A performance test of the Zaurus PDA

Jacques Weewer
Student number: 0542682
Email: 0542682@student.hro.nl

Technische Universiteit Delft

HOGESCHOOL ROTTERDAM

Abstract

The first hours after a crisis or disaster are critical period and has to be handled with care and as efficient as possible. In these first hours a lot of victims can be rescued and survive such a disaster. The longer it takes to get to the victims the more chance there will be that they die. The approach that was thought of can reduce more casualties.

When there is a crisis situation almost all rescue workers walk around in a dynamic environment which can be dangerous. This can lead to communication problems between rescue workers. So there is an interesting part to help the rescue workers by giving them some kind of extra tool so they can inform each other. It is possible to do this on a wires available infrastructure. But what if this infrastructure is overload with people calling the emergency number, or this infrastructure is disabled by the crisis/disaster. In these cases there is a need for a wireless solution.

In the past years there have been a lot of developing on the C2000 network. This is a digital radio network which is used by the rescue services in The Netherlands with as purpose to provide mobile communication devices, which can be mobile data terminals and portophones.

At the University of Delft they thought of an other way. In which the PDA plays a central role. These PDA's are connected with each other in a wireless network. In order to communicate in a good manner with each other there must be some special wireless network in which all rescue workers are connected to each other. But how does this network work and does it work. And how does the chosen device handle all the things that it has to do. Is it capable of doing its tasks.

This thesis describes some of the practical things of the chosen device that can be handed out to the rescue workers. The tests that are performed are only to benefit this complete project and to use as an advise report to think about the hardware specifications of the device and network communication layer that is being used.

Acknowledgment

Finally after after all these years of studying I can write this final thesis to close this period. And now I can say that it is all finally finished. Of course there are some people that really helped me through this whole period, just by cheering me up, giving me some inspiration, or just to laugh with. I'm so excited about it and happy to start of a new period in life.

First of all I would like to thank you my supervisor L.J.M Rothkranz for giving me the opportunity to finish my final project at the TU in Delft. I know when I first walked in I wanted to do something with vision. But as soon as we got some discussion about networking and that it was my background from my study it turned in to this project with the Zaurus. And it was fun for me to work with these little PDA's and also the application(s) that where available for the project.

Second I would like to thank you to my supervisor M.M.M. Ab Del Ghany from the Hogeschool Rotterdam. Thanks you for having faith in me and kept supporting me all those years. Also all our conversations sometimes where good to have, sometimes they gave me some inspiration to keep on going.

I would also like to say thank you to Wim Tiwon. Thank you for all your idea's input and other things we discussed. Without your help about the battery issue it would all probably ended up in a disaster. After all the many discussions and possible causes we finally concluded that the old batteries where almost dead. Also some other idea's you had about things really gave me some good idea's. So I think it you deserve it to be in my thank you note.

Also I must not forget to mention all people I worked with in the lab, the people who we all had some great laughs with and also some serious discussions. First Peter Hagendoorn, the news discussions and the talks about sometimes really nothing where sometimes awesome and fun. Huub van Niekerk, although you joint later, we always had some nice discussions about everything and always laughed about the crazy things. Gert-jan, my companion in the Zaurus project with the Zaurus for sometimes giving me some ideas to think about. And of course all the others I forgot.

Finally I would like to thank my family for always supporting me in everything I did. My father for always keeping me motivated with everything, although he was pushing too much some times and I never listened, still I want to say thanks. My mother for always listening and supporting me. My younger brother for all the sometimes inspirational talks we had but also the little wrestling fights we had, those little fights sometimes really cleared my head and made me forget about some of the things I was stuck at at some point. And of course also a little thanks to my dog, for the long walks when I could think about things that were related to the project.

Thank you all!!!!

Table of Content

1. Introduction 1
   1.1 Background 1
   1.2 Structure of this Document 2
2. The Project 3
3. The Device 5
   3.1 The Zaurus 5
      3.1.1 The Processor 6
      3.1.2 General application and Software 7
   3.2 Secure Digital Card (SD Card) 9
   3.3 Network Interface Card (NIC) 9
4. The Network 11
   4.1 General information about AODV 11
   4.2 Ad hoc On demand Distance Vector 12
      4.2.1 Protocol overview 12
      4.2.2 Path setup 12
      4.2.3 Maintenance 16
      4.2.4 Local Connectivity Management (LCM) 16
   4.3 Results of the AODV tests 16
   4.4 Developed applications for this network 20
      4.4.1 ISME 20
      4.4.2 Lingua 21
      4.4.3 PDA Assistant 22
5. Performance of the Zaurus 25
   5.1 The battery 25
   5.2 Processor usage 25
   5.3 System load 26
   5.4 Memory and Swap usage 26
   5.5 Design of tests 27
      5.5.1 Idle test: without any extra hardware components 28
      5.5.2 Idle test: without SD Card and with NIC 30
      5.5.3 Idle test: with SD Card and without NIC 30
      5.5.4 Idle test: with both hardware components 31
      5.5.5 Stress test: copy a file over the NIC to the NAND 32
      5.5.6 Stress test: copy a file over the NIC to the SD Card 33
      5.5.7 Stress test: copy a file between NAND and SD Card 34
      5.5.8 Stress test: full use of the Zaurus with SD Card mounted but without NIC 35
      5.5.9 Stress test: full use of the Zaurus with SD Card and NIC mounted 37
      5.5.10 The TICS application 38
6. Conclusion 41
   6.1 Battery 41
   6.2 Processor usage 41
   6.3 System Load 42
   6.4 Memory/Swap 42
   6.5 The Network 43
   6.6 Recommendations 44
Bibliography 47

