The Study of Mobile Network Protocols with Virtual Machines

D. Rehunathan, S. Bhatti
University of St Andrews
dr,saleem@cs.st-andrews.ac.uk

V. Perrier
clowncoder@clownix.net

P. Hui
Deutsche Telekom Laboratories (Berlin)
pan.hui@cl.cam.ac.uk

ABSTRACT

With the rapid proliferation of wireless mobile devices in today's society, it is becoming increasingly useful to aggregate mobile devices moving together as a single mobile network. It is also necessary to test new mobile network applications and protocols (e.g. NEMO) in realistic scenarios, where an incremental development approach can be adopted in order to experiment and explore. We present a simulation framework, Cloonix-Net, a virtual network tool using User Mode Linux (UML) machines, for the purposes of building and testing such scenarios. We show that studying mobile network protocols with such a framework is a beneficial step towards better understanding network mobility protocols.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Verification*; I.6 [**Simulation and Modelling**]: Applications

General Terms

Mobile Networks, Network Protocols

Keywords

Network Simulation, Virtualisation, Network Mobility

1. INTRODUCTION

Mobile networks are becomingly increasing relevant in today's society. With the rapid advancement of wireless technology there is an ever increasing desire to remain 'connected' to the Internet. Users are increasingly dependent on personal devices that have increased network capability, e.g. smartphones and PDAs that have Wifi and Bluetooth. In such connectivity environments, there are numerous scenarios where the aggregation of multiple mobile devices as a single *mobile network*, is both applicable and efficient.

Mobile networking is applicable in, for example, both military [18] and civilian scenarios [19], such as providing Internet connectivity to passengers on a train or a plane in-flight. Mobile networking is useful whenever there are a large number of mobile nodes moving together in synchronisation. By aggregating individual nodes as a single mobile network (See Figure 1), we can provide network connectivity efficiently and effectively [15]. For example, a single node in a newly formed mobile network can be nominated or selected to act as a *mobile router (MR)*, thereby conserving power for the rest of the nodes network. This configuration makes it possible for nodes that do not have wide area wireless capability to still get access outside the network by using other means (e.g. Bluetooth or LAN) via the MR. It is also useful in smaller devices, such as sensors in a car or even on a person (e.g. in Body Area Networks [20]), which may or may not have wireless connectivity.

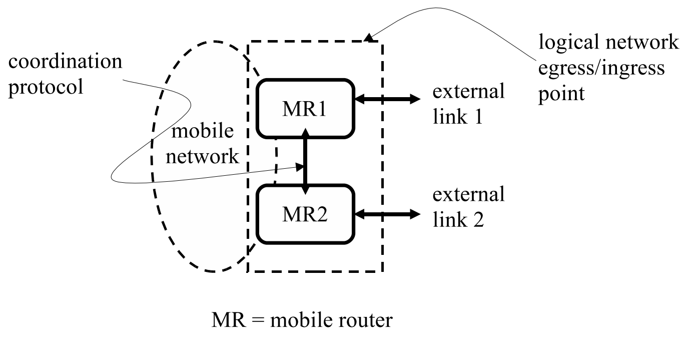


Figure 1: The basic elements of a mobile network. A mobile network may consist of one (or more) mobile router(s) (MR) providing connectivity through one (or more) external links. If more than one are present, a coordination protocol may be used to allow for both links to be used in unison. Nodes that are part of the mobile network gain access by forwarding packets through the MR(s).

Unfortunately, the current network architecture does not support network mobility in the core protocols. New protocols are required to extend existing functionality to incorporate mobile networks. The current IETF standard for Network Mobility (NEMO) [5], still has numerous issues that can be improved, such as route optimisation and multi-homing [15]. Despite being available for a number for years, the NEMO protocol has not seen wide deployment. The large number of proposals for add-ons and modifications [17] to NEMO as well as the possibility of alternative protocols, e.g. [3], is also an indication that users and the research community feel that there is still scope for further developments in this area.

In the study of these new protocols, it is necessary to ensure that they function correctly before deploying them in an operational environment. A safe, sandbox environment is required for evaluating new mobile network protocols, using a platform that allows for simple as well as incrementally more complex experiments to be performed. Currently, there does not exist an adequate number of tools for testing, evaluating and understanding these new protocols. In this paper, we propose such a tool, which we call *Cloonix-net* a modified version of *Cloonix* [16], that provides such functionality.

