1-1-2010

XenITH: Xen in the Hand

Kyle Super
University of Pennsylvania

Jonathan M. Smith
University of Pennsylvania, jms@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-23.

XenITH: Xen in the Hand

Abstract

Usability and portability have been key commercial drivers for increasingly capable handheld devices, which have been enabled by advances in Moore's Law and well as in wireless systems. The nature of such devices makes them extremely personal, and yet they offer an untapped resource for new forms of peer-to-peer and cooperative communications relaying. Taking advantage of such capabilities requires concurrent resource control of the handheld's computational and communications capacities. Virtualization platforms, such as the Xen system, have opened the possibility of multiplexing a handheld device in useful and unobtrusive ways, as personal applications can be used while additional services such as decentralized communications are also in operation. The purpose of this project is to experimentally demonstrate the ability of modern smartphone units to support a programmable network environment. We attempt to validate the system with a series of measurement experiments which demonstrate concurrent use of two operating systems, each using computational and network resources, in two virtual machines. Moreover, we demonstrate an acceptable level of user performance while maintaining a MANET using a programmable network router.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-10-23.

XenITH: Xen In The Hand

Kyle Super
University of Pennsylvania
super@seas.upenn.edu

Jonathan Smith
University of Pennsylvania
jms@cis.upenn.edu

Abstract

Usability and portability have been key commercial drivers for increasingly capable handheld devices, which have been enabled by advances in Moore's Law and well as in wireless systems. The nature of such devices makes them extremely personal, and yet they offer an untapped resource for new forms of peer-to-peer and cooperative communications relaying. Taking advantage of such capabilities requires concurrent resource control of the handheld's computational and communications capacities. Virtualization platforms, such as the Xen system, have opened the possibility of multiplexing a handheld device in useful and unobtrusive ways, as personal applications can be used while additional services such as decentralized communications are also in operation. The purpose of this project is to experimentally demonstrate the ability of modern smartphone units to support a programmable network environment. We attempt to validate the system with a series of measurement experiments which demonstrate concurrent use of two operating systems, each using computational and network resources, in two virtual machines. Moreover, we demonstrate an acceptable level of user performance while maintaining a MANET using a programmable network router.

1. Introduction

Mobile computing and wireless telephony are increasing converging, with many handheld devices, such as the Apple iPhone and the RIM Blackberry, able to assume roles as either cellular telephones or as computers. A key research question that arises as a consequence of this convergence is whether the strategies such as virtualization used for isolation and resource sharing in larger cabled computers will be applicable to machines that are limited by size, computational power, and battery life. This question also has great bearing on the extent to which the National Science Foundation's Global Environment for Network Innovation (GENI) initiative will impact the most numerous network endpoints; the International Telecommunications Union estimated that the number of cellular subscriptions worldwide would reach 4.6 billion by the end of 2009 [1].

The impact of expanding virtualized network capabilities to the mobile edge is broad. Being so pervasive, cellular phones exist in many locations where a more traditional networking infrastructure does not. For example, according to the ITU, less than 0.5 percent of African villages have a PC-based communications network, yet more than half of all Africans use mobile telephony [1]. As such, the creation of mobile ad-hoc computer networks, using cell phones as intermediate nodes, could provide key network services in developing nations or remote locations. Moreover, such capabilities would be useful in environments where a network infrastructure does exist, but whose operational abilities have been limited, such as in a disaster recovery scenario. When used as packet relay devices in an emergency, cell phones could provide local-area emergency response broadcast or emergency responder services. The recent disaster in Haiti demonstrates the need for such capabilities. Following the earthquake, the Inveneo company, known for providing network services to NGO workers in remote location, setup a temporary network using satellites and WiFi to be used by the aid workers in their recovery efforts [2]. The National Association for Amateur Radio even sent HAM radio equipment to fill the communications void left by the earthquake [3]. Such networks could potentially be replaced by a MANET consisting of cell phone nodes, at least as a short term solution to the communication problems caused by a natural disaster.

Finally, the ubiquity of wireless communication systems is an ever increasing phenomenon, as state and local governments realize the value of such infrastructures to all citizens. The Digital Impact Group's Wireless Philadelphia project aims to provide network access to underprivileged households and small businesses through a city wide WiFi network. The service is free to anyone

This work was supported by the National Science Foundation under Award Number CNS-0739347

that can access it, and receives approximately 125,000 unique users each month. However, the project still does not cover over half of the city [4]. A low-cost MANET composed of cell phones could extend the reach of these wireless access points significantly and allow Wireless Philadelphia to achieve its ultimate goal of turning Philadelphia into the nation's most connected city.

