidate ___________________________ signature of supervisor ___________________________

date ___________________________ date ___________________________
Acknowledgements

To my parents Narayanasamy Rehunathan and Shanthi Lim Hua Keng, whom I love very much.

To my aunt Jarojah Narayanasamy, without your love and support, none of this would be possible.

To Saleem Bhatti, my Sensei. Thank you for your wisdom and for taking a chance on me.

To Tristan Henderson, for always setting standards high and just out of reach.

To Greg Bigwood and Angus MacDonald, for making my time in St Andrews an unforgettable one.

To Ran Atkinson, for sharing your experience with me on a great number of things.

To Martin Bateman, for answering my questions in my first year and for free drinks in my second.

To Pan Hui, for giving me an opportunity to do research at T-Labs, Berlin.

To Boon-Chong Seet, my final year undergraduate supervisor and first co-author, thank you for encouraging me to complete my first academic publication.

To Markus Tauber, your “spanners” really tested my patience but in the end, I think they made me a better researcher.

To Fehmi Ben Abdesslem, for making me laugh and giving me a place to stay.

To Yi Yu, for hacking support, late night company and taking videos of me and my trolley at 4am.

To Savio Dimatteo, for late night dinners and advice on programming. Lets meet up in London.

To Iain Parris, for teaching me the value of knowing your stuff.

To Vincent Perrier, for creating Cloonix and letting me modify it.

To Davy Letham, Jim Park and Jose Marques, for all the technical assistance and hardware support.

To Ommena Chandran, you have put up with me and supported me in more ways than I can possibly imagine. I did this for you.
The reader should note that while the major contributions in the design, analysis and implementation of these works were my own, I acknowledge the contribution and guidance of my fellow authors.

D. Rehunathan, S. Bhatti, O. Chandran and P. Hui.

The Study of Mobile Network Protocols with Virtual Machines.
D. Rehunathan, S. Bhatti, V. Perrier and P. Hui.

Application of Virtual Mobile Networking to Real-Time Patient Monitoring.
D. Rehunathan, S. Bhatti.

A Comparative Assessment of Routing for Mobile Networks.
D. Rehunathan, S. Bhatti.

Comparing Network Protocols via Elimination of MAC/PHY Effects. (POSTER)
D. Rehunathan, S Bhatti.

Enabling mobile networks through secure naming.
D. Rehunathan, R. Atkinson and S. Bhatti.

Mobile Networks: Naming vs. Tunneling. (POSTER)
D. Rehunathan and S. Bhatti.
Abstract

As computing devices become increasingly portable, it is becoming necessary to support Mobility as a core network functionality. The availability of devices such as smartphones, tablets, laptops as well as wireless network infrastructure is opening up the possibility of using Network Mobility to cater for multiple mobile nodes simultaneously. Network mobility may be useful in a number of mobile scenarios, where a large number of mobile nodes are moving in unison. A number of operational benefits stand to be gained by aggregating these nodes into a single mobile unit.

Unfortunately, the current state for network mobility support, especially in terms of network layer protocols, is limited. This is in part due to the inherent complexity of mobile network scenarios, the high cost of testing mobile network protocols in operational environments and the difficulties in implementing such protocols.

This thesis looks at how network mobility support may be better enabled by making experimentation with mobile networks more accessible. It shows this by first showing how analytical approaches can be useful in mobile network applications, as they abstract away from experimental details and allow for more straightforward protocol comparisons. It then goes on to look at the tools available to study mobile network protocols, where it introduces and extends an existing tool that uses virtual machines to allow for the study of mobile network protocols. Finally, it demonstrates a practical method in which mobile network support may be easily enabled in a practical setting.
List of Figures

2.1 Network Mobility with NEMO ........................................... 20
2.2 NEMO: VMN Initialisation ............................................... 20
2.3 NEMO: Testing Reachability ............................................ 21
2.4 NEMO: VMN to CN Route Optimisation ............................. 21
2.5 NEMO: MR Handover ..................................................... 21
2.6 Network Mobility with ILNP ............................................. 23
2.7 ILNP: VMN Initialisation .................................................. 24
2.8 ILNP: MR Handover ........................................................ 25
2.9 ILNP: CN Update ........................................................... 25
2.10 ILNPv6 Address Format .................................................. 26
3.1 Network Mobility with OptiNets ....................................... 34
3.2 OptiNets: VMN Initialisation ............................................. 35
3.3 OptiNets: Testing Reachability ........................................ 35
3.4 OptiNets: VMN to CN Route Optimisation ........................ 36
3.5 OptiNets: MR Handover .................................................. 36
3.6 Passenger Handovers to Passenger Mobility Ratio .............. 39
3.7 Total Number of Passengers ............................................. 40
3.8 Passenger Registration Load to Passenger Mobility Ratio .... 41
3.9 Passenger Handover Load to Passenger Mobility Ratio ....... 42
3.10 20% Passenger Movement Ratio ...................................... 43
3.11 50% Passenger Movement Ratio ...................................... 44
3.12 90% Passenger Movement Ratio ...................................... 45
3.13 ILNPv6 Packet Overhead Per Person ................................. 46
3.14 OptiNets Packet Overhead Per Person ............................... 47
3.15 NEMO Packet Overhead Per Person .................................. 48
3.16 ILNPv6 Bandwidth Overhead Per Person .......................... 48
3.17 OptiNets Bandwidth Overhead Per Person ....................... 49
3.18 NEMO Bandwidth Overhead Per Person ............................ 49

4.1 Mobile IPv6 Topology ................................................. 60
4.2 Cloonix Mobile IPv6 Equivalent .................................... 60
4.3 NEMOv6 Topology ..................................................... 61
4.4 Cloonix NEMOv6 Equivalent ....................................... 61
4.5 Nested Mobility Topology (Mobile IPv6 within NEMOv6) .... 62
4.6 Cloonix Nested Mobility Equivalent ............................... 63
4.7 Network Stack Model ............................................... 63
4.8 Real Testbed and Gate-1 used for Wired and Wireless experiments .... 65
4.9 Gate-2 used for Wired and Wireless experiments ............... 66
4.10 Real Mobile Network ............................................... 67
4.11 Cloonix Virtual Throughput Results ............................... 68
4.12 Cloonix Wired Throughput Results ............................... 68
4.13 Cloonix Wireless Throughput Results ......................... 69
4.14 Cloonix Virtual Packet Loss Results ......................... 69
4.15 Cloonix Wired Packet Loss Results ............................. 70
4.16 Cloonix Wireless Packet Loss Results ......................... 70
4.17 Cloonix Virtual Jitter Results ................................... 71
4.18 Cloonix Wired Jitter Results ................................... 71
4.19 Cloonix Wireless Jitter Results ................................ 72

5.1 Overview of Proposed vNurse Architecture ....................... 78
5.2 Flow of Information in WBAN ..................................... 80
5.3 Network Consumption of Smartphone .......................... 85
5.4 Throughput Variation against Packet Size and Bandwidth .... 86
5.5 HTC Hero, Battery Level ....................................... 87
5.6 HTC Hero, Power Level ....................................... 88
5.7 Route taken by NEMO MR ..................................... 89
5.8 HTC Desire Power Level ..................................... 90
5.9 HTC Desire Voltage Level ................................. 91
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Summary of Early Classic Approach Protocols.</td>
<td>12</td>
</tr>
<tr>
<td>2.2</td>
<td>Use of names in ILNP and IP.</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>London Circle Line Data from Tubeprune [93].</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Passenger and train movement used in emulation.</td>
<td>35</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction 1

1.1 Network Mobility ................................................. 1
  1.1.1 Applications of Mobile Networks ......................... 2
  1.1.2 Advantages of Mobile Networks ......................... 3
  1.1.3 Challenges in Network Mobility Support ............... 4

1.2 Motivation .......................................................... 6
  1.2.1 Complexity of Mobile Network Scenarios ............... 7
  1.2.2 Difficulty in Implementing New Network Mobility Protocols ... 7

1.3 Thesis Outline .................................................... 8

1.4 Thesis Structure .................................................. 8

2 State of Network Mobility Support 11

2.1 Host Mobility ....................................................... 11
  2.1.1 Phase One: Tunnelling ....................................... 11
  2.1.2 Phase Two: Naming and Addressing ..................... 14

2.2 Network Mobility .................................................. 17
  2.2.1 NEMO .............................................................. 17
  2.2.2 ILNP ............................................................... 19
  2.2.3 LIN6-NEMO ....................................................... 28
  2.2.4 HIP-NEMO ........................................................ 29

2.3 Summary .............................................................. 30

3 Impact of Architecture Approach in Network Mobility Support 31

3.1 Introduction .......................................................... 31

3.2 Route Optimisation for NEMO ................................. 31
3.3 OptiNets .......................................................... 33
3.4 Experiment ......................................................... 33
  3.4.1 General Equations ........................................... 36
  3.4.2 Passenger Emulation ....................................... 38
3.5 Results ............................................................ 39
  3.5.1 Generated Passenger Mobility Traces ...................... 39
  3.5.2 Effect of Passenger Mobility on Overhead .................. 41
  3.5.3 Effect of Passenger Correspondence ....................... 42
3.6 Discussion ......................................................... 43
3.7 Conclusion ......................................................... 45

4 Studying Mobile Network Protocols with Virtual Machines 51
  4.1 Introduction .................................................... 51
  4.2 Problem ........................................................ 51
  4.3 A Comparison of Existing Tools ................................ 53
    4.3.1 Use of User-Mode Linux ................................. 53
    4.3.2 On the Use of Simulators ............................... 54
    4.3.3 On the Use of Testbeds ................................ 54
    4.3.4 On the Use of VMware ................................ 55
    4.3.5 On the General Use of Virtualisation .................... 56
  4.4 Cloonix-Net .................................................... 57
    4.4.1 Using Cloonix-Net ....................................... 58
    4.4.2 Modelling traffic effects ............................... 59
    4.4.3 Modelling wireless effects ............................. 60
  4.5 Performance Analysis ......................................... 61
    4.5.1 Experiment ................................................ 61
    4.5.2 Testbed .................................................... 63
    4.5.3 Results ..................................................... 64
    4.5.4 Discussion ................................................ 65
  4.6 Conclusion ....................................................... 66

5 Enabling Network Mobility Support for Remote Health Monitoring 73
  5.1 Introduction .................................................... 73
5.2 Remote Patient Monitoring ........................................... 73
  5.2.1 Wireless Body Area Networks ................................. 74
  5.2.2 Virtualisation on Smartphones ................................. 75
5.3 Proposed Architecture .................................................. 76
  5.3.1 Overview .......................................................... 76
  5.3.2 WBAN Component ............................................... 78
  5.3.3 User-Mode Linux ............................................... 79
  5.3.4 NEMO on Android .............................................. 81
  5.3.5 3G connectivity challenges .................................... 81
5.4 Evaluation .............................................................. 84
  5.4.1 Phase One: Feasibility Testing ............................... 84
  5.4.2 Phase Two: Operational Testing .............................. 87
5.5 Conclusion ............................................................. 87

6 Conclusion ................................................................. 93
  6.1 Contribution and Summary ......................................... 93
  6.2 Discussion ............................................................ 96
  6.3 Future Work .......................................................... 97
Chapter 1

Introduction

As portable computing devices become more capable and affordable, they are becoming common items in our day to day lives. Smartphones, tablets and laptops can be found all around us. With regards to enabling mobility, it is efficient to provide network support not as individual mobile devices but as collections of mobile networks that contain mobile nodes (moving in unison). As the density of mobile computing devices and wireless networking infrastructure increase, mobile networking becomes more applicable to various scenarios. However, because network mobility is not supported as a core functionality in the TCP/IP stack, additional mechanisms are required that implement network mobility support.

1.1 Network Mobility

The term Network Mobility Support, is used to describe how such mobile network functionality may be enabled. Network mobility support is defined as: “...concerned with managing the mobility of an entire network. This arises when a router connecting a network to the Internet dynamically changes its point of attachment to the fixed infrastructure, thereby causing the reachability of the entire network to be changed in relation to the fixed Internet topology. Without appropriate mechanisms to support network mobility, sessions established between nodes in the mobile network and the global Internet cannot be maintained after the mobile router changes its point of attachment. As a result, existing sessions would break and connectivity to the global Internet would be lost.” [1]

Mobile networks are networks that dynamically change their point of network attachment. The simplest case of a mobile network would consist of a mobile router (MR) and a mobile network node (MNN). As the MR changes its point of attachment, not only are all existing network connections of the mobile nodes unaffected but the MR and its MNNs are still connected to the external network, despite having changed topological location.
1.1.1 Applications of Mobile Networks

The study of network mobility support is becoming increasingly relevant as computing devices are becoming increasingly portable. As a result, a growing number of users are able to access the Internet, while on-the-move. The rising popularity of devices such as smartphones and tablets, coupled with the adoption of wireless technologies such as IEEE 802.11 and 3G also reflect this. As a result, hardware manufacturers and network service providers are responding with a greater variety of powerful mobile devices and data plans. If this trend continues, it is likely that the desire for mobility and the ubiquity of portable computing devices and wireless infrastructure, will make it possible to realise the following proposed mobile network applications [2]:

1. Personal Area Networks or networks attached to people
2. Networks of sensors and computers deployed in vehicles
3. Access networks deployed in public transportation
4. Ad hoc networks connected to the Internet via a Mobile Router.

**Personal Area Networks**

Personal Area Networks (PAN) are networks which connect devices on a person’s body. These devices may use various communicating protocols such as Bluetooth\(^1\) or ZigBee\(^2\) and may not necessarily use IP. Possible PAN devices could include smartphones, music players and wireless sensors. PANs may be used for inter-device communication or external communication.

**Vehicular Sensor Networks**

In such networks, the vehicles are themselves mobile networks, which provide network access to passengers or on-board sensors. These could include personal network devices of the driver such as a mobile phone or laptop. It could also include sensors within the car, such as a geo-positioning service (GPS) or performance sensors within the engine. The availability of a mobile power supply is also an advantage as there will be less power constraints.

**Public Access Networks**

Public Access Networks are networks that provide wireless network connectivity on public transportation (e.g. trains and buses). These networks allow IP-enabled devices to connect and use the network. The transport vehicle usually consists of one or more Mobile Routers and the number of connecting users may range from tens in buses to hundreds in planes and trains. Available infrastructure may be leveraged to assist connectivity in some cases.

---

\(^1\)https://www.bluetooth.org/About/bluetooth_sig.htm  
\(^2\)http://www.zigbee.org/LearnMore/WhitePapers.aspx
Mobile Ad Hoc Networks

Mobile Ad Hoc Networks (MANET) are self-configuring, decentralized networks formed of mobile devices that have no inherent infrastructure for routing. As a result, all nodes within the network act as routers and the network structure is dynamic. One possible example of mobile networks in this area is when passengers on a mobile train decide to form an ad hoc network to share files. There is also potential for network mobility support solutions to be extended to support additional functionality in MANETs. For example, in [3], it shows how network mobility may be used to support global reachability of MANET nodes. It is interesting to note that the similarity of some of the problems found in these two domains have led to potentially new areas of research with goals towards integrating network mobility and MANET support [4].

1.1.2 Advantages of Mobile Networks

Network Mobility is particularly applicable to the above scenarios because of the following advantages [5]: (i) reduced transmission power, (ii) reduced handoffs, (iii) reduced complexity, (iv) reduction in bandwidth and location update delays and (v) increased manageability. These benefits are accrued by aggregating multiple mobile nodes into a single entity.

With regards to (i), because the mobile router transmits to the fixed infrastructure on behalf of the mobile network nodes, conceivably, it allows the mobile network nodes to save energy as they transmit over a comparatively shorter distance to the mobile router as opposed to the fixed infrastructure themselves.

With regards to (ii), instead of each node having to manage itself, the mobile router manages the mobility of all the nodes within the network. As a result, at each handover, only the mobile router needs to handoff, all the mobile network nodes can maintain their existing connection with the mobile router regardless.

With regards to (iii), aggregating multiple mobile network nodes into a single mobile entity greatly simplifies aspects of mobility management. When the mobile network changes location, only the mobile router needs to configure a reachable address. This reduces the software and hardware complexity on the mobile network nodes and also makes it possible to have non-mobile aware nodes in the mobile network.

With regards to (iv), instead of all the mobile nodes sending a location update to the home agent, a single mobile router is able to update on their behalf. This results in less number of location updates and a reduction in bandwidth consumption for such messages.

With regards to (v), the mobile router forms a natural central point for managing the mobile network and the mobile network nodes. Being the central point, it would be easy to push out updates to the rest of the nodes or to implement network policies. For example, a device may be nominated in a newly formed mobile network to act as a mobile router, thereby conserving power for the rest of the nodes network. This configuration also makes it possible for nodes that are not mobility-aware to still get access outside the network by using other means available (e.g., Bluetooth or LAN). This could potentially be useful for smaller devices such as sensors. Aggregation may be viewed as an extension of current network architecture to top down centralisation (which is really an extension of the current network topology).
1.1.3 Challenges in Network Mobility Support

Despite the potential applications and advantages to be gained in network mobility, these cannot be reaped with the current TCP/IP protocol stack, as it does not support network mobility as a core functionality. While limited forms of network mobility support do exist, the cost of research this field (e.g. experimentation, implementation) is very high. As a result, there are still a number of open challenges, due to the difficulties in enabling this.

Dominance of the Tunnelling Approach

Currently, the dominant network mobility protocol is the NEtwork MObility protocol (NEMO). It is the current IETF standard for network mobility support and is an extension of Mobile IPv6 [6]. Work on the NEMO protocol is currently being progressed in the IETF Mobility EXTensions (MEXT) working group [7]. This working group is working towards a unified mobility architecture for IP, by integrating NEMO with Mobile IPv6 (for mobile hosts), MONAMI [8] (for multi-homing) and IKEv2 [9] (for managing key exchange for security). The primary goal of this working group is to enhance IPv6 mobility such that it will be suitable for large scale deployment scenarios.

NEMO, like Mobile IP, uses a Tunnelling Approach to support network mobility. This approach uses two IP addresses instead of one, one is static and represents a home address and the other is dynamic and depends on a mobile host’s current location in the network. Both these IP addresses are managed by network entities that act as harbours for the mobile hosts and also as intermediaries, tunnelling packets to and from each other.

The popularity of this approach can be seen in the number of early mobility proposals such as [10] [11] [12] and [13]. Between 1991 and 1994, most major mobility solutions subscribed to the Tunnelling approach [12] and differed mainly in practical implementation (e.g. example protocols such as MosquitoNet did not implicitly require a Foreign Agent, even though this functionality was evident [14]). The research of this period culminated in what we know today as Mobile IP [15], which has been accorded Standard status by the IETF and is available in almost all major operating system network stacks.

NEMO-specific Challenges

Despite the popularity and maturity of the NEMO protocol, it is a basic support protocol and still contains a number of open research issues.

Sub-Optimal Routing

One artefact of the Tunnelling approach adopted by NEMO is that the mobile network is not actually observed to be mobile to the outside world, the mobility of the mobile network and its internal nodes is ‘hidden’ behind a topologically-fixed agent/router. The mobile router uses tunnelling to redirect packets to the mobile network. As a result, this makes it impossible for mobility-aware nodes in the mobile network, such as those running Mobile IPv6 (MIPv6), to employ route optimisation, resulting in sub-optimal routing. The usage of the bi-directional tunnel also adds an additional 40 bytes of overhead per packet, which may cause fragmentation of packets in other parts of the end-to-end
network path [16].

Mobile network prefix delegation

A NEMO MR requires one or more mobile network prefixes to setup a bi-directional tunnel between itself and its Home network. These prefixes are advertised to MNNs within the mobile network to ensure that traffic from the mobile network is forwarded and received correctly. Currently, these mobile network prefixes are assigned statically [5].

Nested Mobility

This occurs when there is more than one level of mobile network, i.e. when one MR, with its own mobile network, attaches to an existing mobile network. With NEMO, when this happens, forwarding tunnels of the inner level network are setup within the external mobile network’s tunnels. This results in an overall overhead of 80 bytes per packet and greater complexity in routing exchanges and establishing forwarding paths [17].

Optimised Handoffs

With NEMO, handoffs are handled by the MRs. When the MR moves to a new point of attachment, it has to re-establish its tunnel with its home network, to receive and send packets. This form of handoff is a hard handover, as the previous tunnel is torn down and a new one is made. This could lead to loss of packets. Another form of mobility is the use of soft handovers, which allows for the setting up of the new tunnel before the old tunnel is torn down. This allows packet loss to be reduced (possibility eliminated altogether) during handover. However, soft handoff is not currently supported in NEMO [18].

Multi-homing

This capability allows for multiple connections to mobile entities. By being multi-homed, such mobile nodes or networks have more robust connectivity to the network. It is also easier for mobile systems to maintain and support ubiquitous connectivity across multiple heterogeneous communication mediums. Such connectivity provides for redundancy, load-sharing and policy routing [19].

