The Swiss Army Smartphone: Cloud-Based Delivery of USB Services

Adiseshu Hari, Manoj Jaitly, Yuh-Jye Chang
Alcatel-Lucent Bell Laboratories
Murray Hill, NJ 07974, USA
{Adiseshu.Hari, Manoj.Jaitly, Yuh-Jye.Chang}@alcatel-lucent.com

Andrea Francini
Alcatel-Lucent Bell Laboratories
Mooresville, NC 28115, USA
Andrea.Francini@alcatel-lucent.com

ABSTRACT

A smartphone can be configured to look like any Universal Serial Bus (USB) peripheral and can be managed remotely through its wireless data connection. By virtue of these features, smartphones are ideal vehicles for the delivery of a variety of brand-new, USB-powered services that support the management and troubleshooting of mobile laptops. We provide examples of such USB services and describe a general architecture for their implementation. The services are easy to deploy, because they can be extended to remote laptops without prior installation of new software, and well-suited for delivery through virtualization in a cloud infrastructure. While our examples target mostly the enterprise, USB services, especially virtualized ones, can easily be tailored to suit a broad set of consumer applications.

Categories and Subject Descriptors

B.4.3 [**Interconnections (Subsystems)**]: Interfaces. K.6.2 [**Installation Management**]: Computing equipment management.

General Terms

Design, Management, Performance.

1. INTRODUCTION

Computers are standard equipment for large portions of today's workforce. While greatly enhancing productivity, they come with a heavy maintenance burden. Software and hardware components can suffer myriad failures, from program malfunctions to virus infections, from hard-disk corruption to boot-up issues. For traveling workers, network connectivity may be sporadic and laptop thefts are not unusual. In all cases where the computer has issues, it is vital for the user to have it checked and serviced without delay. This is a problem for remote workers, who may not have immediate access to a corporate service center.

Fortunately, most computer users have access to a mobile phone, which ever more often is a data-enabled smartphone with Internet connectivity. Can this increasingly ubiquitous device enable new maintenance and troubleshooting schemes where computers no longer have to be taken to service centers?

This paper describes a solution that morphs the smartphone into a variety of remotely-controlled Universal Serial Bus (USB) peripherals, creating the mobile handheld equivalent of a Swiss Army knife. The peripherals deliver services that enclose an off-site computer within a secure IT infrastructure with immediate access to advanced troubleshooting, management, and remediation applications.

The delivery of USB services via smartphones results from our integration of four key enablers. First, all personal computers today support USB peripherals. USB devices are available in a variety of device classes that support a broad range of functions, such as networking, storage, video, smart card, keyboard, and mouse. Since current versions of operating systems like Linux and Windows support drivers per USB device class, a new USB device that falls within one of the standard classes does not even require a dedicated driver. For drivers that are not already built into the host operating system, the wide availability and support of USB driver frameworks in current operating systems make their custom development a trivial task.

Second, all smartphones are now capable of acting as USB peripherals. While the type of USB connector may vary across smartphones, each type comes with a USB cable that universally connects it to a computer.

Third, virtually all smartphones can act as *any* USB peripheral. In the smartphone, the central processing unit (CPU), the memory, the flash storage controller, and the USB device controller are all integrated in one system-on-a-chip package. The CPU can emulate any type of USB peripheral with software that can be controlled both locally (directly on the phone) and remotely (by an external entity deep in the network).

Fourth, smartphones are continuously attached to a wireless wide area network (WWAN) when powered on, and always ready to establish Internet connectivity through the same WWAN or a surrounding wireless local area network (WLAN) upon request. They are therefore amenable to remote access from a networked management station, provided that the management station can establish Internet Protocol (IP) connectivity to the smartphone. Contacting a smartphone over the

MobiHeld '11, October 23, 2011, Cascais, Portugal.

Internet is not possible when the smartphone is not connected to the Internet or its IP address is not known to the management station. However, the management station can always reach a powered-on smartphone that is without Internet connectivity using a text message delivered through the short message service (SMS) component of the WWAN infrastructure. A special SMS message can request the smartphone to initiate the IP connection to the management station [6].