List of Illustrations

Illustration 1: Zaurus in normal mode....5
Illustration 2: Zaurus in tablet mode....5
Illustration 3: PXA chip....6
Illustration 4: QTopia application launcher....7
Illustration 5: Secure Data Card....9
Illustration 6: The Network Interface Card....9
Illustration 7: tcpdump clear ahbogdan....11
Illustration 8: tcpdump clear ahleon....11
Illustration 9: Setup AODV....13
Illustration 10: Finding neighbors....13
Illustration 11: Route Setup....15
Illustration 12: tcpdump directly after the AODV is started....17
Illustration 13: Route table....17
Illustration 14: Two Zauruses broadcasting the AODV protocol....17
Illustration 15: Route table change after 2 Zauruses broadcasting....18
Illustration 16: Multihop....18
Illustration 17: tcpdump ahsiska....18
Illustration 18: route table ahsiska....19
Illustration 19: tcpdump ahbogdan....19
Illustration 20: route table ahbogdan....19
Illustration 21: tcpdump ahleon....19
Illustration 22: route table ahleon....20
Illustration 23: ISME Client....20
Illustration 24: Lingua application....21
Illustration 25: PDA Assistant application....22
Illustration 26: Picture that can be send/recieved with the application....22
Illustration 27: Battery....25
Illustration 28: Data storage pyramid in speed order....26
Illustration 29: The memory management simple version....27
Illustration 30: Idle without SD and NIC - Battery usage....29
Illustration 31: Peak idle test half way....29
Illustration 32: Idle with SD and NIC - Processor load....32
Illustration 33: Copy over NIC to NAND - User/System processor usage....33
Illustration 34: Copy over NIC to SD – Processor load....34
Illustration 35: Copy NAND to SD - Memory usage....35
Illustration 36: Full use without NIC - Processor load....36
Illustration 37: Full use with NIC - Battery usage....37
Illustration 38: TICS try 1 - Memory/Swap usage....38

Index of Tables

Table 1: FPU benchmark results....7

1. Introduction

This introduction will describe the background of this thesis and also how this document is setup.

1.1 Background

Lets step back into our recent history, over the last 6 years we had some disasters and terrorist alarms. Something that is very common these days. We shall name a few of them not all. One of the major disasters in The Netherlands was the fireworks disaster in Enschede.

It was 25 May of 2000, what started just as a little fire the firefighters were trying to put out the fire. Nobody knew that there was a garage with all kind of fireworks in it. So the firefighters were trying to put out the fire when suddenly all the firework lit. After this everybody started running and screaming searching for cover. A few seconds later there was a big explosion and a complete part of a neighborhood was completely destroyed. Because of all this screaming the rescue workers weren't able to communicate in a good way with each other. Also people didn't investigate the scene well because of they did it could have saved lives.

On a particular day on the 11th of September in 2001 a group of terrorists hijacked some passenger airplanes and used them to flew into the Twin Towers in the center of New York. Also a plain flew into the Pentagon in Washington D.C. Also there was a 3rd plane probably on it’s way to the white house, but never made it that far. Later that day the 2 massive Towers collapsed and many people lost their lives. When the 2 towers collapsed there was a sudden brake of communication, because a lot of it was stored in that neighborhood and was destroyed by then. There was no way they where able to communicate with the rescue center nor each other. So there was no way of using your cellphone, using a phone booth, everything was just dead. Because there was no way to communicate, no one knew what an other group was doing and so a lot civilians and rescue workers lost their lives.

An other major disaster was the tsunami on the 26th of December in 2004. A tsunami, which is just a sea flood, is cause by an earthquake at the bottom of the sea. First it takes away the water for hundreds of meters and then suddenly a big wave is heading for the shore and destroys everything on his path. This is what happened and this day. This natural disaster caused approx. 200.000 people their lives. This death count was spread over different countries that where in the line of the big wave. This were countries like India, Indonesia, Malaysia , Thailand and a few more that I might forgot to mention. In all of these areas the there was a lot of destruction done by the sea, houses where destroyed, power lines, and communication lines. Because these communication lines were destroyed there was a total chaos on where people were found and if people were dead. So it was very hard to get in contact with these areas. The only way to get information from there was to send people over there and report things.

