Short-term solution for 3G networks in Linux: umtsmon

Klaas van Gend
MontaVista Software, Inc.
klaas.van.gend@mvista.com

Abstract

When not at home and in need of Internet access, one can search for an open Wifi access point. But if there is none available, you'll have to set up a connection through a mobile network—GPRS, EDGE, UMTS, HSDPA, or WCDMA networks. For laptops, there are special PCMCIA cards that can do that; most mid- or high-end mobile phones can do this through USB or Bluetooth as well. To manage such a connection, one needs special software—it works differently from a regular phone dial-up, Wifi, or Ethernet connection.

`umtsmon` was created to address this need. The author wanted to have a simple tool that works for all current Linux distributions. `umtsmon` is not the final mobile network manager application—at some point, the functionality of umtsmon will have to be integrated into Network Manager and its GUI apps. But umtsmon will definitely serve as a playground to find out what users really want, so only the really used features will be implemented the right way into Network Manager. Also: umtsmon is available *now* as a simple download and run—it is usable for existing Linux users. The integrated Network Manager will only help future distributions.

In this talk, Klaas will discuss mobile networks in general, how the various brands of pccards work, and what umtsmon can and cannot do yet.

1 Short history of 3G networks

For Western European consumers, mobile communication networks started in the early 80s. Back then, most countries had their own analog standards and equipment was heavy—mainly due to the batteries and antennas. It was also rather easy to listen into conversations and it was expensive.

The organisation GSM, Groupe Spécial Mobile, was started in 1982 by the joint European telecom operators to address the issues. By the end of the eighties, the GSM standard was close to finished and was tranferred to a standardization body, ETSI. The new standard used digital transmission, performed frequency hopping, and prevented tapping of conversations.

The first network to go live was in Finland in 1991; by the end of 1993, networks were operational in 48 countries with a total of one million subscribers.

The popularity of GSM surprised many—by now every person in the Netherlands (including newlyborns and the elderly) owns more than one mobile handset. This unexpected surge in users caused telecom operators to start hunting for expansion of their networks—they feared overloading of their networks.

Also, GSM features a direct tunnel from handset to local cell antenna, and users are charged for the duration of the connection. This is less useful for data connections, so an extension to the GSM standard was made: GPRS. This allowed for a packet-switching-based connection, therefore not requiring a tunnel. Charging per amount of data was possible. However, GSM/GPRS only allows for small bandwidths—typically comparable to old analog modem speeds—max 28k8 kbits/s if you are lucky.

UMTS was created to address these two needs—relieving the load of the GSM networks and addressing higher bandwidth data communication. Different radio technology required new antennas, thus requiring new radio frequencies and equipment. Governments across Europe saw their chance and auctioned the new radio frequencies for astronomic sums of money. Several telecom operators nearly went bankrupt because they were forced to buy into expensive frequencies in several countries.

A deafening silence followed.

UMTS requires a lot of antennas. To get a decent coverage, there are far more antennas required for UMTS than there are for a normal GSM network. The year 2004 through the first half 2006 saw a heated debate on the dangers of radio transmitters in cell phones and antennas to the public health. All kind of research either suggested that one would get sick from living close to an antenna, or would get a heated brain from using a cell phone. Also some people advised against wearing a cell in one's trouser pockets as it might reduce virility (?!). This made several local municipalities refuse UMTS antennas within their territory. These "white spots" in UMTS coverage might endanger the timely roll-out of the UMTS networks...

So, telecom operators invested in frequencies, invested in equipment. And just some Internet junkies bought into it and bought PCCards for laptops or expensive phones that used the new network. At the same time, most operators added new GSM antennas to their new UMTS stations. By creating more cells, the average number of users in a cell decreased—the GSM technology was saved from overload.

PCCards and especially Blackberries were the first killer apps for UMTS—these devices use UMTS (if available) to get their users access to their e-mail. The reduction of the rates helped adoption of UMTS, but still many consider it too expensive. Nowadays, most telecom operators are on break-even for their 3G networks—when not counting in payment of the original radio licenses.

By now, telecom operators see the danger of competing technologies like WiMax and Wifi. So they are moving their UMTS networks forward to HSPA (HSDPA and HSUPA) which require yet new devices.