Taking advantage of all four properties, we can attach a smartphone to a computer and make it behave as a USB peripheral, and then, whenever a smartphone interface can support IP connectivity, we can control the peripheral remotely. This provides the foundation for a range of brand-new, zero-touch, always-on services. Each service is tied to one of the USB peripherals found in this "Swiss-Army smartphone": the console peripheral supports remote management and troubleshooting of the computer; the storage peripheral supports network storage and backup, managed boot-up, and enhanced data-delivery functions; the smart-card peripheral supports login revocation and other managed security functions. The range of possible applications for smartphone-enabled USB services is broad in the enterprise IT context and can become much broader in the consumer IT space.

Active sessions of USB services bring tremendous value to the enterprise, but are typically sporadic and short-lived. The installation and operation costs of infrastructure equipment that in rare occasions may be required to sustain relatively large volumes of users and data traffic, but most of the time stays idle, could easily dissuade enterprises from deploying such services. The virtualization of the management station in a cloud environment, offered on a pay-per-use basis by cloud service providers, eliminates all the economic drawbacks of private deployments and expands the set of players that can profit from USB services.

The rest of the paper is organized as follows. In Section 2 we review the USB technology. In Section 3 we outline our reference connectivity framework for the computer, the smartphone, and the virtualized management station. In Section 4 we describe a representative set of USB services. In Section 5 we discuss the benefits of our solution over possible alternatives. Finally, in Section 6 we summarize our contributions.

2. USB OVERVIEW

USB [7] is by far the most popular technology for connecting peripherals to computers. All computers and phones of recent release support it.

The USB communication infrastructure is organized as a tree with the host controller as the root and the USB devices as the intermediate nodes and leaves. Data transfers occur over logical pipes that connect the host controller with respective endpoints in the USB devices. The host controller initiates all transfers, by polling the devices for data. Within a USB device, the endpoints are grouped into interfaces. Each interface delivers a single function, such as networking or storage. Every USB device carries identifiers for the interface types that it supports. Many interfaces are now standardized as USB device classes: a USB device that carries the identifier of a standardized class does not need a special device driver to be installed in the host software in order to function correctly, as long as the host operating system (OS) supports the generic driver for that device class.

The following standardized USB device classes are most common: USB Mass Storage Class, for flash drives, digital cameras, audio players, external drives, and memory card readers; USB Communications Device Class (CDC), for communications peripherals like modems, faxes, serial interfaces, and network interfaces; USB Human Interface Device (HID) Class, for peripherals like keyboard and mouse; USB Audio Class, for speakers and microphones; USB Video Class, for webcams; USB Smart Card Class; and USB Printer Class. We focus on USB services that leverage the USB Mass Storage Class and the USB CDC.

A single USB device can support multiple interfaces, which appear to the host controller as separate logical devices. A USB device that simultaneously exposes multiple interfaces is called a *composite device*. While support for composite devices has started appearing in computer operating systems like Linux and Microsoft Windows, it is poorly provided in smartphones and completely absent in pre-boot environments like the basic input/output system (BIOS) of computers. Consistently with the lack of pervasive support for composite USB devices in smartphones of current generation, this paper focuses on the use of smartphones as single-interface USB devices that deliver a single USB functionality at a time.

3. USB SERVICES ARCHITECTURE

In this section we describe our reference architecture for the implementation of USB services in the enterprise IT environment. The architecture can be easily extended to support consumer IT applications.

3.1 Reference Architecture

The platform that delivers USB services, shown in Figure 2, consists of three components: the computer, the smartphone, and the *service management center* (SMC). The SMC is virtualized in a cloud service provider network, where multiple virtual machine (VM) instances can be allocated on demand, and constitutes the single point of entry for management of all the smartphones that support USB services in the enterprise. The web interface of the SMC includes

commands for provisioning and tracking service-enabled smartphones. Each smartphone connects to the SMC via a secure IP medium, most commonly an IPsec [5], SSH [8], or TLS/SSL [2] tunnel. Encrypted tunnels also protect the connections between the SMC and the administrator consoles within the enterprise network.