In Section 2, we give a summary of mobile network research at the network layer, and give an overview of NEMO, which is the current IETF standard for network mobility. We also introduce the use of virtualisation and User Mode Linux (UML), highlighting the benefits and limitations of Cloonix. In Section 3, we explore the work that has already been carried out in this problem space by reviewing some of the existing solutions. In Section 4, we introduce our solution, *Cloonix-net*. In Section 5, we describe a performance analysis and discussion of Cloonix-net. In Section 6, we summarise our contributions and conclude.

2. TESTING NETWORK PROTOCOLS

When developing a new network layer protocol, the researcher is interested in experimentation to explore 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.

For an experimental tool, the practical costs of such protocol testing – additional requirements such as resources overhead, ease of use and re-usability – can be important. Ideally, a good experimentation tool should provide control of such features to the researcher.

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 comparative performance analysis, i.e. 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.

As network mobility protocols are often intended for use in realistic scenarios, we need to consider the situation there may be large numbers of mobile nodes: indeed, that is often used to justify the use of mobile networks in the first place. Any experimentation tool should thus allow the tester to construct realistic scenarios, with a reasonable number of mobile nodes. In this aspect, we recognise that there is a limit 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 typically operates within kernel space, so any errors in the protocol often halt the kernel and may cause overall system failure. This makes debugging on a real machine in a testbed extremely laborious and difficult (in some cases, it may be practically impossible). There are two common ways around this problem. The first is to have separate debugging machine monitoring the experimentation machine, the two being connected via a serial cable. The second method is to virtualise the experimentation machine and run it as a VM image within a test machine. While both methods allow debugging to be undertaken in a controlled, manner, 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.

An important consideration in any simulation is the model that should be created: what characteristics should be included and what excluded. In this case, we consider the exclusion of wireless effects in our tool. While wireless effects are certainly evident in any real world scenario for mobile networks, we believe that adopting an incremental approach towards protocol testing, at the early stages of development, it becomes convenient to reduce complexity and validate the simplest possible case. Once that has been done, we then move on to more complex and realistic cases. By not including 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 advantages 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. A full discussion and justification of this position (with quantitative analysis), is given in [19].

3. BACKGROUND

3.1 Virtualisation with User Mode Linux

Modern computers are capable of using virtualisation to execute multiple instances of many smaller virtual machines (VMs), each running a separate operating system instance [4]. Additionally, We also see a trend towards the creation of person mobile wireless networks as smart-phones are quickly reaching the point where they too can run VMs [1], with companies such as VMware planning to release versions of their virtualisation platform for mobile phones[2], and plans for Android (Froyo 2.2)[3], while Audi recently built a car that has mobile Wifi built-in [4], via their new *MMI Touch and Infotainment System*.

The current landscape of smart phones today are released with proprietary operating systems already installed. While it is possible in some cases for users to install alternative 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 incorporate 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, effectively, run as a single process, allows the user to manage the resources (e.g. CPU time and hardware interfaces). This means that

[1]www.qualcomm.co.uk/products_services/chipsets/snapdragon.html
[2]www.vmware.com/company/news/releases/mvp.html
[3]www.techcrunch.com/2010/05/20/froyo-android/
[4]www.audiusa.com/us/brand/en/exp/progress/Upcoming_Models/new_a8.html

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.

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 [6]. 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 UML kernel by making system calls as usual [7]. As of Linux 2.6.0, UML has been integrated into the main kernel source tree. While UML was designed for 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 [5]. *"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 images are used to encapsulate OS-specific requirements and configurations. This is to allow for multiple users of the test-bed to use the common hardware easily. For our experiments, we use UML for hosting virtual machines in our testbed, it is also used for kernel development due to its sand-boxing 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 [22], the following points have been observed and need to be considered for our experiments:

Long Duration. Virtualisation has minimal effects on UDP experiments when the experiments are carried out over a long period of time. Experiments that measure instantaneous throughput are often inaccurate as a result of the virtualised platform, due to a considerable increase in variance of throughput close to saturation (e.g. $\sim$30Mbps).

Packet Size. Virtualisation creates a limitation on experiments that require small packet sizes and high bit rates (less than 30,000 packets per virtualisation platform). The recommended packet size is 1470 bytes. Other important factors that might skew our results are the impact of running multiple UML instances on a single machine and interference between them. From [22], we see that two UML instances running on the same machine have some impact as the packets are buffered for a random amount of time, for the UML instances to be context-switched, before they are sent over the wireless interface.

