Hardware Evaluation Indoor Testbed

Doris Reim
(doris@net.t-labs.tu-berlin.de)

Project FG INET,
Technische Universität Berlin

Abstract

The MagNets testbed aims at deploying a next-generation high-speed wireless network access and research infrastructure at the campus of TU Berlin. This student work describes the process of finding and evaluating available wireless embedded hardware, finding usable software, building the indoor testbed, and creating centralized maintenance processes. Further it gives a short description of an early experiment on MagNets, TV-streaming over mesh networks and its results.
Contents

1 Introduction – The MagNets Project

2 The Different Testbeds
   2.1 Server Room Testbed – Smoketest
   2.2 Indoor Testbed
   2.3 Outdoor Testbed
   2.4 Backbone

3 Available Hardware and Software
   3.1 Requirements
      3.1.1 Access To Hard- And Software For Experiments
      3.1.2 Outdoor Requirements
      3.1.3 Indoor Requirements
      3.1.4 Computing Power And Storage
      3.1.5 Multiple Radios
      3.1.6 Software
   3.2 Reality Check: Available Hardware
   3.3 Available Software

4 Evaluation And Comparison Of Single Wireless Routers
   4.1 Mikrotik Routerboard
   4.2 Avila
      4.2.1 Wired measurements
      4.2.2 wireless measurements
   4.3 Asus

5 Using The Indoor Testbed With Mikrotik Routerboard
   5.1 OpenWRT on Mikrotik Routerboard
   5.2 Replacing The Firmware For Indoor Testbed
   5.3 Building Preconfigured Images For The Indoor Testbed
   5.4 Saving The Current Testbed Setup Configuration And Replaying A Setup
      Configuration

6 State Of the MagNets Project And Outlook
1 Introduction – The MagNets Project

"The MagNets testbed aims at deploying a next-generation high-speed wireless access infrastructure in the city of Berlin. The network is designed as a wireless access network supported by an operator to perform research, but access is given for free to the students of the Technical University of Berlin to create a semi-productive environment. Moreover, a key feature characteristic of MagNets is heterogeneity along several dimensions: nodes in the network featuring multiple wireless interfaces with different technologies, such as 802.11, FlashOFDM, 802.16, UMTS and BlueTooth; diverse link characteristics; nodes with varying degrees of processing and storage capabilities; interconnection of multiple mesh networks with disparate routing protocols.” [Mag2007]

The goal of the part of MagNets presented here, was to find and evaluate the right hardware for MagNets, build up the Indoor testbed and make it controllable in a centralized and comfortable way.

This work describes the problems and some solutions we found while evaluating the hardware and software to build the different testbeds.

The second chapter briefly introduces the testbeds and their goals. In the third chapter we define our requirements and have a look at reality, what is available and how we chose the different parts. In the fourth chapter we compare the performance of three different hardware types. The fifth chapter gives an introduction the functionalities of the indoor testbed and how to use this testbed. The sixth chapter gives the status of the indoor testbed and a roughly look at possible future work. The appendix is a bonus. It shows the results of one of the first experiments we already did on the testbeds.

2 The Different Testbeds

MagNets consists of several different testbeds, each with a certain purpose. These are described in the following chapter. The indoor testbed is also described in detail in Chapter 5.

2.1 Server Room Testbed – Smoketest

As the first step of doing experiments or testing hardware there is a small environment in the server room. In this testbed we have a serial terminal server and a PoE switch. So any software or hardware setup can be tested without the need to climb up a roof, when the setup is crashing. No default setups for the machines in smoketest are provided since it is used for arbitrary tests by the users.

2.2 Indoor Testbed

In the indoor testbed we want to test things which are known not to crash (by testing it in smoketest for example); crashing in the sense that starting a programme leads to an instant kernel freeze. So there we can do more complex tests to find bugs before we bring them into an outdoor environment where it can be hard to reach the node again. The indoor testbed is deployed at the 16th and 17th floor in the Telefunken building. It consists of nine nodes, each with two routers, one for routing the second for monitoring. The idea of an indoor testbed is to have more than a simulation, a realistic testbed with full control. That means the outdoor testbed will be semi-productive, so for example we can not swamp the mesh with artificial traffic to measure the layer 2 connectivity. In the indoor testbed we already did such measurements.
2.3 Outdoor Testbed

The goal of the whole MagNets Project is to ultimately have a mesh network in a real environment, usable by real users. The outdoor testbed will form a mesh which is accessible by TUB students. It is a big part of the MagNets project. But we will not go deeper into the details of it, because it is not in the scope of this work.

2.4 Backbone

The first MagNets testbed part was built as a backbone with directed antennas. "the high-speed WiFi backbone connects 5 high-rise buildings in the heart of Berlin. The backbone is composed of 6 PC based routers and 12 Access Points (AP) (10 indoor and 2 outdoor). The APs consist of Intel IXP420@266 MHz (indoor) and IXP425@533 MHz (outdoor) programmable network processors (NP) as CPU, and Atheros 5213/5112 chipset for their WLAN interfaces, and run a proprietary operating system called LC.OS." [IPC 07]

More Information about the Backbone can be found in [Mag 07]

3 Available Hardware and Software

To come as close as possible to the goals of the whole project, e.g. doing experiments in a productive mesh network with real users or experiments in a controlled environment like the indoor testbed, we had several requirements on the hardware and on the software. For this project we have a limited budget, so we are restricted to a few hundred Euros per node, which must be taken into account when defining the requirements. This chapter will describe these requirements and have a look at some of the available hardware and software and how many of the requirements they meet.

3.1 Requirements

We want to use the testbeds to develop, implement, and test new or just other than the common protocols. Along with that we also want to combine different protocols and technologies.

The next sections describe the requirements that follow from the planned abilities the testbeds shall provide.

3.1.1 Access To Hard- And Software For Experiments

In any experiments with networking and especially in the area of wireless experiments, access to the network stack is often needed. This access is in many operating systems restricted to few parameters like IP addresses, routing strategies or frequency of the Wifi cards. This behavior perfectly fits the needs of a private customer or the administrator of a company network, but not those of a research network. Wireless routers are often sold with preinstalled proprietary firmware that shows the above restrictions. Further we may even need to change or rewrite the drivers for the hardware. With such firmware it is impossible to do this.