2. Related Research

Despite outnumbering PCs, little research effort has been put forth to develop virtualization for a cell phone platform in the past. A large hurdle to overcome is the fact that cell phones typically operate on ARM-based architectures, while most virtualization software is designed for the x86 architecture found in almost all PCs and server machines. However, recently several commercial software companies have begun making investments in this area. VMWare, a developer of virtualization software, has announced plans to release a version of their product for cell phones in 2012, the purpose being to allow cell phone users to maintain separate environments for different uses, such as one for personal use and one for work [5]. Moreover, OK Labs, another virtualization specialist, currently offers the OKL4 microvisor, which supports Linux, Android, Symbian, and Windows embedded operating systems. The OKL4 software is currently in use on the Motorola Evoke, the first commercially available virtualized cell phone [6]. Finally, the VirtualLogix company has developed the VLX virtualization system, which can run on a variety of ARM-based platforms [7].

The development of an open-source virtualization solution for an ARM-based platform has been somewhat less successful. The VirtualBox software package, a product of Oracle, only supports the x86 architecture. Moreover, the software requires a host operating system on top of which guest operating systems can be run, making VirtualBox unsuitable for our purposes. Another common open source virtualization package, the Xen virtual machine monitor, developed by the University of Cambridge, allows users to run multiple instances of operating systems, such as Linux, NetBSD, Solaris, and Windows, on top of the Xen hypervisor, the lowest layer of the system that interfaces with the hardware. However, the latest version of Xen only supports the x86 architecture, with ports to the IA64 and PowerPC architectures underway. Two active Xen community projects are attempting to port Xen to the ARM architecture as well, but these efforts are in their infancy. The first, Embedded Xen on ARM Platforms, is intended to be run on embedded and real-time systems. The project currently has an alpha release designed to run on the Colibri PXA270, a SODIMM-sized computer module that features an Intel Xscale PXA270 processor [8]. The second community project, Secure Xen on ARM, is more mature than the first project and is being developed by a team of researchers from Samsung lead by Sang-bum Suh. Secure Xen has an alpha release as well, intended to run on the Freescale i.MX21 development board, as well as a patched version of the Goldfish Android emulator. However, the version for the Android emulator currently only supports a toy operating system, not a full Linux system [9].

3. Experimental Setup

The purpose of the experiment is to demonstrate the effectiveness of virtualizing the wireless network connection of a resource constrained computing environment, similar to that present on modern smartphones. We considered a number of potential experimental setups, rejecting many for various reasons before deciding on a workable version. First we discuss those setups that we rejected.

3.1. Rejected Experimental Setups

The first platform we considered for the experiment was the HP iPaq hw6945 smartphone device, due to prior experience with the device and availability of the units. The device employs the Intel Xscale PXA270 processor, and so could potentially be used with the Embedded Xen alpha software release. More importantly, the device has built-in 802.11b wireless capabilities, making it suitable to be used as a MANET node. However, the iPaq hw6945 has limited support for the Linux operating system due to a proprietary hardware configuration, and uses the Windows Mobile Pocket PC operating system natively. Since the Xen hypervisor does not support running Windows under the privileged first domain, we ruled out the HP iPaq as a viable experimental platform.

We next considered a Google Android based solution. Applications for Android are written in the Java language, but are compiled into a proprietary bytecode format that runs on the Dalvik virtual machine. Each application runs on its own instance of the virtual machine, and so the applications are sufficiently sandboxed from each other, making the use of another underlying virtualization solution such as Xen unnecessary. Moreover, unlike most other cell phone platforms, Android allows developers to write applications that do not have a user interface [10]. Such a capability could be used to develop a MANET

This work was supported by the National Science Foundation under Award Number CNS-0739347

routing service that runs in the background, not interfering with normal user applications. However, in order to make use of an existing software router, the software would first have to be ported to use Java and the Android application programming interface. Such a development would be too great an undertaking to warrant the use of Android and its Dalvik virtual machine.