Network Pricing

Basic pricing models exist for mobile nodes, such as WiFi hotspots, where users pay a fee for a specific allotment of time on the network. Such models will not be economical for a mobile router, as it may contain multiple nodes within its network resulting in overall larger bandwidth consumption.

Alternative metrics to charge users, may include bandwidth consumption, whereby users pay only for what they use. However, this would be simple only if one user was responsible for a mobile network. This might arise, where the mobile network seeking network connectivity consists of different unrelated users.

As network mobility becomes more valued, competing service providers might cooperate to allow for vertical handovers across different service providers. This would require some form of recording individual user’s and network’s usages across providers, to res-
ult in one bill.

Nested mobility, where mobile networks may connect to other mobile networks, also presents a problem with respect to pricing. It would be possible for mobile networks to pay for connectivity and subsequently charge other mobile networks to nest.

**New Application Layer APIs**

The usage of tunnelling to enable network mobility, hides mobility from mobile networks, mobile nodes and correspondents. Alternative network mobility protocols, may make it possible for applications to detect mobility of the host, which would otherwise not have been possible with protocols such as NEMO.

As a result, new application layer APIs are needed that would make it possible for applications to anticipate the effects of mobility and take suitable action. For example, in the case of non-seamless handovers, where some loss of connectivity and packets is to be expected, applications should sense when these events are about to occur and accommodate to ensure no effects are noticeable for users.

**End-to-End Security**

IPsec Security Associations, include both the source and destination IP addresses. This means that if a node moves, or a network moves, then the existing IPsec Security Associations will cease to be valid. This constraint exacerbates existing concerns about the scalability of key management for IPsec devices. It also means that, regardless of what changes might be proposed for the Internet Key Exchange (IKEv2), support for mobility and multi-homing will remain limited and hard to deploy in the tactical environments where these capabilities are so crucial.

**1.2 Motivation**

While the list of technical challenges in network mobility support mentioned above are important, I will be looking at those which make mobile network experimentation less accessible to researchers. One of the contributing factors to the existence of these open challenges, is that the cost of researching network mobility support is high. Some reasons for this are: (i) complexity of Mobile Network Scenarios and (ii) difficulty in Implementing New Network Mobility Protocols.
1.2.1 Complexity of Mobile Network Scenarios

Dynamic Nature of Mobility

As mobile nodes and networks move, the network mobility support protocol has to ensure that all existing connections are maintained. Mobile network scenarios are especially dynamic as a mobile network contains various kinds of MNNs. MNNs do not all have the same requirements. Some are fixed, they are referred to as local fixed nodes (LFN). Those that are mobile, are referred to as local mobile nodes (LMN) or visiting mobile nodes (VMN). The latter being mobile nodes from foreign mobile networks.

Issues of Scale

Mobile networks usually contain multiple nodes and may contain up to the hundreds, depending on the mobile network scenario. Network mobility support has to take into account network size and ensure that performance is not affected as the scale of the mobile network increases.

Nested Mobility

In the case of mobile networks, it should be possible for MRs to join other mobile networks. This form of nested mobility may consist of mobile nodes as well as other mobile networks. Depending on the method used to enable network mobility, this may result in performance costs. For example, in NEMO, while sub-optimal routing is an acknowledged problem, it is made significantly worse with increasing levels of nested mobility.

1.2.2 Difficulty in Implementing New Network Mobility Protocols

Writing Code

Writing network protocols is hard. Existing network stacks such as the one found in the Linux kernel are known for their complexity. Writing network protocols in these environments requires experience and a deep understanding, which is typically not available to the average network researcher.

Testing Protocols

One method of testing mobile network protocols is through simulation but a more ideal test would be to implement the protocol into a kernel and carry out realistic experiments with an actual testbed. Experimental testbeds of mobile networks have so far been limited to areas of VANETs and MANETs, whereby researchers have turned cars into mobile networks, providing network access to its passengers. While such testbeds do provide the most realistic testing environments, they come at a high cost. Typically, depending on the scale of experiment targeted, mobile network testbeds are expensive and resource intensive.
1.3 Thesis Outline

This thesis argues that while the demand for mobile network support is rising, the current state of network mobility support is inadequate to meet this demand. It shows that new innovative techniques such as, the abstraction of architecture from engineering in simulation and the modification of existing virtualisation platforms can be utilised to better enable network mobility support.

This thesis looks at the following questions:

1. How can existing analytical approaches be leveraged to provide insight into mobile network application scenarios?

2. Can existing virtualisation platforms be modified so that their benefits may be applied to mobile network problems, especially in the case reducing experimentation cost?

3. How may these new tools be effectively used with the current mobile network landscape and technologies?

1.4 Thesis Structure

In Chapter 1, I have introduced the concept of network mobility and its potential applications, advantages and challenges. I have motivated the thesis and outlined the general arguments of the thesis, as well as its overall structure.

In Chapter 2, I survey the current state of network mobility support. I introduce the Tunneling Approach from its adoption in end-host mobility protocols to its eventual usage in network mobility support. I then explore an alternative approach, called the Naming Approach. I conclude by saying that there is a growing need for network mobility support and that the cost of providing this support is currently high.

In Chapter 3, my objective is to show that current network mobility support in its current form has some performance issues that suggest that it would be good to consider other architectural approaches. By looking at the impact of architectural approaches with regards to route optimisation in mobile networks, I conduct a comparative analysis of the cost of providing optimal routing, in terms of packet and bandwidth overhead, based on an emulation, using data from the London Circle Line metropolitan railway as a scenario. By looking at the impact of architectural approaches with regards to route optimisation in mobile networks, I show that these different architectural approaches to mobility offer significantly different performance trade-offs in routing for mobile networks, depending on the constraints of the network scenario.

In Chapter 4, I present a simulation framework, Cloonix-Net, a virtual network tool using User Mode Linux (UML) machines, for the purposes of building and testing such network mobility scenarios. I demonstrate that studying mobile network protocols with such a framework is a beneficial step towards better understanding network mobility protocols and enables better network mobility support. I motivate this work by first outlining the difficulty in developing network mobility protocols. I then do a comparison of
existing tools and conclude that there is space for a new one. I then introduce Cloonix-Net and show how it may be used for mobile network scenarios. Finally, I evaluate Cloonix-Net by comparing its performance to that of a real testbed.

In Chapter 5, I extend my work in the previous chapter and demonstrate how mobile network support may be easily enabled and used in an innovative way. I first make the case for the usage of mobile networking in E-Health networks. I then provide an overview of my proposed architecture and its operation. I then proceed to implement it in real-life and evaluate the feasibility of such an approach. I concentrate on enabling the mobile router (MR) and on performance issues, such as the impact on battery life.

In Chapter 6, I summarise my motivations, reiterate my arguments and state my contributions towards enabling network mobility support. I go on to discuss in further detail, the overall ideas and implications of my research. I end with a description of potential future work.
Chapter 2

State of Network Mobility Support

In this chapter, I survey the current state of network mobility support. In the first section, I look at the fundamentals of mobility and its key objectives. I then explain how these objectives are achieved at the Network Layer. I look at the evolution of the Tunnelling Approach and its significance to Mobile IP and NEMO. Subsequently, I explore the Naming Approach and alternative proposals to network mobility support.

2.1 Host Mobility

The historic development of end-host mobility solutions can be broadly categorised into two phases. The first phase includes the period leading up to and including the standardisation of Mobile IP by the IETF. The second phase carries on from the first phase till today. These two phases are characterised by the types of end-host mobility proposals which can be categorised into two distinct approaches, the Tunnelling approach characterised by phase one and the Naming approach characterised by phase two.

2.1.1 Phase One: Tunnelling

Representative of this period of research in mobility is the Tunnelling approach or what I will refer to as the Classic Mobility Approach. This approach uses two IP addresses instead of one, one is static and represents a home address and the other is dynamic and depends on a mobile host’s current location in the network. Both these IP addresses are managed by network entities (usually a Home Agent and a Foreign Agent), that act as harbours for the mobile hosts and also as intermediaries, tunnelling packets to and from each other. This is of course a very broad description and there can be many varieties of implementation that all use this approach.

The Tunnelling approach is a result of a practical solution adopted to enable mobility. There is no fundamental restructuring of the underlying semantics of the Network Layer. This approach reuses existing entities to get round the problem of addresses changing due to mobility. It is a unique example of the flexibility of the Internet. There are usually numerous solutions to a particular problem and each has its own advantages and disadvantages. For the case of the Tunnelling approach, one may argue that it does not resolve the crux of the problem. However, the advantage is that, this solution does not require
any changes to the current protocols being used and is a relatively quick and simple fix, to what could otherwise be a complex solution with many implications.

The popularity of the Classic approach can be seen in the number of early mobility proposals such as [10] [11] [12] and [13]. The research of this period culminated in what we know today as Mobile IP [15], which has been accorded Standard status by the IETF and is available in almost all major operating system network stacks. Unfortunately, this protocol has not been widely accepted, as seen by its rare deployment and the fact it is usually not turned on by default.

Between 1991 and 1994, a substantial number of mobility solutions subscribed to the Classical mobility approach [12] and differed mainly in practical implementation (e.g. example protocols such as MosquitoNet did not implicitly require a Foreign Agent, even though this functionality was evident [14].) [20] analysed the protocols architecturally and distilled them down into the specific functionalities required such as a Forwarding Agent, a Address Translation Agent and a Local Directory. It described the Classic approach as a two tier addressing scheme. It also distinguished mobility as an Internet Naming and Addressing Problem and asserted that as a result, mobility was best handled at the Network layer.

Another survey paper is [21], which did a detailed comparison on of existing mobility proposals based on the criteria of: performance, security, deployment, scalability and robustness. [22] took a different approach by analysing the scope of mobility possible with each protocol, and focused on analysing how well each protocol scaled. It also did a comparative analysis with system parameters although not much detail is given. It compared the mobility solutions in terms of design implications, with an emphasis on system performance analysis rather than an architectural one.

Regarding the implementation of mobility within the network, [23] asks the fundamental question of how best to support mobility and revisited the TCP/IP layer. At the network layer, it used two examples, Mobile IP (Classic Mobility Approach) and LIN6 (Split Locator-Identifier Approach) only. It ends with a comparison of the different paradigms of Internet mobility support, the functional and performance aspects, as well as the changes required for implementation of the solutions.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Layer</th>
<th>Characteristic</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>MosquitoNet</td>
<td>Network Layer (IP in IP)</td>
<td>Mobility Aware</td>
<td>MH, HA</td>
</tr>
<tr>
<td></td>
<td>Link Layer (ARP Gratuitous)</td>
<td>Software Entities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network Layer (IP in IP)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMHP</td>
<td>Link Layer (ARP Gratuitous)</td>
<td>&quot;</td>
<td>MN, HA, FA, CN</td>
</tr>
<tr>
<td>Harvard System</td>
<td>&quot;</td>
<td>&quot;</td>
<td>MN, HA, FA, CN</td>
</tr>
<tr>
<td>Columbia</td>
<td>&quot;</td>
<td>Embedded Mobile</td>
<td>MSR</td>
</tr>
<tr>
<td></td>
<td>&quot;</td>
<td>Router Network</td>
<td></td>
</tr>
<tr>
<td>Sony Scheme</td>
<td>Transport Layer</td>
<td>Virtual &amp; Physical Layer</td>
<td>Modified Router</td>
</tr>
<tr>
<td>Migrate</td>
<td>&quot;</td>
<td>TCP Migrate Option</td>
<td>FQDN &amp; Connection ID</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of Early Classic Approach Protocols.
Mobile IP

Mobile IP is a mobility extension of the existing Internet Protocol. Originally called IP Mobility Support for IPv4, it was defined in [24] then updated in [25], and subsequently [26]. The following section is a direct summary from the referenced articles, and is still in the process of being summarised.

Mobile IP has the following key features [27]:

1. mobile devices can change their physical network attachment method and location while continuing to use their existing IP address.

2. the overall scheme for addressing and routing as in regular IP is maintained. No new routing requirements are placed on the inter-network.

3. mobile IP devices can still send to and receive from existing IP devices that do not know how Mobile IP works, and vice-versa.

4. the changes made by Mobile IP are generally confined to the network layer.

5. changes are required to the software in the mobile device, as well as to routers used directly by the mobile device.

Mobile IP packet re-direction is what enables mobility of nodes. Packets from a Correspondent Node (CN) will always travel towards a Mobile Node’s (MN) Home Address, where it will be received by the mobile node’s Home Agent (HA). If the MN is not in its Home Network and is instead in a Foreign Network, the HA will tunnel the CN packet (IP over IP) to the Foreign Agent (FA), which is the equivalent of a HA on a Foreign Network. This FA will then forward the packet to the MN. The packets from the MN to the CN travel directly to the CN, unless of course the CN is also mobile. In this case, the packets will be sent to the CN’s HA. This form of forwarding results in what is known as triangular routing. Although, it enables the network connections to get through over the mobility of the nodes, it results in inefficient routing of packets. This increases the latency of packet travel, especially from the CN to the MN. This path asymmetry may affect the protocol behaviour in the higher layers(e.g. TCP ACK clocking behaviour for rate control) [28].

Mobile IPv4 and Mobile IPv6 are based on the same underlying concepts, but the implementation details are somewhat different. Similar to Mobile IPv4, a CN that wishes to communicate with a MN sends packets to the MN’s Home Address. The HA located on the subnetwork of the Home Address will forward traffic to the MN. The MN’s current location is indicated by its Care of Address, which is used as a locator. Traffic then tunnelled (IP over IP) between the Home Agent and the Care of Address. The HA also responds to IPv6 Neighbour Discovery protocol messages, including Duplicate Address Detection (DAD), that are intended for the MN. Because DAD greatly increases the network handoff time, the IETF is exploring various methods to specifically reduce this inefficiency [29, 30]. Mobile IPv6 introduces a new Mobility Header which is used to carry various mobility-related control messages between the MN and the HA. These control messages permit the MN to inform the HA of any changes to its current location, including when the MN comes home to its Home Address.
Unlike Mobile IPv4, packets from the MN are tunnelled back to the Home Agent, decapsulated from the tunnel by the HA, and then forwarded to the packet destination. This IPv6 tunnelling incurs a fixed 40 byte overhead per packet tunneled. There is no triangular routing in IPv6. This ensures that packets from the MN will not be dropped due to ingress IP address filtering [31]. There is a trade off in that this method of sending packets is computationally expensive, increases latency, and causes packet fragmentation.

In order to improve efficiency, Mobile IPv6 has an optional mechanism to provide Route Optimisation. With this mechanism, the MN informs the CN of its actual location within the network by exchanging binding update (BU) messages. This optimisation reduces the chance that packets will need to be fragmented, and generally reduces the round-trip time, but the additional overhead of the Home Address Option or routing Header means that some packets will still need to be fragmented prior to transmission and reassembled upon receipt [28].

2.1.2 Phase Two: Naming and Addressing

Fundamentals

With my experiments, I have chosen to focus specifically on the network layer as network mobility support is about managing the mobility of a network. While there are a myriad of ways in which this may be achieved, one of the strongest existing paradigms sees mobility as an Internet Naming and Addressing Problem best handled at this layer [20].

Objectives

Network mobility can be described as the ability to initiate, receive and maintain existing network flows, while changing network attachment. To be able to initiate network flows, a mobile node has to have a routable address in its new network location. To receive packets, the correspondents of the mobile node have to be informed of its new IP address. To maintain existing network flows, the current correspondents have to be signalled to receive and send packets to the new IP address.

Information

By seeing mobility as a Naming and Addressing problem, we see that all network flows require two pieces of information, identity and location. They are key pieces of information that are required for network routing. They are defined as follows.

1. **Identity:** This is a name that acts as a fixed point of reference, and is used to uniquely identify a host regardless of where that host happens to be: it is an invariant that allows end-to-end state to be maintained. So, this is what applications and/or end-to-end protocols (e.g. transport layer and application layer protocols), should use.

2. **Location:** This is a name that provides information about the location (topological) of the mobile host, and is used for routing. As such, this should be updated as a host moves.
Functions

Similarly, for mobility, it can be seen as to require two key functions: (i) locating the mobile host for initiating communication; and (ii) preserving communication sessions when a node moves [32]. I have chosen to represent these key functions as two distinct two phases:

1. **Initialisation**: when a mobile entity (node or network) encounters a network access point to wider connectivity beyond the scope of itself.

2. **Handover**: the process of maintaining communication as the mobile entity changes location.

After the Handover phase a mobile entity should be able to continue using existing network flows and also be able to initiate and receive new network connections from correspondents. Both Initialisation and Handover may involve a mobile node, or a mobile network. In the following experiments, this idea is extended to model mobile network handovers.

The phrase Naming and Addressing is used here to recognise the purpose of the Loc/ID split, and that is to have unique semantics for the name of a network node and the address of a network node, as defined in [33]. The Locator is simply the Address of a node and the Identifier is its name. [34] also recognises the importance of naming and addressing for next generation architectures. It supports the split, with one reason being that location is intrinsically dynamic, such as is the case for mobility. This approach also tackles one of the main causes of routing scalability which is being faced today [35]. It is important to note that Name and Address are not the only two identifiers that can be distinguished in the Internet. On a macro-level, it is possible to define the Internet as an example of a multiple-identifier network with a hierarchy of identities [36]. On a micro-level, it is also possible to further distinguish Addresses into Location Specific Identifiers and Location Independent Identifiers [37]. Because of its potential, this approach is gaining popularity not just for end-host mobility, but in other problem spaces as well. For example, the ITU-T Study Group 13 has been discussing the ID/locator split concept and has recently approved a Recommendation (ITU-T Y.2015) that outlines the general requirements for introducing the concept in the NGN functional architecture [38].

However, this approach is not without its own issues. One main problem is managing the mapping between Name and Address [39]. Two other possible problems are (i) Locator Path Liveness Problem and (ii) State Synchronisation Problem. For the first problem, an example of a multi-homed site is used. In this example scenario, the problem is defined as how to detect that a path is down, detect an available alternative path and resume network traffic. The second is related to network-based Map-and-Encap architectures (e.g. LISP) and refers to the transfer of traffic state between one Ingress Tunnel Router (IGR) and another, in the event that the first is down [40].

Below is a summary and overview of relevant split Loc/ID end-host mobility proposals in chronological order.
GSE

The GSE proposal can be said to be one of the earliest split locator/id proposals. Originally conceived as the 8+8 protocol [41] and later updated to GSE, it was certainly the first seriously considered for adoption by the IPng Working Group in 1997. An interim meeting was held for its consideration, but it was ultimately not accepted [42]. The GSE proposal is an updated version of the 8+8 proposal by Mike O’Dell, originally conceived by David Clark.

GSE is a routing architecture for IPv6 which distinguishes (like all split Loc/ID proposals) Location and Identity. It splits the existing routing space into Site routing space and Global routing space. The task of resolution is handled by the site border routers. Because site routing bits are redundant in the global space and vice versa, the site border does not include both within IP packets at any time. The result of this is that the full IPv6 address is hidden from the node, a node is only aware of its site local routing address and its end system designator. The result of two separate routing spaces allows for aggressive routing aggregation in the global routing space and as well as homing independence for site administrators.

Split Naming and Forwarding Architecture

The SNF architectural proposal [43] splits the existing network layer into two distinct domains/spaces, a Naming Layer and a Forwarding Layer. This semantic separation allows for naming and forwarding functionality to be implemented by different routing systems based on the necessary requirements.

SNF uses IP at the forwarding layer and DNS at the naming layer. The SNF prototype Sing, uses three addressing units: the name, the locator, and the ephemeral correspondent identifier (ECI). In the SNF proposal, a host can resolve a name (FQDN) into a location from DNS. ECI together with port numbers are used in the transport layer to identify different packet flows.

This clean separation of functionality into distinct spaces allows for cleaner semantics. It also allows for easier development and replacement of each implementation. And this ties in with the design for change principle and designing for centrality of tussle space [44].

SHIM6

The Shim6 proposal [45] specifies a shim approach within IP. The IP layer is split into a IP Endpoint Layer and an IP routing Layer. The Endpoint Layer space consists of endpoint identifiers, while the routing Layer space consists of locators. The shim layer provides a set of associations between endpoint identity pairs and locator sets. 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 the endpoint.

The important distinction with other split Loc/ID proposals is that all mapping and resolution is carried out within the end point shim layers. Essentially local address rewriting
is carried within the network stack itself.

Shim6 is specially designed for ease of deployment and no changes are required for applications. There is no need for a global database for the location and identification values, as all information is host-centric. All connection state is stored at the fate-sharing endpoints of a session, to this end the authors claim that this approach is fully compliant with the Internet end to end principle.