In 2005 there was this big hurricane called Katrina striking New Orleans. Many people were warned, but not everyone listened to the advise to evacuate. This hurricane had a

devastating force and destroyed a big part of the city, not only 1 the houses but also the power lines, phone lines, some people survived while they stayed at their house some people died, because of the dead power lines no one was able to call to rescue service nor where the rescue workers to really communicate to each other in that area, most of the time this was just done by shouting and marking a house when they searched this area.

These were just a few things that happened over the past 5 years, of course not everything is mentioned but the point is made. These things happened for real and one of the things discovered is that communication really is necessary when lives are at stake. If there is a way that we can all communicate in such a scene a lot of lives can be saved and things wouldn't end up in a chaotic way.

To prevent this chaos the Technical University of Delft together with DECIS worked on the project called COMBINED. This project was setup to develop a system that can be used in these kind of crisis situations like described above. They developed an infrastructure, called a MANET(Mobile Ad hoc Network), that can work wireless and needs no big setup. So every node in the network is free to walk around and in some way it can find an other node to communicate with. A bit higher is the software running on the device, this device is a PDA. The software can be used to to sent data over to a center that has been setup. This center receives all data that rescue workers collect and have sent over. This way they can decide what is needed in a specific area or how many rescue workers are needed in that area.

From kind of software there have been created a lot of types and versions. These were most of the time simulations or were created as an idea. Unfortunately there hasn't been a final version nor solution yet. And still at this moment of writing this thesis I know that there is someone writing an other software version for this project. Different people wrote software for this project with all a different vision on situations and thinking patterns.

But the software is not the biggest point at this present time. There has never been tested with the selected hardware for this project, which is a proof of concept. To experiment with they selected a PDA from Sharp, the Zaurus SL-C760. There where never things tested like how does the Zaurus react in different idle situations, nor was the network strength, range tested. All practical things that was never looked at. This thesis covers the experiments that have been done and describe the findings.

1.2 Structure of this Document

This document will have the following build up. At first the project will be discussed. After that there will be a little introduction about the mobile device that has been used and all of its components. Then the chosen network type (Ad-Hoc) will be discussed and how it will be setup. After that has been done the performance of the Zaurus will be described.

2. The Project

The University of Delft and especially the Man Machine Interaction (MMI) group has a main project on crisis management. Within this group there are different approaches on how the software has been build and thinking ways on how the software should work. This software facilitates the communication between rescue workers with the help of a mobile device. This mobile device is a Sharp Zaurus S-CL760 PDA.

The reason this complete project was setup was to let the rescue workers chat with each other in crisis situations when there is a lack of insufficient communication which leads to wrong decisions. And we all know that wrong decisions can lead to situations were there can be casualties.

This project is is used to investigate the selected hardware and it's components. The chosen Sharp Zaurus S-CL760 is a PDA with a Linux Operating System on it and it was capable of running the Java runtime environment for the developed applications. So there were some things that was never looked at. And those things were the following.

- How does the Zaurus react on different working situations
- How is the network performance

When we look at how the Zaurus reacts in different testing situations there were a set of factors that need to be monitored throughout these tests. These factors were:

1. The memory
2. The processor usage
3. The system load
4. The battery usage

These factors determine how long a PDA can do it's work. And therefor one of the key things is to make a good overall view of how things are related to each other when for example when a Network Interface Card is used with the mobile device will it have a different uptime then when I only use it with an Secure Disk Card or with both components.

One other thing is the network setup. The used and chosen network is an ad-hoc network. Which means there is no need of an access point and all PDA's within the network work as a little special router. The workings of an ad-hoc network can be found in chapter 4. There are also some key things on this part of the project. There for there was created a little list of problems.

1. Is there overhead on the network
2. Does the AODV protocol work like it should work
3. What is the network performance

These things are also interesting to know because a lot of overhead can lead to an instable network. And in critical situations we don't want that to happen. So the project was to look at these problems and find out if this will have some kind of success rate with them.

So the biggest point is to make a helicopter view of what has been done. And what has been done already is this working the way it is suppose to work.

3. The Device

This chapter and the following chapters will describe the hardware that has been used while working on this project. The device it self will be explained with some of its internal components and also the extra hardware devices that are being used. And finally we describe a bit on the software that is running on the device.