We also considered using the patched Goldfish Android emulator from the Secure Xen on ARM project. In this setup, we could avoid using the Dalvik virtual machine altogether by running Android in one virtual environment and the Linux kernel and MANET routing software in another, both running on top of the Secure Xen hypervisor. Unfortunately, the patched Goldfish emulator only supports a mini toy operating system on top of the hypervisor at the moment, and efforts to make the hypervisor operable with the Linux kernel failed. Thus, until the patched Goldfish emulator becomes more stable, it remains unsuitable for our experimental needs. In addition, the Android emulator appears to be largely inefficient, and so such an environment may be too underpowered to be of use in an experiment even if it were capable of supporting a full fledged operating system.

Another platform we considered was the Freescale i.MX21 development board on which the Secure Xen on ARM software was developed. The board features a 266 MHz ARM9 processor and 64 megabytes of SDRAM. The board does not have built-in WiFi capabilities, but does have a PCMCIA expansion slot, allowing a wireless card to be added. Unfortunately the board is extremely difficult to find, due to United States import restrictions. After months of trying, we did eventually get our hands on the development board, but found it difficult to use. First, due to the immaturity of the Secure Xen system, the Xen hypervisor, as well as the Linux kernels for the different virtualized domains, must be directly loaded into memory, requiring the u-Boot bootloader. However, the u-Boot bootloader does not support the i.MX21 development board natively. We contacted the research team at Samsung in an attempt to get their custom version of the bootloader, but received no response. Second, while the Secure Xen software package contains everything needed to create Linux kernel binaries designed for the ARM architecture, the package does not contain a ported filesystem. Thus, a full Linux system would need to be cross-compiled for the ARM architecture. Moreover, the MANET routing software we chose would need to be ported as well. Finally, the i.MX21 system is obsolete compared to modern smartphone platforms, which contain far more powerful ARM-based processor cores. The combination of these difficulties led us to conclude that the creation of an experimental setup using the i.MX21 development board would be too time consuming, and so we abandoned work with the system.

3.2. Selected Experimental Setup

In order to simplify our experimental setup, we finally decided to abandon an ARM-based system in favor of an Intel Atom-based netbook. The Atom processor is a low-power implementation of the x86 instruction set architecture, meaning the mature Xen hypervisor will run on Atom-based devices without modification. Specifically, we chose an Asus Eee PC 1101HA netbook to serve as the test environment. The netbook features an Intel Atom Z520 processor operating at 1.33 Gigahertz and 1 Gigabyte of DDR2 RAM. This test system resembles some of the high-end smartphones on the market today, such as the Google Nexus One. The Nexus One features a Qualcomm QSD 8250 ARM processor operating at 1 Gigahertz and 512 Megabytes of RAM. In fact, this Qualcomm 'Snapdragon' platform is to be used in the Lenovo Skylight smartbooks due out sometime this quarter [11]. Moreover, at CES 2010 earlier this year, LG announced their plans for the LG GW990, an Atom-based smartphone device [12]. While the company has yet to release the technical specifications or release date for the device, the announcement indicates that an Atom-based netbook platform is a reasonable proxy for a smartphone device. Finally, the specific Atom processor in the 1101HA netbook supports Intel Virtualization Technology, meaning that our virtualization software will be able to take advantage of the efficiency afforded by hardware virtualization extensions.

The Xen virtualization software served as the underlying software platform for this experiment, on top of which ran multiple instances of the Ubuntu Linux software distribution. One of these instances served as the user operating system (i.e.: provided the interface with which the user directly interacted). On an actual smartphone platform, this user operating system would be the native operating system, such as Google Android. The other Xen environments contained a stripped down version of the Ubuntu distribution. These instances provided the network infrastructure for a mobile ad-hoc network by sharing the Atheros 802.11b wireless network card on the netbook with the user operating system. Such an infrastructure could be used in disaster recovery scenarios or other settings in which a network either does not exist or has been disrupted.

This work was supported by the National Science Foundation under Award Number CNS-0739347