**HIP**

HIP [46] proposes a new name space for host identity. This space is globally unique and is chosen to be the Public Key of a Public/Private Key pair. It introduces a separation between the host identity and the location identity. The IP address remains as the locator. Although its original specifications do not include the usage of DNS, recent experiments have encouraged the author to consider its usage. The reason being that it is difficult for any user to remember all other Hosts Identities, resulting in a scalability problem.

The fact that HIP uses a cryptographic Identifier has larger architectural and engineering implications in its implementation. Its difference with other split Loc/ID is in these security implications.

**LISP**

LISP [47] is designed to be an incremental, network-based protocol that separates Internet address space into identifier (EIDs) space and locator (RLOCs) space. This has the effect of semantically unbinding name and location within the IP address space. It requires no changes to host stacks but some changes to existing database infrastructures. LISP implementation approach of Map-and-Encap specifies the need for two kinds of tunnel routers, egress and ingress, which are the main instruments for crossing the identifier and locator space. LISP enabled routers perform RLOC to EID mappings, where EID is the site local IP address. The egress router appends a globally routable RLOC on all site generated packets, vice versa for ingress routers and incoming packets. However, LISP only supports two levels of tunnelling (because of overhead and looping).

**2.2 Network Mobility**

With the potential of mobile network applications, there was a corresponding push for the development of mobile network protocols that supported them. I survey two main protocols here. The first is NEMO, which is an extension of Mobile IP adopts the same fundamental approach of tunnelling. And the second is ILNP which sees mobility and a Naming and Addressing problem and adopts and Locator and Identifier approach.

**2.2.1 NEMO**

NEMO is an extension to Mobile IPv6 that provides continued connectivity for nodes within mobile networks. Currently, work in NEMO is being progressed in the Mobility
Extensions for IPv6 (MEXT) working group of the IETF. The main purpose of this group is to create a more complete mobility solution for IPv6. It is a mature RFC and implementations of it already exist.

To date, there are two main implementations of the NEMO protocol that have been validated and are freely available. The first is NEMO Platform for Linux (NEPL)\(^1\). This implementation is for Linux kernel v2.6 and was originally based upon a previous Mobile IPv6 implementation for Linux MIPL2\(^2\). NEPL has been developed and tested together by the Go-Core Project (Helsinki University of Technology)\(^3\) and the Nautilus6 Project (WIDE)\(^4\). The second implementation of NEMO is for the BSD platform, and is named SHISA\(^5\). This implementation includes source for Mobile IPv6 as well as NEMO. The current version of SHISA is based upon [49]. SHISA is supported by FreeBSD, NetBSD and OpenBSD. Both the SHISA and NEPL implementations are written to conform to network mobility as defined for the IETF NEMO WG\(^6\). This is a general specification and there are still some issues that remain unresolved that impede it from being adopted more widely\(^7\).

In order to better understand the open issues presented earlier, in the upcoming sections, I show how mobility is enabled in a mobile network using network mobility support protocols such as NEMO and ILNP.

**NEMO Operation**

NEMO enables network mobility by using an additional IP address, the *Care of Address (CoA)*, for the Mobile Router (MR). The CoA can be seen as a temporary address used by the MR as it moves. The CoA allows packets to be routed to the current location of the MR. The CoA acts as a topological *locator* for the mobile network. Meanwhile, the MR maintains another IP address that is available via DNS, its *Home Address (HoA)*, topologically located at its ‘home network’ (the IP sub-network to which the HoA belongs), and this is used for maintaining session state with Correspondent Nodes (CNs). The HoA acts as an invariant *identifier*, and is used for transport layer state. When the MR is not at its home network, the Home Agent of the MR (HA\(_{MR}\)) acts as a proxy for the MR, forwarding packets received at the home network (using the HoA) to the MR (using the CoA), via a bi-directional, IP-in-IP tunnel. Traffic from within the mobile network is sent to the MR. This traffic is encapsulated through this tunnel back to the HA where it is decapsulated and forwarded. To correspondent nodes (CNs), the mobile network appears to be within its home network.

This approach allows the MR and its Visiting Mobile Nodes (VMNs) to maintain pseudo-end-to-end connectivity despite changing network attachment points. The VMN achieves this by keeping its own Home Agent (HA\(_{VMN}\)) updated with its new CoA, using Mobile IPv6, as it moves. This approach does not change the way the IP address is used today, and there is no impact on the IP address structure. There are also no additional changes required to the IP architecture. The location of the mobile network is inconsequential so long as the MR and its HA\(_{MR}\) can set-up and maintain the bi-directional tunnel between them.

\(^1\)http://www.nautilus6.org/implementation/index.php
\(^2\)http://go.cs.hut.fi/
\(^3\)http://www.nautilus6.org/
\(^4\)http://www.mobileip.jp/
When a MR running NEMO migrates to a foreign network, it replies to any Routing Advertisements it receives from the local Access Gateway (AG), to receive a new CoA on the visited link (we assume the MR is operating as a mobile router and not as a mobile host). It is this AG that provides network connectivity for the MR and the nodes within it. The MR then sends a Binding Update (BU) message to its HA$_{MR}$, informing it of its change of CoA (See Figure 2.1 step (1)). The HA$_{MR}$ updates its HoA-to-CoA cache for that MR and replies with a Binding Acknowledgement (BA). This act sets up and maintains the bi-directional tunnel between them.

Packets meant for the MR are received by the HA$_{MR}$, which then uses IP-in-IP encapsulation to forward the packets to the MR at its latest CoA. All egress packets from the mobile network, sent from each VMN to its CN, must follow the same return path through the MR-HA$_{MR}$ tunnel first before proceeding to its own respective HA$_{VMN}$ (See Figure 2.1 step (6)).

A mobile host has its own Home Address (HoA$_{VMN}$), which is always returned when a DNS lookup is performed for that mobile host. When this host becomes a VMN and joins a NEMO mobile network, it first must receive its new CoA (Figure 2.1 step (2)). It then updates its HA$_{VMN}$ with its CoA by sending a Binding Update (BU) message (See Figure 2.1 step (3)). The HA$_{VMN}$ responds with a Binding Acknowledgement (BA). If the VMN is communicating with any MIPv6-aware CNs (and they are mobility-aware), it will execute a return routability test (RRT) (Figure 2.1 step (5a)) and subsequently update its CNs with its new CoA, via a BU/BA exchange (Figure 2.1 step (5b)).

Upon receiving its CoA, a VMN running MobileIPv6 maintains its own bi-directional tunnel between itself and its own HA$_{VMN}$. Operationally, the VMN-to-HA$_{VMN}$ tunnel exists within the MR-HA$_{MR}$ tunnel. Mobility of the MR and VMN is hidden as all traffic eventually is sent to/from their respective HA$_{VMN}$s.

If the MR changes location, it will again negotiate and receive its new CoA and update its HA$_{MR}$ with its new location (Figure 2.1 step (4)). The HA$_{MR}$ then updates its Binding Cache and the bi-directional tunnel is maintained as it forwards MR packets to the new location.

As for the VMN within the mobile network, it will be unaware of its own mobility as the MR ensures that address on its ingress interface remain unchanged. The mobility of the MR only affects its egress interface. As a result, the VMN will not execute any handovers with its HA$_{VMN}$ or its CNs (if any).

### 2.2.2 ILNP

The Identifier Locator Network Protocol (ILNPv6) [28] is an experimental extension to IPv6. The term Identifier-Locator Network Protocol for IPv6 (ILNPv6) is used, as it can be engineered as enhancements to IPv6 [28, 51–53]. It splits the IP address into two parts, the Identifier and the Locator. The Identifier is used for end nodes through the transport layer and the Locator is used to send packets to the destination sub-network through the network layer. This will benefit the network architecture as it removes the semantic burden on the IP address. With ILNP, the identification of an end node is no longer bound to its topological location on the network. This is believed to aid node mobility, multi-homing, and security. It is important to note that the idea of splitting the IP address is
Figure 2.1: The phases of initialisation and handover for a VMN (running Mobile IPv6) and MR (running NEMO). Step (1) shows the MR updating its $HA_{MR}$ via $AG_1$. Step (2) shows a VMN arriving at the mobile network and registering an IP address gained from the MR. Step (3) shows the VMN updating its own $HA_{VMN}$. Step (4) shows the MR moving and conducting a handover by informing its $HA_{VMN}$ of its new CoA. Step (5a) shows the VMN executing a RRT with its CNs. Step (5b) shows the VMN updating its CNs with a new CoA.

Figure 2.2: 2 packets, total of 288 bytes

With regards to the network layer, ILNP maintains additional session cache that holds all the Identifier and Locator mappings. Only the Identifier is exchanged between the network layer and the transport layer above it. In addition, the ILNP extension uses ICMP Locator Update messages to handle mobile hosts and multi-homing. When a host changes Locator, it sends out these messages to its clients/servers to communicate this change. The host is also able to process/receive Locator Update messages accordingly. (ICMP Locator Update messages can also be used to trigger the end host to get a DNS update.)
With the ILNP extension to IP, TCP will have to be modified slightly, because the network layer will only pass the Identifier to the transport layer. This means that only part of the IP address is required and used. This means that even if a host changes its location, the TCP connection will not break. This is one of the fundamental benefits of ILNP and the splitting of the IP address. And with regards to implementation, the most significant 64-bits of the ILNP address, the Locator, coincides with the IPv6 routing prefix. As a result
the core routers and routing protocols do not have to change. Nodes that are attached to the mobile network have DNS LP records that point to a common DNS L record covering the entire mobile (sub-)network. The common L record would be updated by the MR whenever its uplink moves to a different layer-3 IPv6 network.

**ILNP Operation**

Let us assume that the mobile network has an external link with Locator $L_1$ at access router AG$_1$. This will be held in a DNS L record pointed to by a DNS LP records for each host in the mobile network (Figure 2.6 step (1)). Within the mobile network, localised addressing is used through Locator rewriting in ILNPv6. That is, a local (private) Locator value, $L_L$, is used by all nodes in the mobile network, and for all egress packets, the MR rewrites $L_L$ to $L_1$, and performs the complimentary operation for ingress packets, i.e. this is the ILNPv6 equivalent of NAT, but unlike IP, does not violate end-to-end state and is completely transparent to all ILNPv6 nodes [52]. So, *Initialisation* for a VMN occurs through a VMN receiving Router Advertisements containing information about $L_L$ and the L record name for the mobile network, updating its LP record to point to the L record of the network (Figure 2.6 step (2) and Figure 2.7).

Now, let us assume a *Handover* is triggered for the link currently using $L_1$. A signal is detected in the new cell and a new Locator value, $L_2$, is attained from the Access Gateway (AG$_2$). This can be done through normal IPv6 discovery mechanisms, as Locator values are identical to IPv6 network prefixes. We will assume that the radio cells providing coverage, $L_1$ and $L_2$ overlap. Then, the MR updates the DNS L record (currently holding value $L_1$) to $L_2$ (for new sessions) (Figure 2.6 step (3) and Figure 2.8). It then starts updating the state of existing sessions using value $L_1$ to using value $L_2$, by issuing Locator Update (LU) messages (synonymous to Binding Update message in IPv6) for correspondents using $L_1$ (Figure 2.6 step (4) and Figure 2.9). It then transitions sessions from $L_1$ to $L_2$ using Locator rewriting. When no more packets arrive from remote locations using $L_1$ within a given time period (i.e. all sessions have transitioned to $L_2$), the connection is considered to have completed handover. This is a *soft handover* at the ILNPv6 layer, something that is not currently defined for IPv6 or NEMO. Note that the MR is providing this capability efficiently for the whole mobile network. Note also that during this time, it would also be possible to have another MR and have the whole mobile network multi-homed [52].

It is also possible to use ILNPv6 for normal handover, simply by switching to $L_2$ as soon as possible. Any packets in flight addressed to $L_1$ may be lost, but can be recovered through the retransmission capability in TCP, for example, albeit this would be inefficient, as it may invoke the congestion control behaviour of TCP as TCP ACKs are lost or delayed.

**ILNP and DNS**

The DNS [58, 59] provides a globally distributed name resolution system for the Internet. Today, for practical purposes, DNS is essential for the operation of the Internet: without DNS, many network services would appear to be disconnected. Given the global presence and functionality of DNS, many have proposed utilising DNS to enable host and network mobility [28, 32, 60–63]. Indeed, the proposal in [60] takes advantage of the fact that a host name lookup is ubiquitously performed by most applications that originate
Figure 2.6: This figure shows the 2 phases of Initialisation and Handover for a VMN and MR for ILNPv6. Step (1) shows the MR arriving at a new location, receiving an address from AG\textsubscript{1} and updating its DNS L record with its latest location. Step (2) shows a VMN arriving at the mobile network, receiving a new (local) Locator and name of the L record for the network, then updating its DNS LP record. Step (3) shows the MR moving to a new location, receiving a new Locator from AG\textsubscript{2}, and updating its DNS L record with this new location. Step (4) shows the MR updating all existing sessions between its VMNs and their CNs.

communication with a network host, and uses the DNS name as the invariant, rather than the IP address. This use of a FQDN is consistent with the recommendations for use of names in RFC 1958 [64].

DNS utilises 3 records, they are A (IP address of a domain, 32bit), AAAA (IP address of a domain, 128bit) and PTR (domain name to IP address, used for reverse lookup). ILNP adds an additional 4 new resource records, the I, L, PTRL and PTRI. The I and L records are the Identifier and Locator records associated with a domain name. The PTRL record is used to name the authoritative DNS server for a given Locator. The PTRI record is used to find the domain name for a given Identifier in the context of a specific subnetwork (Locator). Keeping the records for Locator and Identifier separate is more efficient as different hosts will have different levels of mobility. It also makes it easier to assign longer Time To Live (TTL) to records that are more likely to be static. Security is provided by the existing DNS Security specification [65]. Dynamic DNS Updates can be provided by the existing Secure Dynamic DNS Update specifications [66].

In ILNPv6, the mobile network ‘site’ uses private addressing internally (to the site network) and the network’s Mobile Router (MR) rewrites the Locator values of nodes within the site as packets transit that MR. (Note that Locator re-writing does not affect end-system state, as only the Identifier is used by the Transport layer.)

Some common arguments against the use of DNS for mobility are that (i) DNS will not be able to cope with the additional load of large scale mobility; (ii) DNS is too slow for loca-
tion update to propagate within DNS in a timely manner; and (iii) that DNS is insecure. However, we take the position that DNS can not only be used to enable host/network mobility, but that it will be extremely capable of doing so:

1. DNS is robust, as lookups and updates are distributed across administratively-delegated, replicated DNS servers [60]. Use of DNS for mobility is as secure as regular DNS, since Secure Dynamic DNS Update [66] is standardised and widely implemented [53].

2. Traffic caused by mobility will be relatively small, as DNS today deals with a load where close to 50% of DNS traffic is caused by misconfigurations, aggressive retransmissions and poor caching [67].

3. Current implementations of DNS are suitable for use in mobility solutions that require DNS updates at rates as frequent as once per second [68]. Experimental results from [69] show that BIND implementations of DNS with dynamic update can support mobility solutions.

4. Findings from [70] suggest that DNS performance will not be degraded with the widespread use of dynamic, low TTL A-record bindings commonly associated with mobility. Large scale mobility will effect leaf DNS servers, and will have little or no effect on root, top-level-domain (TLD), or even the top-of-the-user-domain DNS servers [53].
So, there are strong indications from previous work to suggest that DNS would be suitable for supporting mobility.

**IPv6 Enhancements**

The IPv6 packet header and the ILNPv6 packet header are deliberately made similar. Essentially, in ILNPv6, the IPv6 address is broken into two separate components, a Locator (L) and an Identifier (I). Significantly, the IPv6 Interface Identifier is replaced by an ILNPv6 Node Identifier (I), with slightly different semantics. The Naming Approach recognises explicitly the dual roles of IP addresses today – as a Locator and as an Identifier. The Locator
names an IP (sub-)network: this is used only in routing, and not by the upper layers (e.g. not used by TCP or UDP). In practise today, the L value in ILNPv6 packets is exactly the same as the top 64 bits of the IPv6 address, and includes the routing information (see Figure 2.10). The Identifier is only used for node identity (e.g. for TCP or UDP session state).

The idea of an Identifier/Locator split is not new, but the ILNP particular approach is new and is specified in more detail than preceding proposals [55,71,72]. ILNPv6 supports the recommendations of RFC 1958 [64], that applications should use fully-qualified domain names (FQDNs), wherever possible. A summary of the difference between the use of names in IP (v4 and v6) and the use in ILNP is given in Table 2.2.

<table>
<thead>
<tr>
<th>Protocol layer</th>
<th>ILNP</th>
<th>IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>FQDN</td>
<td>FQDN, IP address</td>
</tr>
<tr>
<td>Transport</td>
<td>Identifier, I + port no.</td>
<td>IP address + port no.</td>
</tr>
<tr>
<td>Network</td>
<td>Locator, L</td>
<td>IP address</td>
</tr>
<tr>
<td>Link</td>
<td>MAC address</td>
<td>MAC address</td>
</tr>
</tbody>
</table>

Table 2.2: Use of names in ILNP and IP.

The Locator (L) is an unsigned 64-bit value carried in the upper portion of the IPv6 address and is equivalent to an IPv6 address prefix (see Figure 2.10). The (Node) Identifier (I) is an unsigned 64-bit value carried in the lower portion of the IPv6 address. The I value names a (virtual) node itself, rather than the network interface of a node. An end-system may use multiple I values and multiple L values simultaneously. For the duration of a given ILNP session, its I value should remain constant. For practical reasons, the I value is normally formed from one of the MAC addresses associated with the node. This is represented in the IEEE’s EUI-64 syntax, and is very likely to be globally unique as well. This usage is consistent with the IPv6 Addressing Architecture [73]. Strictly, the I value must be unique only within the scope of the L value with which it is used. However, for practical purposes, having an I value that is likely to be globally unique is very useful, and allows us to dispense with IPv6 Duplicate Address Detection (DAD), which in turn greatly reduces the time required for a node to execute a location change.

IPv6:

<table>
<thead>
<tr>
<th>3</th>
<th>45 bits</th>
<th>16 bits</th>
<th>64 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+---------------------------------+---</td>
<td>---------------------+</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>global routing prefix</td>
<td>subnet ID</td>
<td>Interface Identifier</td>
</tr>
<tr>
<td></td>
<td>+---------------------------------+---</td>
<td>---------------------+</td>
<td></td>
</tr>
</tbody>
</table>

ILNPv6:

<table>
<thead>
<tr>
<th>64 bits</th>
<th>64 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locator</td>
<td>Node Identifier</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
</tr>
</tbody>
</table>

Figure 2.10: IPv6 address format (from RFC 3587 [74]) as used in ILNPv6: Locator values are IPv6 routing prefixes.

Current IPv6 address allocation practices provide sites with IPv6 address blocks that are 48-bits long, which leaves 16 bits for intra-site subnetting. As the ILNPv6 network name
(ILNPv6 Locator) is the same as an IPv6 routing prefix, ILNPv6 packets can travel across existing deployed IPv6 backbones. Only the host’s IPv6 stack has to be enhanced to enable ILNPv6 on that host (i.e. to deal with Node Identifier values). ILNPv6 Neighbour Discovery (ND) still uses the full 128-bits of the combined I:L value. So IPv6 ND can also be used without change.

**ILNP Security Considerations**

In IPsec [75] today, the IPsec Security Associations (SAs) are bound to full IP addresses at the local and remote sites as a form of end-system identity. So, IPsec requires that the IP addresses at each end-point of the communication remain fixed. For mobility (as well as use of localised addressing and multi-homing), this may not remain true, and so IPsec has had to be modified, retrospectively, in order to cope with these functions.

With ILNP, however, IPsec SAs are bound only to the Identifier, never to the Locator. This makes it easy for the IPsec Security Association – and the related secure communications channel – to remain operational even if the end-points move.

For DNS, the existing Secure Dynamic DNS Update standard [66] permits a mobile node to update its $L$ records when the node moves. Widely used systems, such as Microsoft Windows or the BIND software used with UNIX, already include support for Secure Dynamic DNS Update [76].

So, ILNPv6 simply uses existing security standards for enabling mobile networks, and does not introduce any new security risks compared to IPv6.

**Mobile Network Operation**

In ILNPv6, the mobile network ‘site’ uses private addressing *internally* (to the site network) and the network’s Mobile Router (MR) rewrites the Locator values of nodes within the site as packets transit that MR. (Note that Locator re-writing does not affect end-system state, as only the Identifier is used by the Transport layer.)

Nodes that are attached to the mobile network have DNS $LP$ records that point to a common DNS $L$ record covering the entire mobile (sub-)network. The common $L$ record would be updated by the MR whenever its uplink moves to a different layer-3 IPv6 network.