3.1 The Zaurus

The Zaurus [2] is a little handheld computer from Sharp. The Zaurus is as they call it a "personal mobile tool". The model that is used throughout these tests is the Sharp Zaurus SL-C760, which can be seen on the picture below. It looks like a tiny laptop, with some extra option to turn in into a tablet form device by twisting the screen around and close over the keyboard. The screen of the Zaurus is a 3.7" screen and has a resolution of 640x480 and can display up to 65.000 colors.

Illustration 1: Zaurus in normal mode

Illustration 2: Zaurus in tablet mode

The processor is from Intel, an Intel XScale PXA255 running on 400Mhz. It also has a flashcard expansion slot, SD card slot, infrared receive/send option, speaker. This little device is a little multimedia PC that can be put away in your pocket. The flashcard expansion can be used for example for a little camera or a Network Interface Card (WiFi or Ethernet). The device it self has 64Megabytes of SDRAM and a 128Megabytes of storage where 65Megabytes is for the user to use the rest is needed for the operating system.[2]

The power that the Zaurus can deliver really depends on the battery. The battery that is being used is a 1800mA 3.7v Rechargeable Lithium-Ion (Li-ion) battery. This type of battery is a sort we find in a lot of mobile devices. These kind of batteries always need to

have a little bit of charge in them when they reach the 0% power line, otherwise these batteries will have a shorter life period then when they are used the correct way.

3.1.1 The Processor

The processor used in this device is an Intel XScale PXA255 [8] running on 400MHz. There are 3 different version of this type of processor released. Running on 200, 300 and 400 MHz. To understand more about this processor the Intel documents that can be found on their website were read partial. This way I found out and could defiantly conclude that this processor misses a lot of mathematical calculation power because of the missing Floating Point Unit (FPU). To show this there has been performed a simple benchmark which is a bit below here. The processor it self is an ARM Processor. In which ARM stands for Advanced RISC Machine. While RISC on its own is also an acronym for Reduced Instruction Set Computer.

Illustration 3: PXA chip

Standard ARM processor misses a lot of extra functions that we do find in the x86 architecture machines. Like said earlier one of the important things is the FPU. But because of the missing functions these processors aren't made for high end applications or tasks. This machine does have one co-processor but this co-processor is only for the video and sound editing/listening/watching on this machine.

The reason to put an ARM processor in such a little device is because it is cheaper to manufacture. There for it is a good CPU type for things you want to do on the personal level. There is also a sales pitch that sounds a bit like this: "The ARM architecture has the best MIPS to Watts ratio as well as best MIPS to $ ratio in the industry; the smallest CPU die size; all the necessary computing capability coupled with low power consumption of which a highly flexible and customizable set of processors are available with options to choose from, all at a low cost." [7]

There are some disadvantages for using this processor. If you create a binary on an x86 architecture machine it will not be able to be executed on the Zaurus which I've tried in the very beginning. So to be able to run programs on the device you need to setup a cross compile environment in which all development libraries and compilers are that are also used on the Zaurus. I think this is also why people chose to program Java because then they do not have to cross compile for the system.

Floating Point Unit (FPU)

The ARM Processor needs to emulate a FPU. This means that it misses a lot of mathematical power to make big calculations. This has been tested with an old benchmark called a whetstone benchmark. This benchmark has been performed on 3 systems, 2 of these systems where i386 architecture processors and the other one was an ARM-architecture processor. The processors used on this benchmark are:

- AMD K6-2 - 350Mhz
- Intel XScale PXS255 – 400MHz

The 2 i386-architecture processors have an built in FPU and have been tested in 2 ways. In the first test we booted the system with the FPU, in the second case we booted the system without a FPU, because it where 2 Linux systems this was zone easily by giving the extra boot parameter *no387*. When the boot parameter was removed we shall see that in this case the FPU will be emulated and it will depend on the processor speed. the table below here shows us the benchmark times that have been measured while performing the whetstone benchmark. In all cases the same amount of cycles (20000) have been used to perform this test.

Table 1: FPU benchmark results

System	***Time with FPU***	***Time without FPU***
Intel Xscale PXA255 - 400Mhz	(none)	1282 sec.
AMD K6-2 - 350Mhz	16 sec.	1792 sec.

If we look a the normal x86 structure processor, in this case the AMD K6-2 that is running on 350Mhz we see that if we enable the FPU we see that it calculates the benchmark in about 16 seconds. When we have enabled the FPU on this processor it is clear to see that it takes a very long time before it also has the calculation done. So in those cases we really must calculate on the amount of MHz and the FPU emulator that is compiled within the kernel. So we see that for a lot of daily use things and heavy calculating things we depend on this FPU unit. Also when starting a simple Java swing application with just 1 button on it. On the Zaurus it tool about 20 to 30 seconds to start it, and on the AMD system it was done within the second.

