Pocket Hypervisors: Opportunities and Challenges

Landon P. Cox
Duke University
lpcox@cs.duke.edu

Peter M. Chen
University of Michigan
pmchen@umich.edu

Abstract

In this position paper, we explore the opportunities and challenges of running pocket hypervisors on commodity mobile devices through four proposed applications: secure operating systems, security services, mobile testbeds, and opportunistic sensor networks. We believe that pocket hypervisors can benefit mobile computing, but that mobility presents several important and unique challenges to virtualization.

1 Introduction

In recent years, hypervisors for commodity operating systems [16] have helped address problems in such diverse systems domains as secure-logging [8], honeypots [15], kernel-debugging [19], and cluster-computing [4]. Hypervisors' popularity across domains is a result of their ability to *encapsulate* the state of a running system, *mediate* all interactions between the hardware and software, and *isolate* concurrently running software components [13].

Thus far, only hypervisor encapsulation has been practical for mobile computing. For example, Internet suspend/resume [14] and SoulPad [18] use mobile devices to help transport suspended virtual machines between desktop (or laptop) endpoints, which actually execute the virtual machines. However, running hypervisors on the devices themselves has traditionally been infeasible or unusable: embedded processors have lacked support for dual-mode operation and an MMU or been too slow to accommodate the additional overhead of virtualization.

Fortunately, these constraints are disappearing; processors for many commodity mobile phones and PDAs increasingly possess the hardware support and capacity to virtualize a complete hardware interface for guest operating systems. The TRANGO hypervisor [26] and efforts to port Xen [21] to ARM-based processors demonstrate this trend. With hypervisors for commodity mobile devices becoming feasible and practical, can they be used to solve mobile computing problems? Our position is that mobile computing can benefit from *pocket hypervisors*, but that several challenges stand in the way.

We will explore some of these benefits through several proposed applications:

Secure operating systems Secure operating systems for mobile devices protect the device's core state from malicious code. For mobile phones, this core includes telecommunications functionality as well as private keys and personal information stored on the device.

Security services A hypervisor can be used to perform security services on behalf of the main operating system, such as virus scanning, intrusion prevention or detection [20, 2], encryption of disk data [25], and computer forensics [8].

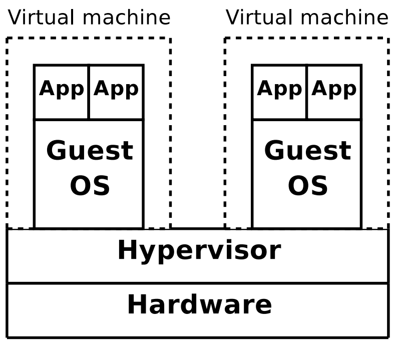

Figure 1. Basic structure of a hypervisor system.

Mobile testbeds Mobile testbeds provide an experimentation and development platform similar to PlanetLab [11]. Pocket hypervisors would allow researchers to safely instantiate virtual machines on the devices of live volunteers, creating opportunities for large-scale experiments under actual physical and network conditions rather than relying on simulations under mobility models [27] and static traces [22].

Opportunistic sensor networks Opportunistic sensor networks allow users to instantiate a temporary environmental monitoring infrastructure on co-located devices. These networks could be used for a variety of social and engineering purposes including notifying the source device when a friend is nearby or establishing an ad-hoc network path to an otherwise out-of-range wireless hotspot.

The common responsibility of hypervisors in each application is to protect host devices' core state (both software and hardware) from the effects of running external code. Developing a hypervisor to meet these goals will require addressing several challenges unique to mobile devices.

First, unlike desktop computers, devices have limited power resources. Thus, pocket hypervisors must accurately account for and allocate battery power among their guest virtual machines. Furthermore, mobile devices possess a physical context (e.g. a location, set of nearby devices, or list of available wireless networks), which is both privacy-sensitive and critical to many distributed mobile systems. Pocket hypervisors' interface must balance the need to protect host device owners' location privacy and guest applications' need to interact with a host's physical context.