Let us assume that the mobile network has an external link with Locator $L_1$ at access router $AG_1$. This will be held in a DNS $L$ record pointed to by a DNS $LP$ records for each host in the mobile network (Figure 2.6 step (1)). Within the mobile network, localised addressing is used through Locator rewriting in ILNPv6. That is, a local (private) Locator value, $L_L$, is used by all nodes in the mobile network, and for all egress packets, the MR rewrites $L_L$ to $L_1$, and performs the complimentary operation for ingress packets, i.e. this is the ILNPv6 equivalent of NAT, but unlike IP, does not violate end-to-end state and is completely transparent to all ILNPv6 nodes [52]. So, initialisation for a VMN occurs through the processing of inbound Router Advertisements containing information about $L_L$ and the $L$ record name for the mobile network. The VMN then updates its $LP$ record to point to the $L$ record of the network (Figure 2.6 step (2) and Figure 2.7).

Now, let us assume a handover is triggered for the link currently using $L_1$. A signal is detected in the new cell and a new Locator value, $L_2$, is attained from the Access Gateway.
This can be done through normal IPv6 discovery mechanisms, as Locator values are identical to IPv6 network prefixes. We will assume that the radio cells providing $L_1$ and $L_2$ overlap. Then, the MR updates the DNS $L$ record (currently holding value $L_1$) to $L_2$ (for new sessions) (Figure 2.6 step (3) and Figure 2.8). It then starts updating the state of existing sessions using value $L_1$ to using value $L_2$, by issuing Locator Update (LU) messages (synonymous to Binding Update message in IPv6) for correspondents using $L_1$ (Figure 2.6 step (4) and Figure 2.9). It then transitions sessions from $L_1$ to $L_2$ using Locator rewriting. When no more packets arrive from remote locations using $L_1$ within a given time period (i.e. all sessions have transitioned to $L_2$), the connection is considered to have completed handover. This is a soft handover at the ILNPv6 layer, something that is not currently defined for IPv6 or NEMO. Note that the MR is providing this capability efficiently for the whole mobile network. Note also that during this time, it would be possible to have another MR and have the whole mobile network multi-homed [52].

ILNPv6 maybe used for normal handover, simply by switching to $L_2$ as soon as possible. Any packets in flight addressed to $L_1$ may be lost, but can be recovered through the retransmission capability in TCP, for example, albeit this would be inefficient, as it may invoke the congestion control behaviour of TCP as TCP ACKs are lost or delayed.

### 2.2.3 LIN6-NEMO

LIN6 [77] is an application of LINA (Location Independent Network Architecture) to IPv6. LIN6 is a protocol supporting mobility and multi-homing in IPv6. LIN6 creates an additional separate routing space for routing aggregation. It uses additional Mapping entities to handle resolution across the identifier name space (LIN ID) and the routing name space (Network Prefix). The distributed network of Mapping Agent entities is essentially a distributed resolution service, somewhat akin to the Home and Foreign Agent entities in Mobile IPv6.

If a node needs to communicate with another node across the network (assuming it already has the destination node’s FQDN), it would first approach the local DNS server with this FQDN to map it to the location of a Mapping Entity and the destination node’s LIN ID. The source node would then contact the Mapping Agent with the destination LIN ID to receive the network location of the destination node.

LIN6-NEMO [78] extends LIN6 to enable network mobility. It was also designed to solve the problems of routing redundancy and header overhead found in MIPv6-NEMO. The key features of these solutions are the implementation of a node identity and location in the forms a LIN ID a network prefix, as well as the usage of Mapping Agents (MA) that manage these dynamic mappings when it comes to mobile nodes and mobile networks.

The basic network mobility using LIN6-NEMO starts when a MR registers its current network prefix to the nearest MA (i.e. location registration). In the case of nested network, the MR registers its network prefixes (assigned by upper MR), as well as the LIN-ID of the upper MR. The inner MR must also inform all of its own mapping entries to the upper MR. As a result the MR has all mapping entries of all nodes within its own mobile network.  

When a correspondent node needs to send a packet to a mobile node. It will first resolve

\[ ^5 \text{LIN6-NEMO uses extended router advertisements which specifies the LIN6 ID of the MR.} \]
the mobile node location with existing MA using its LIN ID. If the mobile node is within a mobile network, the correspondent node will first resolve the LIN ID of the MR. Eventually, the MA will return the location of the MR of the mobile network. The correspondent node then creates the LIN6 address using this MR location and the LIN ID of the mobile node, after which it sends the packet. Upon receipt of the packet, the MR overwrites the MR location with the network prefixed assigned to the mobile node and forwards the packet. For outward going packets from the mobile node, the MR rewrites the network prefix to that of its own location.

Advantages

- No tunnelling mechanism used
- Optimal routing

Disadvantages

- CN update burst
- Currently does not support fixed nodes (nodes that dont run LIN6)
- No security mechanism
- Currently does not support mobile CNs

2.2.4 HIP-NEMO

The Host Identity Protocol (HIP) [46] is a multi-addressing and mobility solution for the IPv4 and IPv6. It proposes a new name space for Host Identity, as well as a new protocol layer. The Host Identity namespace consists of Host Identifiers (HIs), which are public keys of an asymmetric key-pairs. These are used to perform authentication and create IPSec security associations between hosts, without the need to access to certificates or a public-key infrastructure. The HIP layer is added into the TCP/IP stack between the network and transport layers, it objective is to aid in the separation of identifiers from locators. Nodes running HIP first establish session keys with the HIP Base Exchange. These are protected using IPSec ESP. HIP supports mobility and multi-homing through a readdressing mechanism.

HIP-NEMO [79], is an extension of NEMO that supports network mobility (including nested mobility) and multi-homing in a secure way. This idea is based on a HIP-based micromobility solution [80], where a new network entity, called the Local Rendezvous Server (LRVS) was introduced. The LRVS is a HIP enabled gateway router, which enables mobility for mobile nodes connected to it. HIP-NEMO extends the functionality of LRVS to allow for mobile networks.
Advantages

- All advantages from HIP are inherited
- Signaling delegation and hierarchical micro-mobility architecture should mean less signalling and packet overhead than MIPv6-NEMO
- No need for tunneling and encapsulation
- Optimal routing

Disadvantages

- Method is not transparent to MNNs, they have to register at mL RVS
- root mL RVS has to update all CNs

2.3 Summary

For many years, the tunnelling approach has been the approach adopted towards enabling mobility. Despite the maturity of these protocols, they have not been widely adopted as mobility solutions. As the potential for network mobility increases, alternative protocols have been proposed that look promising in tackling the existing challenges and open issues (See Section 1.1.3). Given the potential application of network mobility and its advantages, these new approaches should be explored. In the next chapter, I explore the impact of these alternative architectural approaches examined here.
Chapter 3

Impact of Architecture Approach in Network Mobility Support

3.1 Introduction

In the previous chapter, we looked at the current approaches to network mobility support, particularly the Naming and Tunnelling approaches. In this chapter, I analyse the impact of these architectures in relation to optimal routing. For the Tunnelling approach I use NEMO and for the Naming approach, I use ILNP. In addition to these, I also use OptiNets [81]. OptiNets extends NEMO and seeks to address NEMO’s sub-optimal routing issues.

This chapter focuses on route optimisation [5] as it is one of the key areas of concern with regards to NEMO and its real-life applications [82]. Several solutions to route optimisation for NEMO have been proposed, e.g. most recently [83,84]. One proposal, OptiNets, builds upon NEMO and was designed to address its sub-optimal routing issue. While other route optimisation solutions have been proposed for NEMO, I have chosen to use OptiNets as it has been previously analysed [16], and is a suitable candidate to consider for this study.

3.2 Route Optimisation for NEMO

One of the main issues of the NEMO Basic Support Protocol is the presence of non-optimal routing paths or dogged-legged routing. The problem is described in [85] and the solutions explored in [86]. Because NEMO enables mobility in a similar way to MIP, it has also inherited all its drawbacks [87]. Several solutions to route optimisation for NEMO have been proposed, (e.g. most recently [83,84]). These observations are also experimentally observed in [16].

In section 1.1.3, the problem of nested was briefly introduced. A nested mobile network may occur in two ways:
Mobility within a mobile network

If we consider the mobile nodes within the mobile network to be capable of mobility themselves (e.g., if they run Mobile IP), then we have a scenario that is more realistic in terms of public transport networks. We could imagine individual passengers to be mobile nodes, when they use laptops or mobile phones, and the public transport vehicles themselves to be the Mobile Routers.

Even though NEMO and Mobile IP use the same general approach of tunnelling to enable mobility, there is still currently no standard built-in mechanism that allows for both protocols to co-exist simultaneously in a mobile network, while at the same time ensuring routing optimality for its mobile entities. Binding storms [88] also occurs in a mobile network that contains Mobile IP nodes upon handover.

The problem of routing optimality is further aggravated in large scale mobility scenarios. [89] discusses some of the potential problems when using MIPv6 and NEMO. It introduces the idea of Home Agent migration, where Home Agents are disengaged from their home link and distributed throughout the Internet topology.

Some proposed solutions such as [90] extends Proxy MIPv6 (PMIPv6) to incorporate network mobility. [83] proposes a new architecture called NEMO-PMIPv6 (N-PMIPv6), which integrates mobile networks in PMIPv6-localised-mobility domains.

Networks within networks

It is easy to picture that as the nested level of mobile networks increases, the routing path in an unchanged network becomes correspondingly longer. [91] classifies solutions into two categories, (i) network layer approaches (e.g. HMIPv6) and (ii) application layer approaches (e.g. SIP-NEMO). Since my research focuses on the network layer, I shall describe some of the network layer approaches.

There are certainly many different ways of approaching the problem. Some work by extending NEMO such as [92], which adds Tree discovery, NINA and RRH. These extensions are then used to efficiently route packets in nested NEMO network topologies. Other solutions seek to unify existing protocols in a hybrid approach such as [91], which introduces a protocol named HMR-NEMO, which integrates ideas from MIPv6 and HMIPv6 into NEMO-BSP. Protocols such as NERON (NEst Rout e Optimisation for NEMO) [84], seek to make the mobile nodes within a mobile network aware of the possibility of nested mobile networks. NERON enables nodes behind nested mobile networks to use optimised communication paths with zero tunnelling overhead and end-to-end delay.
3.3 OptiNets

The OptiNets protocol extends NEMO and makes route optimisation possible by having the MR advertise topologically correct network prefixes. As a result, all mobile nodes within the mobile network have topologically correct CoA(s). This allows VMNs running MIPv6 to execute Route Optimisation (RO) with RO-aware CNs, via a Return Routability Test (RRT) (Figure 3.3) and a Binding Update (BU) (Figure 3.4).

Let us analyse what happens when a VMN joins a mobile network running OptiNets. Like NEMO, in OptiNets, a mobile node has its own Home Address (HoA_{VMN}), which is always returned when a DNS lookup is performed for that mobile node. When this node becomes a VMN and joins a OptiNets mobile network, it must first receive its new CoA (Figure 3.1 step (2)). It then updates its HA_{VMN} with its CoA by sending a Binding Update (BU) message (See Figure 3.1 step (3) and Figure 3.2). The HA_{VMN} responds with a Binding Acknowledgement (BA). If the VMN is communicating with any MIPv6-aware CNs and carries out routing optimisation, it will execute a return routability test (RRT) (Figure 3.1 step (5a) and Figure 3.3) and subsequently update its CNs with its new CoA, via a BU (Figure 3.1 step (5b) and Figure 3.4).

If the MR changes location, it will again negotiate and receive its new CoA and update its HA_{MR} with its new location (Figure 3.1 step (4) and Figure 3.5). The HA_{MR} then updates its Binding Cache and the bi-directional tunnel is maintained as it forwards MR packets to the new location. The HA_{MR} then broadcasts its new address prefix via a routing Advertisement to its mobile network nodes on its ingress interface.

In a NEMO/MIPv6 setup where the VMN does not carry out any form of routing optimisation, upon receiving its CoA, the VMN running MIPv6 maintains its own bi-directional tunnel between itself and its own HA_{VMN}. The VMN-to-HA_{VMN} tunnel exists within the MR-to-HA_{MR} tunnel. In NEMO, mobility of the MR and VMN is hidden as all traffic is sent to/from their respective HA_{VMN}(s). Whereas OptiNets allows VMN-to-CN communications to bypass the MR-to-HA_{MR} tunnel entirely.

3.4 Experiment

The purpose behind the experiment is to bring to light the cost of providing for route optimisation (for mobile networks) with different approaches. I chose to factor out the contribution of the wireless layer in our experiments. As we are interested only in the architectural differences between the Naming and Tunnelling approaches, exclusion of any wireless effects allows us to confidently draw conclusions based on differences in protocol architecture only. This provides to a well-defined comparison (based on fewer variables). This also makes the emulation less complex. However, the reader should note that simulations for specific scenarios (e.g. use of WLAN for MAC/PHY) would be required for operational evaluations.

The mobile network scenario I have chosen, is that of the London Circle Line, which is a line on the public metropolitan rail system in the heart of London, UK. I have regarded the passengers boarding and leaving the Circle Line trains as VMNs. I have also regarded each train as a separate mobile network, and each arrival of the train at a new station as a movement of the mobile network that requires it to establish a new network point of
Figure 3.1: The phases of initialisation and handover for a VMN (running Mobile IPv6) and MR (running OptiNets). Step (1) shows the MR updating its HA*MR* via AG. Step (2) shows a VMN arriving at the mobile network and registering a topologically accurate IP address gained from the MR. Step (3) shows the VMN updating its own HA*VMN*. Step (4) shows the MR moving and conducting a handover by informing its HA*VMN* of its new CoA. This step also include the MR broadcasting its new Address Prefix to its Ingress interface. Step (5a) shows the VMN executing a RRT with its CNs. Step (5b) shows the VMN updating its CNs with a new CoA.

attachment. Note that this is not a simulation study. I have not used a mobility model and do not maintain state for individual nodes in our evaluation.

As detailed passenger mobility traces for the London Circle Line are not available, I will generate our own passenger mobility traces based on available information such as yearly passenger statics and train arrival times from Tubeprune [93] and Transport for London [94]. The raw data used from Tubeprune is summarised in Table 3.1 and derived data used in the generation of the mobility traces is summarised in Table 3.2.

The next step is to emulate the actual handovers and route optimisation of the different mobile network protocols using the passenger mobility traces. In Sections 3.4.1 to 3.4.1, I have formulated general equations for packet and bandwidth cost, for NEMO, OptiNets

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours of service per day (N_d)</td>
<td>18</td>
</tr>
<tr>
<td>No. of trains per station per hour (N_t)</td>
<td>7</td>
</tr>
<tr>
<td>No. of stations per hour (N_s)</td>
<td>27</td>
</tr>
<tr>
<td>Mean no. of passengers on a weekday (N_w)</td>
<td>218136</td>
</tr>
<tr>
<td>Total no. of passengers a year</td>
<td>68850600</td>
</tr>
</tbody>
</table>

Table 3.1: London Circle Line Data from Tubeprune [93].
and ILNP (See Equations (3.1) to (3.9)). These were partly obtained from the previous detailed analysis in [16] as well as my own study of the protocols, resulting in the timeline diagrams in Figures 3.2, 3.3, 3.4, 3.5.

The variables considered in this experiment are (i) the number of stations a passenger travels through, which is equal to the number of handovers ($N_h$) and (ii) the number of unique CNs per passenger ($N_{CN}$). For OptiNets, (i) affects the VMN handovers (Figure 3.2), and (ii) affects the number of VMN-to-CN updates (Figure 3.3). There are also a number of packets which have to be generated due to VMN initialisations and MR-to-HA handovers, regardless of variables (i) or (ii). VMN initialisations are dependent on the number of passengers ($N_p$). MR-to-HA handovers are dependent on the number of train stations ($N_s$). I have assumed that handovers and initialisations of VMNs occur during the time in which a train is at a station ($T_h$).

| No. of passengers (VMNs) per hour per train, Weekday ($N_p$) | 448 |
| Handover/stop time at stations per train ($T_h$) | 60s |

Table 3.2: Passenger and train movement used in emulation.
Looking at the general equations, we see that the common unknown is the duration that each passenger stays on the railway system, i.e. the Number of station hops ($N_h$). This value I will derive from our Passenger emulation.

### 3.4.1 General Equations

**NEMO Analysis**

The overhead generated by NEMO per passenger per train per second, $C_{NEMO}$, is calculated as:

$$C_{NEMO} = \frac{K_1.N_p + K_2.N_p.N_{CN} + K_3.N_s}{N_p.N_s.T_h} \quad (3.1)$$

where $K_1$, $K_2$ and $K_3$ are constants. There are three parts to the right-hand side of the numerator of this expression. The first part ($K_1.N_p$) refers to the VMN initialisation (see Figure 2.2). As each VMN will have to join the mobile network, and we have assumed each passenger only makes one trip, thereby joining only once, this number is dependent only on the number of passengers ($N_p$).

In the second part, I assume VMN executes route optimisation with its existing CNs. This includes a return routability test (RRT) (See Figure 2.1 step (5a) and Figure 2.4) and
a BU/BA (See Figure 2.1 step (5b) and Figure 2.3).

In the third part of the expression, \((K_3.N_s)\) is the overhead generated by the MR (see Figure 2.5) per train per hour for a handover. Given that there are 27 stations, \(N_s\), on this route, and that each train takes approximately an hour (on average) to finish one circuit we calculate the number of MR handovers generated per train per hour to be \(K_3.27\).

In Eqn 3.1, if we replace \(K_1\), \(K_2\) and \(K_3\) with the appropriate packets counts or byte counts (See Figures 2.2, 2.3, 2.4, 2.5), we get the packet overhead, \(N_{NEMO}\), and bandwidth overhead, \(B_{NEMO}\):

\[
N_{NEMO} = \frac{2.N_p + 6.N_p.N_{CN} + 2.N_s}{N_p.N_s.T_h} \tag{3.2}
\]

\[
B_{NEMO} = \frac{340.N_p + (560 + 280).N_p.N_{CN} + 208.N_s}{N_p.N_s.T_h} \tag{3.3}
\]

**OptiNets Analysis**

The overhead generated by OptiNets per passenger per train per second, \(C_{OPTI}\), is calculated as:

\[
C_{OPTI} = \frac{H_1.N_p + H_2.N_p.N_{CN}.N_h + H_3.N_s}{N_p.N_s.T_h} \tag{3.4}
\]

where \(H_1\), \(H_2\) and \(H_3\) are constants. There are three parts to the numerator of the right-hand side of this expression. The first part \((H_1.N_p)\) refers to the VMN initialisation (Figure 3.2). As each VMN will have to join the mobile network, I have assumed each passenger only makes one trip, thereby joining only once, this number is dependent upon the number of passengers \((N_p)\). For the second part, I assume VMN executes route optimisation with its existing CNs. This includes a return routability test (RRT) (Figure 3.1 Step (5a) and Figure 3.3) and a BU (Figure 3.1 Step (5b) and Figure 3.4), every time the MR changes location and thus cannot exceed the total number of train stations \((N_s)\). For the third part, I evaluate the overhead generated by the MRs (Figure 3.5). Given that there are 27 stations \((N_s)\), on this route, and that each train takes approximately an hour (on average) to finish one circuit, I calculate the number of MR handovers generated per train per hour to be \(H_3.27\).

In Eqn 3.4, if we replace \(H_1\), \(H_2\) and \(H_3\) with the appropriate packets counts or byte counts (See Figures 3.2, 3.3, 3.4, 3.5) we get the packet overhead, \(N_{OPTI}\), and bandwidth overhead, \(B_{OPTI}\):

\[
N_{OPTI} = \frac{2.N_p + 5.N_p.N_{CN}.N_h + 3.N_s}{N_p.N_s.T_h} \tag{3.5}
\]

\[
B_{OPTI} = \frac{280.N_p + (480 + 96).N_p.N_{CN}.N_h + 388.N_s}{N_p.N_s.T_h} \tag{3.6}
\]
ILNPv6 Analysis

The overhead generated at handover by ILNPv6 per passenger per train per second, \( C_{ILNP} \), is calculated as:

\[
\]  

(3.7)

where \( J_1 \), \( J_2 \), and \( J_3 \) are constants. There are three parts to the numerator of the right-hand side of this expression. The first part \( (J_1.N_p) \) refers to the VMN initialisation (Figure 2.7). As each VMN joins the mobile network, I have assumed each passenger only makes one trip, thereby joining only once, this number is dependent only on the number of passengers \( (N_p) \). The second part is the overhead generated by the handover of the MR, updating its location (Figure 2.8). This is dependent upon the number of train stations visited along the route \( (N_s) \). The third part is the overhead generated by the MR to each unique CN of the resident VMNs to update existing sessions (Figure 2.9). This is directly dependent on the number of passengers \( (N_p) \), the total number of unique CNs for the mobile network \( (N_{CN}) \) as well as the number of train station handovers \( (N_h) \).