Manufacturers of such hardware often have the policy that their architecture is a company secret and do not disclose the specifications.

To avoid these problems we use open source software and hardware with open specifications.
3.1.2 Outdoor Requirements

Running any hardware outdoors leads to several problems. First of all the hardware must be weatherproof. Outdoors, it is also often a problem to get a good power connection, so there should be the possibility of a Power over Ethernet (PoE) supply and low power consumption.

Not every location is easy to reach, so maintenance should be possible remotely. This remote work includes the possibility to reboot the hardware from an undefined state e.g. after a kernel freeze. So it might even be necessary to do changes on the Bios, or to have a serial interface.

Further the hardware should come without moving parts like a cooler with fan because such parts tend to break very often which would cause additional maintenance.

Outdoors there is sometimes limited space, so the ideal would be a very small embedded device.

3.1.3 Indoor Requirements

Indoor deployment means in our case that we want to setup the testbed at the office floors in certain distances to avoid interference and to simulate the real world. Using the same hardware indoors and outdoors enables the indoor testbed to be used as an outdoor simulation. So the indoor testbed can be seen as the first step to the outdoor deployment, but under easier conditions.

The outdoor requirements call for a small embedded device without mechanical cooler, remote controllable and easy to power up. In an office it must also be silent. The loudest part on any PC is usually the cooler. As we do not allow mechanical coolers this problem is in line with the outdoor requirements.

3.1.4 Computing Power And Storage

At first sight, computing power does not seem to be an important issue. Embedded hardware usually has slow CPUs. The tasks of such machines are often very simple or small compared to desktop PCs. Small wireless routers are not designed to do major calculations or to intensively measure traffic. In all of the testbeds we want to run our own software – which is often only an unoptimized prototype – and we want to do measurement as well as tests under difficult conditions and stress tests, e.g. run an experimental routing protocol that produces a high CPU load only with calculating its forwarding tables.

Therefore there is the need for fast CPUs even on the small routers.

Doing measurements sometimes leads to the situation that a part of the collected data must be stored on the router before it can be transmitted to a server. For this we need additional storage capacities, such as a USB storage or an additional compact flash storage.

3.1.5 Multiple Radios

We want the outdoor nodes to be access points and mesh nodes at once with different frequencies for each purpose. Later we also want to add other radio technologies, like UMTS cards. We want to find out how these things can be combined, what the behavior is in terms of interference, protocols, and so on.

To do these things we need the possibility to add more than two cards to one router.

3.1.6 Software

As said before one of the goals of the MagNets project is to test new protocols and methods in wireless networks and for this we sometimes need to do changes in the kernel land. An
open source OS would be the best solution, and to foreclose that, we decided to use a Linux based OS. Among all common open source OSs it is the one that is most widespread and readily available for embedded systems. This leads to the requirement that any hardware we choose must be supported by Linux. As we want to do research and use some new technologies – e.g. newest chipsets in Wifi cards – the hardware must be supported by the 2.6 kernel version, which is the latest version and features the most comprehensive hardware support. Any hardware running Linux but only version 2.4 would not fit.

Which Linux or Linux-like software to use also depends on the possibilities for cross-compiling. Embedded systems usually do not have enough power or memory to compile software. Thus we must be able to create a complete system (image) offline and flash it remotely to a wireless node.

Further the software must fit an embedded hardware, so it must provide full functionality even for very restricted storage, e.g. a complete glibc would in some cases need the entire available flash-storage.

### 3.2 Reality Check: Available Hardware

There are many wireless routers to buy, e.g. all those small and cheap devices for small home networks, like Linksys WR T series, Asus, and so on. For many of them exist possibilities to install a Linux OS. The reasons not to choose them as the main testbed hardware are very similar for most of these small embedded devices, as we will see below. Their purpose is to provide all necessary functionality to small networks like access points and gateways to a LAN. These devices have limited memory and CPU power and often very little storage. Many of them neither have the possibility to exchange the Wifi cards nor to add any cards.

Examples are the Linksys WRT54G series or the Asus WL-500g. The WRT54GL is a widely-used, cheap wireless router with Broadcom architecture, one integrated Broadcom WiFi card, onboard flash between 4 and 16MB and up to 32MB memory. It is easy to see that this device, though it has some Linux support, does not meet our requirements. There is no possibility to add or exchange WiFi cards and the integrated Broadcom card has only binary drivers for Linux kernel version 2.4.

The Asus WL-500g is also currently a popular wireless router. It has similar features as the Linksys routers and two additional USB ports. They also have the WiFi card not integrated but in a mini-PCI slot, so the card can be exchanged. This sounds to be quite near to many requirements, but there are some issues. Asus routers have only one mini-PCI slot, the requirement is two or more. They have a Broadcom 300 MHz MIPS32 processor which is quite little processing power. Linux 2.6 is known to run unstable on this hardware.

This Asus model has some features which make it a good machine for small jobs inside the testbed like passive monitoring. This router will be evaluated in more detail in Chapter 6.

A really good solution are software radios, which provide full access to all layers of the network stack. Any imaginable experiment would be possible with this kind of machine. However, there are drawbacks. The first disadvantage of a software radio is, that they need extensive power, CPU capacity and strong coolers. The second issue is, it is already possible to buy them, but they are that expensive that we could only buy a few of them for this project. As described we want to build up larger mesh networks and therefore buy up to 100 and more nodes. Respecting the budget we had to find a solution that is between the software radios and other real expensive wireless routers, and the small, cheap home network routers.

The named small routers are mostly in the range of 60 to 100 Euros. Better routers than these start at 100 Euro but come without cases, Wifi cards and other accessories. Some of these boards have more than two mini-PCI slots and provide the possibility to add more flash storage, which makes them more flexible. Usually they also have more
RAM than the small routers. Most of them have RS-232 serial ports which helps a lot when testing experimental software. They have very different architectures from Mipsel (Mikrotik) to Intel x86 (Wrapboard) and Intel XScale (Avila). Mipsel architectures for example are not that powerful.

