Mobile Device Offloading Using Enterprise Network and Cloud Resources

Aaron Gember, Aditya Akella
University of Wisconsin Madison
{agember,akella}@cs.wisc.edu

1. INTRODUCTION

Smartphones have become an important element of enterprise computing infrastructure. In a recent survey, 34% of respondents reported using their smartphone more than their computer to conduct business [1]. Paralleling this usage increase is an ever broadening range of mobile applications: speech-to-text, image recognition, etc. Mobile devices need more processing, memory, and battery resources to support these applications, but increases in hardware capabilities are not keeping pace with application demands. In addition, an asymmetry remains between the computing resources of desktop PCs and smartphones.

We propose an offloading framework that takes advantage of local resources in enterprise networks to simultaneously address the resource, performance, and security demands of mobile applications. Idle desktops serve the needs of mobile device by executing arbitrary offloaded applications. A remote cloud can supplement local resources for offloading security- and delay-insensitive applications. Which applications to offload and what secondary computing resources to use is determined by a central enterprise-wide controller configured with appropriate policies. The controller considers resource demands, resource availability, and the network topology to seamlessly leverage all possible computational capacity and elastically adjust to changing conditions.

2. DEMONSTRATION

Our demonstration focuses on the application migration component of our offloading system. Mobile applications are assumed to execute in a virtual runtime environment, creating a relatively clean separation between operating system and application execution state. Our prototype offloads Java applications running in a modified Dalvik Virtual Machine (VM) on a Google Android smartphone. The Dalvik VM is modified to capture a running application's execution state (thread stacks and objects), transfer the state and application code across the network, and resume the application in a new Dalvik VM on an idle desktop (Figure 1).

We offload an application from an Android phone to a laptop (representing an idle desktop) connected via a mini OpenFlow network. After the application is launched on the phone, we manually specify an offloading destination—automatically selecting a secondary resource is future work. The destination is communicated to a NOX controller which installs the appropriate flow entries in the OpenFlow switch. The Dalvik VM on the Android phone is halted and the execution state is sent across the network to the idle desktop (laptop). An agent on the idle desktop receives the state and launches a new Dalvik VM. The Dalvik VM loads the execution state and starts execution from where it was halted. After the application is executing on an idle desktop (laptop), we imagine resource conditions have changed; we migrate the application to a different laptop.

Figure 1: Application migration to idle desktop

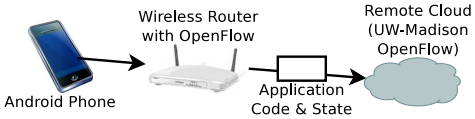

Figure 2: Application migration to remote cloud

The second half of the demonstration offloads an application to a remote cloud. It follows the same procedure as above, but application state traverses the Internet to arrive on a desktop connected to the University of Wisconsin–Madison OpenFlow network (Figure 2).

3. FUTURE WORK

Work on our system prototype is ongoing. In the future, the decision of what and where to offload will be made by a central decision module which considers resource usage information from mobile devices and desktops and selects an offloading destination based on an administrator specified policy. The decision module will communicate with the NOX controller to install the appropriate flows or gather network topology details.

[1] RingCentral. Smartphones changing the way business professionals work and live. http://blog.ringcentral.com/2010/04/smartphones-changing-the-way-business-professionals-work-and-live.html, 2010.