The service software that runs on the smartphone includes a Management Agent (MA) and a User Agent (UA). The MA receives from the SMC the commands that select and configure the specific USB device class stack that supports each service (USB device classes can be switched at run-time, with no need to restart the smartphone). The MA also oversees the encryption parameters that protect all data exchanges with the SMC and the computer that pertain to USB services with enhanced security features. The UA exposes a graphical user interface (GUI) that provides the smartphone user with direct access to some of the smartphone controls, such as the USB interface selector and the user authentication interface.

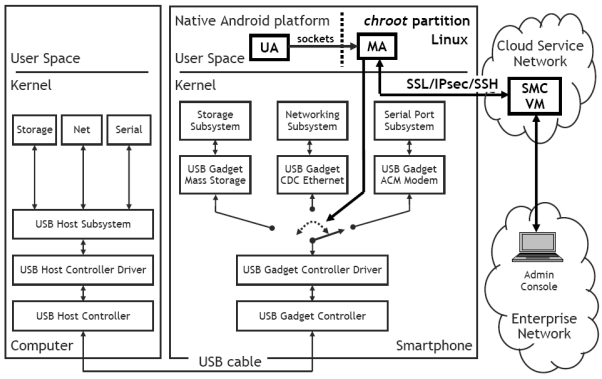

Figure 2. Architecture for USB services.

All USB-service transactions between the smartphone and the computer must occur over a USB cable. This condition does not limit the usability of the services: USB cables are commonly available in small retractable formats and do not restrict the operation of the computer in stationary work settings. Of course, the USB cable can always be unplugged as soon as a service-delivery session reaches completion.

3.2 Implementation of the Architecture

In this section we describe our implementation of the reference architecture for USB services over the Android platform [1] for smartphones.

3.2.1 Service Management Center

The SMC is the point of entry for administration of the USB services. It includes a web-based GUI for provisioning, management, and troubleshooting of user profiles and smartphones. It also provides VPN termination facilities (IPsec or SSL/TLS) for the secure connections to the smartphones.

While its instantiation in a dedicated server within the enterprise is always possible, the SMC is particularly well suited for implementation as a virtual server in the cloud. A provider of cloud services can allocate SMC instances on demand for different enterprise customers, adjusting the number of SMC instances to the widely fluctuating load of connected remote users. This arrangement benefits the cloud service provider with an important expansion of the product portfolio for the enterprise market. On the other hand, the enterprise avoids the addition of new physical components to its IT infrastructure, eliminating the upfront capital expenses for the purchase and the runtime expenses for real estate, power, and maintenance. The flexibility of the cloud model also eliminates the extra cost of provisioning the added infrastructure for peak traffic conditions. We can conclude that the virtualization of the SMC rakes in all the advantages of cloud computing without apparent drawbacks.

3.2.2 Smartphone

We use the G1 (or HTC Dream) smartphone with Android OS [1] as the platform for installation of the MA and UA software modules. Android is an excellent match for most of our needs, because it is open-source and its kernel is Linux-based. However, Android's preferred support for Java applications rather than C applications (it uses a non-standard C library for its core user space components) collides with our plan to reuse C code from prior projects for implementation of the VPN and MA functions. We overcome this obstacle by creating a *chroot* environment that runs a complete regular Linux distribution, cross-compiled for the Advanced RISC Machine (ARM) Instruction Set Architecture (ISA) of the Android smartphone (since we can replicate the same approach over any other smartphone with Android OS and ARM ISA, our solution is not strictly tied to the G1 model). The *chroot* environment is large enough (750 MB in our implementation, at a time when 64 GB Micro SD cards have started hitting the market) to contain all the libraries that are needed for running C applications, such as the standard GNU *libc*.

The cross-compilation of our application C code base for the ARM ISA and its installation in a *chroot* environment adds to the Android platform a standard Linux user space with no impact on its native capabilities. We implement the UA as an Android application outside the *chroot* environment and install the rest of the software, including the MA, in the *chroot* environment. Since *chroot* isolates the file system but not the kernel nor the network, the UA can still communicate with the MA using network sockets.

