Wireless Streaming Video on a Handheld

Version 05
23 Maart 2002

Jean-Paul Saman
Ordina Technical Automation
e-mail: jean-paul.saman@planet.nl

INDEX

0 Introduction

Multimedia in a networked world is almost as normal as having a telephone or possessing a car. Traditionally a streaming solution needed a fast network powerful servers and bulky clients. By standards of today what once was considered bulky is now normal desktop. This makes that streaming video solutions come within reach of mobile devices such as a handheld or portable digital assistance (PDA).

Also wireless solutions will enter the equation. Multimedia and wireless will go hand in hand with upcoming technologies like: GPRS, UMTS, ADSL, Cable. Consumer products like DVD players, settop boxes, PDA's and WebPads are combined with content providers and access architectures to create an end-to end solution. The infrastructure presented in Figure 1 is provides the simplest form of a multimedia gateway backbone that provides services for mobile devices.

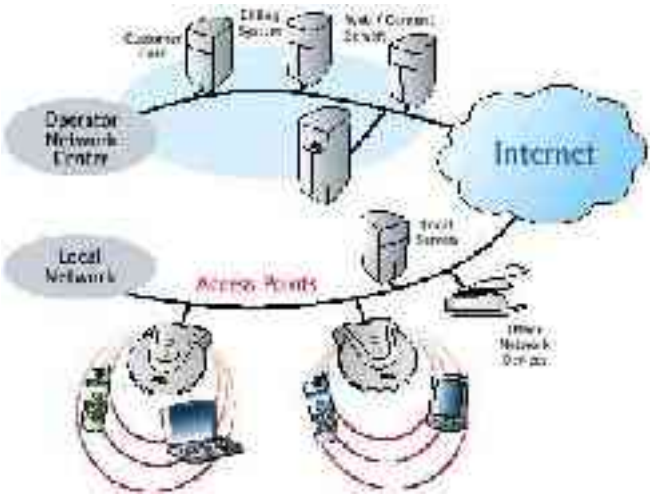

Figure 1: Network infrastructure

The goal of up coming discussion is to provide a low cost multimedia gateway implementation for wireless clients, they connect to a backbone (internet, intranet or office-). For this sample implementation the scope is limited to wireless LAN infrastructure and a PDA. All components, (except the wireless LAN Access Point) run Open Source and Linux software.

01 Wireless Multi-media Portable Digital Adapters (PDA)

Users that have a PDA with wireless capabilities (either Bluetooth or Wireless LAN) can participate in any wireless network. First they have to logon and authenticate themselves before using networks services. On the PDA multimedia information can be accessed or a secure connection with a back-office system can be established.

The wireless user uses several interfaces into the network. He can use any webbrowser, view streaming video, listen to his favorite radio station, listen to his own music collection or participate in a video conference. All this is possible with Linux and multimedia today.

Figure 2: Yopy showing PDA capabilities

02 Internet Server provides media contents

The current network infrastructure will be enhanced with access points for wireless devices. They participate as a normal network node. A device's connection with the network will be through Wireless LAN or Bluetooth. See for more information on these technologies appendix A.

An internet server will have to provide content that can be viewed, listen to on devices with limited screen size and network bandwidth. Today's web developers are used to create flashing internet sites using the latest technologies. All these sites cannot be viewed on a PDA or WebPad because they are too bulky.

Figure 3: 3Com AirConnect Wireless LAN Access Point

Creating web sites and multimedia content for wireless users is a different trade. Instead of a screen size of 800x600 pixels a PDA only has 240x320 pixels. Web developers should also take wireless users into account. Every site should have a small version suitable for 240x320 pixels screens.

Figure 4: Axis Bluetooth Access Point

03 Making Business

All information in this document is about creating a demonstration system with today's technology. Using Embedded Linux and Open Source Software gives the opportunity to develop fast and focus on the streaming video application. See also appendix A for information on Open Source and License issues. In this way there is no time lost in creating our own custom operating system solution and basic framework. Thus shortening the development circle and going to market earlier then the competition.

The application streaming video gives an insight in system requirements that are needed for video conferencing, screen phones, GPRS and upcoming UMTS solutions for wireless devices. Instead of a wireless LAN another wireless technology could be used. Today the most interesting technology is GPRS. Other technologies like UTMS are upcoming and in its early phases of implementation.

In our demonstration there is no real Intellectual Property. A company could make money on the multimedia gateway implementation providing web page conversion for mobile devices, web access to a back office system, video conferencing server software, complete system roll-out and integration.

031 What is GPRS?

GPRS is based on the old GSM technology. It even shares its infrastructure. GSM only deals with voice transmission, GPRS only deals with data transmission. This makes that both technologies complement each other. Currently a GPRS link is unreliable, because GSM transmissions have priority above GPRS transmissions. Another problem is that GPRS subscriptions are expensive and charge fee per byte data sent. Once these problems have been solved, GPRS will be usable beyond business use. Thus having a handheld always online is no longer science fiction.

04 How to use this document?

In this white-paper a wireless streaming video solution for a handheld is described and implemented using only Open Source Software. The big problem is to find and integrate all these separate software packages into a working system.

The following steps correspond to chapters in this document. It teaches how to implement a streaming solution based on Open Source software. All chapters can be read in no particular order, except for chapter 5 which brings all the former chapters together in a final test.

Steps to Take

1. Install Linux distribution on a handheld (PDA)
2. Install and configure wireless LAN drivers
3. Network topology
4. Streaming Video Solution
5. Putting it all together (Integration)

1 Install Linux distribution on a handheld (PDA)

The handheld or PDA is an iPaq, a product from Compaq. It is standard delivered with a Microsoft OS, WinCE for Pocket PC's. This will be replaced by Linux distribution. There are several around each with its merits.

11 Linux distributions for handhelds

In the Open Source community there are several Linux distributions targeted for small devices such as handhelds. These distributions are limited in their use of flash-disk, some require an additional drive called micro-drive. A short description of some distributions will follow.

111 Familiar

The Familiar Project is composed of a group of loosely knit developers all contributing to creating the next generation of PDA OS. Currently, most of our development time is being put towards producing a stable, and fully featured Linux distribution for the Compaq iPAQ h3600-series of handheld computers, as well as applications to run on top of the distribution.

Currently Familiar's Linux distribution supports some of the following key features:

- Entirely based on XFree86's Tiny-X server, which includes the latest RENDER extension.
- Anti-Aliased True-Type Font support in rxvt, blackbox, and fltk (this is extended to any X application using the Xft APIs).
- OpenSSH's ssh and sshd included by default.
- Include JFFS2 support, which enables read/write access to the iPAQ's Flash.
- Integrated Python v2.0 w/ PyGtk and PyGDKImlib.
- Binary and Library compatible with Debian's ARM distribution. In most cases, programs (as long as their dependencies are met) can be taken from Debian and executed on the iPAQ without issue.
- Full package support based on ipkg.