3.1.2 General application and Software

Standard the Zaurus comes with a bunch of software and an Operating System. The Operating System (OS) running on the Zaurus is a Linux OS running a 2.4.18 kernel with some patches put into the vanilla source so it runs without major problems on the device. This Linux distribution is listens to the name of openPDA. On top of the Operating system runs a Graphical User Interface (GUI). This GUI is the Qtopia environment that is created by TrollTech. This company has created several interfaces for mobile devices, for example also for phones. So the Qtopia framework is a proven framework for these kind of devices.

Within this Qtopia environment are also a big set of standard applications. These applications that are meant for the normal average user. Within these ranges of applications we can find an E-mail client, a Word processor, a spreadsheet tool, a scheduler, a to do list and many more applications.

Illustration 4: QTopia application launcher

A lot of the standard applications that are listed in the application launcher like seen in the image on the right are listed in a special process on the Zaurus. This is the Launch process. This process got a list of child processes from applications.

This is needed to launch an application instantly when the application is being tabbed on by the touch screen. This launch process will then start the select application. To make it even more clear the launch process is the main process for the GUI. If you would kill this process the screen will go black and restart the Qtopia environment again. This parent process controls the complete GUI

There are also a lot of standard Linux processes. This are mainly the first processes in the proces list which can be seen when a terminal is started and the command '*ps*' is being used. One of the processes is the init proces. This proces always gets number 1. This is an essential part of the Linux system. When a kernel is started and loaded into the memory and all device drivers and data structures are loaded, it will complete it's own bootproces by starting a program on user level. This program is init. When the init is started and completed, the boot process will continue with a few other administrating jobs. Like for example cleaning up the /tmp directory of all temporary files and starting some network services. In case of the Zaurus it will start a process that monitors the battery and warns us when it's getting low on power, starts the swap procedures, and some other processes.

The used Java version that is installed for the application TICS, which is created by Paul Klapwijk, is a 1.3.1 Java version. It is not the official version from Sun, but a ported blackdown edition. This package only contains the runtime libraries to run an application, so compiling Java application on the device it self won't work.

One of these other processes is the kapm-idled[6] process. This process is used to save battery when the machine stands idle. This process will always run in the background. When the screen of the Zaurus is touched or a key is being pressed this process will stop working, but when the machine is going to an idle state a gain this process will return to save battery power. There is a little bug around this process. If the process is running we see that most of the time this process uses about 97% of CPU power. This is not true, this process takes no CPU power, so this can fool a user by thinking the CPU is very busy at that present time.

Sometimes you are also in need of some custom software, installing this isn't really a problem. There are 2 ways:

- Use a precompiled binary in the form of a ipk package
- compile it yourself

In the case of installing a precompiled binary you can download it from a site that offers them for a download. You can install them by using the ipkg tool. By using the following command: '*ipkg install packagename*' or '*ipkg install packagename.ipk*' to remove the package just change install into remove. This way of installing has a little advantage over compiling a package from source because this way it will be easy to remove it from the device.

If you compile a package by from source you need to setup a cross compiling tree on the system that uses the same libraries and binaries as the Zaurus uses. Otherwise the compiled packages will be incompatible with the system and you will not be able to use them. In some cases cross compiling applications can take up a certain amount of time

because there might be a lot of debugging work lying ahead of you.

3.2 Secure Digital Card (SD Card)

The Secure Digital (SD) Card[4] is used to read and write data from it or to it. So for the Zaurus it is used to store some extra data on it, because the NAND does not support a very large storage space for data. Within the Zaurus and test environment this SD Card is also used to store a swap file on it, this way the memory can write some of its parts to this file and create some free memory for new data. If things are needed from the swap file it will start to swap memory data with data from the swap file.

These 2 operations will take a certain amount of time on the Zaurus. The IO of the SD Card is about 3 Megabyte (MB) per second this is for reading and writing. It can reach up to a clock speed of 25Mhz and besides all that it also has a certain amount of read/write cycles before maybe a failure can occur. This is after about 1.000.000 read/write cycles. With the use of the swap file on the Zaurus this might be reached very fast.

Illustration 5: Secure Data Card

The power consuming cannot be told in a way like with the NIC. In fact the SD Card can take up to 10045mA. Which is far greater then the battery of the Zaurus can deliver. So power that the SD Card needs really depends on how the programming has been done for this device. This little thing is the factor in how the SD Card will be used by the Zaurus.

The SD Card it self can also be set to a read only mode this can be achieved by setting the switch on the SD card it self to lock. This way all data on the SD Card will be protected.

3.3 Network Interface Card (NIC)

The NIC is a model from Linksys [3], the WCF12. This device is the hardware component that is used to build the Ad Hoc network. This device has the possibility to turn the radio signaling on or off. A problem might be that the Linux kernel module which delivers the support for this device does not support this feature. This will mean that when the network is setup you cannot set the radio off and still keep the device powered. So when the device is loaded the power and radio will always be turned on and this way it will keep using the PDA battery power. The only way to let it not use any battery power is to shutdown the complete network, shut the NIC down. When the NIC is turned on it can use up to 3.3V and 250mA.