At the beginning of the project we already had some Mikrotik Routerboards, namely Routerboard 532a. They come with 128MB onboard NAND-flash, a CF slot and 2 onboard mini-PCI slots. The Routerboards have a Mipsel architecture which was clear to not be very powerful, but they provide as a big plus the possibility to add so-called small daughterboards with up to four additional mini-PCI slots. Routerboards have a proprietary bootloader and the Mikrotik Router OS. A 2.4 kernel for the boards is provided by Mikrotik but no 2.6. So the question was if we could get a 2.6 Linux variation working. During the hardware evaluation process we also tested the Wrapboard. At that time it was sold as a complete package with case and antennas by Saxnet at a price of more than 500 Euro. The Wrapboard is an x86 architecture, it has two mini-PCI slots without any possibility to expand. The preinstalled OS is Linux Debian. The possibility to install plain Linux distributions is a big advantage. Disadvantages are the high price (avoidable by buying the board and all parts single on different shops) and that the board is not extensible.

The market for such wireless routers is constantly changing. The available boards are getting better, faster, and also cheaper. In 2007 the Gateworks Avila boards appeared. Avila has an Intel XScale processor and four mini-PCI slots onboard. They can be ordered with extended memory. Additionally they are very open. Avilas have the redboot bootloader and are supported by ddWRT and therefore usable with Linux. So it is possible to choose any imaginable system and way to boot.

In Chapter 4.2 we will have a closer look on the Mitkrotik Routerboard and Gateworks Avila. We deployed Routerboards in the Indoor testbed. When the outdoor part started, Avilas have already been available and are deployed there. The Indoor testbed will be equipped with Avilas as well in the future but is now running on Routerboards.

3.3 Available Software

Above we said what the requirements for the software are and decided to use a Linux-like System. This chapter gives a small overview on the available systems and their pros and cons. The experiences which are described here were all made with the Mikrotik Routerboard. But at the time we had to choose a system we needed to get it running on the Mikrotik hardware first which had a 2.4 Kernel image downloadable from Mikrotik but no further support for 2.6 kernels.

- Mikrotik Router OS:
  The Mikrotik Router OS seems to have a Linux based system inside. The appearance of the Router OS resembles Cisco’s IOS, but provides limited functionality.
  It is a proprietary and closed system, so it is not possible to add functionality.

- VxWorks:
  This is an OS especially for small embedded systems like wireless routers. VxWorks is known to be a good OS and to have a somewhat good performance. The disadvantages are, it is not free and very expensive. Because of that it was never thought to be part of the project and not tested.

- DDWRT:
  DDWRT is an opensource project basing on Linux providing a toolchain and also ready firmware images for several wireless routers and other embedded hardware like PS2. We did not test DDWRT then because at that time they did not support Mikrotik Routerboard.
OpenWRT:
"OpenWrt is described as a Linux distribution for embedded devices. Instead of trying to create a single, static firmware, OpenWrt provides a fully writable file system with package management. This frees you from the application selection and configuration provided by the vendor and allows you to customize the device through the use of packages to suit any application. For developer, OpenWrt is the framework to build an application without having to build a complete firmware around it;" [OWRT]
OpenWrt is open source software under GPL2 and includes the uclibc. Support for the Routerboards is only available for the still not stable Kamikaze release. That means working on a permanently changing developer trunk. But most of the time it works out of the box and image building is quite comfortable. The system is open and so it allows to include own software. We decided to use this system, because it is Linux based, it includes the uclibc, it is a build kit that allows adding any software and at last it was the first one that supported Mikrotik routerboard with a 2.6 Kernel.
More about how to work with OpenWRT in Chapter 5.

Plain Linux:
We wanted to use Linux and the distributions of Linux provide different cross-toolchains. So why not just use plain Linux with uclibc?
In the beginning we tried. The first was Gentoo which brings a good cross-toolchain (crossdev) with many options to support a lot of different hardware. Also the uclibc is included. At that time there was a cross-compiling bug in gentoo with some parts of the libc. We tried to get that working for about one week and then gave up.
The second thing to try was Debian with the cross-toolchain Debian supports. This is also a well working toolchain, but the support for Mipsel was apparently so me the years ago.
There was finally a Linux 2.6 kernel prepared for Mikrotik Routerboard by a user of the Mikrotik webforum. We focused on that, but it turned out that this needed more work. When we managed to get it running and to boot over Ethernet about two weeks later OpenWRT introduced an out of the box working version for the Routerboards.

After evaluating all these OSs we decided to choose OpenWRT. Though it is sometimes not stable, it provides good support for all our hardware. We established good contact with a few of the developers, which prove quite valuable, as we can get direct feedback for problems and can discuss future features in the system.
4 Evaluation And Comparison Of Single Wireless Routers

In this chapter we give more details about the hardware we have chosen for the project and compare them.

To get an idea what the different architectures can achieve, we did a measurement test on Mikrotik Routerboard, Gateworks Avila and Asus.

*original 64MB

Testsetup:
The router is connected to two other PCs, each over an own Ethernet interface. Between the interfaces we switch on routing with a fixed forwarding table. One of the PCs works as server and one as client. Then we run iperf over this setup in the following way: We are measuring two minutes on certain bandwidths, e.g. 1MBits/sec, 10MBits/sec, 60MBits/sec, 100MBits/sec starting at 40 Bytes packet size UDP traffic. The packet size is every time increased up to the maximum of 1400 Bytes. Each run is repeated three times. To examine the impact of the traffic on memory and interrupt load, we only execute a tool that measures the CPU load [CPUSAGE].

4.1 Mikrotik Routerboard

As said before when starting with the indoor testbed we already had a few Mikrotik Routerboards 532a. At that time they had the advantage to add a lot of Wifi cards and their price was within the limits of the budget.

But they also brought some problems despite the weak CPU. The proprietary boot loader that is installed on the board has very restricted possibilities.

The goal was to have a system for the indoor testbed, that gives the possibility to exchange test setups quickly. There are two use cases which must be taken into account, (i) only the configuration on each node is to change, (ii) the whole image must be replaced. With the Routerboards we could realize the first case (see Chapter 5 how to use the indoor testbed). To realize the second use case, several approaches are possible. One solution we thought about was to boot over Ethernet, mount the CF-card and copy the new image to the flashcard. To do this there would need to be a configuration that has the rule, try always Ethernet first, in case of fail boot from CF. The boot loader of the Routerboards does not provide this possibility and this cannot be changed without trying to hack the boot loader which could lead to complete destruction of the board.