In Eqn 3.7 if we replace \( J_1 \), \( J_2 \) and \( J_3 \) with the appropriate packets count or byte count (See Figures 2.7, 2.8, 2.9) we get the packet overhead, \( N_{ILNP} \), and bandwidth overhead, \( B_{ILNP} \):

\[
\]  

(3.8)

\[
B_{ILNP} = \frac{1362.N_p + 1362.N_s + 144.N_p.N_{CN}.N_h}{N_p.N_s.T_h}
\]  

(3.9)

3.4.2 Passenger Emulation

The emulation emulates the London Circle line using Equations (3.1) to (3.9). Our software emulates all 27 train stations as well as the individual train arrivals and departures. It also models each specific passenger boarding the train at a random selected train station, and alighting at a randomly selected station. Specifically, the train movements and passenger numbers follow the available statistics. For the passenger movements, I used the total number of passengers for a year and divided this equally to all train stations for a year (taking into account the passenger differences of weekdays, Saturdays and Sundays). At the start of each run, a train arrives at the first station with 0 passengers. A uniform random distribution is used to emulate the arrival of passengers from this pool for a given day and train station. As a train arrives, the passengers board that train and at each subsequent stop, I randomly select passengers to alight the train. This selection is based on the passenger mobility ratio, \( R_P \), which we define as the ratio of number of passengers in the train that alight to the number that remain on board.

Take for example, it is a weekday and \( N_p = 448 \). I have assumed that for each train arrival, \( N_p/N_t = 64 \) passengers need to board. If there is sufficient room for all passengers to board, the final number of passengers travelling each year will be 68850600. In total, I
executed each emulation 3 times, each with a different value of $R_P$. Each run emulates one year of train and passenger movements. I have assigned CN a value of 2 arbitrarily, as it suits our experimental purposes. Even though this is not a real mobility model, as my study overall is comparative, and the same model is applied to OptiNets and NEMO, I believe that it is sufficient.

Looking at Equations (3.1) to (3.9), we see that they share the same denominator, $N_p, N_s, T_h$. This term is defined as the time frame in which all registrations and handovers must be completed per passenger per station, i.e. the period that two cells would overlap is the hand-off period. We have defined $T_h$ as 60s (the average time a train spends at each station) and assumed that handovers occur at the station. The output of each equation will be framed as the cost in terms of packet overhead and bandwidth overhead, separately, per passenger per station, evaluated during the handover period, $T_h$. I have also assumed that the NEMO protocol has enabled IP Authentication Header [95].

### 3.5 Results

#### 3.5.1 Generated Passenger Mobility Traces

![Average Number of Handovers per Passenger](image)

Figure 3.6: Average Number of Handovers in Relation to Passenger Movement Ratio, when CN=2. The error bars shown here are calculated to standard error.

Figure 3.7 shows the total number of passengers in the emulation, given a Passenger Mobility Ratio. Again, we see that as $R_P$ decreases, the total amount of passengers also decreases. We know that as passengers are less mobile, a larger number will remain on
the train and not get off. This leave less room for new passengers to board. The reader should note that the line on Figure 3.7 is a visual aid and only links the data points, so that their significance is made clearer.

Figure 3.7: Total Number of Passengers to the Passenger Mobility Ratio

Figure 3.6 shows how the average number of handovers (Hops) for the passengers change with the variation of the Passenger Mobility Ratio ($R_P$). We see that as $R_P$ decreases, the average number of handovers increases exponentially. This correlates with the fact that when passengers are less mobile, they are more likely to remain on the train. The reader should note that the line on Figure 3.6 is a visual aid and only links the data points, so that their significance is made clearer.

Given the general equations, we know that boarding passengers add registration overhead and passengers that do not alight contribute handover overhead. Figures 3.8 and 3.9, show the relationship of these overheads to $R_P$. In Figure 3.8, we see that as $R_P$ decreases, less passengers are able to board and the overall registration overhead decreases. We note that for a given passenger mobility ratio, the station at which passengers are no longer able to board is different, with regards to $R_P$. This lag is due to the fact that the trains arrive at the first station empty and pick up more passengers (who remain on board) as they travels along.

In Figure 3.9, we see the same lag in the total number of passengers for a train. As a train stops at more stations, the overall number of passengers increases to a steady state. We see that for $R_P$ values of 10% and below, the steady state population is significantly less than the maximum limit of 500. We see that this limit is approached at higher values of $R_P$. We also see that because of the nature of passenger movement on a train. The
fact that passengers need to alight before others may board, means that train registration overhead is easily affected by low values of $R_P$.

![Train registration overhead](image)

Figure 3.8: Passenger Registration Load to Passenger Movement Ratio, when CN=2

### 3.5.2 Effect of Passenger Mobility on Overhead

Since my source gives the total number of passengers as approximately 68850600, I have focused my later results on $R_P$ values from 20% and above. Using the formulated overhead equations for NEMO, OptiNets and ILNP, we have calculated the bandwidth and packet overhead for each.

In Figures 3.10, 3.11 and 3.12, we see the relative overhead costs of each protocol. In these results, we have calculated the overhead with the assumption that $N_{CN}$ is equal to 2. By observing how OptiNets and ILNPv6 behave with different values of $R_P$, we see that ILNPv6 is much less sensitive to the change in number of handovers ($N_h$), compared to OptiNets. When $R_P$ is set at 50% (Figure 3.11), both ILNPv6 and OptiNets have similar RO costs. However, when $R_P$ is set at 90% (Figure 3.12), the cost of OptiNets increases by an order of magnitude in comparison to the increase of ILNPv6. As a result, in mobile scenarios of high passenger mobility fluctuations, the OptiNets approach will possibly lead to much larger variations in bandwidth usage compared to ILNPv6.
Figure 3.9: Passenger Handover Load to Passenger Movement Ratio, when CN=2

3.5.3 Effect of Passenger Correspondence

From Figures 3.1 and 2.6, we see that the protocol exchange for ILNPv6 is simpler than that for NEMO/OptiNets, and the data path that results is also simpler, compared with NEMO which requires two sets of tunnels. Additionally, we find that ILNPv6 leverages existing DNS infrastructure for naming, whilst NEMO/OptiNets must introduce additional network entities (the HA and FA) in order to function. Also, the use of the tunnels creates potential inefficiency in packet forwarding, and system complexity, as a result of tunnels and redirection through home networks.

Having observed the impact of varying $R_P$, I next study the impact of varying $N_{CN}$. Using the expressions for packet overhead in Eqns (3.5), and our expressions for bandwidth overhead in Eqns (3.6), we vary the value of $N_{h}$ from 1 to 14 (half a circuit of the Circle Line), and the value of $N_{CN}$ from 1 to 20 (i.e. every VMN has 20 unique CNs). For reference, we also include the overhead for NEMO/OptiNets without route-optimisation, i.e. all VMNs using tunnels via the $MR_{HAR}$.
From Figure 3.10, we see that, compared to NEMO, the packet overhead of OptiNets is much greater - a factor of \(~10\). From Figure 3.17, we see similar increases for bandwidth overhead - a factor of \(~10\) for OptiNets compared to NEMO. On a side note, we observe that despite not having detailed passenger mobility traces, we are not only able to generate passenger traces that are verifiable to some extent, but the calculated average hop/handover count for each passenger can be used to predict the control overhead of each protocol for a range of $N_{CN}$.

### 3.6 Discussion

These results are based on a profile of mobile network public access network scenario. The results show the importance of considering the nature of the operational scenario...
when choosing a mobility solution. They also demonstrate a need to consider the type of network service being provided to the users (passenger in this case). In this case for example, if the users did not require optimal routing or if bandwidth for the provider was critical, then the NEMO solution would be the best choice.

This scenario does not consider the type of network traffic that occurs with mobile users. It is possible that passengers may use the network to view simple websites or access emails. However, given the rise in alternative real-time and bandwidth intensive forms of communication such as Skype and other on-line communication platforms, this might very well change.

In my experiments, I have also considered each passenger to be a mobile node. Given the nature of some kinds mobile networks, (i.e. personal networks or body area networks), this might not always be the case. The passenger might be running his own body area network and be a mobile router instead, which would result in nested mobile networks. This would add an additional layer of routing between the mobile nodes within the private mobile network and the external correspondent nodes. One potential result is that it would make it harder to predict the mobile node to passenger ratio, thus making network provision more complex.

Now let us consider an alternative application scenario such as a vehicular ad hoc mobile network on an expressway. In the experimental scenario, providers have an advantage of the fact that trains have schedules and travel on tracks and are thus very predictable. The same can be said for train passengers with regards to time and location. Enabling
network mobility support for cars on an expressway is much more complex. One of the major hurdles would be the lack of infrastructure support. It is trivial to install gateways at each train station, but much more complex to provide connectivity along all roads. As a result, each vehicle would have to become part of an ad hoc network and would have to provide access to its passengers and connect through other vehicles nearby. Another issue is the variability in the number of nearby vehicles. On urban roads, this number may be quite high, but on rural roads it might be extremely low.

Such a scenario would most likely have different results. For example, the dynamic nature of the vehicles interacting with each other, would result in a higher number of mobile handovers. Any route optimisation mechanism in place would likewise be affected. Also, in this scenario each mobile network is smaller but has the potential of having a larger number of mobile nodes (depending on the capacity of the vehicle). Depending on the needs of the passengers, a tunneling approach might be more suitable, as it hides the mobility of the nodes.

### 3.7 Conclusion

I have compared NEMO, OptiNets and ILNPv6, which are three very different approaches to optimised routing for mobile networks. I have created an emulation of a network scenario, focussing on – initialisation and handover – and derived analytical expressions for packet overhead and bandwidth overhead for NEMO, OptiNets and ILNPv6.
By using a scenario based on data from passengers and trains numbers on the Circle Line metropolitan railway in London, UK, I have evaluated the expressions for packet overhead and bandwidth overhead by varying two key characteristics of a mobile network - its degree of mobility, $N_h$ (which we measure by the number of handovers as trains move between stations), and the number of external communications from the mobile network, $N_{CN}$ (which we measure by the number of unique CNs for each VMN).

I have quantified the cost of providing for optimal routing in terms of packet and bandwidth overhead by deriving analytical expressions for packet overhead and bandwidth overhead for NEMO, OptiNets and ILNPv6. With these expressions, I chose two variables $N_h$ and $N_{CN}$ to test their respective performance. I then wrote a java-based emulation to create passenger mobility traces on the assumption of a uniform random distribution of passenger arrivals (assuming $N_{CN}$ is 2) to calculate an approximate value of $N_h$.

I have shown, with respect to control overhead, that there exists a trade-off between provisioning for mobility and having optimal routing paths for mobile traffic flows. NEMO may be better suited for mobile networks that do not have mobility aware nodes. For OptiNets, we see that RO for one level of tunnelling has a higher overhead in comparison and would be worse if multiple levels of tunnelling exist. And for ILNPv6, it might be better suited where the mobile network is a mix of mobile and static nodes.

The results show that there is scope for the study of alternative architectural approaches to network mobility. It also shows that the current dominant Tunnelling approach to enabling mobility for mobile networks does not cater for all mobile network applications. There may be benefits gained from studying alternative mechanisms of network mobility support as they could perform better, in certain applications such as route optimisation. The success of each approach is very much dependent upon the mobile network scenario and its unique mobility conditions.

In this chapter, I have shown the value of a purely analytical study in highlighting areas that may require additional attention in mobile network support. However, practical
experimentation is also required, and this is a considerable challenge due to the overhead of setting up, operating and maintaining a testbed. In the next chapter, I look at the study of mobile network protocols and the current tools available. I introduce a new tool that is less costly and useful in studying and providing for network mobility support.
Figure 3.15: NEMO - Eqn (3.2) (min/max: 0.01/0.08)

Figure 3.16: ILNPv6 - Eqn (3.9) (min/max: 0.99/25.79)
Figure 3.17: OPTI - Eqn (3.6) (min/max: 0.55/99.75)

Figure 3.18: NEMO - Eqn (3.3) (min/max: 0.66/9.66)
Chapter 4

Studying Mobile Network Protocols with Virtual Machines

4.1 Introduction

In the previous chapter, we looked at how there is a need to study and evaluate alternative architectural approaches to network mobility support at an analytical level. While alternative network mobility protocols have been proposed, the difficulty in implementing and evaluating these protocols, is a huge obstacle towards their eventual use. This chapter looks at how the study of new network mobility protocols may be aided with virtualisation technologies, where it is not possible to use a testbed. It also addresses the concern that there does not exist an adequate number of tools for testing, evaluating and understanding network mobility protocols. I propose Cloonix-Net, which is a modified version of Cloonix [96], that simplifies the study of mobile network protocols.

4.2 Problem

When developing a new network layer mobility protocol, the architect is interested in the following properties:

1. initialisation, rendezvous and handover messages
2. protocol overhead of (1) under various load conditions
3. scalability of (1), i.e. the relationship of (1) and (2) when a large number of mobile nodes are introduced, or when there is large degree of mobility taking place.

If we consider the practical costs of such protocol testing, additional requirements such as resources required, ease of use and re-usability of the tool can also be important. Ideally, a good tool should provide these features to the architect.

The requirement of operational flexibility can be defined in two parts. The first should be the ability to construct realistic scenarios that reflect how the protocols will be used in an operational context. The second, is the ability to have multiple entities/roles represented
within the user-defined network. This allows for direct (absolute) performance evaluation and also opens up the possibility for conducting performance comparisons between different protocols. This is in line with an incremental engineering approach: by comparing protocols in simple, realistic scenarios, we can attribute with greater confidence the results of any experiment to the character/nature of the protocols (being compared) as opposed to artefacts of the experimental environment.

Network mobility protocols are often used in realistic scenarios, where there are large numbers of mobile nodes: indeed, that is often used to justify the use of mobile network aggregation. Any testing tool should thus allow the tester to construct realistic scenarios, with a reasonable number of mobile nodes. In this aspect, I recognise that there is a limit as to how much this can be scaled with the use of virtual machine (VM) images.

Another requirement is ease of debugging. The network protocol stack operates within kernel space, so any errors in the protocol often halt the kernel and thus the operating system. This makes debugging on a real machine in a testbed extremely laborious and difficult (in some case it may be practically impossible). There are two common ways around this problem. The first is to have separate debugging machine monitoring the test machine, the two being connected via a serial cable. The second method is to virtualise the test machine and run it as a VM image within the test machine. While both methods allow debugging to be undertaken in a controlled environment, the latter approach offers a greater degree of flexibility, faster development cycle and greater scale of experimentation than dealing with multiple real hardware test machines. At the same time, while greater scale may be achieved in simulation, the VM based approach allows us to execute the actual protocol code that would be used in an operational scenario, which may not be possible in a pure simulation framework. Virtualisation also allows the user to continue to use existing tools for debugging, making it convenient for re-usability, load testing and interoperability testing.

Another important consideration in any simulation is the model that should be created: what characteristics should be included and what excluded. In this case, consider the exclusion of wireless effects in our tool. While wireless effects are certainly evident in any real world scenario, I believe that adopting an incremental approach towards protocol testing, at the early stages of development, is convenient to reduce complexity and validate the simplest possible case. One example of this could be a comparative analysis of two mobile network protocols. By excluding a wireless layer in our initial model, we remove the following potential effects:

1. MAC layer collisions and back-off/retransmit behaviour
2. transmission errors resulting in silent packet discard
3. fading effects (resulting in errors) due to radio interference

While these effects are essential to consider for real world testing (e.g. when considering a deployment), it would be advantageous to have a more controlled and predictable environment which gives mobility without the wireless ‘noise’ in order to perform a baseline analysis of the protocol behaviour.
4.3 A Comparison of Existing Tools

4.3.1 Use of User-Mode Linux

User-Mode Linux (UML) is a port of the Linux kernel to Linux. It implements a Linux virtual machine running on a Linux host. Its hardware is virtual, being constructed from resources provided by the host. UML can run essentially any application that can be run on the host [97]. To the host kernel, the UML instance is a normal process. To the UML processes, the UML instance is a kernel. Processes interact with the kernel by making system calls, which are like procedure calls except that they request the kernel to do something on their behalf [98]. As of Linux 2.6.0, UML has been integrated into the main kernel source tree. While UML was designed for the x86 processors, it has since been ported to other architectures including IA-64 and PowerPC. One example in which UML is being used for large scale network experimentation is ORBIT 1. “ORBIT is a two-tier laboratory emulator/field trial network testbed designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings.”

UML enables Linux virtual machines (VMs) to be run as Linux applications within a host Linux OS instance. This allows the user to run different copies of Linux safely in user space without affecting the overall stability of the host operating system. It is also possible to further restrict a virtual machine in UML to specific hardware thus improving host stability. The UML instances contain a full Linux OS and so can have great flexibility and functionality for experimentation. In Cloonix-Net, UML functionality also allows us to run different VMs with different Linux kernels. Thus, it is possible to create a virtual network with different entities, each having its own unique role, communicating together as if in a real network testbed, but operating on a single Linux host.

Other network emulators that utilise virtualisation technologies exist - these use paravirtualisation as opposed to full virtualisation like UML. Such tools include IMUNES (Integrated Network Topology Emulator/ Simulator) [99] and CORE (Common Open Research Emulator) [100]. IMUNES is a network emulator model based on FreeBSD that allows for the existence of multiple virtualised network stacks within the kernel. Processes in user space can be grouped to these virtual stacks and subsequently communicate through virtual links between them. These groups are the virtual nodes.

CORE (which is based on IMUNES), is a lightweight, real-time network emulator that allows the user to create topologies that consist of real and virtual nodes. It is able to emulate nodes (e.g. routers and PCs) and the links between them. CORE supports wired and wireless links to a certain extent. It does not virtualise Layers 1 and 2 but instead focuses on Layer 3 emulation. Like IMUNES, it uses FreeBSD and implements an actual TCP/IP stack. But because both use paravirtualisation, all the virtual images share the same kernel, filesystem, processor, memory, clock and other resources. With regards to mobile network protocol testing, a custom network stack is required as certain network layer protocols need to be modified (e.g. NEMO and Mobile IP). As a result, the separation of kernel and filesystem offered by UML through full virtualisation provides for a simpler and more flexible platform to undertake such testing, though at the price of scalability.

1www.orbit-lab.org/
4.3.2 On the Use of Simulators

One popular tool for mobile systems is simulation, e.g. using OMNET\(^2\) and NS2\(^3\). While simulation is used widely within the research community, there is a reliance on that community to support collaboration and reproduction of new functionality. This may be sporadic in development and the maturity of code that results may be variable. In order for the existing mobile network standards such as NEMO to be used effectively in these platforms, there has to be considerable investment of time, to port the existing Linux code into the simulators, and then test and maintain existing code ports to the simulator as well as development of the main operational code base. Also, there needs to be effort in setting up connectivity scenarios, calibration and subsequently running the simulation. However, given that there is already existing code that has been verified and shown to work, it seems a natural progression to use that code directly where possible, as the results based on the ported code may not be the same as those of the actual code. With virtualisation (via UML), we have the ability to run real working code in a controllable and reproducible environment. While newer simulators such as NS3\(^4\) are planning to permit incorporation of system code, such tools still lack maturity. (Note: OMNET does include a module of a working FreeBSD stack with real code).

4.3.3 On the Use of Testbeds

Another option is the use of testbeds to conduct experiments. Often, accurate and realistic testbeds are expensive and time consuming to procure, setup and configure. Also, a designated, controlled space has to be allocated and managed, especially for wireless testbeds, where interference is a big concern. Additionally, for large testbeds, one must also consider the overall management and upkeep of the operational nodes, both in terms of finance and the manpower needed. In testbed experiments, some form of tuning or calibration is required every time new hardware is incorporated (e.g. after a hardware upgrade, or to replace a failed component), before experiments can begin. Some freely accessible testbeds exist, such as ORBIT\(^5\) and Emulab\(^6\). However, these still have limitations in terms of configuration, customisation, privacy and scheduling. PlanetLab\(^7\) is another example. It has issues regarding congestion, especially when it gets closer to certain conference deadlines. I am in favour of the usage of testbeds, and I believe that they should be incorporated whenever possible. However, I take the position that using the VM approach using UML would allow researchers to be better informed before progressing to a full test-bed: the VM approach is complimentary to a testbed experiment. VM results, where the effects of wireless transmission are not considered, could potentially offer a ‘best case baseline’ for comparison with testbed results. Also, results from a VM approach could potentially be used to target and narrow down specific areas of interest, which can then be fully explored with testbed experiments, thereby saving time and effort.