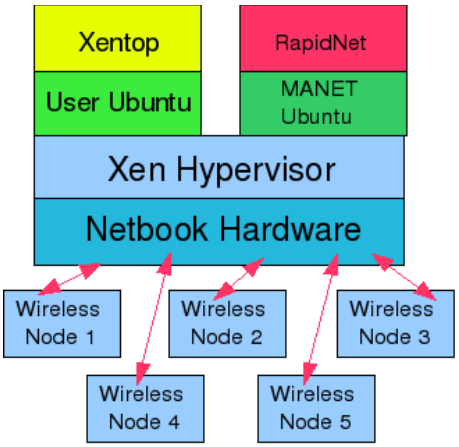


Figure 1: Experimental Setup 1

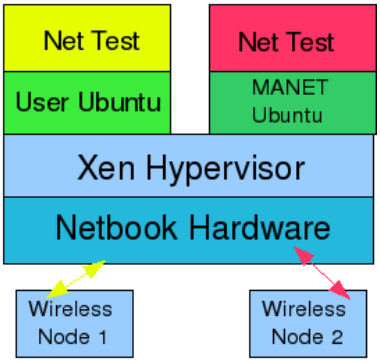


Figure 2: Experimental Setup 2

On top of the stripped down Ubuntu distributions and providing the actual network routing was the RapidNet declarative language based network system developed at the University of Pennsylvania. The system makes use of routing protocols written in Network Datalog, a distributed recursive query language for querying networks. Prior research has shown the effectiveness of RapidNet and Network Datalog in implementing a wide variety of proactive, reactive, and epidemic MANET routing protocols, as well as hybrid versions of these protocols. However, the purpose of this experiment was not to validate the effectiveness of RapidNet, but rather to demonstrate that such a system can operate in a virtualized, resource-constrained environment, without greatly impacting the user experience.

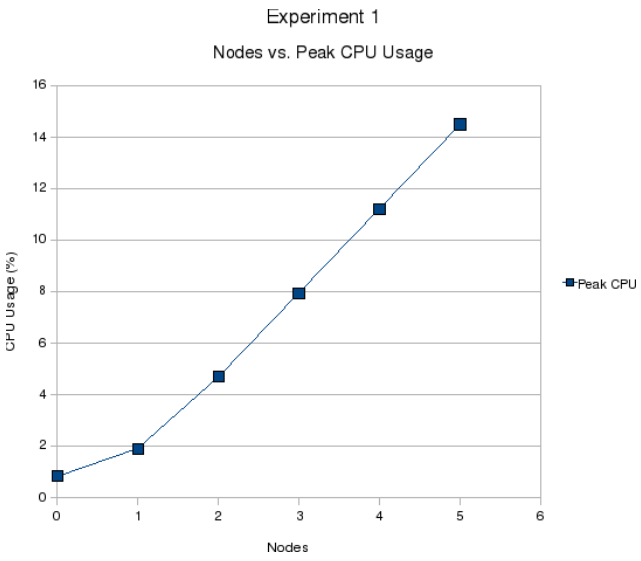
In order to measure the impact of RapidNet and the second virtual machine, we conducted a series of three experiments; the first two measured the CPU usage of the RapidNet system under varying operating environments, and the third measured the ability of the two concurrent virtual machines to share the available network bandwidth. The first two experiments used the same basic setup (see Figure 1), consisting of five wireless nodes, which were commodity laptops available to our lab, as well as the Asus netbook. In the first experiment, we measured the peak CPU usage of the second Xen virtual machine while the RapidNet system composed MANETs using a simple path vector protocol. We varied the number of wireless nodes included in the system between zero and five. In the second experiment, we measured both peak and average CPU usage of the second virtual machine while randomly adding and removing wireless links from the MANET at varying, regular intervals. Note that in both experiments, we measured the CPU usage of the entire second virtual machine, not just the RapidNet applications. This measure provides a better idea of the total overhead from such a concurrent system. In both cases, we used the XenStat Perl application in conjunction with the Xentop utility to poll CPU usage.

In the third experiment, we used a second experimental setup (see Figure 2), consisting of two wireless nodes and the Asus netbook. Here, we measured the total bandwidth available to the user virtual machine while varying the bandwidth utilization of the second virtual machine. To do so, we composed a simple network application capable of sending dummy traffic to another wireless node at a specific rate. The application running on the user virtual machine communicated with one wireless node, while that running on the second virtual machine communicated with a second wireless node. Note that Xen maintains the ability to limit outgoing bandwidth used by different virtual machines. While we did not make use of this capability in our experiments, it would most likely be useful in a real-world setup.