We put a HowTo for the configuration of the bootloader to the OpenWRT wiki http://
As said above the Mipsel CPU is not very powerful. For a wireless router this is usually not important. The router must only get packets in and out. As seen in figure 1 the Mikrotik routerboards reach the full CPU load already at 10MBit with packets smaller than 300 Bytes. In this area the CPU was even not able to schedule the CPU measurement program anymore. This can be seen, because the traffic still was sent, but we got no data for the CPU.

Sending 100MBit it is independent from packet size, the CPU load can only incidently be measured.

Comparing the plot 1 with pps to the one 2 with throughput, we can see that the Routerboards’ performance problems are not caused by the memory. It seems that the boards can not handle this many interrupts. In detail we see in figure 1 that the load is very high when the packets are small. At the speed of 100MBit/sec it good to see that this load increases very slowly from big to small packets. In figure 2 we can also see that the throughput never is higher than about 55MBit/sec. This is an indicator that also with big packets on higher bandwidths the interrupts are a problem.

In the following sections it can be seen that we did measurements also with 60MBit/sec. We did not do that for the Routerboards because at 30MBit/sec we already can see, that it is not a big difference. The important things can be read from the plots as they are.
Figure 2: Mikrotik Routerboard 532a: Throughput in relation to CPU-load
4.2 Avila

4.2.1 Wired measurements

After we had already decided to deploy Mikrotik Routerboard we found the Gateworks Avilas. They do not have the possibility of extending to more than four Wifi cards, but four seems to be enough for our purposes. In summary Avila has all the advantages the Routerboards have plus the open Redboot boot loader and much more capacity in computing power.

Assembling and maintaining the Avilas is done by the MagNets outdoor team. Informations can be found in the outdoor section of the wiki [SVNI].

The performance measurement on Avila was done on machines with the original version with 64MB RAM, to see their real performance.

Figure 3 shows how big the difference between the architectures is. As it can be seen in the figure, the CPU-load is very high especially with small packets. There are gaps in the CPU measurements. It is not clear why. It might be an instability in the OS or the load is too high to be scheduled.

Comparing figure 3 and 4 we can see, that with smaller packets the throughput decreases considerable while the pps increases. This is an indicator that the Avilas can handle the interrupt load very well.

Also on high packet sizes the CPU-load is under 20% and the throughput reaches almost 100%. So it seems that the Avilas do not have any problems with their memory.

Figure 3: Gateworks Avila: Packets per second in relation to cpu-load
4.2.2 wireless measurements

As the Avilas show such good performance, we also wanted to see how they work in wireless mode. The difficulty with such a measurement is, that working wireless always can show some effects that can not be simply explained. There is also a special challenge with Atheros WiFi cards. The drivers currently used in OpenWRT are not stable at all and because of that the behavior is not predictable. Though these two barriers, we got some nice results.

To do the wireless measurement, we took two avilas communicating over wireless (Atheros WiFi cards), each of them with a wired connection to a PC. The only job of the PCs was to generate the traffic to be measured. The mode of the WiFi cards was 5GHz.

Figure 5 shows, two things. The first is, that the CPU load goes only over 50% at a packet size smaller than 500 Bytes. Again, it might be that the CPU would perform even better with more stable drivers.

The second thing is, for IEEE 802.11a it is said, that usual 50% of the maximum bit rate of 54MBit/s can be reached. In our experiment we reached almost 35MBit/s, which would not happen in reality, because the two Avilas were located in 10 cm distance to each other. But it shows, that the Avilas have a very strong architecture, that they can even handle unstable wireless drivers up to a certain point.

The Avilas are now used for the outdoor testbed and perform quite well. In future they might be also used for the indoor testbed.
4.3 Asus

As said above the Asus routers belong to the small and cheap routers. They would not be suitable as routers for the productive side of the testbeds, but they are good as monitor nodes in the mesh. For this purpose they are the cheapest boards with exchangeable radio and USB. We have for the 100 planned outdoor nodes 40 as double nodes, with one Avila and one Asus. The Asus are only there to passively listen for the traffic on the mesh radio. With this the traffic is not influenced by any measurement.

To give a rough insight how they perform we did a measurement test with the same setup as for Avila and Routerboard, see figure 6 and figure 7. The OpenWRT version for Asus was not stable at the moment we did the measurement. So it was a problem to get any data because of the many crashes and as can be seen, any bandwidth higher than 10MBits/sec shows a very unpredictable behavior.

In this case, when we have a closer look on figure 6 and figure 7, we can see that with the full speed of 100MBit/sec we can send neither very small nor very big packets. Indeed we should even avoid to send on a rate of 60MBit/sec. On that bandwidth we could not send plain ACK-packets because the Asus cannot handle packets smaller than 50Bytes.

Doing the same comparison as before, we can see that at 60MBit/sec the pps is even decreasing again while the throughput at that point looks good. This might be an indicator, that the Asus has there a real problem handling interrupts. Additionally it seems that – having again a look on the 100MBit/sec curve – the Asus also has some memory problems.

For the project that means, we made the right decision not to use the Asus as mesh
Figure 6: Asus WL-500g: Packets per second in relation to CPU-load
Figure 7: Asus WL-500g: Throughput in relation to CPU-load
routers. But with their features like the USB slot they are the ideal monitoring boards.

5 Using The Indoor Testbed With Mikrotik Routerboard

This chapter will give an overview about the indoor testbed, how to use it and how to work with OpenWrt on the Routerboards.

5.1 OpenWRT on Mikrotik Routerboard

For Routerboard it is necessary to use the unstable OpenWRT Kamikaze version directly from the development trunk. The advantage is that this trunk contains always the newest driver, kernel and software versions. The disadvantage is, it is sometimes a problem to find a revision that is compiling and running without kernel crashes. When we built the indoor testbed there were some bugs in the OpenWRT build system for Routerboards, so we had to go back to a then quite old revision 10170. A general HowTo for OpenWRT Kamikaze that also works for Routerboard is now in our internal wiki https://svn.net.t-labs.tu-berlin.de/projects/magnets-intern/wiki/KamikazeHowTo. There are also all needed links to other OpenWRT documentations.