The rest of this paper is organized as follows. Section 2 discusses related work in hypervisors and virtual machines. Section 3 describes how pocket hypervisors enable secure operating systems, security services, mobile testbeds, and opportunistic sensor networks and discusses some of the challenges that arise when implementing these applications. Section 4 provides our conclusions.

2 Background and Related Work

Hypervisors, or virtual machine monitors, were first developed by IBM in the 1960s. Figure 1 shows the basic structure of a hypervisor system, in which a hypervisor runs on the bare hardware and guest operating systems run above the hypervisor. There has been a recent revival of interest in this structure due to projects such as Disco [7], which begot VMware, and Xen [16]. Hypervisors are useful because they can encapsulate a running system, including kernel and hardware state, within the abstraction of a virtual machine, they mediate all interactions between hardware and guest virtual machines, and they provide correctness and performance isolation among virtual machines.

Scylla [17], Maté [23], and Java also offer virtual machine environments for embedded devices, but provide a more restrictive computational and programming environment than hypervisors such as TRANGO [26]. The Scylla, Maté, and Java virtual machines are bytecode interpreters that are incompatible with other programming languages and native code. On the other hand, hypervisors export a complete hardware interface with support for native legacy code, including operating systems.

Exporting a complete mobile device interface to guest code is crucial for our applications. First, securing mobile devices requires running code below the kernel to contain vulnerabilities like the operating system's Bluetooth stack and to protect services like intrusion-detection. Second, mobile testbeds and opportunistic sensor networks require precise power accounting of guest applications. Otherwise, the opportunity costs of hosting guest code will be too great for device volunteers. Xen [16] and other hypervisors have demonstrated that a low-level approach to resource accounting is simpler than relying on bytecode interpreters running above a shared kernel. Finally, unlike bytecode interpreters, hypervisors do not restrict application developers to any particular programming language or operating system. Thus, rather than force applications to be written in Java or to run on top of a power-accounting operating system like ECOSystem [9], devices can remain compatible with applications written in C or developed for Windows Mobile.

3 Applications

In this Section, we describe several applications for pocket hypervisors: secure operating systems, security services, mobile testbeds, and opportunistic sensor networks.

3.1 Secure Operating Systems

As mobile phones' hardware has tended toward PCs', so too has the software they run. Users routinely take photographs, play games, listen to music, watch videos, instant message, and word process with their mobiles; mobile phones have evolved into a form-factor- and power-constrained platform for personal computing. Unfortunately, this evolution has also brought with it many of the problems associated with personal computing, including security threats [1].

For example, vulnerabilities in several implementations of the Bluetooth stack have been discovered in the past year [24]. If a mobile phone's Bluetooth stack contained such a vulnerability, it might be exploited by an attacker to gain control of other software on the phone, including its operating system. Once in control of the operating system, the attacker may be able to mount a denial-of-service (DOS) attack on the phone's telecommunication functions or extract the private keys located in the phone's SIM card [10]. Even without compromising the operating system, an attacker could mount a DOS attack by running resource-intensive processes to drain the device's battery or gain access to sensitive personal information like the user's photos, address book, and SMS messages.

Hypervisors can help thwart such attacks by allowing devices' functionality to be partitioned among several guest virtual machines. For example, a mobile phone could run a core virtual machine containing its authentication and voice communication software and a separate, personal computing virtual machine with the device's end-user applications.

These virtual machines could be given access to different hardware resources: the core virtual machine might have exclusive access to the phone's SIM card and GSM radio, while the personal computing virtual machine might have exclusive access to the Bluetooth radio. By partitioning access to hardware in such a way, a compromised Bluetooth stack in the personal computing virtual machine could not be used to access the phone's SIM card or disable its GSM radio.

Other hardware resources such as the battery, processor, physical memory, display, speaker, and microphone would have to be multiplexed among each virtual machine. Ensuring "fair use" of these resources relies on hypervisor performance isolation.