For more information see: http://familiar.handhelds.org

112 Intimate

The Intimate project is a fully blown debian-based (debian is another desktop distribution) Linux distribution for the Compaq iPAQ. Taking the work being done by the Familiar Project and combining it with fully blown debian package management, and access to the thousands of existing debian arm packages. The goal is simple: Provide a best of both worlds. The disadvantage is ... it won't fit in the 16MB Flash. A micro-drive is needed to use this distribution The minimum requirements are currently around 140MB of storage for the base image.

For more information see: http://intimate.handhelds.org

113 Pocket Linux

PocketLinux is a complete operating system solution targeted at small, Internet-enabled devices. The PocketLinux "application framework" gives developers options on how they deliver content and application services to their target consumer market. It makes it possible to develop applications using the best tool for the job.

Developers can use XML to easily develop applications that integrate cleanly and completely into the PocketLinux environment. For more advanced needs, the full power of the Kaffe Java Virtual Machine is available and exposed, making available all the modularity, portability and elegance that is the hallmark of Java.

Performance critical code and bindings to hardware interfaces can be implemented natively (eg. using C, C++ and assembly). Pre-existing non-Java code can be ported to use the same graphics and input libraries as the rest of the framework to ease integration. Libraries to support multimedia types such as MP3 music files, MPEG movies, and flash animations are implemented this way.

The PocketLinux user experience is designed from the ground up to be personal. It has full support for "theming". The entire user interface is customizable using simple XML without the need for programming. It is designed to run on small devices such as PDAs, cell phones, pagers, etc. while retaining the ability to be useful for other classes of devices. PocketLinux is distributed as Open Source under the GNU General Public License.

For more information see: http://www.pocketlinux.com

114 Qt Palmtop on an iPAQ

The Qt Palmtop for iPaq is based upon the Familiar distribution. It provides a GUI windowing system based upon Qt-libraries specifically tuned for small devices such as a handheld. The Qt-libraries are also called Qt/Embedded. Qt is used in the popular desktop windowing application KDE. See http://www.kde.org for more information.

For more information see: http://qpe.sourceforge.net

115 Century Embedded PIXIL

PIXIL provides a complete PIM suite for Internet appliances. Developed for small devices with requirements for wireless communications, networking, web browsing, power management and GUIs. PIXIL is based on the open standards of embedded Linux. It can be used in PDAs, WebPads, set-top boxes, cellular phones, smart handhelds and thin clients.

For more information see: http://embedded.centurysoftware.com/pixil/index.php

12 Installing a distribution

Our project chose the Familiar v0.4 distribution for Linux. To make things more complicated the standard Familiar v0.4 distributions kernel was not used, because it did not support our wireless LAN cards sufficient. The Orinoco driver did not got through its initialisation most of the time. Therefore we switched to a more recent kernel 2.4.6-rmk1-np2-fam4 instead. Even then we needed to make small changes to the driver. See next chapter for more details.

Instructions to install the Familiar distribution can be found on: familiar.handhelds.org . It explains the process of flashing from a Windows workstation. The document "How to flash an iPaq from Linux" describes the same process using Linux tools only. [still to be written]

References:

- Familiar distribution: http://familiar.handhelds.org
- Intimate distribution: http://intimate.handhelds.org
- PocketLinux distribution: http://www.pocketlinux.com
- QT/Embedded distribution: http://qpe.sourceforge.net
- Century Embedded PIXIL: http://embedded.centurysoftware.com/pixil/index.php
- KDE desktop: http://www.kde.org
- Handheld information: http://www.handhelds.org
- Pocket PC iPaq information: http://www.compaq.com
- Linux HowTo's: http://www.linuxdoc.org

2 Install and Configure Wireless LAN drivers

Wireless LAN refers to (computer-) networks without physical wires. The transport mechanism through the air is always RF (radio frequency). The 2.4 GHz frequency to be precise. The wireless technology delivers a network LAN oriented connection that can carry UDP or TCP/IP protocols. There are several wireless standards available today IEEE 802.11a, 802.11b and 802.11g. See chapter "Technology explained" for more in depth information. In this document Wireless LAN refers to the IEEE 802.11b standard. It gives a network bandwidth of 11 Mbps.

Most wireless cards in use today are manufactured by Symbol Technologies and are sold under an OEM license by others. 3Com resells a Prism II card from Symbol Technologies as their 3Com AirConnect product. Other similar products supported under Linux are: Lucent/Agere Orinoco, Compaq Aironet, Symbol Technologies Spectrum24t.

21 Choosing hardware

The choice for a wireless solution is made upon availability and support of the hardware on the target OS. Under Linux the drivers are usually part of the kernel and distribution. If not then finding an active Open Source project is your next best bet. Make sure that the preferred hardware is supported by the distribution or kernel and driver that is going to be used. We chose a 3Com Wireless LAN solution for the hardware.

Linux 2.4.6-rmk1-np2-fam4 does support Prism II based cards. It also supported our 3Com AirConnect card, because it is based upon a Prism II card from Symbol Technologies. In the open source community there are several drivers available for these cards:

- Spectrum24t: http://Spectrum24t.sourceforge.net
- David Gibson Orinoco driver: http://ozlabs.org/people/dgibson/dldwd/

After some experimentation and overcoming difficulties with both drivers, it became apparent that the Orinoco driver of David Gibson is the best choice. It still needed some minor tweaking for our 3Com card, but is better and more actively supported then the Spectrum24t driver. On top of that it supports a whole range of WLAN cards that are quite similar. The rest of this chapter teaches how to build wireless LAN drivers and cross-compile a Linux kernel for the ARM architecture.

22 Cross-Compilation

A cross-compile environment is needed for Linux before it can be compiled for the ARM architecture. The easiest way this can be done is by selecting a pre-built one, e.g. the skiff-toolchain as used by Familiar distribution. (See references chapter 1.) Install it under the root directory.

> A **cross-compiler** is a compiler that runs on cpu A and creates binaries that run on cpu B The act of compiling source code for another cpu-architecture then where it is compiled is called **cross-compiling**

It is also possible to build a cross-compiler from scratch. The GNU project offers all needed sources (binutils, gcc, glibc) on their website http://www.gnu.org. The document "Linux Cross Development Environment" describes how to setup a cross-compiler and an environment for ARM hardware reference board. Beware, it is not a task for the faint hearted.

221 Linux kernel

The standard linux kernel can be downloaded from http://www.kernel.org or one of the mirrors. It does not have support for the ARM architecture yet. The ARM port of the Linux kernel is still fairly new, with other words in experimental. To use it for our purpose, kernel-patches that support the ARM hardware should be applied. These can be found on the ARM port website maintained by Russel King. He and Nicolas Pitre are currently the main developers and driving forces behind the port of Linux kernel to the ARM hardware. To get those patches, download them from http://www.arm.linux.org.uk. The files are named patch-2.4.6-rmk1.gz and diff-2.4.6-rmk1-np2.gz. How to apply these patches on top of the standard Linux kernel is described shortly. First login as root.