5.2 Replacing The Firmware For Indoor Testbed

As already said, one of the goals was that everyone doing experiments should be able to work with her own images. For Routerboard the way by booting over Ethernet does not work. The second possibility is the kexec programme. With this tool it is possible to boot a default kernel and then boot into a new one. If that fails the machine can fall back on its default kernel and continue to work. Unfortunately this works not for Mipsel architectures.

The third possibility is a program called sysupgrade written by one of the main OpenWRT developers which saves the main part of the system into the RAM and copies then the new image to the CF-card. This would be a great thing for the indoor testbed, but in the revision we had to use it was not available for Mipsel. It seems that we could now upgrade to the newest trunk revision and get this functionality. This is still work to do.

5.3 Building Preconfigured Images For The Indoor Testbed

The indoor testbed consists of nine nodes with two boards each, that makes 18 images to configure and copy to the boards. In our wiki [SVNI] it is described how to preconfigure and build images in general and one by one.

We did not want to do these steps for 18 boards. So we first created the topology with fixed IP addresses and forwarding tables and then wrote a bundle of scripts to build all images from a configuration file at once. The scripts to do this including a README file explaining how to use the scripts can be found in the MagNets internal SVN repository https://svn.net.t-labs.tu-berlin.de/svn/magnets-intern/indoor/tools/image-builder/.

5.4 Saving The Current Testbed Setup Configuration And Replaying A Setup Configuration

All the nodes are reachable via a management server over Ethernet. In OpenWRT all necessary files for configuration are stored in /etc/config. Additionally we need in our case also /etc/init.d. To save the configuration of a setup we just need to download
these directories via Panther from all nodes. The same thing the other way round works.

for getting a configuration back to the nodes. For this we also have some scripts and a

README how to use them in the repository https://svn.net.t-labs.tu-berlin.de/
svn/magnets-intern/indoor/tools/config-recovery/

6 State Of the MagNets Project And Outlook

Until now we have a fully operational backbone. The backbone needs some system updates

which is already work in progress.

The outdoor testbed has now five nodes. There are still some problems with the power

provisioning, because the PoE supply is difficult to get over distances like we have when

getting from a room on the ground floor up the board over the 10th floor. But a solution

seems to be found and is yet in development.

The indoor testbed which is the major focus of this work is working. All 9 nodes are

deployed. We can exchange the configuration and store it for the next time we want to

continue with an experiment. One thing that is still not working is the desired feature

to be able to exchange automated the whole firmware image for one setup and get back

to the old one again. With the newest Kamikaze versions this should be possible. So the

future work on the indoor testbed is, to test that possibility in the smoketest testbed and

then install new images on all 9 nodes.

It further seems that with the current version that there are some strange effects on the

MadWifi software, which is responsible to manage all Wifi related things. We hope that

this will get better with a new Kamikaze version.

There are certain things we learned while doing this project.

First of all, the heterogeneity among embedded devices is much higher than among e.g.

PCs. This has the effect, that it sometimes takes a long time and some hints by others to

find the machine that matches our requirements best. The second thing is, instabilities

and side effects which can often hardly be explained are the daily work when dealing

with wireless technologies. We often needed a lot of patience to find one single bug, that

stopped the whole testbed and found it on places we would have never expected.

Third inside this project is a huge amount of knowledge that was never written down

before. Especially OpenWRT has a lot of hidden features which their developers just

do not have the time to document. This project needed the open source and wireless

community to get to the point we are now. The take away is, in such a project it is often

not a question of programming skills or technical knowledge but about to know when and

who to ask.
References

[CPUSAGE] http://www.net.t-labs.tu-berlin.de/~fabian/sources/cpusage-0.31.man.html


[SVNI] https://svn.net.t-labs.tu-berlin.de/projects/magnets-intern/


[DIOWRT] http://downloads.openwrt.org/kamikaze/docs/openwrt.html#x1-310002


Ubiquitous TV delivery to the masses

Roger P. Karrer
Deutsche Telekom Laboratories
10587 Berlin (Germany)
roger.karrer@telekom.de

Doris Reim
TU Berlin
10587 Berlin (Germany)
doris@net.t-labs.tu-berlin.de

Abstract

The vision of ubiquitous delivery of TV streams to the masses at high quality is tantalizing. However, today, TV streaming is either limited to fixed wired outlets (IPTV), or it is distributed over DVB-x but the required receivers are not built into their devices. Wireless mesh networks have the potential to bridge this gap, as they can receive the TV from either technology and forward it over multiple hops to the user devices that are today equipped with 802.11 cards by default. This paper describes the set up of a TV streaming over the Magnets mesh network deployed in the city of Berlin. Initial performance evaluations on the Magnets backbone and the Magnets indoor mesh show that a combination of careful engineering and multi-card wireless routers can provide sufficient bandwidth to support TV streaming over wireless networks.

1 Introduction

The vision of ubiquitous high-speed access to the Internet is tantalizing. With the trend to all-IP networks and services, users will not only be able to surf the web, but also have access to a plethora of voice (VoIP) and streaming services, such as TV (IPTV). In fact, forecasts for 2012 predict that Mobile TV will be at least successful if not the “killer application” for next-generation mobile systems.

For the distribution, various technologies are available today, including WiFi, 3G and Digital Video Broadcast, e.g. via terrestrial antennas (DVB-T). The use and the advantages of the different technologies depend on the availability of the technology, the content and the data access. For example, DVB-T is suited to broadcast the same data to a large user group, but is a one-way data communication that does not allow interactive data access. In the city of Berlin, e.g., three DVB-T antennas are sufficient to broadcast digital TV in a way that within the city area and its near surroundings, TV channels can be accessed indoors and outdoors with a small indoor antenna. Beyond this “inner circle”, TV programs can be received with outdoor or roof-mounted antennas. As a result, 30 channels are available today at good quality for free (excluding general fees), a net save of tens of dollars compared to TV via cable.

But how does the data that arrives on wired Internet connections via IPTV or via DVB-T a large user group? IPTV is only available at the outlet of the wired connections. DVB-T, in contrast, is widely available, however only few portable devices such as laptops or home entertainment devices are equipped with DVB-T receivers. Even though indoor boxes and, most recently, USB-enabled DVB-T receivers are available, it is unlikely that all devices will be equipped with such receivers given that their costs ranges from several tens to hundreds of dollars. Therefore, in today’s environments, a gap exists between the vision of pervasive TV access and its realization.