We provide the smartphone with the needed USB device classes by porting the respective modules from the regular Linux kernel into the Android kernel, where

some classes, such as the USB Mass Storage Class, are natively missing. We do not modify any of the existing USB gadget modules of Android. On the networking side, the smartphone runs a DHCP server and uses the *iptables* framework of Linux for network address translation (NAT). For VPN connectivity to the SMC, we install a commercial IPsec implementation for Linux, consisting of a kernel IPsec driver and a user-space IKE utility controlled by the MA.

We partition the storage space in the Secure Digital (SD) card of the smartphone to create static allocations for the *chroot* Linux environment, for the encryption keys, and for the data to be stored when the smartphone operates as a USB Mass Storage Class device. We compile the *iptables*, USB, and VPN software using open source code for the G1 Android smartphone that is publicly available.

3.2.3 Computer

Our architecture does not require the installation of service-specific software in the computer. Drivers for the USB device classes that implement the services are generally present by default in current OS releases. We emphasize that the absence of software installation requirements for the computer is a major differentiator for our USB services, since it makes their deployment possible without involvement of the end user.

4. EXAMPLES OF USB SERVICES

We describe here a representative selection of USB services that rely on a variety of USB device types.

4.1 USB Internet Tethering

Tethering is the extension of the smartphone's WWAN connection to an attached computer. It makes Internet access possible when more typical media for network connectivity, such as Ethernet or WiFi, are not available to the computer (e.g., while traveling).

In a conventional tethering service the smartphone exposes a Bluetooth or USB modem interface, which the computer uses for dial-up networking (the WiFi option is also available, but flawed by much higher power consumption). The USB modem interface method connects the computer directly with the Internet, bypassing the smartphone. This configuration interferes with the synchronization of applications between the computer and the smartphone and prevents computer access to any files stored on the smartphone.

In our USB Internet tethering service, the MA configures the smartphone to work as a router and to expose a USB Ethernet interface (USB CDC) to the computer, so that the computer can communicate with both the Internet and the smartphone. The *iptables* framework provides NAT functionality, which enables IP masquerading. The *dnsmasq* utility issues the private IP address to the computer (DHCP server functionality) and relays the DNS settings from the WWAN base station. With *lighttpd*, a lightweight web server, the computer can access files on the smartphone through a web-based interface. By activation of the SSH server daemon we allow the computer to log into the smartphone and transfer files to and from the smartphone using an encrypted interface.

4.2 USB VPN Tethering

Internet tethering offers connectivity to the outside world, but enterprise users also need to access their corporate network through a VPN connection, secured by user authentication and data encryption. IPsec [5] is typically the VPN technology of choice. The Point-to-Point Tunneling Protocol (PPTP) [3] and SSH [8] are far less popular alternatives. TLS/SSL tunnels [2] are gaining popularity, mostly because they can be opened with a web browser and do not require the installation and configuration of a software client, but they only support web-based applications.

In solutions for remote enterprise access that adopt technologies other than TLS/SSL, a VPN client is included in the pre-imaged computer of the user. With this approach, remote access is not possible if the pre-imaged computer is broken or not within reach. Other computers cannot be used because they lack the necessary VPN client with proper configuration.

Just like the USB Internet tethering service, our USB VPN tethering service allows the user to connect *any* computer to the smartphone. However, with VPN tethering the computer connects not to the general Internet, but to the VPN gateway of the enterprise, using a VPN client installed in the smartphone that offloads authentication and encryption from the client-free computer.

The user invokes the VPN tethering option from the UA on the smartphone and provides the necessary authentication credentials to establish the VPN tunnel between the smartphone and the enterprise. When the user connects the computer to the smartphone using the USB cable, the computer perceives the smartphone as a USB Ethernet device. The *dnsmasq* utility on the smartphone relays to the computer its enterprise IP address (obtained from the VPN gateway at the edge of the enterprise network) along with other network parameters like the identifiers of the DNS and WINS servers. Since the smartphone also acts as the default router for the computer, the computer directs all of its plain-text traffic to the smartphone, where it is encrypted and dispatched over the access connection that currently serves the smartphone (whether WWAN, WLAN, or Bluetooth).

Opening VPN access to a generic computer could in theory increase the exposure of the enterprise network to threats associated with dual connectivity, where the computer relays a rogue connection from one of its other network interfaces, and with corrupted files

residing in the computer's hard disk. To avoid dual connectivity, the MA can force all network traffic of the computer through the smartphone when VPN tethering is enabled. Disk scan software installed in the smartphone and updated irrespective of computer attachment can exclude computers with unsafe software from the enterprise perimeter.