2 Analysis of a PCMCIA UMTS card

All telecom operators sell PCCards for laptops. However, all are OEM products, manufactured by small specialised companies like Option (Belgium), Novatel Wireless (USA), 3GSystems (Germany) and Sierra Wireless (USA).

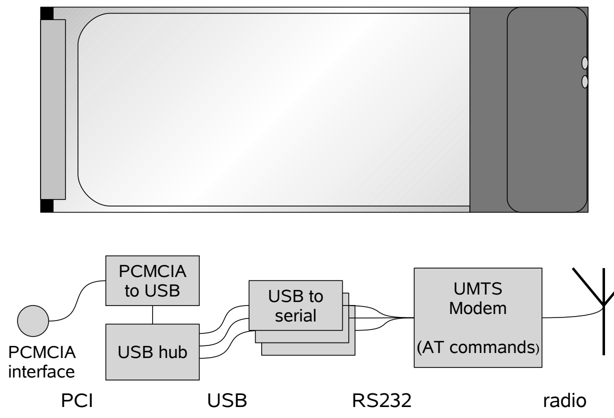

Figure 1: a PCMCIA card and its contents

The first generation of UMTS cards as pioneered by Option, have an interesting hardware architecture as depicted in Figure 1. If one inserts the card into a laptop running Linux, Linux will recognize a new USB interface. Three (or four) usbserial devices are connected (via a hub) to that USB interface. Standard Linux kernels do not recognize the VendorID/Product ID for usbserial, but if forced to load by hand, three new serial ports appear on `/dev/ttyUSBx`.

Most other vendors have cards with similar design—this is mainly due to the fact that there are very limited manufacturers of chipsets for UMTS data. In most cases, this is QualComm.

QualComm decided to go a different route for the HSDPA cards. No longer serial2usb interfaces, but specialised connections that require a specialised driver. With help from other people, Paul Hardwick from Option ported the existing Windows driver to Linux. This driver is now known as the Nozomi driver. Its path to inclusion in the Linux kernel is interesting—of course it was rejected for inclusion because it was not written the "Linux way." With help from Greg Kroah-Hartman, the driver is taking shape now.

3 Analysis of a "Zero CD" USB UMTS brick

To reduce packaging costs, some vendors now ship a technology called "ZeroCD." It was pioneered by Option in their ICON USB box (see Figure 2), but there are also PCCards available. Essential is that the device

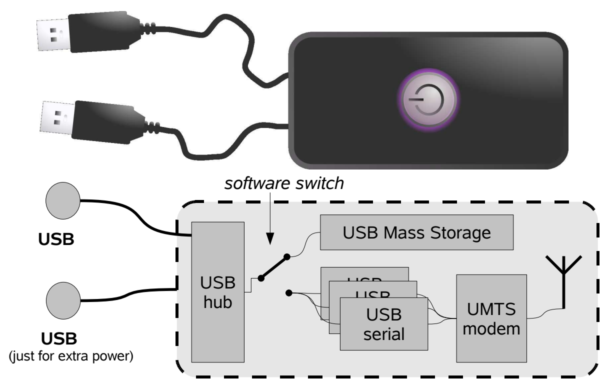

Figure 2: a "ZeroCD" USB box and its contents

boots up as a USB Mass Storage Device. For Windows users, an `autorun.inf` file will automatically take care of installing the software. Afterwards, the driver will send a magic string to the USB Hub. The USB Hub then disconnects the Mass Storage and connects the modem ports, either through USBSerial or the above mentioned Nozomi. For this type of device, vendors do not need to ship installation CDs. For users, the installation is simplified: just plugging it in is enough.

The major disadvantage of the current ZeroCD chipset is the power consumption—it requires two USB plugs because one USB plug can only carry 5W... Be careful and remove the device if your laptop runs on battery power!

4 Back to the AT commands era

Whether the card supports serial or nozomi, after loading the right driver, you will run into serial ports on `/dev/ttyUSBx`, `/dev/ttyACMx`, or `/dev/nozomiX`.

Adventurous Penguins immediately type `minicom /dev/ttyUSB0` to see what's on the serial devices. Hopefully, they are old enough to remember the good old Hayes-compatible modem era, where one could play with AT Commands and PPP settings for ages before the first connection to the outside world was successful.