3.2 Security Services

Hypervisors make an excellent platform for adding security services because they have a narrower interface and smaller code size than commodity operating systems. Several such security services for desktop and laptop computers have been built, and we believe the operating systems on mobile devices would also benefit from these types of services.

One type of hypervisor security service inspects the state of the operating system running above it and detects malicious software such as viruses [2, 20]. Because such services run in the hypervisor, they can inspect all application and operating system state. They can also gain control on key events, such as system calls or specific application events. The main difficulty with these types of services is the semantic gap between the abstractions available to the hypervisor (the virtual hardware) and the abstractions used in the software running above. The semantic gap can be bridged by re-implementing parts of the higher-level software or by leveraging the functionality already present in the higher-level software.

Another hypervisor security service is instruction-level replay [8], which is useful for computer forensics and debugging. This service allows step-by-step replay of long periods (months) or execution, yet costs little overhead in time or space.

A third type of hypervisor security service encrypts the data generated by applications or the operating system before it is saved on disk or sent over the network. As with other hypervisor security services, this service does not depend on the application or operating system to be written correctly; it can provide a VPN or an encrypted disk without support from higher levels of software.

3.3 Mobile Testbeds

The insurance provided by hypervisor-enforced isolation enables several new services that utilize mobile devices safely executing external code. One such service is a federation of mobile testbeds composed of hundreds of volunteers willing to let researchers execute virtual machine images on their phones. To incentivize university students and faculty to participate, testbed virtual machines could run in the background as the BOINC screensaver does. Virtual machine images could be uploaded either via SMS or strategically placed wireless hand-off points around campus. Cryptographic naming techniques such as those used by Internet suspend/resume [14] can reduce the overhead of uploading virtual machines by taking advantage of data already present on devices.

Similar to PlanetLab [11], these networks would provide a platform for large-scale mobile experiments and distributed system development that currently does not exist. In addition to scale, the main advantage of mobile testbeds is that researchers could run experiments with live users, rather than relying on simulations involving mobility models [27] and static traces [22]. This approach is complementary to other emerging testbeds such as ORBIT [5] and could provide valuable insight into how experimental systems behave under actual network conditions and mobility patterns.

The greatest barrier to these testbeds is the opportunity cost to volunteers for participating. Specifically, a testbed in which experiments consume too much of a volunteer device's battery power will not be sustainable. However, a recent study of laptop battery consumption found that many laptops carry significant excess battery power between charges [6]. While we are unaware of any similar studies of mobile device battery use, the laptop result suggests that device owners may have excess power to donate to a mobile testbed.

However, if a device's spare capacity is insufficient, mobile hypervisors can help ensure that normal usage is unaffected by enforcing restrictions on the portion of a device's power consumed by experiments. The accounting and enforcement mechanisms used by the ECOSystem [9] operating system provide a promising starting point. Which policies hypervisors should enforce is an open question. They might consider such variables as the affect of deep-cycling on a battery's lifespan, the rate at which a virtual machine is consuming power, and predictions of the next time the device will be charged.

3.4 Opportunistic Sensor Networks

Our final proposed service for mobile devices is opportunistic sensor networks. Within these networks, mobile users could instantiate virtual machine images on any accepting co-located mobile devices. This network could safely self-propagate, searching for resources specified by the user such as an Internet hotspot, the presence of the user's friends, or the user's car in a crowded parking lot.

These networks are a generalization of mobile testbeds; instead of allowing a small group of trusted researchers to instantiate virtual machines, users would accept virtual machines from arbitrary users. Because of this, opportunistic sensor networks combine many of the issues of secure operating systems and mobile testbeds. However, relaxing assumptions about the trustworthiness of guest virtual machines also has implications for users' privacy.

For example, Place Lab [3] uses WiFi access point identifiers to infer a device's location. Stalkers and marketers must not be allowed to track users by simply uploading such an application to their mobile phone. This is a radically different threat model than previous location-privacy work, which has focused on devices being tracked from without [12]. Here, the danger is that a device could be tracked from within.