Illustration 6: The Network Interface Card

This NIC supports 2 types of networks the Managed network, where you use an access point to build the network. Or you can set it in Ad Hoc mode, the network type used for this whole project. The NIC can reach up to a data transfer rate of 11Mbps and can reach a distance of about approx. 100m. There is also an option to encrypt the data that is being send with a so called Wired Equivalent Privacy (WEP) Key. This WEP key can be set to a 64-bit or 128-bit encryption. This encryption is used to protect your data so people that will try to hack the network aren't able to read out the information and change it in to something bad.

This WEP Key is a part of the IEEE 802.11 standard that was ratified in September of 1999. With the WEP key, if it is the 64 or 128-bit version, there are 24-bits used as an initialization vector(IV). In the 64-bit key this means that 40bits are used for the key and with the 128-bit version 104 bits are used for the key. You can make a bigger key , because a bigger key is harder to crack, but in this sort off cases it's just a case of trying harder to crack it. There are other weaknesses, the possibility of IV collisions and altered packets, this cannot be changed by creating a longer key. But for this WCF12 card we cannot use bigger keys for better security so we can only stick to a limit of 128-bit if needed.

4. The Network

The network in this project is also a key element. Rescue workers must be able to keep in contact with each other via the mobile device they are using. So the network has to be setup in an instant. And via this network rescue workers can inform each other for dangerous situations or casualties. So the network in general needs to be a wireless network using a WiFi standard protocol. The protocol that is being used in this project is the IEEE 802.11b standard for the network.

When we start monitoring the network with 2 Zauruses and they are doing nothing we can see that there is no traffic passing by the interfaces at all. This means that there is no useless data passing by the interfaces so the network it self is pretty clean from junk data. This can be seen in Illustration 7 and 8.

```
bash-2.05# time tcpdump -i eth0
tcpdump: listening on eth0

0 packets received by filter
0 packets dropped by kernel

real    4m18.969s
user    0m0.010s
sys     0m0.050s
bash-2.05#
```

Illustration 7: tcpdump clear ahbogdan

```
bash-2.05# time tcpdump -i eth0
tcpdump: listening on eth0

0 packets received by filter
0 packets dropped by kernel

real    2m59.871s
user    0m0.030s
sys     0m0.020s
bash-2.05#
```

Illustration 8: tcpdump clear ahleon

The network isn't doing a thing although the network interfaces are up and waiting for receiving data. The only thing is that neither of them is sending or receiving data at this moment. So when doing nothing this network is pretty clean it self.

4.1 General information about AODV

An Ad Hoc Network is a collection of mobile devices which don't need a centralized access point and are engaged to each other in a cooperative way. All the hosts keep track of the other hosts that they can reach and route the packet to it if this is needed. Each host is some sort of special router.

Why do we need AODV then? We need AODV in situation when there is no fixed infrastructure. The mobile devices all use the OADV routing protocol and this way there is no need for administrative because the devices can decide the route on it's own. There are also a few reasons for this.

1. Maybe some situation does not permit you the installation of an infrastructure.
2. It's not always economical to establish an infrastructure.
3. Establish an infrastructure in a physical way is not possible.

Off course these type of networks can be used in different situations. I will not name them all but the case this report is being written is for when a group of rescue workers are at the scene of for example a disaster. In these cases an Ad Hoc network can be pretty practical.

4.2 Ad hoc On demand Distance Vector

4.2.1 Protocol overview

The Ad Hoc On demand Distance Vector (AODV) [10] is a routing protocol that is intended for us by mobile nodes(the Zaurus) for routing data in Ad Hoc networks. Originally OADV is an extension of the Distance Sequenced Distance Vector (DSDV) routing protocol. We will not discuss that protocol here, we will keep with just AODV. Although it should be mentioned that AODV is designed to improve upon the performance characteristics of DSDV.