3.2 Achieving Network Mobility

Previous work [23] identifies fundamental requirements for mobility, including two key functions: (i) locating the mobile host for initiating communication; and (ii) preserving communication sessions when a node moves. For a mobile network scenario, we have chosen to represent these key functions as three distinct operational phases:

1. *Initialisation*: when a mobile entity – a visiting mobile node (VMN) or a mobile router (MR) – discovers a network access point to wider connectivity and configures itself to communicate via that access point.
2. *Rendezvous*: the process by which a correspondent node (CN) can initiate communication correctly with a remote mobile network (i.e a node within a mobile network).
3. *Handover*: this is the process of maintaining existing communication sessions as the mobile network changes its network point of attachment.

These three phases are critical in all forms of mobile networking. In general, all mobile networking protocols have to perform these functions successfully, in one form or another, for mobility to occur. It is how these operations are enacted (which is protocol-dependent) that specifically defines a particular approach (architecture) to realising mobility, and it is this that requires definition, analysis and evaluation within a research context.

3.3 The NEMO Protocol

The current IETF standard for mobile networking is the NEtwork MObility protocol (NEMO). Work on the NEMO protocol is currently being progressed in the IETF Mobility EXTensions (MEXT) WG [10]. This WG is working towards a unified mobility architecture for IP, by integrating NEMO with Mobile IPv6 [2] (for mobile hosts), MONAMI [9] (for multi-homing) and IKEv2 [14] (for managing key exchange for security). The primary goal of this WG is to enhance IPv6 mobility such that it will be suitable for large scale deployment scenarios. (On a broader scale, this involves dealing with issues regarding harmonisation of mobility with multi-homing and security also: an item for future work.)

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)*[6]. This implementation is for Linux kernel v2.6 and was originally based upon a previous Mobile IPv6 implementation for *Linux MIPL2* [12]. NEPL has been developed and tested together by the Go-Core Project (Helsinki University of Technology)[7] and the Nautilus6 Project (WIDE)[8]. The second implementation of NEMO is for the BSD platform, and is named *SHISA*[9]. This implementation includes source for Mobile IPv6 as well as NEMO. The current version of SHISA is based upon [11]. 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 [5]. This is a general specification and there are still some issues that remain unresolved that impede it from being adopted more widely [15].

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

[5]www.orbit-lab.org/

[6]http://www.nautilus6.org/implementation/index.php

[7]http://go.cs.hut.fi/

[8]http://www.nautilus6.org/

[9]http://www.mobileip.jp/

MR. This traffic is encapsulated through the tunnel back to the HA where it is de-capsulated and forwarded. To correspondent nodes (CNs), the mobile network appears to be within its home network.

3.4 Challenges in NEMO

Currently, there are five broad areas of research for NEMO, as listed below. All these are potential areas where the standard NEMO architecture can be improved and would require experimentation.

1. Route Optimisation within NEMO [17]. The NEMO protocol works by creating proxies between Mobile Routers (MR) and their home networks. Ingress packets meant for the MR are forwarded via an IPv6-in-IPv6 tunnel. Similarly, egress packets from the mobile network are forwarded back to the home network. This approach hides the actual mobility of the network by making it appear as though it never left the home network. Relaying via the home network in this way may lead to sub-optimal routing. Another problem is that this 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.

2. Multi-homing with NEMO [8]. Network Mobility allows for multiple nodes to remain connected to the network. This capability encourages the ubiquity of the Internet into everyday objects. However, it follows that to remain ever-connected to the Internet, there must be service available at all times. To maximise the possibility of this, the ability to multi-home across multiple heterogeneous communication mediums is an important one. Such connectivity provides for redundancy, load-sharing and policy routing.

3. Nested mobile networks with NEMO [13]. 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.

4. Optimised handoffs for NEMO [21]. With NEMO, handoffs are handled my 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 then 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.

5. Applications of NEMO in MANETs and VANETs [24]. One use of mobile networks is in vehicular area networks (VANETs) mobile ad hoc network (MANET) scenarios. We can think of vehicles as a mobile network, as they are mobile, have their own power supply and can contain multiple network devices. This scenario has characteristics such as the rectilinear paths that vehicles often take within modern cityscapes, and the possibility of using fixed infrastructure (e.g. lampposts) to aid connectivity and mobility.

4. A COMPARISON OF EXISTING TOOLS

4.1 User-Mode Linux

User Mode Linux (UML) is virtualisation functionality within the Linux operating system. It 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 increasing host stability even further. However, 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.