4.3 USB Smart Card

Compared to a traditional logon procedure based on username and password, a smart card increases the robustness of the user authentication process for remote network access by joining the requirement for "something the user knows" (the personal identification number, or PIN) with the requirement for "something the user has" (the smart card). The cryptographic certificates stored in the smart card are necessary not only for user logon after the host OS is loaded, but also to enable the execution of the boot-up procedure. Unfortunately, smart card authentication for user logon is by itself not sufficient to secure a computer, since the computer hard disk can always be read by attaching it to another computer. Foolproof security requires using smart card authentication not only for booting but also for full disk encryption.

The USB smart card service configures the smartphone as a smart card with network connectivity. The user must connect the smartphone to the computer via the USB cable in order for the computer to boot up and for the logon procedure to move on. The smartphone is also the repository of the cryptographic keys for encryption of the computer's hard disk. An implementation that complies with smart card standards can interoperate with different full disk encryption solutions, starting with the pre-boot authentication procedure.

If the computer is stolen without the smart card, the hard disk contents cannot be read. If the theft includes the smartphone, the IT administrator can remotely revoke the encryption keys and the authentication certificates the first time the smartphone attaches to a network again, blocking access to the hard disk. Still, the hard disk contents are not lost forever, because the administrator can save the disk encryption keys before revoking them, and restore them in the eventuality that the hard disk is recovered. This is true even if the original smartphone is lost, because the same keys can be installed in a different smartphone. Note that different regulations may apply to the duplication and remote storage of cryptographic keys depending on their use. Typically, the duplication of encryption keys is permitted but the duplication of authentication keys is not.

For implementation of the USB smart card service we choose the USB CDC Ethernet device class instead of the more obvious USB Smart Card class. This way the smart card can coexist with the VPN tethering service for remote enterprise access. Also, the USB Smart Card class only supports card readers. Instead, the enhanced computer protection enabled by our service requires the entire smart card, with keys and certificates, to be instantiated in the smartphone. The end result is a smart card that is remotely controlled by the IT administrator, who can install and remove encryption keys and certificates via the network.

4.4 USB Serial Console

To troubleshoot issues on a computer, a technician needs to log into it. If an issue disrupts network connectivity, the technician cannot troubleshoot the computer remotely.

Many computers and most network equipment support serial-console access, which does not require a functional networking stack and a healthy network infrastructure. In setups for remote serial-console access, the serial console is connected to a serial console server, which in turn is accessed remotely through *telnet* or SSH sessions. However, these setups are ineffective when the network problems that make the computer or network equipment unreachable also prevent access to the serial console server.

The ideal solution provides out-of-band access to the serial console independently of the operational state of the main network, and can be found in a smartphone that acts as a serial console. The USB CDC ACM specifies a USB serial port with full support for all modem control signals. The Linux kernel includes a USB gadget serial module, which implements a subset of the USB CDC ACM class. The USB serial module can also be configured as a raw serial port, with no modem control signals: this mode is sufficient for operation of a serial console for Linux systems, whereas the ACM mode is needed for serial port access to Microsoft Windows systems.

When the end user selects the managed USB serial console service from the UA, the smartphone loads the USB serial module from the Linux kernel. When the smartphone is connected through a USB cable, it appears to the computer or network equipment as a serial console terminal. Once the smartphone is available as a serial console terminal, the enterprise IT administrator can log into the console over the smartphone's data connection. If the managed USB VPN tethering service is also implemented, the login session is encrypted. The result is an out-of-band secure connection to off-site equipment that is always ready for establishment, even at times when regular network connectivity is not available.

4.5 USB Mass Storage

When configured as a USB Mass Storage device, the smartphone acts as a flash disk drive attached to the

computer. In our implementation, we create a partition on the SD card of the smartphone and map it to the USB Mass Storage device. The MA has the ability to move data files in both directions between the mapped partition and the network. As a result, the smartphone appears to the computer as a virtual network drive in the enterprise network.