Extract the linux kernel source code in a directory using tar.

```
tar -xzvf linux-2.4.6.tgz
```

The tar file creates a directory called linux. In there patches can be applied (see http://www.arm.linux.org.uk/developer/machines). Copy them into the linux directory:

```
cp patch-2.4.6-rmk1  diff-2.4.6-rmk1-np2.gz linux/
cd linux/
```

Now it is time to patch the kernel with ARM and StrongARM specific patches. The order is important! (See the inset below.)

> All **patches** of diffs are differences against the original source code A patch called patch-246-rmk1 tells the reader that it is a patch against the Linux kernel 246 The suffix ***rmk1*** means Russel M King patch version 1 The ***np*** stands for Nicolas Pitre
> Thus a patch called diff-246-rmk1-np2 needs the Russel M King patch version 1 to be applied first After that it can be applied Failing to respect this order in applying patches will result in **rejections** of part of a patch or the complete patch A rejection will be saved into a file called *rej The original file will be called *orig

```
patch -p1 < patch-2.4.6-rmk1
patch -p1 < diff-2.4.6-rmk1-np2
```

If the patches are applied without rejections, then the next command will find no *.rej files.

```
find . -name *.rej
```

Before continuing with the next step a special compiler is needed. The cross-compiler nick named 'skiff-toolchain' is found on http://www.handhelds.org and should be installed under the root-directory. Do a:

```
make mrproper
```

We need to replicate the kernel configuration from the Familiar distribution in our own copy. During this step it is important to know which hardware is in the handheld, because this information is needed in the kernel configuration. See also the documentation in the kernel "Documentation" directory.

The Linux kernel comes with its own configuration tool. There are three interfaces a text based (make config), menu text based (make menuconfig) and a menu based (make xconfig). Each interface provides the same information and allows for the same level of detail. The easiest method is the X-configuration interface of the kernel source. Configure the kernel by typing:

```
make xconfig
```

Go to the menu where network devices are listed and make sure that for the drivers hermes, orinoco and orinoco_cs the option modules (the option 'M') is selected. This ensures that the driver will be built as modules instead of statically linked into the kernel. Save the new configuration and continue the next steps.

The configuration is saves as a ".config" file in the same directory as you typed "make xconfig". In the very same directory there is a Makefile. Issue the next command to setup all dependencies for ARM architecture and kernel:

```
make dep
```

Next some changes in the Makefile are needed. At the first four lines it says:

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 6
EXTRAVERSION =-rmk1-np2
```

Change the last line into :

```
EXTRAVERSION =-rmk1-np2-fam4
```

The EXTRAVERSION information will be compiled into every kernel module. It tells the kernel for which version the modules are compiled and is a guard for interface changes. Now tell the Makefile where to find the cross-compiler. Change line 22 from:

```
CROSS_COMPILE = arm-linux-
```

into:

```
CROSS_COMPILE = /skiff/local/arm-linux/bin/arm-linux-
```

Beware! The last '-' sino error. It should be there, because this macro is used as a prefix to every binary in the makefile.

At the end of this document a chapter "Creating a patch" can be found. There a patch for the wireless LAN driver can be found. This patch is needed to let 3Com AirConnect cards with firmware 1.50 operate consistently. Copy this patch to the ARM-kernel directory and apply the patch:

```
cp patch-2.4.6-rmk1-np2-jps1 linux/
patch -p1 < patch-2.4.6-rmk1-np2-jps1
```

The patch mechanism works in the same way as applied earlier in this document. There may not be any *.rej files left in the source tree. Compile the kernel by typing:

```
make dep bzImage modules modules_install 2>&1 compile_errors.txt
```

Now it is time to relax and drink a cup of coffee. Depending on the CPU speed of the computer this step may take 5 to 20 minutes. On slow (probably older) computer it takes half an hour or longer.

If there are no compile errors in the file compile_errors.txt, then the kernel compiled correctly. In the directory /lib/modules/2.4.6-rmk1-np2-fam4 you find the modules needed for our 3COM AirConnect WLAN card, they are named hermes.o, orinoco.o and orinoco_cs.

222 Wireless LAN drivers

The '`make modules_install`" step of previous section installs all modularised drivers in a directory under `/lib/modules`. Thereby the following logic will be used:

- If the directory does not exist yet, then it will be created.
- If it does exist, then it will be overwritten.

Remember the first four lines of the Makefile mentioned above? It contains some numbers and text, which indicates the kernels version. These will be concatenated together to form the new kernel version and the new directory where all driver modules will be installed.

The drivers for the Wireless LAN card are in the directory:

```
/lib/modules/2.4.6-rmk1-np2-fam4/kernel/drivers/net/wireless
```

Copy these drivers or modules to the handheld and put them in the same directory. If there are older drivers present on the handheld, remove them first (only hermes.o, orinoco.o or orinoco_cs.o). Make sure you logged in under the root account and run the following command:

```
depmod -a
```

Depmod checks all modules that are installed under the current kernel modules directory. If kernel versioning information and interface bindings are correct. If the checks reveals some

inconsistencies, then it will be printed on the console otherwise it will remain silent. This tool helps ensuring that kernel modules are compiled for the kernel they are going to be used in.

The PCMCIA subsystem consists of a PCMCIA driver inside the kernel a user space daemon, card database (config and .conf files) and some scripts with options files (.opts).

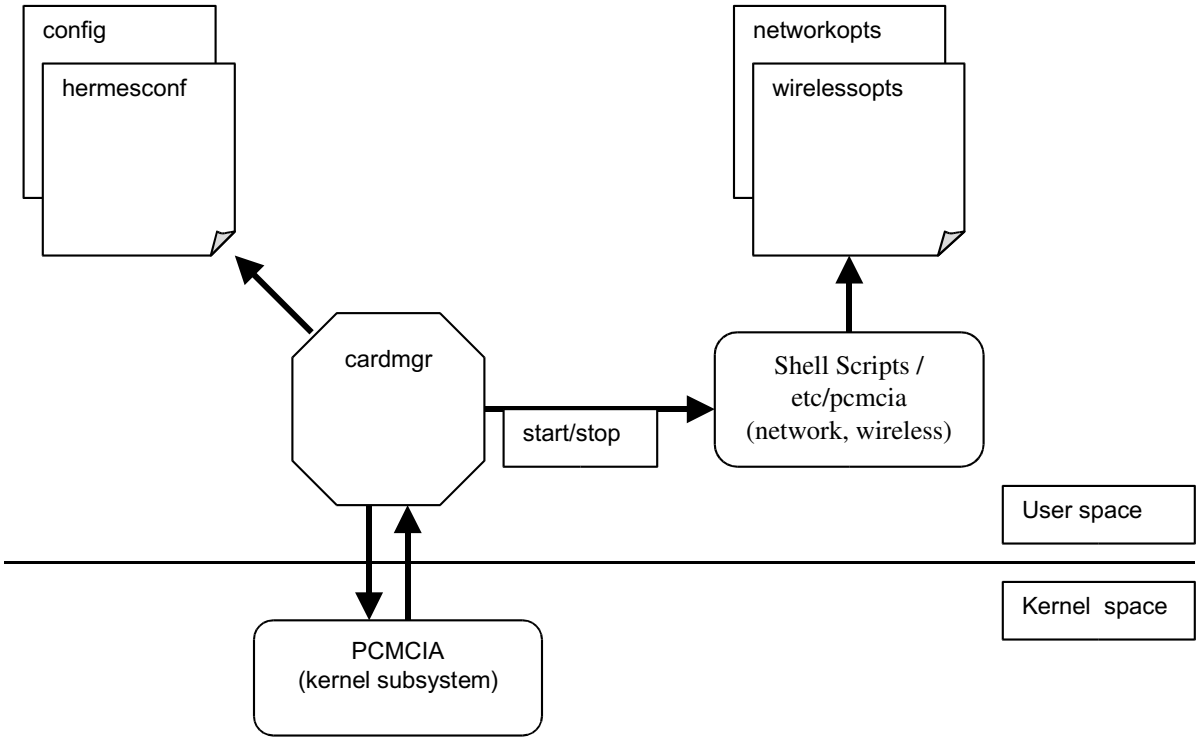

Figure 5: PCMCIA subsystem

The kernel driver sole responsibility is to react on physical interactions (e.g.: insertion, ejection) of a card and sending commands to a card manager daemon (cardmgr), which runs in user space. This daemon reads the card identification strings and looks these up in his database of cards (the files "conf "and "hermes.conf" in the figure). If a match can be found, then it will load the drivers associated with the card and run a shell script. In case of a inserting a network card, the command "`. /network eth0 start`" will be executed. All these steps should successfully complete, before the PCMCIA card can be used.

The PCMCIA subsystem needs to be taught how to recognise the 3Com cards and which drivers it needs to load. A file named hermes.conf is needed, it is part of David Gibson Orinoco drivers source code archive. If it cannot be found, then create a file in the /etc/pcmcia directory with the following lines in it.