IMUNES (Integrated Network Topology Emulator/Simulator) [25] is a network emulator 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 (Common Open Research Emulator) [1], 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.

However, both IMUNES and CORE use paravirtualisation: all the VM images share the same kernel, filesystem, processor, memory, clock and other resources directly. For experimentation, a single custom network stack incorporating the required network layer protocols (e.g. NEMO and Mobile IP) needs to be built, which may be complex. 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.

4.2 Use of Simulators

Many mobile simulation systems exist, e.g. OMNET[10] and NS2[11]. While simulations are used widely within the research community, there is a reliance on that community to support to collaborate and reproduce 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 a considerable investment of time, to port the existing 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, we have the ability to run real working code in a controllable and reproducible environment. While newer simulators such as NS3[12] are planning to permit incorporation of system code, such tools still lack maturity.

[10] `http://www.omnetpp.org/`
[11] `http://www.isi.edu/nsnam/ns/`
[12] `http://www.nsnam.org/`

(Note: OMNET does include a module of a full FreeBSD stack.).

4.3 Use of Testbeds

Often, accurate and realistic testbeds are expensive and time consuming to procure, setup, configure and maintain. 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 also 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 [13] and Emulab [14]. However, these still have limitations in terms of configuration, customisation, privacy and scheduling. PlanetLab [15] is another example. It has issues regarding congestion, especially when it gets close to certain conference deadlines. We are generally in favour of the usage of testbeds, and we believe that they should be incorporated whenever possible. However, we take the position that using VMs 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 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.

4.4 Use of VMware

One of the first tools we considered was VMware[16]. VMware specialises in virtualisation software. Under an 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 software (which runs on all major platforms), we spent some time exploring 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.

[13] `http://www.orbit-lab.org/`
[14] `http://www.emulab.net/`
[15] `http://www.planet-lab.org/`
[16] `http://www.vmware.com/`

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 (i.e. network switches) to to allow suitable topologies to be configured. The second obstacle was to figure out how to emulate mobility 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). The overhead required to manage these VLANs dynamically for an experimental scenario (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 our purposes.

4.5 On the General Use of Virtualisation

The following advantages make it worthwhile for using virutalisation in experimentation of mobile network scenarios:

1. It is quick to get up and run for experimentation, especially with the available Mobile IPv6 and NEMOv6 network configurations 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 capable 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 experimental results.
3. Use of VMs means that any calibration or setting up of a network entity (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, either after completion, or 'snapshots' of experiments in progress.
5. In the case of UML, independence of wireless effects makes use of the experimental configuration and interpretation of experimental results easier in the early stages of experimentation. However, there will come a point when those wireless effects will be needed to be investigated also (see Section 5.2), but here the VM can be re-used (see point below).
6. Migration from virtual experiments to emulation and testbed experiments is a natural progression, and could use the existing VM images for deployment on real testbed systems. The kernel of existing VM images 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 with not 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 performance bottlenecks occurring as a result of the operating system of the host machine, rather than effects/behaviour of the experimental VMs. These may be difficult 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 experiments (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.

5. CLOONIX-NET

Cloonix[17] is a configurable virtual network that uses UML. It is recommended as a tool for the study of NEPL [18]. Cloonix is openly available software that creates 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.

The main advantages of Cloonix are the following:

- Plug and play test of network functions.
- Build re-playable demos 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: it still not possible to pause a running scenario and start it again. The main concern for our network protocol research is that Cloonix restricts the user to a single kernel for Debian and Fedora operating systems. Multiple filesystems 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) file. This limitation of one kernel makes it difficult to create a testbed of different network entities.

Cloonix-net is our modification of Cloonix-net 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 for multiple kernels and multiple 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. We 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 (Figures 2(a) and 2(b)), the second represents a NEMOv6 handover scenario (Figures 2(c) and 2(d)), and the third is a Nested Mobile Network, where a Mobile IPv6 node joins a mobile network running NEMO (Figures 2(e), 2(f)). For our experiments, the host machine specifications were:

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

Each UML VM instance had the following specifications:

- total diskspace 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

5.1 Using Cloonix-net

Cloonix provides a WYSIWYG interface for managing the virtual network topology. This topology may be edited in real-time via the GUI (See Figures 2(b), 2(d) and 2(f)) Green circles represent the network interfaces of the VM images. Grey circles are used for the connection of LAN links, which are represented by grey links. Links may be added by selecting the desired interfaces represented by the grey circles. Clicking the green circles will bring up a shell of the selected live VM.