\(^2\)http://www.omnetpp.org/
\(^3\)http://www.isi.edu/nsnam/ns/
\(^4\)http://www.nsnam.org/
\(^5\)http://www.orbit-lab.org/
\(^6\)http://www.emulab.net/
\(^7\)http://www.planet-lab.org/
4.3.4 On the Use of VMware

One of the first tools, I considered, apart from Cloonix-Net was VMware\textsuperscript{8}. VMware specialises in virtualisation software. Under the academic license, VMware software was readily available for use, and there is a large and growing community of researchers contributing to libraries of `VM appliances’, custom configured VMware images that can be used ‘off-the-shelf’. Using VMware virtualisation software (which runs on all major platforms - Windows, Linux and Mac), I explored its capabilities in fulfilling our requirements for mobile network testing. VMware has the advantage of a large community, which provides support for the ‘VM appliances’. It is also very easy to create and share VMware images. Ultimately however, our main concern was that there was not enough operational flexibility for mobile network testing.

Below is a brief description of the three basic modes of network connectivity in VMware:

1. Bridged: where VMware uses the physical interface of the host to directly emulate the virtual interface in the virtual machine. Here the virtual machine appears as a separate machine on the (real, physical) LAN.
2. Host-only: where a virtual interface is created on the host, which is connected to the virtual interface in the virtual machine. This sets up communication between the host and virtual machine only.
3. NAT: where a virtual interface on the host is connected to the virtual interface on the virtual machine. The host acts a Network Address Translation (NAT) router for all egress and ingress packets. In this mode, the virtual machine does not appear visible (at least at the IP layer) to the outside world.

These modes made it difficult to create useful network topologies for experimentation. The only option available was to create a virtual experimental network, with all virtual machines on the Bridged mode. This created two main obstacles. The first was to setup and configure Virtual LANs (VLANs) on the real, physical network to allow suitable topologies to be configured. The second obstacle was to figure out how to emulate mobility of the machines, to trigger initialisation, rendezvous and handoff events. This had to be implemented by dynamically tearing down specific existing VLAN connections (to simulate when a mobile node leaves a network) and simultaneously setting up new VLAN connections (to simulate when a mobile node joins a network). It was found that the effort required to manage these VLANs dynamically for an experimental scenario (which requires sending specific commands to an ethernet switch and waiting for those commands to be activated) introduced a level of complexity of configuration that reduced greatly the appeal of VMware for this purpose.

\textsuperscript{8}http://www.vmware.com/
4.3.5 On the General Use of Virtualisation

For the use of VMs, the following advantages make it worthwhile for use in experimenta-
tion of mobile network scenarios:

1. It is quick to get up and running with experimentation, especially with the avail-
able Mobile IPv6 and NEMOv6 network configuration provided. Also, hardware
requirements will be relatively modest. There is no need to acquire any additional
hardware or special equipment - the basic requirement is a desktop machine cap-
able of running virtual machines, which could easily be an existing machine.

2. It is straightforward to share experiments for the purposes of collaboration or to
allow for other researchers to use your configurations and/or reproduce experimen-
tal results.

3. Using of virtual machines means that any calibration or setting up of a network en-
tity (e.g. mobile router or mobile node) need not be re-done. By simply copying the
existing virtual machine, another entity of similar type can be created and added to
the test network.

4. It becomes a trivial task to make backups and to archive existing experiments, both
after completion, and ‘snapshots’ of experiments in progress.

5. In the case of UML, exclusion of wireless effects makes for simpler experimental
configuration. The interpretation of such experimental results is directed primar-
ily on the architecture of the protocol (which may be of greater concern to the re-
searcher). The VM can also be archived, shared or re-configured easily. There will
however come a point when wireless effects will be needed to be investigated.

6. Migration from virtual experiments to emulation and testbed experiments is a nat-
ural progression, and could use the existing VM images for deployment on real
testbed systems. The kernel of existing virtual machines can simply be copied over
to real world machines, similarly for file-systems.

There are also some drawbacks to consider with the use of VMs:

1. Researchers will have to invest time in learning to use the Linux operating system
(in our case Debian); though these skills will be transferable.

2. For the creation of new VM images, some form of setup and calibration is required
in the first instance, in a similar fashion as for a testbed (though not with the same
level of effort, as we do not have to deal with the nuances of real hardware).

3. There is the possibility of experimental results being effected by artefacts or per-
formance bottlenecks occurring as a result of the operating system of the host ma-
chine, rather than effects/behaviour of the experimental VMs. These may be diffi-
cult to predict, detect or filter from experimental results.

4. Currently, virtual networks (consisting of multiple VMs) have to be run on a single
host machine. As it is not yet possible to distribute the load of hosting these virtual
machines, the number that can be hosted effectively is bounded by the hardware
specifications of the host machine. This may lead to relatively small-scale experi-
ments (e.g. 8-10 nodes) unless especially capable host machines are available.
5. For more complex operational testing, additional management tools might be required. For the creation of network topology we have Cloonix-Net; it is conceivable that if new functionality needs to be added, a new management tool will be required, or Cloonix-net would have to be extended.

4.4 Cloonix-Net

Cloonix is a configurable virtual network that uses UML. It is recommended as a tool for the study of NEPL. By using VMs, there is no simulation of wireless effects in the network scenarios created. When pursuing the development of a new protocol in an incremental fashion, this lack of additional complexity due to wireless transmission and reception effects can be viewed as an advantage. During development, it was an openly available software that created virtual networks using UML virtual machines. It consists of two major components. The primary component is the Virtual Switch, which creates the virtual LANs required to connect the individual UML machines in a user-defined network topology. This switch receives XML messages to configure all the LAN communications, then switches all IP packets according to the network configuration. The network topology is user defined and stored as a text file. The second component is the collection of VM images. These are implemented by pseudo-filesystems, which are available as UML or KVM machines for the Debian and Fedora operating systems. These VM images have ethernet interfaces plugged to 'sockets' that are in turn connected to the Virtual Switch. By using VMs, there is no simulation of wireless effects in the network scenarios created.

The main advantages of Cloonix are the following:

- Plug and play test of network functions
- Build re-playable demonstrations which can be easily distributed and shared
- Allows for scripts that emulate topology modifications
- Easy tests for routing software code
- Educational value for students and file-system is jailed
- Built in GUI to visualise your network, the links and the flows running in the links

Some disadvantages exist, such as its still not possible to freeze an emulation scenario and start it again. The main concern for our network protocol needs is that Cloonix restricts the user to a single kernel for Debian and Fedora operating systems. Multiple file-systems are allowed but these are used only as references. Any writing to file is saved in a separate file called a Copy-On-Write (COW). This limitation of one kernel makes it difficult to create a testbed of different network entities.

Cloonix-Net is our modification of Cloonix that has the added functionality of running multiple UML machines each with its own separate kernel. This allows the user to create custom network topologies that consist of distinct network entities. Cloonix-Net allows

---

9 [http://www.cloonix.net/](http://www.cloonix.net/)
for multiple kernels and filesystems. The user can create a unique testbed topology consisting of very different machines to suit his needs. For example, it is possible to have a VM acting as a Mobile Home Agent and a separate VM as a Mobile Router, thus enabling the user to design experiments that involve different entities of a mobile network architecture and study their interactions. I have also created and shared three pre-configured testbeds which can be used right out of the box. The first represents a Mobile IPv6 handover scenario, the second represents a NEMOv6 handover scenario and the third is a Nested Mobile Network, where a Mobile IPv6 node joins a mobile network running NEMO.

Each Virtual Machine had the following specifications:

- total disk space 4.0GB
- 512 MB memory
- NEPL umip 0.4
- kernel size of 27 MB
- Debian GNU/Linux 5.0
- Linux 2.6.29.5 i686 GNU/Linux kernel

4.4.1 Using Clonix-Net

A Primer for Mobile IPv6 experiments

MN node handovers can be executed by first placing the MN within its home network (See Figure 4.2). Green circles represent the network interfaces of the VM images. Grey circles are used for the connection of LAN links. The MN can be ‘connected’ to its HA by simply clicking on the grey circle of interface 0 of the MIP-HA virtual machine and clicking interface 0 of the MN; this creates a link between them. Communication can then be initiated between the MN and the CN (e.g. using ping). The next step is to cause a handover by moving the MN from its Home Network to AR1 (which is essentially a foreign network). This is achieved by right-clicking on the previously made link and choosing the ‘delete link’ option. This will detach the MN from its HA. The next step is to re-attach it to a foreign network, in this case, interface 1 of AR1. Handover messages can be observed and logged by interacting with the NEPL daemons. The handover can also be re-confirmed by observing the re-establishment and continuation of communication between the MN and CN.

A Primer for NEMOv6 experiments

Mobile network mobility can be done by moving the MR from its home network, NEMO-HA to its foreign network NEMO-R (See Figure 4.4 and instructions in the previous section.) I leave it up to the users to include additional fixed nodes attached to the mobile network or just have the mobile network consist of the MR, as required. Finally, we move the mobile network back to NEMO-HA.

A Primer for Nested Mobility experiments
By combining the topologies of a Mobile IPv6 and a NEMOv6 setup, it is possible to create a nested mobile network scenario. Figure 4.6, shows how a mobile node (MIP-MN) can be configured to leave its home agent (MIP-HA) and join an existing mobile router (NEMO-MR), maintaining its connections with its correspondent node (MIP-MN) as the mobile router moves from access router 1 to access router 2 (AR1 and AR2 respectively).

4.4.2 Modelling traffic effects

Even though Cloonix-Net does not allow for the simulation of wireless radio effects, it is possible to re-create some traffic conditions brought about by the wireless layer using the Linux Network Emulator tool (Netem) 11. Netem is a tool available for emulating certain traffic conditions that can be found in wide area networks. Netem is capable of creating the following network conditions: (i) Network delays, (ii) Delay distribution, (iii) Packet loss, (iv) Packet duplication, (v) Packet corruption and (vi) Packet re-ordering.

By incorporating netem with Cloonix-Net, we have some means for re-creating wireless radio effects (not MAC access effects, however), but in a constrained and repeatable way. This adds stability and makes possible testing of different protocols or different configurations against similar network conditions.

If we look at the three key requirements for a tool that allows for the study of mobile network protocols (Section 4.2), we see that Cloonix-Net fulfils the first requirement. The usage of Cloonix-Net and netem enables us to fulfil the second requirement up to a certain extent. We will be able to make comparative analysis but we will fall short of actual behaviour of a wireless layer. However, as we have discussed previously, in view of an incremental approach, this is certainly acceptable at a development stage and for our purposes. The third requirement is met to some extent. In terms of a large number of mobile nodes (VM images), the original author of Cloonix-Net has tested it with up to 30 VM images. While this number is not large, we argue that it is large enough for preliminary experiments. Cloonix-Net does adequately meet our research requirements and has shown itself to be useful in our preliminary study of mobile network protocols.

4.4.3 Modelling wireless effects

Here I explain an approach in which wireless layer effects may be initially excluded and then later modelled and applied as a function upon previous data for more realistic results. In the simplified communication stack of Figure 4.7, the boxes labelled “Network” represent the network layer protocol. If we assume that the packet transmission behaviour can be expressed by some function, $N$. The boxes labelled “MAC/PHY” represent the MAC/PHY protocol, and its behaviour can be expressed by some function, $M$. If we wish to examine only the comparative behaviour of two different network layer interactions (and not the absolute performance) over the same MAC/PHY layer, then we can eliminate $M$.

If $P$ is the observed packet behaviour of the radio transmissions, we can say that:

$$ P = N \odot M $$ (4.1)
where ⊙ is an operator denoting that \( N \) is modulated by \( M \). If we use the subscripts \( NEMO \) and \( OPTI \) for NEMO and OptiNets, respectively, then for a given MAC/PHY, \( r \), we wish to establish an expression for the comparative behaviour, \( H \):

\[
\begin{align*}
P_{NEMO} &= N_{NEMO} \odot M_r \\
P_{OPTI} &= N_{OPTI} \odot M_r \\
H &= P_{NEMO} / P_{OPTI} \\
    &= N_{NEMO} / N_{OPTI}
\end{align*}
\]

If we wish to make a comparative analysis of bandwidth overhead, we can evaluate and compare the packet level interaction of the protocol (line CD in Figure 4.7) instead of the potentially more complex, and media/technology-specific behaviour due to the MAC/PHY (line AB in Figure 4.7). So, in our evaluation, we will have produced expressions for the network level behaviour in terms of packet transmissions, and assessed directly the comparative packet level overhead of NEMO and OptiNets.

### 4.5 Performance Analysis

#### 4.5.1 Experiment

In order to evaluate the network performance of a Cloonix-Net virtual network, I chose to run a performance analysis experiment across our virtual mobile network topology from the mobile correspondent node (MIP-CN) to the NEMO mobile router (NEMO-MR) (See Figure 4.4). I used **iperf** to measure throughput, packet loss and jitter between these two nodes. I varied the bandwidth and packet size of a UDP flow, as I wanted to replicate a real-time application. Each combination of variables was executed 5 times and the mean value was plotted in the following graphs. To put our results in perspective,
I also performed the same *iperf* tests in two other testbeds with the same topology. The first was a pure wired testbed that was connected with ethernet cables (100Mb/s). The second was a wireless testbed, which is identical to the wired setup with the exception that the last hop between the NEMO Home Agent (NEMO-HA) and NEMO-MR was wireless, 802.11n (up to 150 Mb/s). I have also taken steps to ensure that the results are not affected by the kernel UDP buffers by setting the UDP buffer size to the maximum (using `sysctl -w net.core.rmem-max=8388608`).

For my experiments, I have selected the following topologies; for a mobile node - Figure 4.1, for a mobile network - Figure 4.3, for a nested mobile network Figure 4.5. These topologies are derived from the examples show in [1], which outlines the main terminology and scenarios for network mobility.

Figure 4.1, depicts a basic topology for end-host mobility. It includes the minimum elements required to show MIPv6 mobility. The main connection is between the mobile node and the correspondent node. And the mobility takes place when the mobile node moves between access router 1 and access router 2. Figure 4.2, is the equivalent as seen in Clonix. Figure 4.3, depicts the simplest NEMO mobile network topology. It includes a mobile network, a NEMO Home Agent and an access router. Figure 4.4, depicts its equivalent form in Clonix.

Figure 4.5, shows one possible form of a nested mobility. I have arbitrarily chosen this form for its simplicity and the fact that builds upon previous topologies used. It is a combination of end-host mobility and mobile network mobility. In my experiments, I have chosen to place the mobile node within the mobile network as it moves between access router 1 and access router 2. Figure 4.6, depicts its equivalent form in Clonix.
4.5.2 Testbed

To evaluate the performance of the virtual mobile network topologies. I constructed two identical real-life testbeds to compare it against. The first was a fully wired tested, where all the connections between the machines was with Ethernet cables. The second was a fully wireless testbed where all the connections between all the machines was with 802.11n WiFi.

The host machines had the following specifications:

- Intel Dual Core, Pentium 4 2.80GHz
- 1 GB memory
- Ubuntu 9.04 Desktop
• Linux 2.6.28.1 i686 GNU/Linux kernel

Figure 4.8 shows the NEMO and Mobile Home Agents as well as the main router (RT) and Correspondent Node. It also contains the Access Router 1 machine used for handovers. These were set up and configured in my office for the duration of the experiments. Figure 4.9 shows the Access Router 2 placed at a separate office location in the building, on the same floor. Figure 4.10 shows the mobile wireless network configured for the wireless testbed experiments. I used a Asus EeePC 1000 as the mobile router and an Asus EeePC 900 as a Mobile Node within the mobile network. The mobile router was modified to support 802.11n. I also an Asus 700 for monitoring purposes.

The main challenge in setting up the testbed was the configuration of the 802.11n network on Debian. I had to retrofit the shuttles and there was no existing documentation on how to do this. The card I used were COMPEX iWaveport WLM200NX 2T2R 802.11N a/b/g/n minipci card. The specifications are as follows:

• 2.4/5GHz IEEE 802.11n/a/b/g standard
• Output Power of up to 20dBm @ a/g/n Band
• Support for up to 2x2 MIMO with spatial multiplexing
• 4x throughput of 802.11g and 80.211a
• Wireless Encryption and Authentication Supported
• Transmission Power Control (TPC)
• Enhanced performance with Atheros XSPAN technology Optimized for higher throughput at long range
• High Performance (up to 300Mbps physical data rates and 200Mbps of actual user throughput) with Low Power Consumption
• Dynamic Frequency Selection (DFS)
• Multi-Country Roaming Support (IEEE802.11d)
• 2 X U.FL Antenna Connector

This involved rebuilding the kernel to support the cards as well and installing the appropriate supporting libraries and drivers. I also double checked that the signals being produced with the aid of tools such as Wireshark, Kismet and the Wi-Spy DBx, which is a 2.4/5 Ghz USB spectrum analyzer.

4.5.3 Results

For Figure 4.11, 4.12 and 4.13, we see that the overall throughput of the virtual testbed is not nearly as high as the wired and wireless testbeds. This is mainly due to the performance limitations of the desktop hardware. For example, factors such as the read/write speed of the local disk, the software limits of the virtual UML kernels as well as the limits
of the host kernel. We also notice that the maximum throughput of the virtual testbed (approx. 20Mbs/s) is less than the other testbeds, and occurs when the packet size is 1400bytes and the bandwidth is set at 20Mbs/s. This corresponds to the fact that MTU was set at 1500bytes. The throughput profiles of the wired and wireless testbeds appear to be the same, with slight performance degradation for the wireless testbed at higher bandwidth settings.

For Figures 4.14, 4.15 and 4.16, we see that while packet loss is generally low for the wired and wireless testbeds, the virtual testbed experiences negligible packet loss until it exceeds beyond the range of 10Mb/s bandwidth at 1400byte packet size. It then sustains very high packet loss. Thus we can see the operating limitations of using Cloonix-Net for such kinds of network testing, beyond which it is no longer realistic.

For Figures 4.17, 4.18 and 4.19, as expected the jitter is extremely low for the wired testbed and slightly higher on average in the wireless testbed. The jitter in the virtual testbed is very high after a 10Mb/s bandwidth and 1400 bytes packet size. I suspect that this may be due to the limit of the host operating system kernel. As its load is exceeded, the Virtual Switch does not get enough CPU time to process all the packets and as a result, it starts to drop packets.

4.5.4 Discussion

These results demonstrate how experimental results from Cloonix-Net are comparable to those gathered from the wired/wireless testbeds, given the topologies shown. The caveat being that the operating boundaries are not exceeded. Also, given the limited capacity of the machine running Cloonix-Net, it is conceivable that a more powerful machine would be able to produce accurate results beyond that of the operational boundaries shown.
here.

I note that my experiments did not involve pinpointing the causes of the operational boundaries. One could speculate that they are the result of a combination of hardware and software limitations. I leave this to future researchers as it is beyond the scope of this research.

### 4.6 Conclusion

The future holds tremendous potential where mobile networking can be utilised in various scenarios, e.g. military and everyday-civilian scenarios. While there is already an IETF standard protocol for mobile networking (NEMO), its has seen little deployment, and it is not clear when it will gain widespread usage, despite being available for close to a decade. At the same time, there have been numerous optimisations proposed for NEMO, as well as alternative protocols which utilise very different approaches. Unfortunately, there is a shortage of available tools to adequately experiment and compare these advances.

I have shown that Cloonix-Net allows for the testing of mobile network protocols in realistic topologies. I have also demonstrated the experimental limitations of Cloonix-Net to the user. I have compared Cloonix-Net to identical testbeds, both wired and wireless. The results show that Cloonix-Net is a realistic network emulation tool that allows for the testing of network protocols for network mobility within reasonable limits.

I have contributed pre-configured network topologies of Mobile IPv6 (end-host mobility) and NEMO (network mobility) that allow for simple experiments to be conducted out-of-the-box. I hope that this tool will benefit other researchers (as much as it has ourselves)
in their efforts to understand mobile networks. By comparing the advantages and disadvantages of this tool, to other existing methodologies, I hope to have provided the reader a good platform to begin research into this area.

I put forward that not only is the study of mobile network protocols without wireless effects both a necessary and beneficial step towards a better understanding of network mobility, but also that the modified Cloonix-Net\textsuperscript{12} as a virtual tool is useful tool for preliminary (pre-testbed) mobile network experiments.

In the next chapter, I extend this work on virtual machines and demonstrate how they may be used to enable network mobility in Personal Area Networks, in the context of an E-Health application, using a novel technique of running virtual machines on top of Android. I demonstrate how this is a feasible, simpler alternative to enable network mobility support and the study of network mobility protocols, avoiding the overhead and problems of having to modify existing devices directly in order to enable mobile network support.