```
device "orinoco_cs"
class "network"
module "hermes", "orinoco", "orinoco_cs"
# module "hermes", "orinoco" opts "pc_debug=3", "orinoco_cs"
#
# Wireless network adapters
#
# We should use the manfid (which cover multiple cards),
# otherwise we will
# go crazy listing all cards and their variations !!!

# Second class of device : Symbol & OEM
card "LA4111 Spectrum24 Wireless LAN PC Card"
version "Symbol Technologies"
bind "orinoco_cs"
card "3Com AirConnect"
version "3Com", "3CRWE737A AirConnect Wireless LAN PC Card"
bind "orinoco_cs"
card "Intel PRO/Wireless 2011"
manfid 0x0089,0x0001
bind "orinoco_cs"
card "Ericsson WLAN Card C11"
manfid 0x016b,0x0001
bind "orinoco_cs"
```

The next time the PCMCIA manager is loaded it reads all *.conf files under /etc/pcmcia including our just created hermes.conf. Restarting the PCMCIA manager on the handheld is not enough, it does not re-read its configuration files. Instead simply reboot the handheld with /sbin/reboot to be sure.

During testing it became apparent that the driver had a problem with our 3Com AirConnect cards. Therefore a patch of the drivers was needed. There were some portability problems, because of other data structure alignment on ARM architecture. The chapter “Creating a patch” describes how to create a patch and lists the patch used in this implementation.

At David Gibson’s web site (see references below) the latest version of the orinoco drivers can be found. Version 0.8b and higher incorporates the patch used in this document.

References

- Standard Linux kernel: http://www.kernel.org
- ARM/Linux development kernel: http://www.arm.linux.org.uk
- Spectrum24t WLAN drivers: http://spectrum24t.sourceforge.net
- David Gibson’s Orinoco drivers: http://ozlabs.org/people/dgibson/dldwd/
- Jean Touriless Wireless LAN How To: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/
- Wireless HowTo: http://www.linuxdoc.org/HOWTO/Wireless-HOWTO.html
- Document “Linux Cross-Development Environment”: http://www.ordina-net.nl/
- Linux HowTo’s: http://www.linuxdoc.org

3 Network Topology

A simple network consists of at least one server, one wireless LAN Access Point and one PDA or laptop computer. The wireless LAN Access Point is just another node in this network with its own IP-address. It has the same functionality as a HUB, but then for wireless networks.

The maximum throughput needed for DVD quality in a video stream is 8 Mbps. Over a static 100Mb network this is not a problem, even a 10Mb network will cope. A wireless network is however different. Theoretically it should be capable of 11Mbps, but in practices about half 5.5Mbps can be realised on a good day.

Wireless networks capacity and stability are subject to temperature, air humidity and other RF sources. All these factors will influence the maximum throughput of the network. RF shielding to block unwanted RF in the room and restricting the wireless devices to be within line of sight will help to get a full 5.5 Mbps throughput at all times. Keep this in mind when constructing a wireless network.

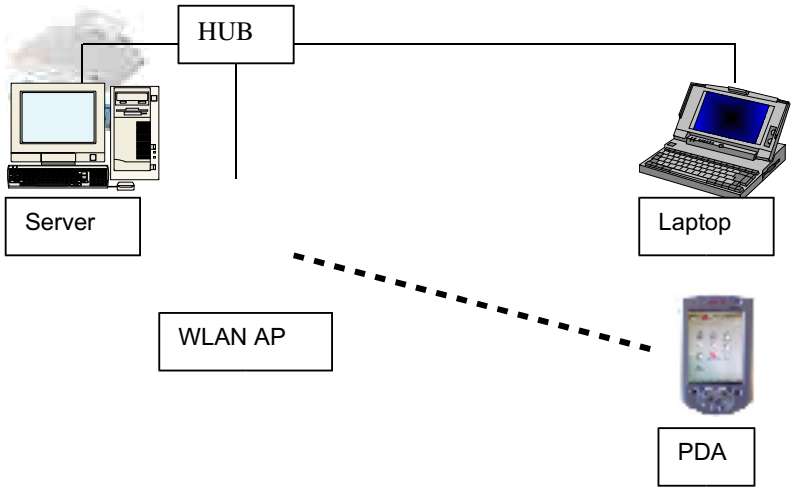

Figure 6: Network Topology Streaming Video

31 Network infrastructure

Most Access Points are capable of running as a DHCP (Dynamic Host Configuration Protocol) server. All wireless clients get a IP-address assigned upon registration at the access point. Security is another issue with wireless networking, because it is easy to listen in. Every single packet can be received either by other devices or access points. In fact by anyone who cares to listen. The Wireless LAN standard provides an encryption algorithm, but it is flawed and has already been broken. Although each AP has security settings like: MAC addresses only, Wireless Encryption Standard (either 40 or 128 bits). If security is wanted, then creating a VPN in a wireless network segment with secure socket layer (SSL) provides a better solution.

I won't explain how to configure the server and WLAN AP, because this is straight forward and documented extensively. See references of this chapter for some resources. Configuring a wireless device is quite similar to configuring a static network, with of course some differences: Access Point name, channel, mode (Ad-Hoc or Managed), bandwidth (1, 2, 5.5 or 11Mbps). In addition to configuring network parameters, it is also needed to give some wireless settings for communication. Read the manual thoroughly.

Choose IP-addresses for all nodes in the network. E.g.: Configure the server with IP-address 192.168.0.1, the wireless LAN Access Point get IP-address 192.168.0.254 and the PDA gets IP-address 192.168.0.10. Or use dynamic IP-addresses with DHCP.

Once the server and WLAN AP are configured and connected through UTP cables, then test if the server can ping the Access Point. Like this:

```
ping 192.168.0.254
```

The Wireless LAN Access Point should respond to this request. If it does respond, then go to the next step. That is configuring networking for the PDA.

32 Wireless Client

In the directory /etc/pcmcia on the PDA there are some scripts. The important files in this step are network.opts and wireless.opts. The first file describes network settings for a PCMCIA device, the second one describes wireless settings. Both are needed to configure a wireless client in such a way that it can register at a wireless access point.

The basic settings for /etc/pcmcia/network.opts are:

```
case "$ADDRESS" in
*,*,*,XX:XX:XX:*) # XX:XX:XX:* should be filled in with MAC-address
      IPADDRESS="192.168.0.10"
      IPDEV="eth0"
      GATEWAY="192.168.0.254"
;;
```

The basic settings for /etc/pcmcia/wireless.opts are:

```
case "$ADDRESS" in
*,*,*,XX:XX:XX:*) # XX:XX:XX:* should be filled in with MAC-address
      INFO="3Com AirConnect"
      ESSID="101"       # can also be "any"
      MODE="Managed"    # type of connection
                        # -"Managed" is through WLAN AP
                        # -"Ad-Hoc" is between two WLAN cards and no AP
      RATE="11M"        # transmission speed, can be 1,2,5,11M
;;
```

Restart the PDA without the wireless card inserted. Before testing verify that the WLAN AP is on and reachable from the server. Use the command "ping" for this. Then insert the wireless card

in the PDA's PCSMCIA jacket. The console should show messages including a './network eth0 start'.

Verify with ifconfig and iwconfig is the card is registered correctly:

- ifconfig should display two ethernet devices, i.e.: lo and eth0.
- iwconfig should display one devices that supports wireless extensions, i.e.: eth0. Check if the MAC address of the Access Point is different from 00:00:00:00. If it is, then the wireless card is recognised correctly.

The last test:

- First ping the WLAN AP:

```
ping 192.168.0.254
```

- And finally ping the server:

```
ping 192.168.0.1
```

If this last test is passed, then the network (wireless and static) is fully operational at the IP level. Now it is possible to use all network related tools and utilities (ssh, scp, ftp, telnet, browsers, etc.) on the PDA. The following step is getting a streaming server operational.

References

- RedHat Linux 7.2: http://www.redhat.com
- Familiar v0.4: http://familiar.handhelds.com
- 3Com AirConnect Access Point and WLAN cards: http://www.3com.com
- Linux Networking HowTo's: http://www.linuxdoc.org
- Jean Touriless Wireless LAN How To: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux
- Wireless HowTo: http://www.linuxdoc.org/HOWTO/Wireless-HOWTO.html

4 Streaming Video Solution

Streaming over a static wired network does not require the video stream to be resized to the receivers capabilities. Tweak the streams resolution to match the screen size of the receiver and lower picture quality. In a static network both sender and receiver usually have plenty of bandwidth and processing power available. In the wireless world however this is not automatically the case. In a previous chapter was told that a wireless 802.11b network has a maximum throughput of 5.5Mbps data in practise instead of the 11Mbps theoretically.

Today's handhelds have a StrongArm, which runs on a frequency of 206MHz and have limited disk and memory sizes. Streaming a DVD quality movie to such a receiver will not work. First of all a DVD easily occupies 8Mbps of bandwidth, which is not available on the wireless link. Secondly screen sizes are usually 240 by 320 pixels. Thus all video streams should be scaled down to limit screen size and bandwidth, requiring a lot of processor power.

The word **streaming** is usually used when people actually mean **broadcast** With broadcast a server pushes, or with another word broadcasts media (several video or audio streams) onto a network to multiple receivers Clients can tune into one stream and watch or listen to the program This is how our TV and radio network operates and is usually based on analog technology Also in digital technology a server can broadcast a stream to multiple receivers this is also called **multicast** Broadcasts to one receiver only is called a **unicast**

Streaming refers to a digital stream of bytes regardless which protocol is used It can either be server pushed or client requested, but is always network oriented Several protocols are in use To name a few:

- TS: transport stream protocol used in satellites and DVB
- PS: packet stream protocol used in DVD and MPEG
- RTP: real-time transmission protocol a IETF standard for real-time transmission over the internet
- HTTP: hypertext transmission protocol used in the world wide web also known as internet

A video server could provide already scaled down movies or do this dynamically. For our solution already scaled down movies are used. Here are some open source solutions to use as server and or scaling of MPEG streams:

VideoLAN is a project of students from the École Centrale Paris. Its main goals are MPEG and DVD playing and broadcasting on the campus, but it also features a standalone multimedia player that can read DVDs and MPEG files. It will also eventually support streams from a satellite or from an MPEG2 encoding card. The solution consists of a streaming video server and client. VideoLAN software runs on multiple operating systems, among which: Linux, BeOS, MacOSX, Windows. Our solution uses their software. Goto http://www.videolan.org/, for more information about this project.

FFmpeg is a complete and free Internet Live Audio and Video Broadcasting solution. It is a "three-in-one" solution.

- FFmpeg includes a soft VCR capable of encoding in many different formats simultaneously, a streaming server for Netcasting multimedia and is available under the GNU General Public License.
- FFmpeg generates streaming files, in many popular formats simultaneously, faster than any other solution.
- FFmpeg can stream live or pre-recorded content to start your own Internet Radio or TV station, or to turn your video camera into a video monitoring system.

FFmpeg is currently in the ALPHA stage of development and, although functional, still has bugs to be worked out. Goto http://ffmpeg.sourceforge.net for more information about this project.

41 Video Server

The VideoLAN project offers several solutions for streaming servers, but they are all still very much in development. There is a videolan server (vls), videolan channel server (vlcs) and a videolan mini server (vlms). Another alternative which is not part of videolan project is ffmpeg. It delivers a streaming server (ffserver) and mpeg encoder (ffmpeg). The simplest tool for our purpose was vlms.

411 VideoLAN Mini Server based streaming

The VideoLAN mini server streams mpeg files as TS packets to its destination address. This can either be a unicast (e.g.: 192.168.0.10) or multicast (192.168.255) addresses.

An example of unicast:

```
vlms -d 192.168.0.10:1234 your.mpg
```

or multicast:

```
vlms -d 192.168.0.255:1234 your.mpg
```

Depending on the Access Point and wireless driver multicast works or not. In our setup it did not work well. So unicast was used.

412 Webserver based streaming

The simplest solution that gave very good results is to use a webserver based video streaming. On the Linux server the apache webserver was configured to export our mpeg video directory. Look into apache documentation or on the internet how to configure the server correctly.

The benefit of this solution is that the server does not convert a video file in another protocol stream, thus eliminating the chance of errors as much as possible. A webserver sends the file as is to the client (basically a ftp download). Luckily our video client accepts http-based input; use it like this:

```
vlc -V x11 http://192.168.0.1/video/your.mpg
```

413 Videolan Server based streaming

The videolan server is still very much in development and is harder to setup and to configure, than its smaller brother vlms. It uses a file called “vls.cfg” for channel configuration and a file called “input.cfg” for declarations of video streams.

The Videolan server can capture streams from DVB devices, TV tuners or other incoming video data. Also DVD and files can be used as input for the server. The Videolan Server uses the Transport Stream protocol for streaming over IP.

Telnet can be used to connect to videolan his configuration interface on port 9999. From within that telnet session streams can be started and stopped.