5.1.1 A Primer for Mobile IPv6 experiments

MN handovers can be executed by first placing the MN within its home network (See Figure 2(a)). Communication can then be initiated between the MN and the CN (e.g. using *ping*). A handover is initiated by moving the MN from its Home Network to the Wireless Gate-1. Handover messages can be observed and logged by interacting with the NEPL daemons using the GUI shell. The handover can also be re-confirmed by observing the re-establishment and continuation of communication between the MN and CN.

5.1.2 A Primer for NEMOv6 experiments

Network mobility can be done by first removing the link between the MR and its Home Agent and then creating a new link between the MR and any one of the Wireless Gates (See Figure 2(c)). This action simulates movement of the MR.

[17] `http://www.clownix.net/`

[18] `http://www.nautilus6.org/doc/nepl-howto/`

5.1.3 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 2(e), 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 and access router 1 (AR1 and AR2 respectively).

5.2 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*) [19]. *netem* can emulate certain traffic conditions that can be found in wide area networks. *netem* is capable of controlling 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 added stability makes possible testing 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 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. The third requirement is met up to an extent. In terms of a large number of mobile nodes (VM images), the original author of Cloonix 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.

6. PERFORMANCE ANALYSIS

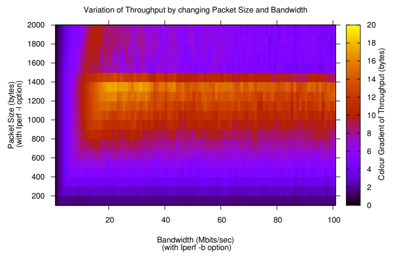
6.1 Experiment

In order to evaluate the network performance of a Cloonix-net virtual network, we 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 2(d)). We used *iperf* to measure throughput, packet loss and jitter between these two nodes. We varied the bandwidth and packet size of a UDP flow, as we wanted to replicate a real-time application. Each combination of variables was executed 5 times and the mean value is plotted in the following graphs. To put our results in perspective, we also performed the same *iperf* tests in two other testbeds with the exact 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 (5GHz, upto to 150 Mb/s). We have also taken steps to ensure that our 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`).

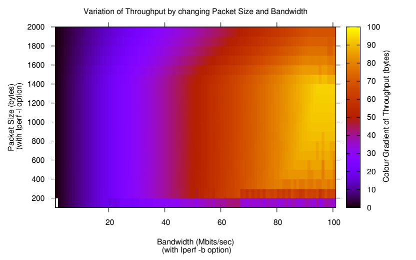
[19]`http://www.linuxfoundation.org/en/Net:Netem`

6.2 Results

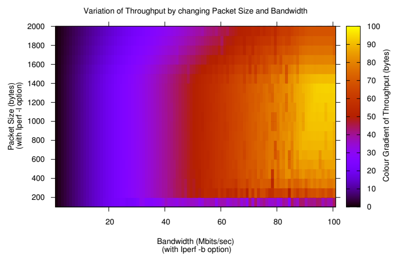
For Figure 3, we see that the overall throughput of the virtual testbed is not nearly as high as the wired and wireless testbeds. This is mainly due to the performance limitations of the desktop hardware, but our chosen hardware has a very modest specification by today's standards. Limiting factors include 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.



(a) Virtual Results



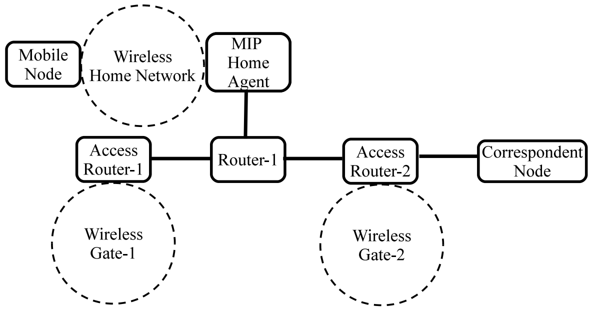
(b) Wired Results



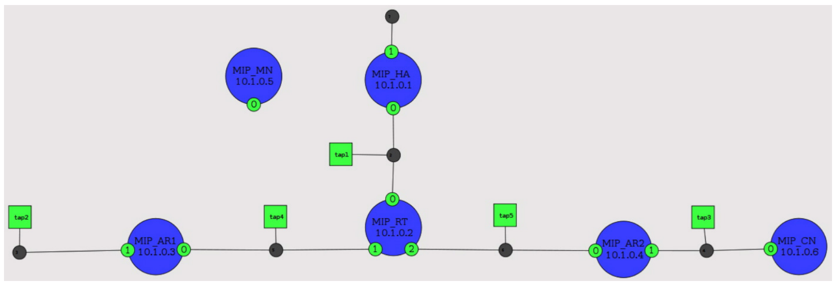
(c) Wireless Results

Figure 3: Throughput graphs of Virtual, Wired and Wireless Testbeds. This figure shows the throughput operating limits (up to 20Mbits/s) of using Cloonix compared to real wireless and wired testbed.