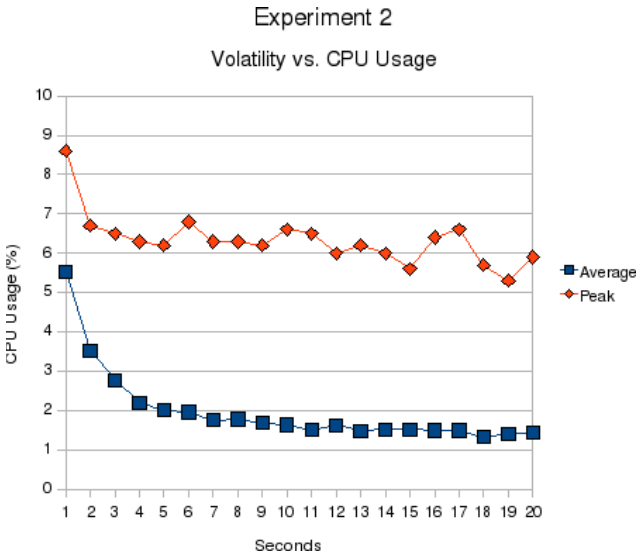
4. Experimental Results

In the first chart, we present the results from experiment one. As can be seen, the peak CPU usage required by the second virtual machine and the RapidNet system scales almost perfectly linearly as we add more nodes to the ad-hoc network. With as many as five nodes entering the MANET concurrently, the total CPU usage peaks at only 14.48%. Note that in compiling these results, each node had links available to every other wireless node, but we modified the link weights to ensure that every route would always proceed through the netbook as an intermediate hop.

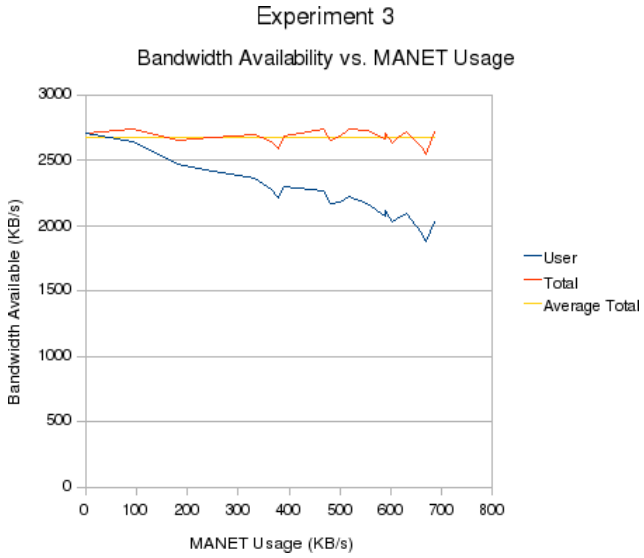
This work was supported by the National Science Foundation under Award Number CNS-0739347



In the second chart, we see the results from experiment two, where we randomly add and remove links from the ad-hoc network at regular intervals (between one and twenty seconds for each addition/removal). We see that for additions and removals that occur at rates of less than once every five seconds, the average CPU usage of the second virtual machine and the RapidNet system remains less than two percent. Moreover, the peak CPU usage remains below seven percent for rates of less than once every one second. Thus, both the average and peak CPU usages of the second virtual machine remain quite reasonable for all but the most volatile mobile ad-hoc networks.



Finally, in the third chart, we see the results from experiment three. Here we send dummy data from both the user virtual machine and the second virtual machine, varying the maximum rate that the second virtual machine can send data. As expected, the total bandwidth available for the user virtual machine drops in concert with a rise in utilization from the second. Moreover, we also see that the total bandwidth utilized by both virtual machines remains relatively constant around 2675 KB/s. This stability indicates that we incur very little overhead by sharing the network connection between two virtual machines. Any variability we did see seemed fairly random and was most likely due to external environmental factors.



5. Conclusion

Considering the results of our experiment, we hope to have demonstrated two properties of virtualized network setups in resource constrained environments. First, at a reasonable level of volatility, the RapidNet MANET system will quickly establish links between nodes without requiring an overbearing amount of CPU consumption. Second, at reasonable levels of bandwidth usage, the RapidNet MANET system will not greatly interfere with the user's personal operating experience. Thus, we see that such a system would be useful in a real-world setting. Specifically, we see that current smartphone technology is capable of dually supporting a normal user operating environment, as well as an environment devoted to the maintenance of a mobile ad-hoc network infrastructure. As discussed earlier, such an infrastructure could provide the basis for network capabilities in disaster recovery scenarios, or in settings in which a more permanent network infrastructure does not exist. Moreover, the capability of smartphone technology to virtualize the built-in WiFi hardware demonstrates that the mobile edge

This work was supported by the National Science Foundation under Award Number CNS-0739347