AODV has some primary objectives:

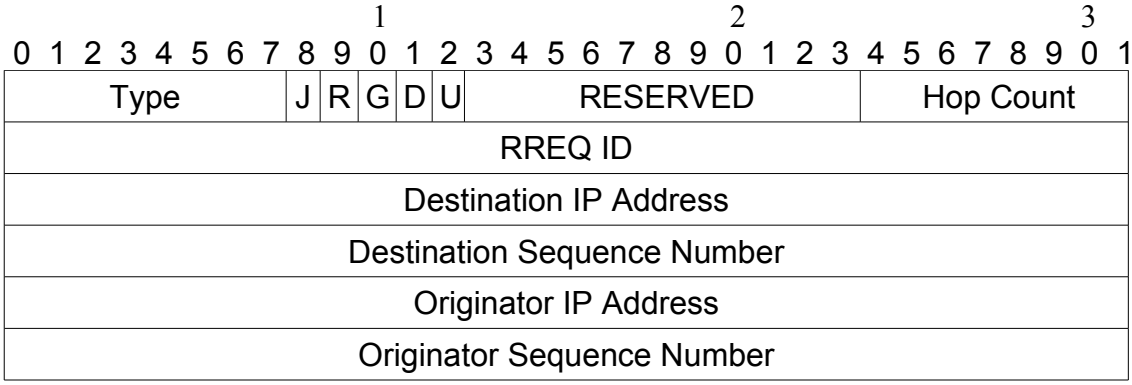
1. To broadcast discovery packets only when necessary.
2. To distinguish between local connectivity management (neighborhood detection) and general topology maintenance.
3. To decimate information about changes in local connectivity to those neighboring mobile nodes those are likely to need the information.

By minimizing the number of broadcasts the control overhead will of AODV will decrease. This is also caused by the use of the pure on demand route acquisition. Each node also got 2 counters to maintain. One counter is the sequence number, and the other counter is the Broadcast-ID. The sequence number will only be increased to keep the information up to date about the reverse route to the source. The Broadcast-ID will only increase when the source issue a new Route Request Message (RREQ). Next to that all hosts maintain the information of reachable hosts with a bi-directional connectivity. All AODV traffic use the UDP datagrams and is being send over port 645.

4.2.2 Path setup

RREP

Below we see how a RREP packet look like and also what the content of the specific packet fields are. Below this all is described what it exactly does.



RREQ packet

Type	1
J	Join Flag; reserved for multicast.
R	Repair Flag; reserved for multicast.
G	Gratuitous RREP Flag; Indicates whether a gratuitous RREP should be unicast to the node specified by the Destination IP Address field.
D	Destination Only Flag; This indicates only the destination may respond to this RREQ.
U	Unknown Sequence Number; this indicates that the sequence number is unknown.
Reserved	Sent as 0; ignored on reception.
Hop count	This is the number of hops from the originator IP address to the node that is handling the request. This will get increased with 1 on every host the packet passes.
RREQ ID	A sequence number uniquely identifying the particular RREQ when taken in to conjunction with the originating node's ip address.
Destination IP	The IP address of the destination for which the route is desired.
Destination Sequence Number	The latest sequence number received in the past by the originator from any route towards the destination.
Originator IP Address	The IP Address of the node that originated the route request.
Originator Sequence Number	The current sequence number to be used in the route entry pointing towards the originator of the route request.

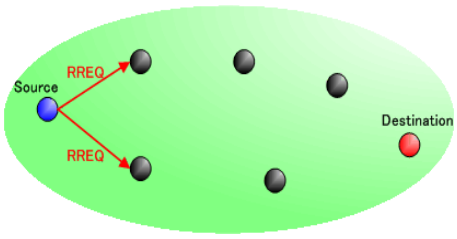


Illustration 9: Setup AODV

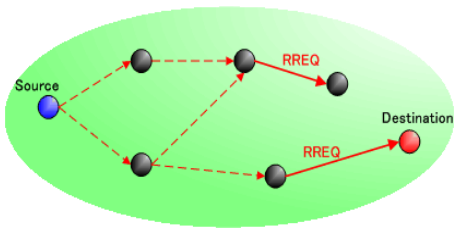


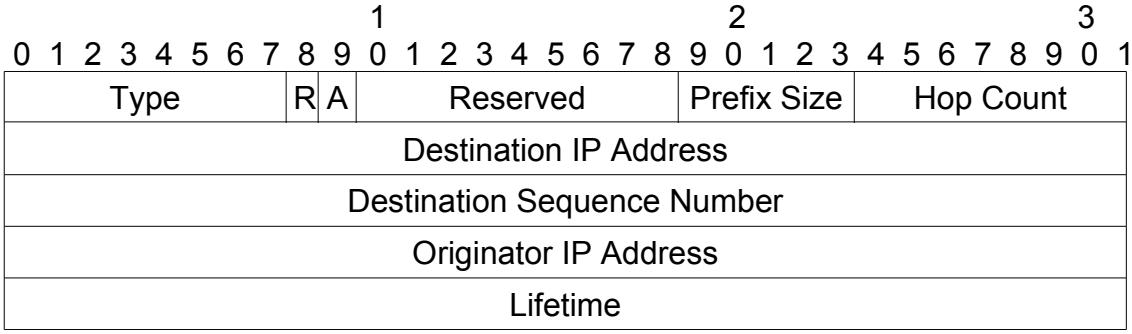
Illustration 10: Finding neighbors

When the path discovery process will start a RREQ packet like above will be send out to all neighbor hosts with all the information that it needs to contain. The Hop Count will be set to 0, and every time it passes a host it will be incremented with 1 value higher. A mode that receives a RREQ packet first checks if it was received over a so called bi-directional link. If it hasn't it will check if it has a route entry to the requested destination. When this bi-directional links isn't the case it will check if has processed a similar RREQ package, if this already happened the packet will be discarded. When the node has a routing entry in its route table for the destination only then will it reply back to the source, but it will only reply