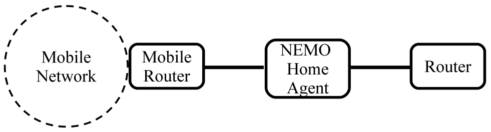
For Figure 4, we see that the while packet loss is generally low for the wired and wireless testbeds, the virtual testbed experiences negligible packet loss until it exceeds the range of 10Mb/s bandwidth at 1400byte packet size, after which it experiences very high packet loss. Thus we can see the operating limitations of using Cloonix-net for such kinds of network testing, beyond which it is no longer realistic.



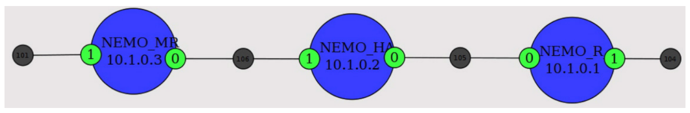
(a) Mobile IPv6 topology



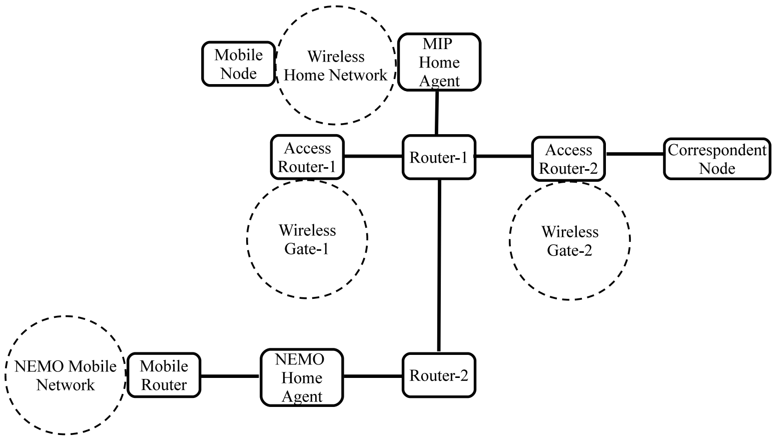
(b) Cloonix Mobile IPv6 equivalent



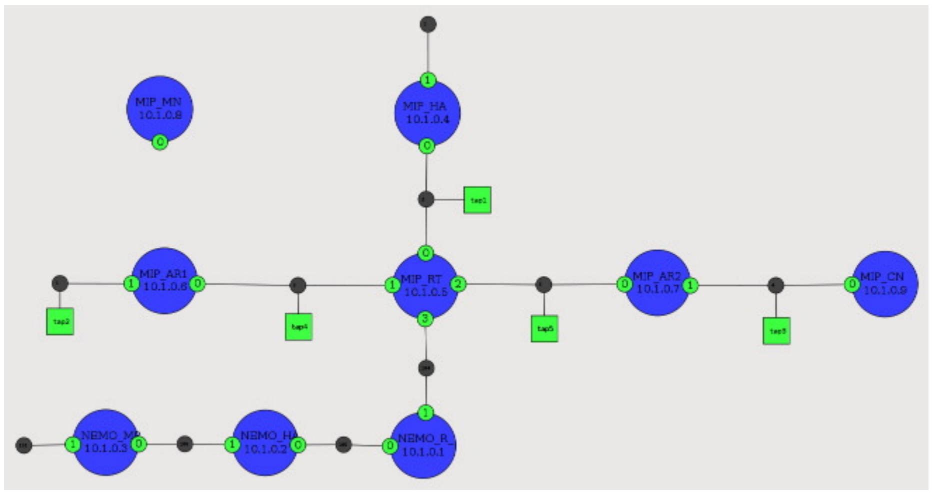
(c) NEMOv6 topology



(d) Cloonix NEMOv6 equivalent



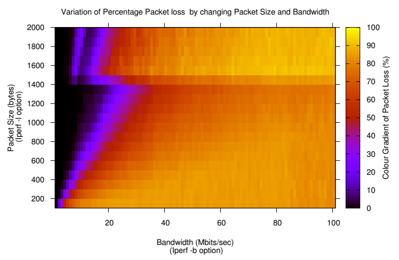
(e) Nested Mobility topology (Mobile IPv6 within NEMOv6)