The IT organization of the enterprise can use the mapped partition on the smartphone to push/pull content to/from the computer even at times when the computer is not powered on or connected to the smartphone. Thanks to enhanced data transfer primitives (described in greater detail in [4]), the data exchanges between the computer and the enterprise servers, which normally require direct IP connectivity between the two endpoints, can be staged in two time-shifted steps. In one step, the computer and the smartphone exchange the content irrespective of the availability of Internet connectivity. In the other step, the smartphone and the enterprise server exchange the content whether or not the computer is powered on and attached to the smartphone.

The main outcome of managed USB Mass Storage services is the extended opportunity offered to critical data content like software patches and drive backups to reach their destinations within short time of their generation. Such time gap reduction is extremely beneficial to the IT security of the enterprise [4], [6].

5. BENEFITS OVER ALTERNATIVES

In existing applications, smartphones act either as USB Mass Storage devices, for audio, video, and picture files, or as USB CDC ACM Modem devices, for conventional tethering. It is rare for a smartphone to support other USB device classes and expose a broad variety of networked peripherals.

WiFi or Bluetooth could be used instead of USB to connect a virtual peripheral in the smartphone to the computer, but they are not as ubiquitous as USB in computers (especially desktops and rack-mounted) and smartphones. They also present a number of important drawbacks. For WiFi connectivity, both the computer and the smartphone must connect to the same network, which means that there must be a WiFi hotspot nearby that both can join. It is possible to create an ad-hoc network between a laptop and a phone, but this requires extra configuration steps. Bluetooth is capable of ad-hoc networking, but also at the expense of extra configuration steps. Even neglecting the configuration hurdle, the power issue remains critical. Turning on the Bluetooth interface increases the power consumption of the smartphone; turning on the WiFi interface increases it even more. This limits the uptime for the smartphone when using either interface. In addition, both Bluetooth and WiFi increase the vulnerability of the enterprise by expanding the set of channels over which the laptop and the smartphone can be attacked. Finally, computers do not support the two technologies at boot time, confining their use to post-boot times only. USB does not suffer from any of these limitations.

6. CONCLUSIONS

We have described examples of new IT services enabled by the Swiss-Army smartphone, a conventional handheld empowered by the versatility of the standard USB device classes. The services add functionality to a computer without requiring the installation of new software and can be managed remotely through the wireless interfaces of the smartphone. They bring convenience to end users and important cost reductions to IT organizations, especially in enterprises of small and medium size where a substantial fraction of the overall workforce is sparsely distributed over wide geographical areas. Further savings are enabled by the virtualization in a cloud environment of the service management center that provides administrative access and secure connectivity to the smartphones. We underscore that our USB services do not rely on specific assumptions regarding the nature and features of the underlying enterprise IT infrastructure and are therefore straightforward to integrate in existing IT solutions. Overall, the ability to instantiate a virtual USB peripheral on a smartphone and control it from a remote location can spawn a large set of creative uses, of which we have only started scratching the surface.

7. REFERENCES

[1] Android. <http://www.android.com>.

[2] Dierks, T. and Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5246. August 2008. <http://www.ietf.org/rfc/rfc5246.txt>.

[3] Hamzeh, K., et al. Point-to-Point Tunneling Protocol. IETF RFC 2637. July 1999. <http://www.ietf.org/rfc/rfc2637.txt>.

[4] Hari, A., et al. Energy-Efficient Data Transfer Primitives for Laptops Using Mobile Handhelds. Proceedings of ACM MobiHeld 2010 (New Delhi, India, Aug. 2010).

[5] Kent, S. and Seo, K. Security Architecture for the Internet Protocol. IETF RFC 4301, December 2005, <http://www.ietf.org/rfc/rfc4301.txt>.

[6] Stiliadis, D., et al. Evros: A Service-Delivery Platform for Extending Security Coverage and IT Reach. Bell Labs Tech. J., 12, 3 (Sep. 2007).

[7] Universal Serial Bus. <http://www.usb.org>.

[8] Ylonen, T. and Lonvick, C. The Secure Shell (SSH) Connection Protocol. IETF RFC 4254. Jan. 2006. <http://www.ietf.org/rfc/rfc4254.txt>.