WiFi mesh networks have the potential to bridge the gap between the locations where TV data is available (wired outlets or DVB-x senders) and the end users. We understand a mesh network as a set of infrastructure-based, power-wired access points (APs) that primarily relays the data over multiple hops (APs) from or to a small subset of the APs that are connected to a data source. In the case of TV, this data source is either a wired Internet connection or a DVB-T receiver. Thus, by forwarding data from the data source over multiple hops, the mesh network allows users within its coverage area to receive TV streams.

However, can WiFi mesh networks really come up to the demands of TV streams? It is well known that many deployed WiFi meshes today have serious performance issues. First, the application-layer throughput of links of the MIT Roofnet [3] or the Technology For All network in Houston [4] is limited to single-digit numbers, even though the WiFi cards have a raw throughput of 54 Mbps. Taking the protocol overhead of approximately 50% of the different layers into account, the measured throughput is simply just a fraction of the expected application-layer throughput. Second, wireless mesh networks show severe performance degradations for flows that traverse multiple hops. Moreover, they show severe unfairness towards flows that traverse more hops than shorter flows. In particular, studies show that flows may completely starve after 3 hops in a
wireless network [5]. Therefore, current mesh networks are not suited to support the bandwidth demands required for TV streaming over mesh networks. But does this shortcoming imply that wireless mesh networks are unable to support TV streams?

This paper describes the set up of a TV streaming over the Magnets mesh network deployed in the city of Berlin [7]. Compared to the above mesh networks, Magnets has two fundamental differences. First, Magnets contains a dedicated, well-engineered high-speed wireless backbone. Initial measurements show that single backbone links are able to support high data rates. However, we have not investigated if the backbone characteristics are sufficient to support TV streams in terms of bandwidth, delay and jitter. Second, attached to the backbone is a WiFi mesh network whose APs are equipped with multiple WiFi cards. Having multiple WiFi cards allows an AP to simultaneously send and/or receive data from and to different APs. In this paper, we provide evidence that the performance degradations found in access points with a single WiFi card only are not visible when multiple cards are used in different frequency bands.

The contributions of this paper are three-fold. First, we describe the design of the Magnets network. In particular, we highlight the design of the backbone and the mesh network. We provide details on the network layout and the mesh nodes used for the backbone and the mesh. Second, we describe the setup to stream TV over the backbone and the mesh network. We provide details on the hardware, the software and the requirements in the mesh and at end systems to support TV streaming over the mesh and the backbone. Finally, we provide initial measurements on the backbone and in the mesh that show that a combination of careful engineering and multi-card wireless routers can provide sufficient bandwidth to support TV streaming over wireless networks.

This paper is organized as follows. Section 2 provides background information. Section 3 describes the architecture of the Magnets network and the setup of the TV streaming. Section 4 presents measurements from streaming TV and TV-like traffic over Magnets. After discussing related work in Section 5, we conclude in Section 6.

2 Background

This section describes the scenario and provides background on the technical details of DVB-T in general and the setup in Berlin in particular.

Figure 1 depicts the scenario we are considering: users (mobile nodes, MN) with laptops or any other device that is able to display TV streams and that is equipped with a WiFi card are connected to an access point of the mesh cloud.

Figure 1. TV streaming over a mesh.

Other APs of the same mesh are connected to a TV streaming source: either a fixed Internet connection where the TV is streamed via IPTV over the core, or a mesh router or server that contains a DVB-T receiver. The mesh forwards the data over multiple hops from this source to the different users.

DVB-T stands for Digital Video Broadcasting - Terrestrial and it is the DVB European consortium standard for the broadcast transmission of digital terrestrial television. DVB-T transmits a compressed digital audio/video stream, using OFDM modulation with concatenated channel coding (i.e. COFDM). Currently, a second-generation specification (DVB-T2) is under discussion.

DVB-T uses MPEG-2 for source coding. The compressed video, compressed audio, and data streams are multiplexed into PSs (Programme Streams). One or more PSs are joined together into an MPEG-2 TS (MPEG-2 Transport Stream). The TS is the basic digital stream which is being transmitted and received by home Set Top Boxes or by DVB-T USB devices. Most recently, H.264 is also used as an alternative to H.264.

The bitrates for the transported data depend on a number of coding and modulation parameters and can range from about 5 to about 32 Mbit/s. In Berlin, the DVB-T senders are set to 16-QAM 2/3, which result in bitrates between 14 and 18 Mbps [1].

Given the scenario and the requirements from the DVB-T streaming, the main challenge we are interested in is: are mesh networks able to support the required bandwidth, latency and jitter for TV streaming? Commodity hardware today achieves a raw data throughput of 54 Mbps or even 108 Mbps via proprietary modes. With the overhead of the different protocols in the Internet stack that accounts for roughly 50% of the raw throughput, the expected application-layer throughput can be estimated to roughly 27 Mbps. However, experimental mesh networks deployed, such as the MIT Roofnet [3] or the Technology For All in Houston [4], reach only single-digit throughputs and are far from achieving the rates required to stream TV.
3 Architecture

This section first provides information about the Magnets testbed, in particular the WiFi backbone and the WiFi mesh in isolation. Then we describe the setup to stream TV over Magnets.

3.1 Magnets backbone

The Magnets project aims at deploying a next-generation wireless access network architecture. Within this network, the high-speed WiFi backbone connects 5 high-rise buildings in the heart of Berlin. The backbone is composed of 6 PC based routers and 12 Access Points (AP) (10 indoor and 2 outdoor). The APs consist of Intel IXP240/266 MHz (indoor) and IXP425/533 MHz (outdoor) programmable network processors (NP) as CPU, and Atheros 5213/5/112 chipset for their WLAN interfaces, and run a proprietary operating system called LC.OS. More information on the topology can be found in [7].

In previous work, we have shown that the Magnets backbone achieves high link speeds over single and multiple hops by three means [6]. First, each link can be activated individually to avoid multi-hop throughput degradations known from mesh networks [5]. To achieve independent link transmissions, each link is operated via a dedicated AP. The APs are interconnected via a Linux PC with multiple network interface cards that acts as a router. Second, directional antennas ensure a high signal level to bridge the distances but also reduce the interference with other links.