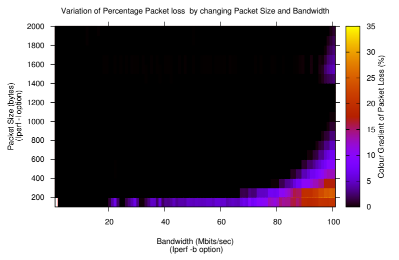
(f) Cloonix Nested Mobility equivalent

Figure 2: Screenshots of both sample topologies.

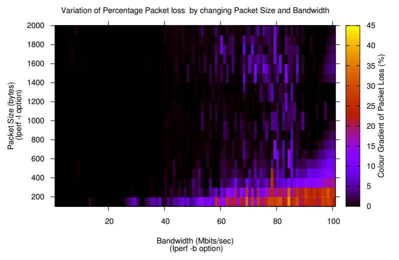
For Figure 5, as expected the jitter is extremely low for the wired testbed and slightly higher on average in the wireless testbed. The jitter in the virtual testbed is very high after a 10Mb/s bandwidth and 1400 bytes packet size, which is in accordance with Figure 4. We suspect that this may be due to the limit of the host operating system kernel: at high load, the Virtual Switch does not get enough CPU time to process all the packets, and starts to drop packets.



(a) Virtual Results



(b) Wired Results



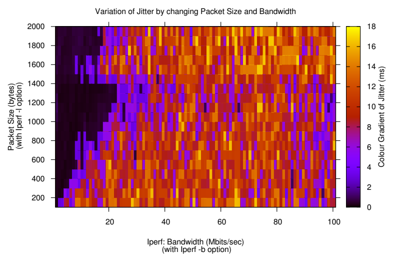
(c) Wireless Results

Figure 4: Packet Loss graphs of Virtual, Wired and Wireless Testbeds. This figure shows the presence of jitter depending on the effective bandwidth and packet size, using Cloonix compared to a real wireless and wired testbed.

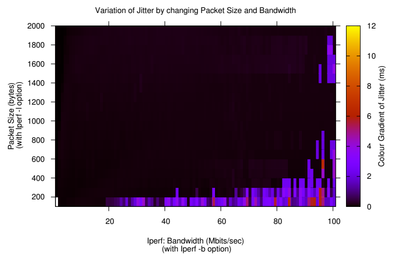
7. CONCLUSION

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), it 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.

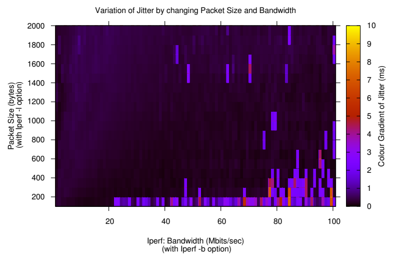
This paper introduces a modification of one of the few tools that now allows for such research to take place conveniently. We have shown that Cloonix-net allows for the testing of mobile network



(a) Virtual Results



(b) Wired Results



(c) Wireless Results

Figure 5: Jitter graphs of Virtual, Wired and Wireless Testbeds. This figure shows the presence of packet loss depending on the effective bandwidth and packet size, using Cloonix compared to a real wireless and wired testbed.

protocols in realistic topologies. We have also demonstrated the experimental limitations to Cloonix-net to the user using very modest hardware. We have compared Cloonix-net to testbeds, both wired and wireless. The results show that Cloonix-net is a realistic network tool that allows for the testing of network protocols for network mobility within certain constraints applied by the host hardware platform.

Cloonix-net has now been integrated into the main Cloonix distribution (from v8.xx `http://clownix.net/`), and includes preconfigured network topologies of Mobile IPv6 (end-host mobility) and NEMO (network mobility) that allow for simple experiments to be conducted out-of-the-box. We hope that this tool will benefit other researchers in their efforts to understand mobile networks. By analysing the advantages and disadvantages of this tool, we hope to have provided a suitable platform for network researchers. We propose that not only is the study of mobile network protocols without wireless effects a beneficial step towards a better understanding of network mobility, but also that the modified Cloonix-net is a useful tool for preliminary (pre-testbed) experiments, which is convenient for use for a wide range of network researchers.