\textsuperscript{12}Since the writing of this chapter, Cloonix-Net has been integrated into the main Cloonix release.
Variation of Throughput by changing Packet Size and Bandwidth

Figure 4.11: Virtual Throughput graphs on Testbed

Figure 4.12: Wired Throughput graphs on Testbed
Variation of Throughput by changing Packet Size and Bandwidth

Figure 4.13: Wireless Throughput graphs on Testbed

Variation of Percentage Packet loss by changing Packet Size and Bandwidth

Figure 4.14: Virtual Packet Loss graphs on Testbed
Variation of Percentage Packet loss by changing Packet Size and Bandwidth

Figure 4.15: Wired Packet Loss graphs on Testbed

Figure 4.16: Wireless packet Loss graphs on Testbed
Variation of Jitter by changing Packet Size and Bandwidth

Iperf: Bandwidth (Mbits/sec)
(with Iperf -b option)

Packet Size (bytes)
(with Iperf -l option)

Colour Gradient of Jitter (ms)

Figure 4.17: Virtual Jitter graphs on Testbed

Figure 4.18: Wired Jitter graphs on Testbed
Figure 4.19: Wireless Jitter graphs on Testbed
Chapter 5

Enabling Network Mobility Support for Remote Health Monitoring

5.1 Introduction

In the previous chapter, we looked at how the study of network mobility protocols can be difficult and my contribution towards enabling network mobility support with Cloonix-Net. In this chapter, I extend this work on virtual machines and demonstrate how they may be used to enable real mobile networks. Specifically, I show how it is feasible to use virtual images on Smartphones to build a mobile Personal Area network for the purposes of remote health monitoring.

I concentrate on enabling the mobile router (MR) as it forms the key architectural point for enabling and managing mobility. I also focus on performance issues, such as the impact of supporting mobile networks via 3G, WLAN and Bluetooth technology, on the battery life of a modern device.

5.2 Remote Patient Monitoring

Increased human longevity and increases in population means that there are increasing demands on health-services globally. In developed regions, as the average lifespan of the populace increases, and health demands increase, there is a shift in health care policy, with demands placed on the existing healthcare systems. For example, an ageing population and preventive medicine requires a higher degree of monitoring. Faced with these conditions, there is a growing need to provide better quality healthcare at affordable prices.

In search of a solution, healthcare researchers have been looking into leveraging technology to see if current clinical methodologies and practices can be augmented or changed. One example of this is patient monitoring. That is, the measurement of certain biological parameters for clinical purposes, e.g., heart-rate/pulse, blood pressure, body temperature, etc. Such biological data forms the basis of diagnosis, treatment and ongoing care for both in-patients – those resident in a clinical facility (such as a hospital) – and out-patients – those not resident in a clinical facility but who have to travel to one for monitoring or
treatment.

It has been shown that there may be advantages to enabling remote monitoring of patients. For example, it is less expensive to monitor stable elderly patients within the comfort of their own homes, as opposed to having them reside in the local hospital ward [101]. As well as lowering costs at clinical facilities, and being more convenient for the patient, there are also possible psychological and clinical benefits of having patients rest and recuperate in the familiar environment of their own home, while maintaining the level of monitoring they might receive in a clinical facility. Potentially, there are significant economic gains to be made in the long run when utilising appropriate technology to reduce cost and improve patient management [102].

The increasing availability of smartphones that have powerful general computing capabilities, coupled with open operating systems (such as Android) and other common-off-the-shelf (COTS) components are paving the way for their usage as personal monitoring platforms. Most smartphones already have a variety of sensors built-in, for example GPS, an accelerometer, camera, microphone, proximity sensor and touchscreen. They are also agile with respect to connectivity in that they usually have more than one network interface such as 3G, WiFi or Bluetooth. Smartphones that use the Android platform are easy to develop for and this has allowed for numerous opportunistic sensing applications to be developed that are freely available on the Android Market such as Tricorder 1.

Mass production and a global market has also greatly reduced the price and increased the availability and capability of medical grade sensors. Some of these have a WiFi or Bluetooth interface which can be connected to a smartphone; the latter can then act as data collector/aggregator and communication router/up-link, forwarding data to a remote site. Small sensors have been designed which could be carried with the user and permit continuous monitoring, for example, the Zephyr HxM Bluetooth ECG sensor 2 which is specifically designed for sports use. Given these factors, it is not surprising to see the rising interest in wireless body area networks (WBANs) in healthcare.

5.2.1 Wireless Body Area Networks

Wireless body area networks (WBAN) are defined as mobile wireless networks that may contain different sensors that are physically placed about a person. The WBAN moves with the subject and the sensors are able to function continuously throughout the subject’s (mobile) activities. Apart from the sensors, all WBANs also have a special device that acts as a mobile router (MR) for the networked sensors. A good summary of WBANs in the field of healthcare can be found in [103].

WBANs are a mature field of research, where the challenges and applications have been explored for quite some time [104] [105]. One of the most promising applications of WBANs is healthcare, where wireless sensors attached to a patient are used to wirelessly monitor patient health statistics and activity [106] [104]. WBANs have been made possible by two major developments. The first is the technological advancements in integrated circuits, wireless communications and physiological sensing, giving us miniature, light-weight, ultra-low battery power, intelligent monitoring devices. And the second are the many advances in wireless technology and communication standards such as

---

1 http://code.google.com/p/moonblink/wiki/Tricorder
2 http://www.zephyr-technology.com/products/consumer-hxm
ZigBee, 802.15.14 and Bluetooth, which are making it possible to form sensor networks from separate sensors with greater potential.

WBANs have an important role to play in the future of mobile medical care: they have been considered for a number of years for potential healthcare applications. One of the first areas specific application has been targeted is cardiovascular monitoring (via ECG sensors). Projects such as CardioNet [107] have developed their Mobile Cardiac Outpatient Telemetry (MCOT) system that enables real-time heart beat monitoring, focusing on arrhythmias. CardioNet also allows for integrated analysis and response, as it has an integrated diagnostic and patient management tool.

Another potential application is in the realm of first aid responders in disaster recovery situations. AID-N (Advanced Health and Disaster Aid Network) [108] is one such example. Its main objective is to allow healthcare providers at disaster scenes and medical professionals at remote hospitals the ability to consult specialists who are geographically distant, on patient issues. Patients wear motes (MICAZ), which record their vital signs and transmit the data to a central database. Motes within an area form a wireless ad hoc network along with an on site portable tablet PC which acts as a hub. AMON [109] is a similar system that looks at patients with cardiac and respiratory problems. It works in the same way through wrist-worn sensors, with continuous data collection and evaluation of multiple vital signs.

The usage of sensors in a WBAN usually calls for a transit/relay device that acts as a data sink and also as a router to the outside world. Usually portable tablet PC or more recently smart phones are used in this way. HealthGear [110] and Mobihealth [111] are two such examples. HealthGear uses Bluetooth to form an ad hoc wireless network between its sensors and cellphone. The cellphone stores, transmits and analyses the data and presents it to the phone user. Mobihealth was developed for the continuous monitoring of patients outside of the hospital environment. It proposes to integrate sensors and actuators to form a wireless BAN. Mobihealth goes further to leverage smartphone connectivity to push data online. One of the purposes of the project was to test the ability of 2.5G/3G infrastructures to support value-added healthcare services.

Another important aspect of WBANs is the placement of the sensor. One novel approach has been to embed the sensors in clothing. Some examples of this are Lifeguard monitoring system [112], HealthGear [110], Wearable BAN [113] and LifeShirt [114]. LifeShirt has the additional ability to share information with peripheral devices and incorporate patient input. This is then collected via a PDA and transmitted online. There are also projects which go further and try to integrate WBANs on a larger scale. The CodeBlue [115] project looks at building an entire wireless infrastructure to support sensing via the WBANs. The BASUMA Project [116] is another such project that focuses on multimedia applications. The MyHeart Project [117] aims to integrate sensing into the average person’s daily routine to achieve user-defined lifestyle goals such as better health and personal health goals.

5.2.2 Virtualisation on Smartphones

Modern computers are capable of using virtualisation to present the illusion of many smaller virtual machines (VMs), each running a separate operating system instance [118]. With advancement in the mobile phone industry, smart-phones are quickly reaching the
point where they too can run VMs\(^3\). Companies such as VMWare are already planning
to release versions of their virtualisation platform for mobile phones\(^4\). We also see a
trend towards the creation of person mobile wireless networks. Android is planning this
for their Froyo 2.2\(^5\) and Audi recently built a car that has mobile Wifi built-in\(^6\), via their
new MMITouch and Infotainment System.

The current landscape of smart phones today are released with proprietary operating
systems already installed. And while it is possible in some cases for users to install alter-
native operating systems, this may void the warranty of the phone or may even be
considered illegal. By leveraging virtualisation technologies, users are able to safely in-
corporate features of other operating systems into their default installed ones. With the
ability to run multiple virtual machines at the same time or switch between them, the
user also gains access to greater functionality and choice.

The use of virtualisation, where the virtual machine is run as a single process, allows
the user to manage the resources (e.g. CPU time and hardware interfaces). This means
that despite having a limited hardware resources, these resources can be fully utilised by
switching between virtual machines. Additional benefits such as portability, flexibility
and security are also inherent characteristics of employing virtualisation techniques.

5.3 Proposed Architecture

5.3.1 Overview

In order to meet the needs of remote patient monitoring through network mobility on
WBANs, I propose the vNurse architecture. vNurse builds upon and extends such tech-
nologies that are inexpensive and readily available to allow for better patient monitoring.
This proof-of-concept system leverages three COTS technologies, namely, (i) virtualisa-
tion; (ii) network mobility; and (iii) wireless sensor networking. It utilises existing wire-
less sensors to collect the relevant patient sensor readings, such as temperature, heart-
rate and GPS location (our system also allows for ambient environment readings, such as
room temperature). These readings can be streamed ‘live’ from the patient to the health-
care practitioner, or stored on the smartphone device (which acts as both a data collector
and mobile router) for later upload or retrieval on request. The smartphone connections
can be maintained even while the patient is mobile (within the home or travelling out-
side), ensuring 24-hour monitoring. This functionality is achieved by aggregating the
WBAN sensors as a single mobile network, utilising mobility protocols such as Mobile
IP\(^7\) and NEMO\(^8\) to achieve this.

This architecture (See Figure 5.1) has been designed to solve the problem of remote mon-
itoring in eHealth scenarios. This system comprises of four main components (i) WBAN,
(ii) virtualisation, (iii) network mobility and (iv) connectivity. Each of these components
provide a key aspect of functionality towards the final goal of the proposed solution,

\(^3\)www.qualcomm.co.uk/products_services/chipsets/snapdragon.html
\(^4\)www.vmware.com/company/news/releases/mvp.html
\(^5\)www.techcrunch.com/2010/05/20/froyo-android/
\(^7\)http://ipv6.com/articles/mobile/Mobile-IPv6.htm
\(^8\)http://datatracker.ietf.org/wg/nemo/charter/
which is to provide low cost, efficient and effective remote patient monitoring.

The implementation of vNurse uses virtualisation on a guest OS virtual machine (VM) image – specifically, a Debian VM image – to encapsulate the WBAN functions within a secure, configurable and self-contained environment, leveraging the Android host OS capabilities for onward communication via the smartphone. The Debian VM image would be produced by technical administrators working under the guidance of healthcare practitioners. Libraries or templates of such images could be constructed for general use, and then customised, under the guidance of a healthcare practitioner, to suit the monitoring needs of a particular individual patient. The patient would simply carry the VM image to a suitable smartphone device. The VM image would be configured so that it can be monitored and controlled remotely by the health care practitioner or by a technical administrator, as required.

I propose to use the smartphone as a data collector (aggregating and partially processing, if required, data gathered from the sensors) and as a mobile router for the sensors. It will host a wireless network via its WiFi interface to provide network connectivity to local wireless sensors. It will provide uplink capability via the existing 3G interface of the phone. As the patients roam, I leverage the prevalence of 3G coverage to provide network connectivity at all times. To accomplish this, the smartphone will also be acting as a NEMO mobile router. As a result, existing TCP/IP connections will be handed over as the smartphone moves from cell to cell. Of course, it is also possible to use the WiFi network as an uplink, if required. Suitable uplink connectivity is available in various forms, for example a wireless broadband gateway in a residential setting.

The smartphone’s Home Network will be the server to which the healthcare practitioner wishes the data to be sent back. A Home Agent is a server in the Home Network which runs the NEMO protocol and has a public IP address that allows the smartphone to connect to. It also serves as a central repository and secure distribution point for the patients’ data to be disseminated to other interested parties such as the patients’ relatives or friends who have been authorised.
5.3.2 WBAN Component

WBANs have a unique set of requirements that should be considered to ensure their successful deployment. Possible requirements such as a bio-sensor design and sensor system design \cite{119} are especially important for our medical scenario. For bio-sensor design, the target scenarios need sterile sensors as the intended target could be someone who is ill or someone who is recovering from illness. Similarly, the sensors should also be as comfortable as possible as they are likely to be attached to the subject for a considerable period of time, both day and night. It would also be beneficial if the sensors were discreet and/or possibly visually unobtrusive to cater to any negative psychological sentiment that might arise.

For the sensor system, the sensors should be made from off-the-shelf components to keep costs low. I believe that an open market and standard architecture will encourage early adoption by hospitals. It will also mean that practitioners and patients will be able to select the appropriate sensors to incorporate into their mobile WBAN. An open market would also encourage a large variety of sensors to be available. Such a market does not exist today, and I take the position that showing the feasibility of a low-cost, easy-to-use mobile network platform, enabling using a consumer device such as a mobile phone, would encourage such a market \cite{120}.
The WBAN component is the lowest component in our architectural hierarchy. This component is very similar to existing WBANs. In our architecture, the physical limitations are those of a PAN (personal area network). Because we are interested in the sensor readings of particular patient, we can assume that the sensors of the WBAN are geographically close to the patient. While we are interested in sensors that are attached directly to the patient, we may also need information about the patient’s environment, e.g. room temperature. As any sensor readings available could potentially be useful (depending on the healthcare requirements), I have tried to make it possible for sensor-location-independence in our proposal. This is achieved through the usage of WiFi and pre-built libraries already configured within the guest OS virtual image.

In this architecture, the patient may wear any number of sensors required. These external sensors along with those on the smartphone constitute our WBAN. As the subject moves from location to location, the readings from the sensors are recorded, possibly encoded (e.g. compressed and/or encrypted) and forwarded to the healthcare practitioner. Including additional environmental sensors, such as those that measure room temperature, allows for some possible correlation between how a patient feels and their varying surroundings in a quantitative way.

Figure 5.2 outlines the basic setup of our mobile WBAN platform. The large box labelled Host OS is an HTC G2 Touch smartphone running Android 2.2 (with a custom 2.6.29 kernel). I assume that the patient carries a smartphone at all times. I chose the Android platform because it has a large growing market share and is the arguably the most open mobile platform available. The box labelled Guest OS refers to a mounted image of a Debian file-system that I built and placed on the SDcard of the phone.

The boxes labelled Network refer to the network connectivity of the Debian and Android entities. I used this connectivity to pass data between the two entities. The oval labelled ASE Server refers to a python daemon that I have written to access local sensor readings through the Android API as well as act as a sink for external sensors wirelessly connected to the phone. The oval labelled Master is another python daemon that is running within the Debian file-system. Its purpose is to push the reading across the network to the monitoring machine. Some data processing may occur here such as pre-analysis or data compression, plus encapsulation for security, e.g. through use of standard libraries such as SSL. Real time readings are also made available to the patient here.

5.3.3 User-Mode Linux

User-Mode Linux (See Section 4.3.1) images are used to encapsulate operating system specific requirements and configurations. This is to allow for multiple users of the testbed to use the common hardware easily. For the experiment, I used UML for hosting virtual machines in our testbed. It was also used for kernel development due to its sandboxing properties.

There are some experimental limitations that a researcher must be aware of when using UML images for networking experimentation, especially for scenarios that look at wireless layer effects such as ours. If we look at common metrics used in the characterisation of wireless networks, such as throughput, delay and jitter, we find some work has been done in ORBIT to quantify the impact of using UML. From [121], the following points have been demonstrated and need to be considered for the experiments:
Figure 5.2: This figure shows the flow of information in our mobile WBAN platform. Line (1a) represents the readings from the local sensors on the phone that is accessible via ASE Python API sensor facade. Line (1b) represents the readings from external sensors attached to the NEMO MR network. Line (2) represents the raw sensor readings being pulled into the Controller application running within the Guest OS. Line (3) represents the stream information that is then sent across the network to the healthcare personnel monitoring the patient. This stream also allows for management information to passed to the controller.

Long Duration

Virtualisation has minimal effects on UDP experiments when the experiments are carried out over a long period of time. In virtualised platforms, experiments that measure instantaneous throughput are often inaccurate due to considerable increase in variance of throughput close to saturation (30Mbps).

Packet Size

Virtualisation creates a limitation on experiments that require small packet sizes and high bit rates (less than 30000 packets per virtualisation platform). The recommended packet size is 1470 bytes. Other important factors that might skew the results are the impact of running multiple UML instances on a single machine and the interference between them. From [121], we see that 2 UML instances running on the same machine have some performance impact as the packets are buffered for a random amount of time, for the UML to be context-switched, before they are sent over the wireless interface.

I used a virtual Debian file-system to encapsulate the information processing, information forwarding and mobility software. By encapsulating relevant information processes and resources into a virtual file-system, I made the solution portable between devices. It would be possible to build an entirely custom virtual file-system and populate it with the required programs for processing, monitoring and forwarding. These file-systems can also be easily shared and distributed to patients who require monitoring. These patients would only have to load the virtual file-system and run it, they do not have to do any configuration before hand. I encapsulate the required functionality into a Debian image for
the following reasons; (i) access to greater Linux functionality that is not available within
the Android hosts OS, (ii) a natural way to port and customise our application directly
without affecting the host operating system, (iii) By manipulating the permissions of the
file image, I can control the access rights to the information stored within the image and
thereby have some level of security control for sensitive information. Similarly it should
be possible to configure the rights of the file-system to protect sensitive information on
the host OS such as the messages and contacts.

The use of the VM image would also allow the monitoring capability to be placed easily
onto the patient’s personal mobile device (assuming he had a suitable device), for fur-
ther convenience, if the patient was not willing to carry an additional device. VM image
sand-boxing would protect the monitoring functionality, as well as prevent the poten-
tially complex configuration of the monitoring system from interfering with the normal
operation of the device if the application was installed on the host OS directly.

5.3.4 NEMO on Android

Network Mobility (NEMO)\(^9\) requires a custom kernel that has specific options (such as
IPv6) enabled. In order to get NEMO working on Android, I had to build a custom
Android kernel for the Hero and flash it onto the phone. Fortunately, the 2.6.29 kernel
source and other required binaries are freely available. For the NEMO userland, because
I am installing the userland within the Debian VM file-system, and am not choosing to
link the Android file-system to it, there is an additional step of installing the new kernel
modules into the Debian VM image. Only with these in place can the NEMO userland
be successfully built and installed. Another issue faced was with the \textit{tun} module. This
module is required for IP tunnelling. In a previous kernel configuration, where the \textit{tun}
module was built-in, I discovered that Android places it at \texttt{/dev/tun} as opposed to the
usual \texttt{/dev/net/tun}. As a workaround, I built \textit{tun} as a module and installed it directly onto
the Debian image, and used \texttt{mknod} to create the appropriate directories. Our procedure
is summarised below:

1. compile custom Android kernel with custom configuration
2. install Android OS onto Hero
3. boot into “recovery mode” and flash custom kernel
4. rebuild WLAN module and link to custom kernel sources
5. flash new WLAN module onto Hero
6. copy linux modules and install them onto Debian virtual filesystem

5.3.5 3G connectivity challenges

There are some engineering issues with 3G connectivity that, whilst not invalidating the
architectural approach I have followed, make practical realisation of some parts of the

\(^9\)http://umip.org/
architecture challenging. Note that all of the issues listed below can be solved by the 3G provider adjusting network configuration as required.

Commercial 3G networks often use Network Address Translation (NAT), so the Server on the VM image may not be easily reachable via egress connection requests. For example, a technician wishing to perform some system maintenance on the VM image would typically initiate a connection request to the VM image for a management application.

Additionally, 3G providers may use transparent proxies, stateful firewalls, or simply block certain ports and protocols. This again may perturb the operation of the certain parts of the application, or at least require reconfiguration of the VM image, the host OS, and/or the 3G network.

Most networks today run IPv4 while vNurse uses IPv6 for NEMO. In the short-term, various solutions for IPv6/IPv4 inter-operation exist, although they add to the engineering complexity and create additional management and operational overhead for any working solution. In the longer term, native IPv6 support will enable greater use of mechanisms such as NEMO.

vNurse uses the existing 3G infrastructure for wireless coverage. However, there are a number of challenges to be faced as 3G usually provides IPv4 addresses that are ephemeral and itinerant [122]. Because NEMO is IPv6 based, some form of translation mechanism from IPv6 to IPv4, back to IPv6 is necessary, to network the NEMO Mobile Router (smartphone) and the NEMO Home Agent (Physician Server). I discuss our choice of mechanism from the available methods below.
Teredo

Teredo [123] is an IPv6 transition protocol. Transition protocols were designed as temporary fixtures to facilitate the crossover from IPv4 addressing to IPv6. This phase is expected to occur as the Internet transitions from IPv4 to IPv6. In order to maintain the connectivity between these 2 otherwise fractured networks, some method for tunnelling across both domains is necessary.