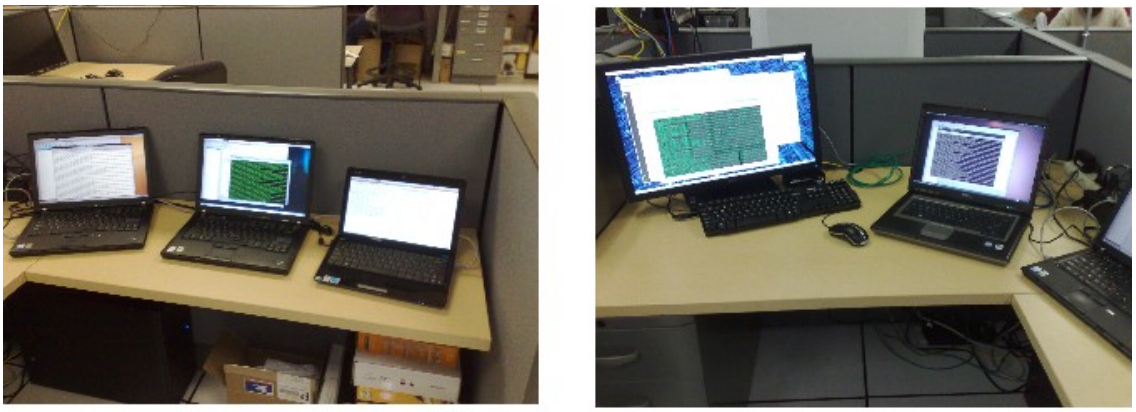


Figure 3: Images of the Experiments

can and should be considered in the context of larger initiatives, such as Wireless Philadelphia and GENI.

Further experimentation related to this project could involve utilizing a larger network testbed, such as the Orbit system, to see how well everything scales as the number of wireless nodes increases. Such scaling would be necessary in larger, real world applications, where nodes could be very densely distributed. In addition, some specific measure of the user operating experience, such as using the e-Model for voice quality when analyzing a user's VoIP connection, could be used to better quantify our results and provide a more concrete analysis of what operating parameters are reasonable from a user's perspective.

Further research not directly related to this project could involve making use of the commercially available cell phone virtualization platforms discussed earlier. While our project demonstrates the effectiveness of virtualizing a resource constrained system similar to a smartphone device, we did not develop anything to be used in a real-world application. As our difficulty in finding a suitable experimental platform shows, currently only commercial solutions seem to be viable for such development. Hopefully the efforts to port Xen to the ARM architecture will eventually provide the tools needed to virtualize actual cell phone devices, but for now any real world development and experimentation will require commercial support.

6. Acknowledgments

Special thanks to University of Pennsylvania Professor Boon Thau Loo, as well as to his graduate students Shivkumar Muthukumar and Xiaozhou Li, for all their help in getting the RapidNet experimental setup up and running.

References

[1] Richard Heeks, "ICT4D 2.0: The Next Phase of Applying ICT for International Development," Computer, pp. 26-33, June 2008

[2] Nancy Gohring, “NGO networks in Haiti cause problems for local ISPs,” PC World Business Center, PCWorld.com, February 2010

[3] “ARRL Sends Ham Aid Equipment to Haiti,” The National Association for Amateur Radio, ARRL.org, January 2010

[4] “About Digital Impact Group,” Digital Impact Group, WirelessPhiladelphia.org, January 2010

[5] Tim Lohman, “VMware developing dual OS smartphone virtualisation,” Computerworld, Computerworld.com.au, December 2009

[6] Chris Walker, “OK Labs Enables World’s First Virtualized Smartphone, with Mobile Virtualization Solution ,” Open Kernel Labs, OK-Labs.com, March 2009

[7] Olga Kharif, “Virtualization Goes Mobile,” BusinessWeek, BusinessWeek.com, April 2008

[8] Daniel Rossier, “Embedded Xen on Arm Platforms,” http://sourceforge.net/projects/embeddedxen/

[9] Sang-bum Suh, “Xen ARM Project,” http://wiki.xensource.com/xenwiki/XenARM

[10] “A Spectrum White Paper: Thoughts on Google Android,” Spectrum Data Technologies, SpectrumDT.com, February 2008

[11] Brian Chen, “Hands-On with the Lenovo Skylight Smartbook,” Wired Gadget Lab, Wired.com, January 2010

[12] Vladislov Savov, “LG GW990 Hands-on video,” Engadget, Engadget.com, January 2010

[13] Changbin Liu, “Declarative Policy-based Adaptive MANET Routing,” University of Pennsylvania, October 2009, http://netdb.cis.upenn.edu/rapidnet/

This work was supported by the National Science Foundation under Award Number CNS-0739347