Third, the APs feature two proprietary protocols to enhance the throughput beyond the 54 Mbps supported by 802.11a/g termed Turbo Mode and Burst Mode that can be enabled optionally. Turbo Mode doubles the channel from 20 MHz to 40 MHz. While, using Burst Mode, the sender only waits for the shorter SIFS (Short Inter-Frame Space) after a successful data exchange instead of the longer Distributed Inter-Frame Space (DIFS) specified in 802.11. The modes should result in a performance enhancement of 10 Mbps for Burst Mode and a throughput doubling for Turbo Mode. These modes are expected to boost the backbone performance without negative impact due to the independent link scheduling and the use of directional antennas. For general (mesh) networks, however, Burst Mode can lead to severe unfairness and Turbo Mode interferes with all other channels in the 2.4 GHz spectrum because it must be centered around channel 6 to stay within the allotted frequency band.

3.2 Magnets mesh

The Magnets mesh network aims at investigating the limitations in terms of capacity and delay, but with off-the-shelf hardware only. Our main interest is to assess how a mesh network can scale to high capacity and thus to high end-to-end throughput if the mesh nodes are equipped with multiple WiFi cards. Towards that goals, we have decided to acquire 20 RouterBoard 532 [2] as an all-in-one integrated communication platform. Such a board features a MIPS32 CPU running at up to 66 MHz and a 32-bit PCI controller at 66 MHz. For networking, the board provides up to 3 Ethernet ports and 2 MiniPCI slots on board. Daughterboards can additionally be attached via on-board connectors. The RouterBoard 564, e.g., is a daughterboard that provides 6 Ethernet ports and 4 MiniPCI slots. Using Atheros 802.11a/g WiFi cards that offer 54 Mb/sec in their standard mode and 108 Mb/sec with SuperAG enhanced technology, the theoretical throughput of a routerboard reaches up to 648 Mb/sec.

We deployed the boards on 5 adjacent floors of our office building. Due to the concrete and steal construction of the building, the connectivity among the nodes shows an interesting behavior. On each floor, the boards are placed at the four corners of the building. While each board “sees” its neighbors along the wall, the connectivity along the diagonal is very low. That is, the connectivity of the 4 mesh nodes forms a rectangle. Vertically, connectivity exists between adjacent floors only, however, there is no connectivity between nodes that are separated by two or more floors.

Given this connectivity, we have several options to configure the routing and thus modify the logical topology of the mesh network. For our study, we use static routing for two reasons: to avoid effects on streaming by route changes and build the different logical topologies. In an extreme case, we can build a linear topology of 20 mesh nodes. To avoid any undesirable interference among the nodes, the channels of the different WiFi cards are separated as far apart as possible. That is, we use the entire free spectrum at 2.4 GHz and in the upper and lower band of the 5 GHz range. The detailed assignment varies with the routing and thus the logical topology.

3.3 TV streaming over Magnets

To set up TV streaming over Magnets, only a few components had to be acquired and configured, as sketched in Figure 2. To receive the data stream sent out from the antennas, we connected a USB DVB-T receiver to a Linux server.
This server acts as the ingress point of the TV stream into Magnets. The hardware on the USB DVB-T receiver captures the TV stream and forwards it to the TV capturing software running on the server. For this purpose, we used the Kaffeine player that already supports the handling of DVB-T devices, and we installed the XviD library to interpret the data stream.

The Kaffeine player also supports the broadcasting of the incoming data stream to a destination. For our purpose, we connected an AP via a wire to the Linux host and configured that the data stream is directly forwarded to the AP. From there, the data stream is forwarded over multiple APs towards the WiFi-enabled laptops that act as sinks. With respect to the APs, we used both the Magnets backbone and the indoor mesh to forward the data. We always used fixed routes to forward the data because at this stage we wanted to investigate if the underlying infrastructure is already able to support the data rates required by the TV stream. A dynamic routing protocol would have mixed the link behavior with the routing behavior and would like probably have resulted in effects that are not easily understood. Therefore, we decided to stick to static routes only.

4 Evaluation

This section describes our measurement strategy and discusses our experimental results.

4.1 Measurement strategy

To assess the ability to support TV streaming over Magnets, we perform two sets of measurements over the backbone and the mesh independently. First, we use iperf, a well-known traffic generator tool, to generate a constant stream of UDP packets at a rate of 16 Mbps with packets of 1kB size. This rate corresponds to the streaming rate of DVB-T in Berlin. Second, we attach the DVB-T stick to the server and forward the real TV stream over the mesh.

This two-staged strategy is motivated by three key facts. The first is that the DVB-T stream does not generate packets at a precise regular interval. Inside the server, the data transfer from the USB card to the Kaffeine player and from there to the AP can interfere as the transfers cross the same bus. In contrast, iperf is the only application running on the server, the mesh and the client and is therefore more precise, in particular to measure jitter and latencies. Second, we set up iperf to have its packets carry a sequence number to detect packet loss. With DVB-T, the identification of packet loss is not straightforward.

On the other hand, DVB-T has the advantages that we can assess the video quality. We modified the client viewer to calculate the number of correctly received frames in time for displaying. That is, packets that are not received in time are not displayed and are therefore not included in these measurements. Moreover, since MPEG gives different priorities to its frames within a Group of Pictures (GOP), the loss of a single packet may have an influence of multiple other packets or frames. Finally, DVB-T measurements take into account that frames may not be displayed if they arrive correctly but too late at the client. Again, this metric is not considered with the iperf measurements. Therefore, DVB-T allows us to measure application-layer metrics whereas iperf relates to network-layer metrics.

When streaming data over the backbone, we consider a two-hop topology from the node at T-Labs via TC to HHI (see Figure 1 in [7]). We chose this topology because links from and to ETF are not reliable enough and we excluded the direct links from T-Labs to HHI to avoid mutual interference among the nodes at HHI that are all in the 2.4 GHz band. For the mesh, we set up the routing on the mesh nodes to form a linear topology. The experiments we show in this paper compare the performance seen by a client that is between 1 and 4 hops away from the IPTV server. All experiments were conducted over 2 minutes time and repeated 20 times.

4.2 Backbone measurements

Figure 3 depicts the throughput at the receiving node when traffic is injected at 16 Mbps using iperf. The x-axis denotes the time of the experiment, the y-axis denotes the throughput in Mbps. The figure shows a randomly selected trace. We anticipate that all traces have similar characteristics in terms of average throughput and standard deviation, so that only the details of when which dip occurs changes among the traces.