Actually, they need to—because that's exactly what you get: all three interfaces are connected to an UMTS modem with an AT Command set. However, standardization committee 3GPP has standardized the AT commands for 3G network modems, so there are standardized commands to talk to the modem to enter PIN codes, select the network of a mobile operator, send an SMS, and such. There are multiple interfaces to the modem to allow one interface to be used by PPP for the actual network connection, whereas one can use another interface to send AT commands to retrieve the status of the modem, network, or SIM card.

A few examples of new AT commands:

- `AT+COPS=?` will (after 30 seconds) return a list of all mobile networks that are available, with info on which networks you are allowed to connect to.
- `AT+CPIN="1234"` to enter a PIN code for the smart card.
- `AT+CSQ` returns the signal strength of the mobile connection.

So we're stuck with AT interfaces. Let's re-install ppp and go back to the old days. Or install umtsmon—which will interact with the modem and do all the AT commands for you.

5 Single Serial Port devices

The most popular cards in Europe all have multiple interfaces. This means that we can, for example, use `/dev/ttyUSB0` to connect PPP to, whilst we can simultaneously send AT commands to `/dev/ttyUSB2`. However, some cards (like the Sony Ericsson GC79, most Sierra Wireless cards, and some of the Novatels) only have a single serial port. This means that AT commands and PPP data need to share the same port. Technically, this is not a problem—software like `kppp` implements this functionality already. However, the OpenMoko project also spawned a new project called the GSM Daemon that also can do this. This might be an interesting road for the future—at the cost of another external dependency.

6 umtsmon system design—dependencies

As stated before, umtsmon was designed to work on as wide a range of Linuxes as possible. This is why we attempt to make as few assumptions about the operating system as possible. Yet we have to rely on a few packages to be present:

- PPP
  We need PPP to make the actual network connection and change routing and such.

- QT3
  The QT library is needed for the UI. `umtsmon` cannot run without it. QT in itself also has a few dependencies, like X.

For versions of umtsmon beyond 0.6, we probably will add a few more requirements:

- pcmcia-utils
  This one obviously is not necessary for the ICON and other USB-only devices. We want to use pcmcia-utils to enable users to reset the card (`pccard eject` and `pccard insert`) and/or to simplify the autodetection code.

- libusb
  At this moment, a lot of the autodetection is done by browsing through the `/sys` filesystem. This is rather complex to code consistently for all Linux kernel revisions. Using libusb should solve that problem for us and again simplify the autodetection code. It might be wise to move the libusb-dependent code into a shared library that is `dlopen()` ed at runtime to prevent umtsmon from trying to run if libusb isn't present or if it is too old to work.

- icon_switch
  This is a small utility that switches the ICON box from mass storage to modem operation. Unfortunately, it is rather unreliable and it needs extensions for the other ZeroCD devices that have different USB IDs and may require different code sequences. `umtsmon` at this moment requires PPP to be SUID—umtsmon calls pppd directly with arguments controlling the connection. `umtsmon` will complain to the user if PPP is not set SUID and ask the user if it is allowed to fix it. This is a security hole: in theory people could start writing malicious dialers dialing expensive foreign numbers. This should be addressed in a future revision by having umtsmon create profiles in the `/etc/ppp/peers/` directory.

7 umtsmon software design

Internally, umtsmon is written to follow the MVC design pattern. MVC means Model View Controller. Basically, this comes down to a separation of concerns—a class should only contain code that represents data (=model), changes data (=controller), or displays it (=view).

Central in the umtsmon 0.6 design are the classes Query, SerialPort, ConnectionInfo, and PPPConnection. Any AT Command sequence that is to be sent to the card is represented as a Query instance. The Query class also contains rudimentary parsing and will strip off echos and such. Query connects to a SerialPort instance for the actual communication.

The following paragraphs discuss some details of the design of umtsmon.

7.1 PPPConnection

PPPConnection is the beating heart of the application. As can be seen in Figure 3, the main GUI class MainWindow subscribes one of its attributes, the MainWindowPPPObserver, to receive any state changes of the PPP daemon. If someone outside umtsmon or umtsmon itself then starts the ppp daemon to make a connection, the PPPConnection class with call all its attached Observers to notify the state changes. MainWindow responds to that by enabling/disabling buttons and menu items.