Thus, the hypervisor must carefully restrict information given to guest virtual machines about a host device's physical context. This is in conflict with the requirements of many distributed mobile applications, which need information about their environment to be useful. How this tension should be resolved is an open question, but several possible techniques exist.

First, virtual machine images might be cryptographically signed by their owners. This would create some disincentives to misbehave by eliminating anonymous snooping and allow users to reject requests from unknown sources. Additionally, signed guests may allow the hypervisor to only forward WiFi and Bluetooth traffic among co-located devices hosting guests with matching signatures.

To enable discovery, the identities of other co-located devices could be "scrubbed" using cryptographic hashes. For example, rather than returning a full list of BD_ADDRs and human-readable names generated by a scan for Bluetooth devices, the hypervisor might return an untrusted guest virtual machine the salted SHA1 hashes of those values. If the guest attempted to communicate with other devices, the hypervisor could translate those BD_ADDRs back to actual addresses, similar to NAT for IP addresses.

4 Conclusions

Pocket hypervisors enable a range of useful applications for commodity mobile devices, including stronger device security, mobile testbeds, and opportunistic sensor networks. However, only by addressing the challenges unique to running hypervisors on mobile devices, such as multiplexing battery power and balancing application needs and users' privacy, will the benefits of these application be realized.

References

[1] First mobile phone virus created. BBC, June 16, 2004.

[2] A. J. et al. Detecting past and present intrusions through vulnerability-specific predicates. In *SOSP*, October 2005.

[3] A. L. et al. Place Lab: Device positioning using radio beacons in the wild. In *Pervasive*, May 2005.

[4] D. I. et al. Sharing networked resources with brokered leases. In *USENIX*, June 2006.

[5] D. R. et al. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *WCNC*, March 2005.

[6] D. T. et al. LLAMA: An adaptive strategy for utilizing excess energy to perform background tasks on mobile devices. In *University of Massachusetts Technical Report TR-06-44*, September 2006.

[7] E. B. et al. Disco: Running commodity operating systems on scalable multiprocessors. *TOCS*, 15(4), 1997.

[8] G. W. D. et al. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In *OSDI*, December 2002.

[9] H. Z. et al. ECOSystem: Managing power as a first class operating system resource. In *ASPLOS*, October 2002.

[10] J. D. et al. SIM trust parameters. In *Intel Developer Update Magazine*, January 2003.

[11] L. P. et al. Experiences building PlanetLab. In *OSDI*, November 2006.

[12] L. P. C. et al. Presence exchanges: Towards sustainable presence-sharing. In *HotMobile*, April 2006.

[13] M. R. et al. Virtual machine monitors: Current technology and future trends. In *IEEE Computer Magazine*, May 2005.

[14] M. S. et al. Towards seamless mobility on pervasive hardware. *Pervasive and Mobile Computing*, 1(2), June 2005.

[15] M. V. et al. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In *SOSP*, October 2005.

[16] P. B. et al. Xen and the art of virtualization. In *SOSP*, October 2003.

[17] P. S.-M. et al. Scylla: A smart virtual machine for mobile embedded systems. In *WMCSA*, December 2000.

[18] R. C. et al. Reincarnating PCs with portable soulpads. In *MobiSys*, June 2005.

[19] S. T. K. et al. Debugging operating systems with time-traveling virtual machines. In *USENIX*, April 2005.

[20] T. G. et al. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *NDSS*, February 2003.

[21] D. R. Ferstay. Fast secure virtualization for the ARM platform, March 2006. University of British Columbia, Master's Thesis.

[22] T. Henderson and et al. The changing uses of a mature campus-wide wireless network. In *MobiCom*, September 2004.

[23] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *ASPLOS*, October 2002.

[24] C. Mertoni and et al. The BlueBag: a mobile, covert bluetooth attack and infection device. In *Blackhat*, August 2006.

[25] R. Meushaw and D. Simard. NetTop: Commercial Technology in High Assurance Applications. *Tech Trend Notes*, 9(4), September 2000.

[26] TRANGO systems.
http://www.trango-systems.com.

[27] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM*, April 2003.