The figure shows that the Magnets backbone is able to support the traffic injected via iperf. In fact, the mean throughput is slightly above 15 Mbps and therefore only minimally below the injected traffic rate. The standard deviation lies at 1.3 Mbps, which emphasizes the stability of the links. The small dips in the performance, however, show that the stability is treacherous, while in fact the medium is
still air and therefore sudden drops in the performance may occur for whatever reason.

Next, with the same setup, but using the DVB-T receiver instead of iperf, we measure the framerate that the video player sees at the destination. The video player logs the number of frames it can display per second. This metric is shown on the y-axis of Figure 4 as a function of time on the x-axis for a random experiment again. Note that the number of displayable frames excludes those frames that arrived at the receiver but can not be displayed because a frame with higher priority was not received correctly. For example, if a P-frame is missing in the stream, B-frames that depend on the P-frame can not be displayed either even though they may have been received correctly. These frames are not accounted for because the player can not display them.

The figure shows that the backbone is able to maintain an almost reliable frame rate. In fact, the average frame rate is 28 frames per second, out of 30 transmitted, with a standard deviation of 2 frames per second. These rates clearly lead to an acceptable if not excellent viewing experience by a human user.

Thus we conclude that the Magnets backbone is able to provide the necessary support to stream TV from a source over multiple hops to a destination.

4.3 Mesh measurements

This section repeats the above measurements, except that the data is now streamed over the indoor mesh rather than over the outdoor backbone. There are two significant differences in the measurement setup that we expect will influence the results. First, the mesh is indoor. Therefore, we expect that the performance of the links is more stable than in the outdoor environment. Second, the mesh network uses omni-directional antennas whereas the backbone used directional antennas. Due to the differences in the environment, the antennas do not have a significant impact on the throughput, but with omni-directional antennas we are able to compare the streaming performance when the APs have one or two WiFi cards enabled.

First, we repeat the above experiment using iperf over the mesh, with an traffic load of 16 Mbps. Figure 5 shows the average throughput as a function of the size of the mesh, which corresponds to the number of hops due to the linear topology on the x-axis. The two bar groups show when the APs have 1 or 2 WiFi cards enabled. The throughput is first averaged per experiment and then averaged over the 20 repetitions.

When the APs have 2 WiFi cards enabled, the average throughput lies above 15 Mbps, independent of the size of the mesh. It can be expected that the throughput would remain at a similar level even if the mesh network contained more nodes. In contrast, with only 1 WiFi card per AP, the throughput degrades significantly with the size of the mesh. For a 2-hop topology, the performance is already reduced to 10 Mbps, and for a 3- and 4-hop topology, it is as low as 2 Mbps. Two factors contribute to the performance degradation. First, each relaying node can only send or receive at a given point in time. Therefore, the "raw capacity" of a node is actually halved, which has a visible impact on the application-layer throughput. Second, all WiFi cards must be set to the same frequency to ensure that two adjacent nodes see each other. As a result, some interference is caused also to those nodes that are not a direct neighbor. Fortunately, due to the building, the mutual interference is limited and thus this parameter is not affecting the throughput as much as the former.

Finally, Figure 6 depicts the displayed frame rate at the receiver as a function of the mesh size and the number of WiFi cards. The figure shows a similar trend as in the previous experiment: the frame rate is almost optimal and independent of the mesh size, whereas it degrades when only 1 WiFi card is used. In contrast to Figure 5, however, the frame rate almost reaches zero for a topology of 3 nodes or more. The combination of low throughput and high jitter (not shown in the figures) leads to the situation that the video player hardly receives a correct set of frames in the time needed.

Thus, based on the above results, we conclude that the
Magnets backbone and the mesh are able to bridge the gap between TV stream availability and its distribution to a broad user community. In particular, the design of the network allows Magnets to obtain sufficient bandwidth to stream TV to the users.

5 Related Work

The vision of using DVB-T as a broadcast medium is shared by other projects, e.g. Daidalos [8]. And, of course, the deployment of DVB-T stations that already broadcast digital TV channels in several cities in Europe. However, it is still unclear how many devices will be equipped with DVB-T receivers to actually profit from the broadcast medium.

Mesh networks are increasingly deployed worldwide. The most prominent mesh networks are probably the MIT roofnet [3] and the TTA network in Houston, Texas [4]. However, many cities today deploy mesh networks, including San Francisco, London, Taipei, etc. Technically, the key distinguishing factor of Magnets is the combination of a high-speed wireless backbone with a wireless mesh network with multiple WiFi cards per mesh node. A key problem that many commercial mesh networks face is that the business cases are still at odds and that applications are missing to attract subscribers.

Therefore, we argue that the combination and integration of DVB-T with a high-speed wireless mesh network that even supports TV streaming is novel. The results highlight the feasibility of streaming TV over a mesh and provide therefore one example of an application that may make mesh networks attractive to users.

6 Conclusions

This paper presents an architecture to bring digital TV to a large user population. The architecture consists of a wireless mesh network that forwards data from a single source that is connected to a TV streaming source to users that are connected to the mesh access points. The streaming source can be a simple wired Internet connection that receives IPTV data, or a DVB-T receiver attached to a wireless mesh node via USB that transforms DVB-T signals into IP packets to forward over the mesh.

An experimental performance evaluation on the Magnets backbone and the indoor mesh shows that Magnets is able to support the data rates required for IPTV. The use of multiple WiFi cards per AP is a key requirement to achieve a sustainable throughput in a mesh network. The comparison of results with 1 or 2 WiFi cards emphasize the need to build mesh networks with multiple WiFi cards. On the other hand, as promising as these results may look, a number of factors may also speak against such a deployment. For example, the free spectrum available today is scarce. What happens if multiple mesh networks, each with multiple WiFi cards, start to interfere? How much free spectrum will we eventually need? Shedding light on these and other questions is a key goal of the Magnets project.

In future work, we will perform more measurements over more complex topologies and even over an outdoor mesh. Moreover, we will look at the control functions within the different layers to understand the limiting factors of the mesh network. Because the presented measurements show the potential of mesh networks to realize the vision of pervasive communication.

References