Teredo is essentially a service that allows IPv6 hosts behind IPv4 NATs to tunnel their IPv6 packets by encapsulating them in IPv4 UDP packets. The Teredo protocol uses a client-server architecture to enable this connectivity. The IPv6 host essentially runs a Teredo client that runs a tunnel between itself through the NAT and into the public IPv4 domain. Here the packets are forwarded to the Teredo server, which provides a tunnel from itself to the IPv6 domain. These servers are stateless and scattered across the Internet.

6to4

6to4 [124] is another transition mechanism that unlike Teredo does not require the setting up of specific tunnels; instead traffic is routed through 6to4 relay servers. These servers route traffic between 6to4 and native IPv6 domains. The anycast 192.88.99.1 address has been allocated for the purpose of sending packets to a 6to4 relay router. This mechanism requires a global IPv4 address in order to work. In this mechanism, the host is entirely responsible for the encapsulation and decapsulation of the IPv6 and IPv4 packets respectively.

DSMIPv6

DSMIPv6 [125] specifies the support of IPv4 address for NEMO and MIPv6. Specifically, it allows for mobile nodes to be assigned both IPv4 and IPv6 addresses. It allows the transport of IPv4 and IPv6 tunnels to the Home Agent, making it possible for mobile nodes and mobile routers to move in both IPv4 and IPv6 domains. It utilises the usage of dual IPv4 and IPv6 stacks within the mobile entities. DSMIPv6 also considers the scenario whereby a mobile node moves within a private IPv4 network, though not all kinds of NATs are fully supported.

m6t Protocol

The m6t\(^{10}\) protocol provides a simpler approach compared to that proposed by DSMIPv6. It proposes to support NEMO/MIPv6 over IPv4 by setting up a direct IPv4 UDP tunnel between the mobile node and its Home Agent. Essentially, a virtual IPv4 m6t interface is created that sits between the IPv6 and IPv4 layer in the host. This approach does not impact the existing implementation of NEMO/MIPv6, unlike DSMIPv6.

\(^{10}\) http://natisbad.org/m6t/
5.4 Evaluation

5.4.1 Phase One: Feasibility Testing

I first conducted some preliminary tests on the feasibility of using a smartphone for sensing. In these tests, I have opted to use the HTC G2 Hero smartphone as our prototype device. It has a Qualcomm MSM7200A CPU of 528 MHz, 512MB of onboard memory and 288MB of RAM. For our purposes, it has the following local sensors: (i) Trackball with Enter button, (ii) Internal GPS antenna, (iii) 5.0 megapixel color camera with autofocus, (iv) G-sensor (accelerometer), (v) Digital Compass and (vi) 3.2-inch TFT-LCD touch-sensitive screen with 320x480 HVGA resolution. It also supports the following wireless networks: (i) Quad-band GSM/GPRS/EDGE/3G (UMTS), (ii) Bluetooth 2.0 with Enhanced Data Rate and (iii) IEEE 802.11 b/g (WiFi).

Network Consumption

In order to gauge the bandwidth consumed by sending sensor data from the smartphone across the network to a server, I wrote a python script that polled all local sensor readings (from a HTC Hero running Android 2.1) and sent them to a specific server across a WiFi network for a duration of 500s. I repeated the experiment with both UDP and TCP protocols. The results are shown in Figure 5.3. We see that the approximate cost in terms of bandwidth consumption is approximately 5 Kbytes/s at a maximum sampling rate.

I next measured the network metrics of the WiFi network itself to gauge the effect of a single sensing smartphone on the available bandwidth. I used Iperf to conduct a series of bandwidth, jitter and network loss experiments. In this case I configured Iperf to send 1Mbyte of data to the corresponding server and varied both the packet sizes and target bandwidths. From Figure 5.4, we see that the throughput achievable in the wireless network is 6Mbit/s. This means that the smartphone consumes on average 0.66% of the available bandwidth.

3G bandwidth cost

Currently, T-mobile UK has a fair use policy of 40Mbytes a day £1. At a local sensor sampling rate of 5 KBytes/s, a fully sensing phone (at maximum sampling rate) would use up its available data in approximately 2.3 hours. At that rate, a full day’s worth of sensing would cost around £10.80. Given that some mobile phone planes do include unlimited internet usage, this is figure seems reasonable.

---

12 http://sourceforge.net/projects/iperf/
13 http://www.t-mobile.co.uk/services/uk/fairuse/
Figure 5.3: This graph shows the total amount of bandwidth consumed by a single smartphone that is polling all of its sensor data and pushing it to a local server across a Wifi network.

**Power Consumption**

I have examined the power consumption of the smartphone in different configurations and scenarios during data collection. As I did not want to bias my evaluation with the power effects of any network handovers, in the following tests, the smart phone remained in a stationary position. Our main method of collecting battery information was through the Python API on the Android Scripting Environment (ASE)\(^\text{14}\), which is available on Android platforms via the Android Market. Because I did not wish to run additional Python scripts outside of the Debian VM image, I exported the “AP-PORT” values into the VM. This made it possible to access the Python Android API from within the shell of the Debian file-system. Using this API, I am able to get information about battery levels in terms of an overall remaining percentage and as well as in terms of power. One potential problem with using this API is that the LCD screen needs to remain lit in order for the sensor readings to be collected. As a result of this, I have forced the LCD to remain lit throughout all our experiments, biasing power consumption figures. As a result, the figures should not be taken as absolute values in themselves but should in fact be considered relatively.

*On the Effect of Wireless Medium*

I also wanted to the gauge the power impact of running the sensor application on differ-
ent physical connectivity. In our proposed architecture, I used 3G for the uplink interface and WiFi for the ‘internal’ interface of the mobile network. In the future, there might exist smartphones with two or more WiFi interfaces. If this were to be the case, it would possible to multi-home between the 3G interface and one of the WiFi interfaces; the other WiFi interface could be reserved for the NEMO mobile network. In such an event, Figures 5.5 and 5.6 show the difference in power consumption between the 3G and WLAN media.

*On the Effect of NEMO*

Here I compare the power consumption of a smartphone running the sensor application over 3G and a smartphone running the sensor application while acting as a NEMO Mobile Router. This basically involves creating a wireless hotspot for wireless mobile sensors. In order to create this hotspot, I enabled the WiFi and 3G interfaces simultaneously and started the NEMO daemon. Figures 5.5 and 5.6, show the additional power costs of running the NEMO Mobile router daemon as well as maintaining the wireless mobile hotspot.

The results show that remote sensing over 3G consumes more power compared to remote sensing over WLAN. Also, as expected, enabling full mobile network functionality consumes the most power out of all the tested scenarios.
Figure 5.5: This figure shows the states of the battery level of the HTC Hero under different conditions.

5.4.2 Phase Two: Operational Testing

In this second phase, I focus on battery consumption in a realistic setting. The scenario is based on a patient moving from his home to his hospital. I tested the battery usage as the smartphone moved from one location to another as it was sensing.

The results are similar to those collected in phase one; power consumption increases as more functions are enabled. While the results are promising in that they show that the vNurse architecture is feasible as a technique and that mobile networking can be applied to remote health monitoring, they also underline the importance of improving and/or increasing the battery lifespan in mobile devices for such purposes.

5.5 Conclusion

Utilisation of commercial off-the-shelf technology in WBANs in healthcare opens up the possibility of using inexpensive and unobtrusive monitoring of patients during their daily activities for prolonged periods of time. The benefits of such applications are numerous. Patients will now be able to have continuous monitoring in the comfort and privacy of their homes. Less medical manpower per patient will be required, which means that healthcare workers will be able to manage their time with greater efficiency, thus providing better care at a reduced cost. Constant remote monitoring also means faster response times in the event of a medical emergency, and the potential for speedier
and/or more accurate diagnosis in some cases. Doctors will also have much more information to make better informed decisions about ongoing care. As the demand for health-services increases, so does the availability and variety of sensors. Today, there exists wearable sensors that have applications in stroke, myocardial infarction rehabilitation and traumatic brain injury rehabilitation [126]. With the technology available today to enable these and the possibility of remote patient monitoring, based on smartphones as the controlling platform, there is a potential for huge revolution/changes to healthcare practice.

In this chapter, I have demonstrated a unique method of leveraging virtual machines to enable network mobility support on Smartphones. The benefits of this technique are (i) easier configuration, (ii) easier testing, (iii) easier verification and (iv) enhanced portability. I have shown that it is feasible to use virtual machines on mobile devices, such as Smartphones to enable network mobility comparatively easily. With this technique, the testing of network mobility protocols is no longer limited to costly testbeds.

Figure 5.6: This figure shows the states of the power level of the HTC Hero under different conditions.
Figure 5.7: This figure shows the route taken by the NEMO MR during the experiment.
Figure 5.8: This figure shows the percentage of power of the HTC Desire over time, in various modes.
Figure 5.9: This figure shows the percentage of voltage of the HTC Desire over time, in various modes.
Chapter 6

Conclusion

6.1 Contribution and Summary

Network Mobility is the ability of a network (consisting of a mobile router and a number nodes) to dynamically change its point of attachment to fixed network infrastructure and yet maintain network connectivity. Network Mobility Support is concerned with the management of the mobility of a network. As network mobility is currently not a core functionality supported by the TCP/IP network stack, additional mechanisms are required to enable it. In this thesis, I have focussed on network protocols of the Network Layer, such as NEMO. This thesis argues that the current state of these mobile network protocols is limited and proposes two innovative techniques that make the study and implementation of network mobility protocols easier.

The increasing portability of computing devices as well as the ubiquity of wireless network infrastructure, is opening up the opportunities for applying network mobility in a number of scenarios (See Chapter 1.1.1). These applications are worth exploring due to the advantages of mobile networks (See Chapter 1.1.2).

However, there is a problem with providing network mobility support, as it is currently not supported as a core network functionality in the TCP/IP stack. While forms of network mobility support do exist (albeit in limited capacities), the cost of research in network mobility support (i.e. Experimentation and Implementation) is very high, resulting in a number of open issues (See Chapter 1.1.3).

A contributing factor to the existence of these challenges, is that researching network mobility support is not accessible. Some reasons for this are: (i) Complexity of Mobile Network Scenarios and (ii) Difficulty in Implementing New Network Mobility Protocols (See Chapter 1.2). This thesis addressed some of the difficulties in studying mobile network protocols, in hopes of improving the current state of network mobility support. In support of these goals, I have made the following contributions:

1. I have shown that analytical approaches are useful in mobile network applications because they abstract the architecture from the engineering, allowing for straightforward protocol comparisons

2. I have shown how existing virtualisation platforms may be used as a practical tool towards studying network mobility protocols
3. I have provided an innovative example of how virtual images may be extended for use in practical settings

In addition to these contributions, I would like to highlight that my work has the following ‘firsts’ that altogether demonstrate how I have made enabling mobile networks easier than it currently is:

- First analyses that shows that alternative approaches to NEMO/tunnelling could work and have acceptable (possibly better) performance
- First integration of a mobile network protocol into a practical, freely available tool (now part of Cloonix)
- One of the first uses of virtualisation of on a mobile phone, and first practical demonstration of mobile network protocols on a smartphone over an existing production network

I will elaborate on my contributions in the order in which they have been presented.

In Chapter 2, I survey the current state of network mobility support and introduce the general mechanisms available, as well as clarify the nomenclature of this field that has been used throughout the thesis. I introduced the fundamental operations required for mobility support, the prerequisites of all mobility support solutions. To provide a background to the architectural discussions in Chapter 3, I survey the history of two approaches to end-host mobility, the Tunnelling and Naming approaches. I end with a glimpse of some of the network mobility protocols that have been proposed.

In Chapter 3, I highlight the limitations of the current state of network mobility support and encourage alternative approaches to be considered that also have advantages. I do this by focussing on route optimisation, as it is one of the open issues in mobile networking. I look at three protocols, NEMO, ILNP and OptiNets. I chose NEMO, as it uses the Tunnelling Approach and is the current IETF standard for network mobility. I chose ILNP as it uses the Naming Approach and I chose OptiNets as it has route optimisation, and has previously been analysed.

To analyse the cost of each protocol with regards to providing route optimisation, the mobile network scenario I chose is that of the London Circle Line, which is a line on the public metropolitan rail system in the heart of London, UK. It is also one of the proposed potential applications of mobile networks introduced in Chapter 1.1.1. I also used real passenger statistics to generate passenger movement traces. These combined with formulated general equations for packet and bandwidth cost, allowed me to work out the cost of providing for route optimisation for each protocol. I also varied the mobility of the passengers as well as the number of individual correspondent nodes, to see how operational conditions could impact the results.

The results show that there exists a trade-off between provisioning for mobility and having optimal routing paths for mobile traffic flows. There are no optimum approaches, but the choice of approach used should consider the mobile application context in which it is to be applied. NEMO may be better suited for mobile networks that do not have mobility aware nodes. And for ILNPv6, it might be better suited where the mobile network is a mix of mobile and static nodes.
In Chapter 4, I look at the current tools available to study mobile network protocols. Having identified some of the limitations on the state of network mobility support, in this chapter, I look at the difficulty in studying and implementing possible solutions. I begin by exploring some of the requirements an architect faces when designing new protocols. I then do a comparison between tools such as simulators, testbeds and virtual networks. I then introduce Cloonix, which is a configurable virtual network that uses UML (User Mode Linux). My contribution in this chapter is a modification of Cloonix, entitled Cloonix-Net, which adds the functionality of running multiple UML machines each with its own separate kernel. This allows the user to create custom network topologies that consist of distinct network entities. Cloonix-Net also allows for multiple kernels and multiple file-systems. The user can create a unique testbed topology consisting of very different machines to suit his needs.

My second contribution in this chapter was to extend the utility of Cloonix-Net as a mobile network analysis tool through modification. I conducted a performance analysis using 3 mobile network topologies; Mobile IPv6, NEMOv6 and a Nested Mobility Topology (Mobile IPv6 within NEMOv6). For each topology, I built an equivalent model in three instances, the first using Cloonix-Net, the second with a real testbed with wired connections and the third with a real testbed with wireless connections. The experiment tested between the mobile correspondent node (MIP-CN) and the NEMO mobile router (NEMO-MR). I used iperf to measure throughput, packet loss and jitter between these two nodes. I varied the bandwidth and packet size of a UDP flow, as I wanted to replicate a real-time application. The results showed the physical limitations of Cloonix-Net in comparison to the wired and wireless testbeds with respect to throughput. The results also showed that beyond the range of 10Mb/s bandwidth at 1400byte packet size, Cloonix-Net experienced negligible packet loss and jitter. While Cloonix-Net has certain operational limits that are below that of a real wired and wireless testbed, I have shown that as a tool, it is still useful to network architects for testing new network protocols and topologies.

In Chapter 5, I built upon the work done with User Mode Linux, and demonstrated how virtual images may be used to easily enable real mobile networks. Specifically, I show how it is feasible to use virtual images on Smartphones to build a mobile PAN for the purposes of remote health monitoring. I focussed on enabling the mobile router as it forms the key architectural point for enabling and managing mobility. I also concentrate on performance issues such as the impact on battery life of enabling network mobility over various mediums such as 3G and WLAN.

First I introduce the subject of remote patient monitoring and E-health networks and show how combined with wireless body networks, it is a feasible problem for the application of mobile network support. I elaborate on this concept by going through three target scenarios (See Chapter ??).

I then discuss some of the important design issues (e.g. mobility, sensor and usability). After a brief background on the usage of virtualisation on smartphones, I give an overview of my proposed architecture, vNurse. vNurse builds upon and extends such technologies that are inexpensive and readily available to allow for better patient monitoring. This proof-of-concept system leverages three COTS technologies, namely, (i) virtualisation; (ii) network mobility; and (iii) wireless sensor networking. The main contribution behind vNurse is this utilisation of virtualisation on a guest OS virtual machine (VM) image – specifically, a Debian VM image – to encapsulate the WBAN functions within
a secure, configurable and self-contained environment, leveraging the Android host OS capabilities for onward communication via the smartphone. This encapsulation, also enables mobile router and network mobility functionality on the smartphone easily.

The evaluation of vNurse was carried out in two phases. The first phase involved preliminary tests carried out on HTC G2 Hero smartphone, as part of an internship with T-Labs Berlin, and the second phase was carried out on a HTC Desire smartphone in London. For the first phase, I measured basic network usage when sampling and the corresponding economic cost to the user. I then measured the power consumption of sensing by varying the wireless medium (3G vs WLAN) and the impact of enabling network mobility on the smartphone, which involved running the NEMO protocol and turning on the wireless hotspot function simultaneously. The results show that remote sensing over 3G consumes more power compared to remote sensing over WLAN. Also, as expected, enabling full mobile network functionality consumes the most power out of all the tested scenarios. In the second phase, the focus was on testing the power consumption in a realistic remote sensing scenario. I enabled the HTC Desire smartphone as a mobile router, and physically moved the phone (as a roaming patient would) and collected power consumption data. The results were very similar to those collected on the HTC Hero in phase one. The results also highlighted the importance of increasing or improving power utilisation to lengthen the lifespan of enabling network mobility for remote sensing.

6.2 Discussion

In Chapter 3, I used a realistic mobile network scenario based on the London Circle Line. I used real passenger statistics, from which I generated passenger mobility traces for my emulation. While this approach is suitable in this case, it is important to understand how the traces impact the results. In this case, other factors must be considered before accepting the validity of the data. For that experiment it was important to validate, the final number of passengers for each run, as well as the number of passengers in each train. These two factors reflect the inherent complexity of mobility scenarios.

In Chapter 4, using identical topologies and testbeds (virtual, wired and wireless), I have shown the operational limitations of Cloonix-Net, for a given set of hardware and software conditions. It is conceivable that these limits vary from configuration to configuration. It is thus appropriate for users of Cloonix-Net to first seek the operation limits of their own setup. The experiments carried out were also between two fixed nodes. It is thus conceivable that experiments carried out between a fixed and mobile entity or between two mobile entities may have other unforeseeable effects on the results. While there is utility in Cloonix-Net, it is prudent to first discern the operation limits of the test in question (as is the case for most experimental tools).

In Chapter 5, I enabled network mobility on an Android smartphone for the purposes of remote health monitoring, using a custom virtual image installed onto the device. While the objective of enabling network mobility support was achieved, the actual use of this method for commercial operation still requires more study. This is reflected in the manner in which mobility was enabled. For example, NEMO requires IPv6, however, currently 3G networks only support IPv4. For the experiment, the m6t Protocol was used as a workaround, but this set up a IPv4 tunnel between the MR and its HA, resulting in zero binding updates, only location updates. As the frequency of location updates
are much more frequent than binding updates, this arguably has limited impact on battery consumption. However, the results do not inform us what would happen when 3G networks do support IPv6.

### 6.3 Future Work

Despite having shown that the operational limits of Cloonix-Net are within useful bounds, there are certainly improvements that can be considered for future work. First, is the ability to export topologies and virtual machines easily as a single file object. Having a single file which can then be archived or shared with other researchers would prove invaluable. Second, is the incorporation of a Physical Layer that allows for mobility and wireless experimentation. While the exclusion of wireless results, certainly aids in some aspects of protocol study, especially in areas where the operation of the protocol is in priority. In such cases, having less compound effects simplifies the results and makes it easier to draw conclusions that depend on the protocol operation only. Including this layer is a natural step forward as mobility support usually involves utilisation of a wireless layer. An alternative solution to this would be to allow mobility traces to be exported outside of Cloonix-Net, which can then be input into other network testing tools.

I have mainly leveraged virtualisation to achieve this. User Mode Linux has been part of the Linux main kernel source since 2.6.0; so long as this continues and the Android kernel does not remove it, tools and architectures such as Cloonix-Net and vNurse will continue to be useful. However, support in User Mode Linux for the purposes of network testing is also limited. If these techniques gain more support, it would be prudent to analyse UML and incorporate improvements that allow for better testing and improved operational boundaries.

In Chapter 5, my purpose was to enable MR functionality. But for the purposes of remote health monitoring, another critical component is that of the wireless sensors. During the planning phase of the experiment, it was originally planned to incorporate such sensors in our setup. However, because there was a lack of WLAN sensors available, we ended up testing with a Bluetooth sensor. This too was unsuccessful; As in preliminary tests, the activation of Bluetooth and NEMO protocol resulted in frequent kernel crashes. One area of future work could be the analysis of the Linux kernel to prevent such errors from occurring and to allow for phones to act as mobile routers.
References


[71] C. J. Bennett, S. W. Edge, and A. J. Hinchley, “Issues in the Interconnection of Datagram Networks.” IEN 1, July 1977. his document is also known as INDRA note 637 and PSPWN 76.