At this moment, only MainWindow is subscribed to receive the PPP state changes. This will change in the near future when we start talking about the NetworkManager integration—that will require another Observer to the PPP state.

7.2 (Inhibiting) ConnectionInfo

ConnectionInfo regularly polls the card to ask for the mobile operator, signal strength, and such. On some occasions, ConnectionInfo must be prevented from sending out Queries, like during the PPP connection setup or whilst `AT+COPS=?` (see Section 4) is running. In such cases, the PPPConnection or NetworkChanger class just creates a ConnectionInfoInhibitor instance. Creation of the Inhibitor instance will increase a counter inside

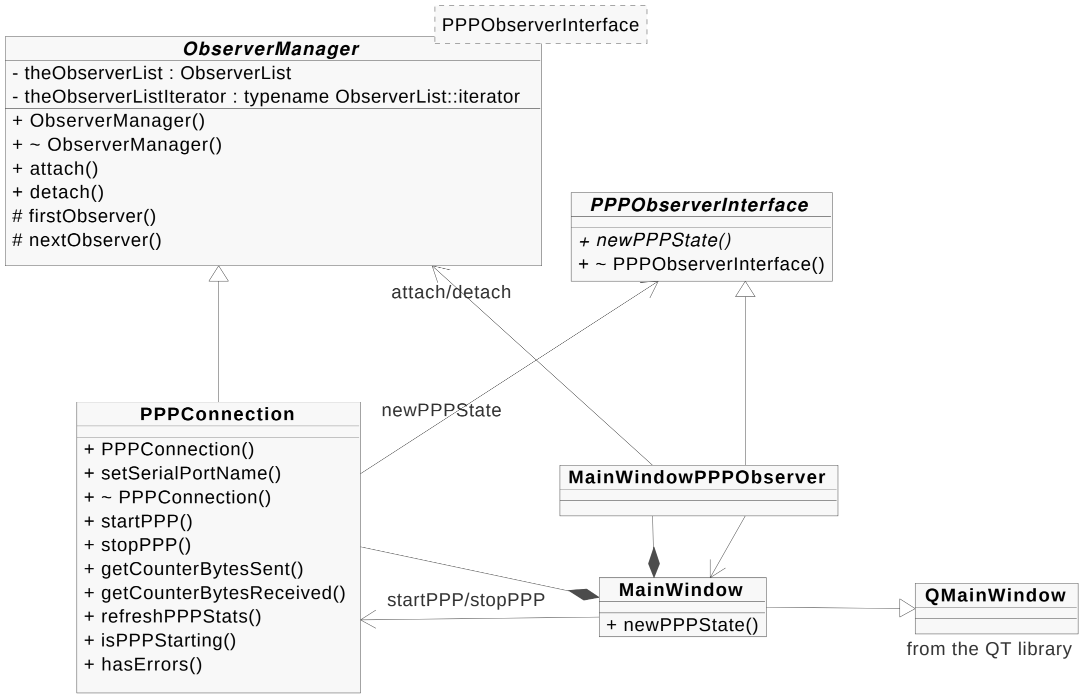

Figure 3: Class Diagram of PPPConnection and interacting classes

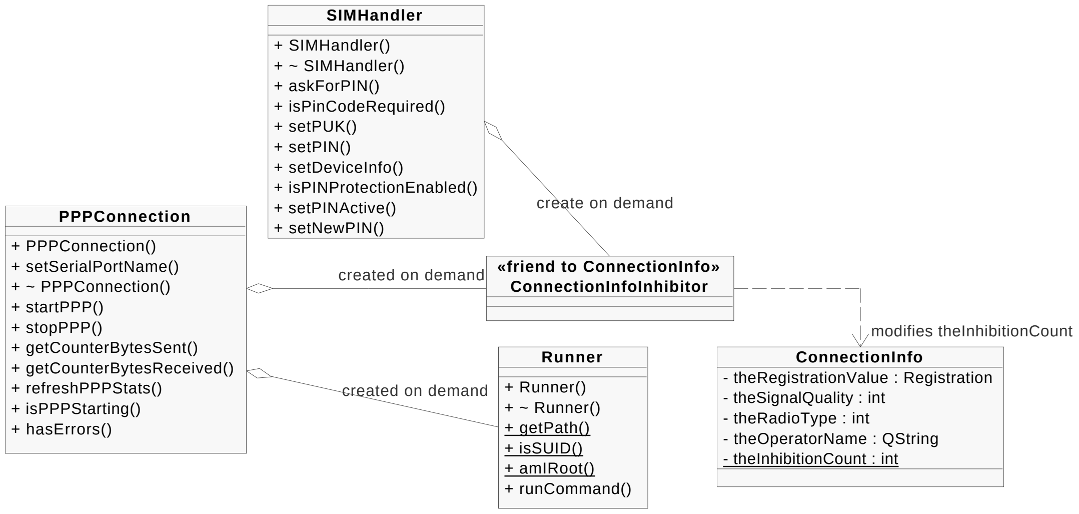

Figure 4: Class Diagram of ConnectionInfo and interacting classes

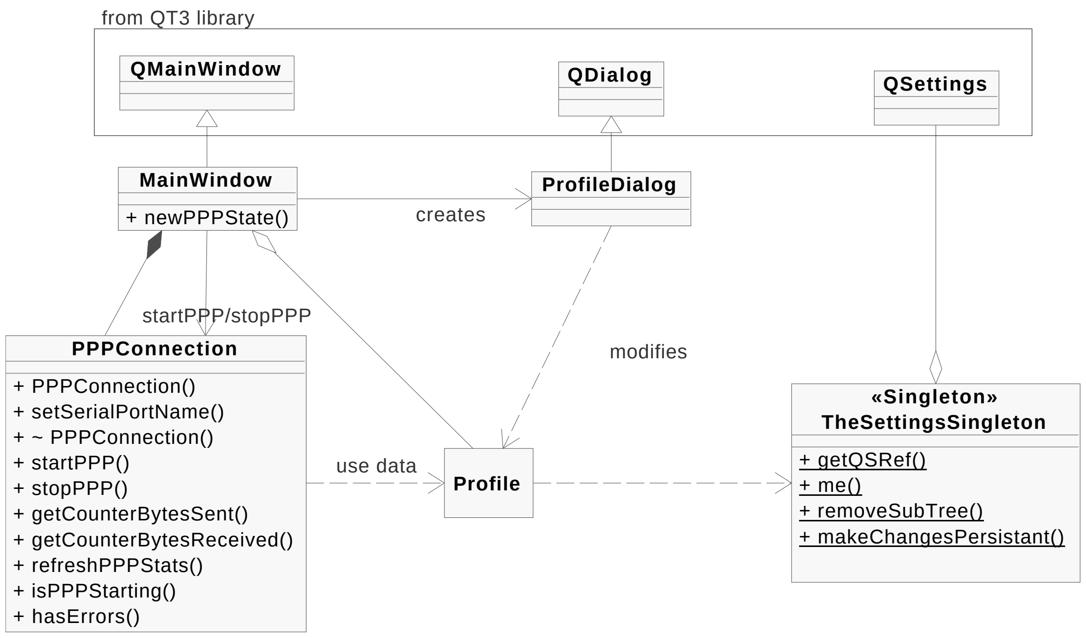

Figure 5: Class Diagram of Profile and interacting classes

ConnectionInfo; upon destruction, the counter will automatically be decreased. PPPConnection uses an instance of the Runner class to manage the execution of `/usr/sbin/pppd`. This is shown in Figure 4. In the case of single serial port cards, the AT commands and the PPP data stream need to be sent over the same serial port. In that case, ConnectionInfo never can run when a PPP connection exists.

7.3 Profile Management

Refer to Figure 5 for a class diagram. Once the user clicks on the connect button in MainWindow, PPPConnection gets called with a reference to a Profile. PPPConnection will use the info inside the Profile class to setup the connection. To change a profile, the user selects Profile Management in the menu in the GUI. The ProfileDialog will be instantiated and filled with the data from the active Profile. Users then can choose to create another profile, change the current one, etc. The data is stored to disk in a `key=data` formatted `~/.umtsmon/umtsmonrc` file. The QSettings class takes care of that. Because settings need to be accessed throughout the program, possibly even before `main()` is started, settings are always retrieved through a Singleton pattern class. This also solves the issue that a QSettings class only saves its data upon destruction. Calling `makeChangesPersistent()` will thus cause the QSettings instance to be destroyed.