8. REFERENCES

[1] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim. Core: A real-time network emulator. In *Military Communications Conf, 2008. MILCOM 2008. IEEE*, Nov. 2008.

[2] S. M. S. Al-Buraiky. Mobile IPv6 with Linux. *Linux J.*, 2008(169), 2008.

[3] R. Atkinson, S. Bhatti, and S. Hailes. ILNP: Mobility, Multi-Homing, Localised Addressing and Security Through Naming. *Telecommunication Systems*, 42(3-4):273–299, December 2009.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proc. 19th ACM Symposium on Operating Systems Principles*, New York, NY, USA, 2003. ACM.

[5] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963 (Proposed Standard), Jan. 2005.

[6] J. Dike. User-mode linux. In *ALS '01: Proc. 5th Linux Showcase & Conf*, Berkeley, CA, USA, 2001. USENIX.

[7] J. Dike. *User Mode Linux*. Prentice Hall, 2006.

[8] T. Ernst and J. Charbon. Multi-homing with NEMO Basic Support. Technical report, In 1st Intl Conf on Mobile Computing and Ubiquitous Computing (ICMU), 2004.

[9] T. Ernst, N. Montavont, R. Wakikawa, C. Ng, and K. Kuladinithi. Motivations and Scenarios for Using Multiple Interfaces and Global Addresses. Interent Draft, Monami6 Working Group, November 2008.

[10] IETF. Mobility Extensions for IPv6 (MEXT).

[11] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[12] R. Kuntz and J. Lorchat. Versatile IPv6 mobility deployment with dual stack mobile IPv6. In *MobiArch '08: Proc. 3rd Intl workshop on Mobility in the Evolving Internet architecture*, New York, NY, USA, 2008. ACM.

[13] H.-J. Lim, M. Kim, J.-H. Lee, and T. M. Chung. Route Optimization in Nested NEMO: Classification, Evaluation and Analysis from NEMO Fringe Stub Perspective. *IEEE Transactions on Mobile Computing*, 99(1), 2009.

[14] N. Lu, H. Zhou, and Y. Qin. A comparison study of ike protocols. In *Mobility '08: Proc. Intl Conf on Mobile Technology, Applications, and Systems*, New York, NY, USA, 2008. ACM.

[15] E. Perera, V. Sivaraman, and A. Seneviratne. Survey on Network Mobility Support. *SIGMOBILE Mobile Computer Communication Review*, 8(2), 2004.

[16] V. Perrier. Cloonix. http://clownix.net/.

[17] H. Petander, E. Perera, K.-C. Lan, and A. Seneviratne. Measuring and Improving the Performance of Network Mobility Management in IPv6 Networks. *Selected Areas in Communications, IEEE Journal on*, 24(9), Sept. 2006.

[18] D. Rehunathan, R. Atkinson, and S. Bhatti. Enabling mobile networks through secure naming, 2009.

[19] D. Rehunathan and S. Bhatti. A Comparison of Routing for Mobile Networks. In *WiMob2010 - 6th IEEE Intl Conf on Wireless and Mobile Computing, Networking and Communications*. IEEE, Oct 2010.

[20] D. Rehunathan and S. Bhatti. Application of Virtual Mobile Networking to Real-Time Patient Monitoring. In *ATNAC2010 – IEEE Australasian Telecommunications and Network Applications Conf.* IEEE, Nov 2010.

[21] T.-L. Sheu and B.-C. Kuo. An analytical model of two-tier handoff mechanisms for a hierarchical NEMO system. *Wirel. Netw.*, 14(6), 2008.

[22] S. Singhal, G. Hadjichristofi, I. Seskar, and D. Raychaudhri. Evaluation of UML Based Wireless Network Virtualization. In *Next Generation Internet Networks, 2008. NGI 2008*, 28-30 2008.

[23] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek. Reconsidering Internet Mobility. In *HOTOS '01: Proc. 8th Workshop on Hot Topics in Operating Systems*, Washington, DC, USA, 2001. IEEE Computer Society.

[24] I. Soto, C. Bernardos, M. Calderon, A. Banchs, and A. Azcorra. Nemo-enabled localized mobility support for internet access in automotive scenarios. *Communications Magazine, IEEE*, 47(5), May 2009.

[25] M. Zec. Operating system support for integrated network emulation in imunes. In Proc. 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure, 2004.