```
telnet localhost 9999
```

The file “vls.cfg” lists two default usernames and passwords. Use the administrator account “bozo” to login on the telnet session. Change the default settings when you are going to deploy this solution onto a network. The start command lets you start a stream.

42 Video Client

The VideoLAN client must be compiled (ported) for a StrongARM cpu-architecture. There are two choices:

- Cross-compilation (remember cross-compilation of Linux kernel)
- Native compilation (compile on an ARM-based computer)

The disadvantage of cross-compilation for user-land applications is, that it requires cross-compilation of all user-land libraries needed by your application. Usually these libraries are not easy to configure for cross-compilation and order does matter a lot here. It is easy to make mistakes in these steps. Therefor doing a native compile is much easier (and preferable).

We choose “native compilation”. Compaq offers a free online service for native compilation through a compile farm. All cpu-architectures Compaq builds are online accessible for this purpose. So is there an ARM-cluster called skiffcluster[1-5].handhelds.org. On the website of handhelds.org information about this can be found. In short, this is how to login on the cluster through telnet:

```
telnet skiffcluster3.handhelds.org
```

It asks for a username and password. The username is “guest” and password is empty, just press return. There are some rules to obey. The most important ones being:

- All your work should be done within a subdirectory that has the name of your e-mail address. In this way the system administrators can send you a mail if necessary.
- Cleanup your directory when you’re done.

Extract the latest vlc source within this directory, then cd into it and type:

```
make distclean;
./configure -prefix=/usr -enable-x11 -enable-gtk \
            -enable-fb -enable-dsp
```

And finally to compile vlc:

```
make
```

When the compile is successfully completed, tar, zip and download the vlc directory to your desktop computer. Use either ftp or wget:

```
wget ftp://guest:@skiffcluster3.handhelds.org/e-mail@address/vlc-x.x.x-ipaq.tgz
```

Copy the files: vlc, qvlc, (go in directory "/usr/bin") plugins/*.so (go in "/usr/lib/vidoelan/vlc") to the iPaq. Add to /etc/ld.so.conf the videolan directory "/usr/lib/vidoelan/vlc" and run /sbin/ldconfig to update the shared library cache.

Or (if available) use the iPaq binary distribution of vlc. At this moment of writing the VideoLAN team is busy porting vlc to the iPaq. See the online information at: http://www.videolan.org/vlc/familiar.html for the current status.

43 Video content

A good quality video stream uses about 5-8 Mbps bandwidth on any network. DVD quality streams require 7-8 Mbps. For a static network of 100 Mb this is usually no problem, but a wireless network is subjected to much more influences (weather, radio interference, etc.). A PDA does not need this high quality video, a stream of 1 Mbps is more then enough.

Handhelds are less powerful then today's desktop computers and wireless networks are subjected to many more influences then a static network. The video streams needs to be converted to a lower quality in order to compensate for these other network characteristics. Tools can help in this step. A nice tool is FlaskMPEG (http://flaskmpeg.sourceforge.net) but it runs on Windows only. Another one is ffmpeg (http://ffmpeg.sourceforge.net) that also run on Linux, but cannot convert all streams currently. It use is targeted at live streams from a TV tuner card or another video device (/dev/videodev).

An example with ffmpeg: (Beware ffmpeg does not do MPEG2 Video encoding yet)

```
ffmpeg -i your.mpg      // input bestandsnaam
       -g 3             // closed grops (small is better for streaming)
       -r 25            // frame rate
       -b 750           // video bit rate
       -s 160x128       // screen size WxH
       -ar 44100        // audio frame rate
       -ac 2            // audio channels (2 = stereo)
       -ab 64           // audio bit rate
```

```
-qscale 15       // quality scaling: 1 (excelent)
                 // and 31 (worst)
your160x128@750.mpeg // output filename
```

At the moment I suggest to use FlaskMPEG for MPEG decoding and encoding. There is enough documentation for that tool, so I won't go in-depth here. Instead go to the website: http://flaskmpeg.sourceforge.net.

References

- VideoLAN: http://www.videolan.org
- Handhelds.org: http://www.handhelds.org
- ARM compile cluster: http://skiffcluster.handhelds.org
- FFMPEG streaming server: http://ffmpeg.sourceforge.net
- FlaskMPEG: http://flaskmpeg.sourceforge.net
- Apache webserver: http://www.apache.org
- Linux Documentation Project: http://www.linuxdoc.org

5 Putting it all together

In previous chapters you learned about flashing Linux on a PDA, cross-compiling an ARM/Linux kernel with wireless drivers, configuring a streaming server, configuring wireless LAN on a PDA, compiling a video client for a StrongARM cpu and to scale down a video sample. Here it all will be combined to a streaming video solution that is ready for the big test.

Make sure a webserver is running and the video directory is accessible. This can be verified with any browser. If unsure try to ping every node in the network. Get the PDA, insert the wireless card and check if the network is up and running. And then it is video time …, run the following command on the PDA:

```
vlc -V x11 http://192.168.0.1/video/your.mpg
```

Sit back and enjoy your own streaming video solution with Linux.

51 Lessons learned

Using experimental pieces of software gives problems and so it did. Three things were experimental in the implementation: ARM/Linux kernel, Wireless LAN device drivers and the VideoLAN client on StrongARM.

Without the support and fast paced development in the Open Source community it wouldn't have been possible to tackle all these problems in the short time available. For us using Open Source software was a big advantage. Giving back patches and feedback to the Open Source community was appreciated.

511 Embedded Linux Development

Developing for embedded Linux is not much different then for normal Linux. All tools and applications are the same. There are only limitations in disk and memory size. Using a Linux development system is highly recommended. The Linux kernel works the same and loading kernel modules works the same. For applications the Linux system is transparent, so it is then possible to develop and debug on the development system. After cross-compilation the applications behaves the same on the embedded device as on the desktop.

Installing Linux on a handheld is not that difficult, but is not something for normal users. This applies to any handheld OS. That is why PDA are shipped with an operating system pre-installed.

Using Linux on a handheld is feasible and also usable for non-technical people. Linux distributions with a good usability factor are QT Palmtop/Embedded, Century Embedded (http://embedded.centurysoftware.com/pixil/pixiloe.php) and Yopy (http://www.gmate.co.kr). Also Linux-only PDA's are available for instance the Yopy from Gmate. The Yopy uses a micowindows (http://www.microwindows.org) user interface. Another microwindows based distribution is from Century Embedded

512 User Interface

A PDA gives more restrictions on the user interface then a desktop. For starters it's small screen size, usually a quarter VGA sized screen (240x320 pixels), limits the amount of screen assets that can be used. It input method, a touch screen manipulated with a stylus pen also occupies screen assets.

Web pages from most sites do not show correctly or not at all on a PDA. One of the problems is the small screen size. Another is the extensive use of frames, Java and JavaScript. All Java VM implementations we tested on our PDA are too slow to be usable.

Each multimedia gateway that provides a web service should translate web pages to formats usable on a small screen of a PDA or cell phone. Such a service should strip out or convert all not usable features like: frames, Java, JavaScript into a format that is usable.

513 Wireless LAN

Developments of the Linux kernel go in a fast pace, especially if the official stable kernel is tracked. For this project the ARM/Linux kernel had to be used. During the demo project there were major rewrites for device drivers in the ARM/Linux kernel. This did give some problems, but it was more inconvenient then a show stopper. At all times it was still possible to use the kernel and device drivers.

Setting up the Wireless LAN link between a laptop in first instance and a handheld later gave us plenty of problems. I already hinted at that in chapter 3. We had to start a bug hunt in Wireless LAN device drivers to fix problems with our WLAN card. It took us more then a week to track down and fix this problem. Eventually we upgraded to a newer driver, called "Orinoco", that supported our card better.

During testing it became apparent that also the Orinoco driver had a problem with our 3Com AirConnect cards, but worked better then the other driver. Therefore a small patch of the driver was needed. There were some portability problems, because of other data structure alignment on ARM architecture and some retries were needed. The chapter "Creating a patch" describes how to create a patch and lists the patch used in this implementation. Restarting the PCMCIA manager on the handheld is not enough it does not re-read its configuration files. Stopping it will print errors on the console. Instead simply reboot the handheld with /sbin/reboot to be sure.

The ARM/Linux kernel is currently so far evolved, that it is going to be integrated in the next major stable kernel release. This means that during the development kernel 2.5 series the ARM architecture will be integrated. Also Wireless LAN support is maturing and going into mainstream use. Without the fast pace of development in the Open Source community it wouldn't have be possible to create this demo in the time we did and with the limited man power we had.

Availability of source code on one platform, does not guarantee an easy migration to another platform. Hardware specific differences will always bite you. Our video client used decoders that uses floating point calculations quite extensively. For Intel systems this is not a problem, because this is done in hardware. For StrongARM based systems this is a problem, because it lacks hardware support for floating points calculations.

Scaling down the video content to a dimension that can be shown on a limited screen of 240 by 320 pixels and with a lower quality did further optimised our use of available network bandwidth. This results in seeing video and not hearing any audio. Another solution was needed.

Most audio and video decoder software use floating point calculations. On a cpu that has no hardware support for floating point these calculations will be done in software only. This comes with a big performance hit resulting in bad audio and video. The solution is to use other decoders that do not use floating points but integer based algorithms for decoding streaming video. With any other operating system the same problem would have come up.

The Videolan Client is extended with a plugin called mad that supports the libmad audio decoder library. The libmad library decodes an mpeg audio elementary stream with an integer algorithm. Together with Internet based streaming video the integer-based decoder contributes to the stability of the demo.

References

- VideoLAN: http://www.videolan.org
- Handhelds.org: http://www.handhelds.org
- ARM compile cluster: http://skiffcluster.handhelds.org
- FFMPEG streaming server: http://ffmpeg.sourceforge.net
- FlaskMPEG: http://flaskmpeg.sourceforge.net
- Apache webserver: http://www.apache.org

APPENDIX: Technology explained

A 1 IEEE 802 11 for Wireless LAN

WLANs typically cover distances from ten to a few hundred meters. This smaller coverage distance allows lower power transmissions that often permit the use of unlicensed frequency bands. Because LANs often are used for relatively high-capacity data communications, they often have fairly high data rates. IEEE 802.11, for example, has a nominal range of 100 meters and data rates up to 11 Mbps. The types of devices normally used with WLANs are ones that have a robust computing platform and power supply; notebook computers, in particular.

IEEE 802.11 technology is being deployed widely for WLAN applications. WLANs permit a degree of mobility and enhance convenience by eliminating the wires used to connect computers to networks. WLANs are increasingly being installed in businesses, shops, airport lounges, homes, and other venues; in nearly all of these cases, IEEE 802.11 is the technology of choice.

IEEE 802.11 is a popular WLAN technology because it was designed for just this purpose. As indicated above, WLANs usually replace or augment existing wired LANs; hence, the devices typically used with WLANs are the same ones used on wired LANs: servers, desktop computers, and notebook computers. For the same reasons, WLAN applications are usually identical to those of their wired counterparts: e-mail, Web browsing, and others that rely on standard Internet-based protocols and technologies.
Earlier we noted that typical WLAN applications consume a fair amount of electrical power for their relatively high data rate, and medium-range transmissions. Portable devices like notebook computers can supply this energy and still maintain reasonable battery life. But smaller portable devices like handheld computers, pagers, and mobile phones can't sustain the necessary power draw for long periods of time without the auxiliary power supplies that make the devices less mobile and convenient.

Several characteristics of WLAN technology, namely IEEE 802.11 technologies are:

- Range: The reach is about 100 meters.
- Raw data rate: 11 Mbps
- Power consumption: is significant for handheld devices, auxiliary power needed for longer duration of use.
- Packet technology: IEEE 802.11 heavily leverages Internet technologies for "Ethernet-style" data networking.
- Spectrum band: 2.4 GHz

The 2.4 GHz spectrum, is unlicensed virtually worldwide. Hence, it is an attractive band for wireless communication technologies that can operate within the constraints imposed by the spectrum. Among the benefits are limits on the transmission power and measures required to deal with RF interference. Also other wireless communications take place inside this band, namely Bluetooth, HomeRF, some cordless phones, and even microwave ovens. It seems that all these devices must hinder each other when operating, but in reality this is not a big problem. A

primary consideration for any technology that operates in this band is that it must expect RF interference and implement measures to deal with it. Performance degradation often is an effect of RF interference.

References

- Wireless Lan and Bluetooth compared: http://www-106.ibm.com/developerworks/wireless/library/wi-phone/?t=gr,p=WirelessBuds
- IEEE: http://www.ieee.org
- Bluetooth: http://www.bluetooth.org

A 2 Open Source

The most widely known Open Source license is called the GPL. It is however not the only license there is e.g: Mozilla Public License, BSD license, Artistic license. See the website of the Free Software Foundation http://www.fsf.org and http://www.opensource.org for more information about Open Source Licences.

The GPL is used in the Linux kernel and in Open Source applications. In general every change to the kernels source code should be send back to the original author. Redistributing a modified kernel requires also the source code to be redistribution. There are some special cases related to drivers. For hardware manufactures this is the most important part of this license.

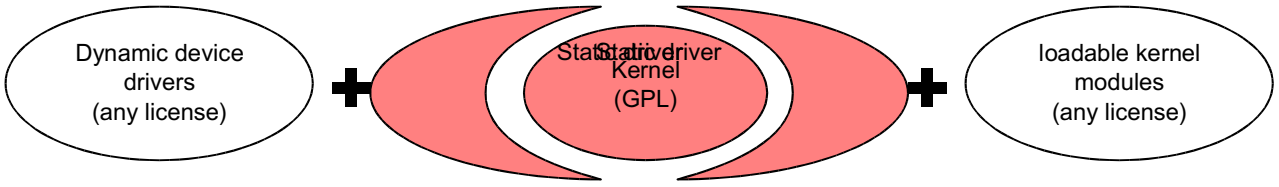

Figure 7: License status of loadable kernel modules

Drivers that are loaded after boot-time are allowed to be binary only distributed. As long as it uses the official driver module interface. The cards firmware is not thought to be part of a the driver and can thus contain Intellectual Property. In the later 2.4.x kernels a driver can be distributed under a dual license. Providing more flexibility in writing drivers for card manufacturers.

The GPL obligation:

- If GPL code is modified and distribute the results, then it is required that:
  - Source code changes are made available on request to customers receiving the (re) distribution.
  - Pass on the GPL license with full rights to distribute, modify and rebuild the source.
- Internal distribution of code is not seen as re-distribution.
- Device drivers written as loadable kernel modules remain proprietary. Releasing under the GPL however helps keeping them up-to date with the kernel and helps fixing bugs.

The libraries however are almost always released under another Open Source license called the Lesser or Library GNU Public License (LGPL). In contrast to what most people would like you to believe an application that uses a library released under the LGPL is not automatically subjected to the same license. Or with other words the application DOES NOT BECOME OPEN SOURCE. The application can still be released binary only. In this way a vendors Intellectual Property is still protected.

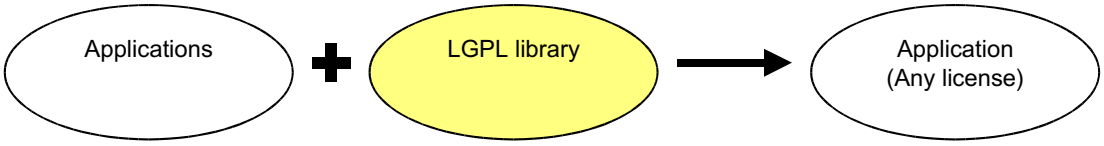

Figure 8: Application binding to LGPL library then any license possible

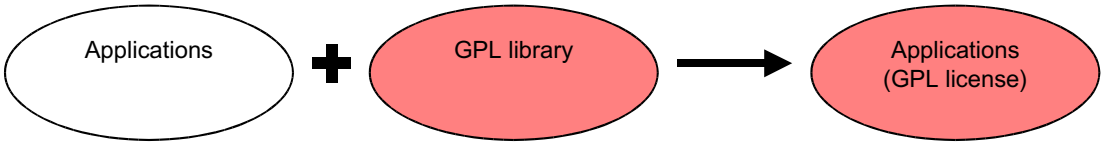

Figure 9: Application binding to GPL library then only GPL license possible

LGPL in short:

- Applications with no GPL code and not linking to any GPL-based libraries remain proprietary even when running on Linux
- Distribution of proprietary applications on the same media as Linux does not cause the application to become contaminated.
- Applications linking to LGPL libraries (such as the standard GNU C library) remain proprietary as well

References

- Free Software Foundation: http://www.fsf.org
- Open Source Organisation: http://www.opensource.org
- GPL and other Open Source licenses: http://www.gnu.org/philosophy/license-list.html
- Open Source projects: http://www.sourceforge.net

A 3 Building a cross-compiler

The ARM/Linux website contains documentation for building a cross-compiler and where ARM toolchain can be found. An online book called “A RM Linux” can be found at: http://www.aleph1.co.uk/armlinux/thebook.html. It contains an extract.

References

- ARM/Linux toolchains/cross-compiler: http://www.arm.linux.org.uk/developer/tools.shtml
- Latest Binutils, gcc and glibc for building your own cross-compiler: http://www.gnu.org

- Pre-compiled cross-compile toolchains: http://www.handhelds.org and http://www.arm.linux.org.uk/

A 4 Creating a patch

Changes to the official Linux kernel are best kept, when a patch is created. In the Documentation directory of the kernel a file called “SubmittingPatches” explains the process. I’ll tell it here in short:

```
wget http://www.moses.uklinux.net/patches/dontdiff
diff -urN -X dontdiff linux-2.4.6-rmk1-np2 246ipaq-linux > patch-2.4.6-
rmk1-np2-jps1
```

Cut and paste the patch below into a separate file and call it patch-2.4.6-rmk1-np2-jps1. Copy the file into the same directory as the kernel files are. Next patch the kernel with:

```
patch -p1 < patch-2.4.6-rmk1-np2-jps1
```

The patch itself:

```
diff -urN orinoco-0.08a/hermes.c orinoco-0.08a-jps1/hermes.c
--- orinoco-0.08a/hermes.c Tue Oct  9 07:48:25 2001
+++ orinoco-0.08a-jps1/hermes.c   Sun Oct 21 13:21:23 2001
@@ -99,9 +99,23 @@
 static int hermes_issue_cmd(hermes_t *hw, uint16_t cmd, uint16_t param0)
 {
       uint16_t reg;
+      int k;

       /* First check that the command register is not busy */
       reg = hermes_read_regn(hw, CMD);
+
+      /* If it is busy give it a second chance, this works better for
+       * 3Com AirConnect version 1.00 and 2.00 of the firmware on StrongArm
+       * It's needed on card insertion (especially when an iPaq backsleeve is
+       * low on power) and with special commands iwconfig, iwpriv..
+       */
+      k = CMD_BUSY_TIMEOUT;
+      while (k && (reg & HERMES_CMD_BUSY)) {
+            k--;
+            udelay(1);
+            reg = hermes_read_regn(hw, CMD);
+      }
+      DEBUG(3, "hermes_issue_cmd: did %d retries.\n", (CMD_BUSY_TIMEOUT-k));
       if (reg & HERMES_CMD_BUSY) {
             return -EBUSY;
       }
@@ -325,7 +339,7 @@

       k = BAP_BUSY_TIMEOUT;
       reg = hermes_read_reg(hw, oreg);
-      while ((reg & HERMES_OFFSET_BUSY) & k) {
+      while ((reg & HERMES_OFFSET_BUSY) && k) {
             k--;
             udelay(1);
             reg = hermes_read_reg(hw, oreg);
```