8 NetworkManager integration and/or takeover

High on the wish list is to integrate at least a little with NetworkManager. The big annoyance is that at the moment, even if a UMTS connection is made, software like Gaim and Firefox will refuse to connect because according to them there is no connection. Apparently they use libnm to ask NetworkManager if the system is on-line or not. As stated before, umtsmon should run, regardless of NetworkManager's presence. However, this is only loose coupling—we're not discussing adding UMTS support to Network Manager yet. In the end, UMTS connections should be just another item that is implemented in NetworkManager and its GUIs. We currently view umtsmon as a playing ground for that. What features are actually used? How to implement stuff? Where to put security constraints? It all is easier to do in a small program than in the collection of binaries that makes up NetworkManager. Yet NetworkManager is the future—it is the most convenient way for users to manage yet another networking connection. It remains to be seen if the current umtsmon team will actually do the NetworkManager implementation.

Brand	model	type	remarks
Sony Ericsson Option	GC79 GT GPRS EDGE	single-port serial	GPRS and EDGE only
Option Huawei	3G Quad E612	three of four port usb2serial	need kernel module usbserial.ko with parameters or the specialised option kernel module.
Option		nozomi	need nozomi kernel module
Option 3GSystems	ICON XSPlug3	external usb box	ZeroCD chipset, need switching
Sierra Wireless Novatel	7xx series U630/U530	PCMCIA serial modem	single serial port
Novatel	XU870	dual serial port, only first usable	first Expresscard to be supported
Kyocera	KPC650	dual serial port	serial ports don't communicate.
various mobile phones	various	connect through either serial, USB or Bluetooth	handled as a single serial port card

Table 1: Hardware support of umtsmon

9 Device, Commercial and Distribution support

At this moment, umtsmon supports a wide variety of hardware. The 0.6 release will support devices from Novatel, 3G Systems, Sony Ericsson, Option, Sierra Wireless, and Huawei. Also mobile phones that have a laptop connection through USB, Bluetooth, or serial can be supported, with a few limitations. See Table 1 for a more complete list.

None of the device vendors is cooperating with the development, however. The development team is currently investigating creating a fund to buy all available hardware and distribute it amongst the developers to ensure that all hardware is supported and remains operational.

Network operator T-Mobile Germany sponsored the development by providing a laptop and devices to one of the developers, who happened to also have done an internship on UMTS on Linux for T-Mobile. The internship resulted in a lot of improvements to umtsmon, thanks Christofer!

At the moment of writing this paper, umtsmon is available as a standard package in OpenSuse (starting umtsmon 0.3 in OpenSuse 10.2) and Gentoo (starting with umtsmon 0.5, currently in ~amd64 and ~x86 only).

10 Conclusions

- Support for UMTS cards is in the same position as hardware enablement projects like ALSA were a few years ago. Several devices work—mostly the devices owned by the developers. Manufacturers don't see the need to cooperate yet, nor to fund development.
- All known 3G mobile devices implement serial interfaces, either directly or through drivers.
- UMTS is just another radio technology reusing existing communication standards: AT commands.
- umtsmon was created to handle the AT commands exchange for the user and start the PPP daemon.
- umtsmon is not really integrated into Linux—it's a standalone application.
- NetworkManager integration should start from scratch, possibly inheriting the GSM multiplexing daemon from OpenMoko to solve the single serial port problem correctly.
- Writing a paper for OLS is a good stimulus for coders to finally write down some parts of their software design.

11 Links

The umtsmon website:
`http://umtsmon.sourceforge.net/`

PharScape (HOWTOs and support forum for all Option based cards and the Nozomi drivers):
`http://www.pharscape.org/`

The 3GPP approved AT command set:
`http://www.3gpp.org/ftp/Specs/latest/Rel-7/27_series/`

Proceedings of the Linux Symposium

Volume Two

June 27th–30th, 2007
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

Dirk Hohndel, *Intel*
Martin Bligh, *Google*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
John Feeney, *Red Hat, Inc.*
Len DiMaggio, *Red Hat, Inc.*
John Poelstra, *Red Hat, Inc.*