back when the Sequence number in RREQ is greater than the Sequence number in its route table. If this isn't the case the RREQ packet will be rebroadcasted. Then a reverse path will will be established as the RREQ travels closer to the destination node.

RREQ

Below here we see how a RREQ packet look like and also what the content of the specific packet fields are. Below this all is described what it exactly does. This packet is the response on an RREP packet.



RREP packet

Type	2
R	Repair Flag; reserved for multicast.
A	Acknowledgment required.
Reserved	Sent as 0; ignored on reception.
Prefix Size	If nonzero, the 5 bit prefix size specifies that the indicated next hop may be used for any node with the same routing prefix (as defined by the prefix size) as the requested destination.
Hop count	This is the number of hops from the originator IP address to the destination IP address. For multicast route requests this indicates the number of hops to the multicast tree member sending the RREP.
Destination IP	The IP address of the destination for which the route is desired.
Destination Sequence Number	The latest sequence number received in the past by the originator from any route towards the destination.
Originator IP Address	The IP Address of the node that originated the route request.
Lifetime	The current sequence number to be used in the route entry pointing towards the originator of the route request.

When the RREQ reached the destination, the destination node will respond with a RREP packet, having almost similar fields as the RREQ packet. A RREP packet travels back to the source using the created reverse path. All nodes in between the source and

destination will create a new route entry for Destination if this is necessary, also set the next hop from which the RREP packet came from, then records the hop count and Sequence number and update the active neighbor list with the node the RREP packet came from. And last but not least the information for the timeout will be updated for both destination and source entries. When the RREP packet reaches the source, the route table entry is modified only if the Destination Sequence number is greater or equal for the new route to the Destination Sequence number in the current route. Otherwise the route will be discarded.

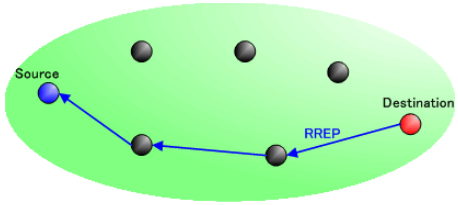


Illustration 11: Route Setup

RERR

The packet described below here is the error packet that will be send when an error occurs on a request. In this case whenever a link breaks . This will cause one or more destinations to be come unreachable from some of the node's neighbors.

0 1 2 3 4 5 6 7	8	9 0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9 0 1
Type	N	Reserved	DestCount
Unreachable Destination (1)			
Unreachable Destination Sequence number (1)			
Additional Unreachable Destination IP Address (if needed)			
Additional Unreachable Sequence Number (if needed)			

RERR packet

Type	3
N	No Delete Flag. Will be set when a node has performed a local repair of the link. When it is set an upstream node will not delete the route.
Reserved	Sent as 0; ignored on reception.
DestCount	The number of unreachable destination hosts included in the message; this value must at least be 1.
Unreachable Destination IP Address	This is the IP address of the destination that has become unreachable due to link break.

Unreachable Destination Sequence number	The sequence number in the route table entry for the destination listed in the previous Unreachable Destination IP address field.

4.2.3 Maintenance

Path

When a source node is moving the complete route discovery procedure must be done again to find a new route to the destination. In case when a node between the the Source and destination move or the destination it self moves then there will be send a new special RREP packet towards the source with a new Sequence number and a hop-count to infinity. The sequence number that will be send is greater then the previous known sequence number.

Route table

The route table on each node contain the same entries that need to be kept up to date. This information is the following:

1. Destination
2. Next hop
3. Number of hops
4. Sequence number for the destination
5. Active neighbors
6. Expiration time

When a new route is being presented the route entry will be updated based upon the information from the hop-count and the sequence numbers.

4.2.4 Local Connectivity Management (LCM)

A node learns from his neighbors in 2 ways.

1. Node receives a broadcast from a neighbor.
2. Node receives a 'HELLO' message with it's identify, sequence number and TTL.

A HELLO message is broadcasted by a node to all it's active downstream neighbors, when it has not received an packets of any of it's neighbors withing the interval. This interval is called the 'HELLO_INTERVAL'. When a node doesn't receive a hello message from its neighbors along an active path the upstream node will receive a link failure notification on the path.

4.3 Results of the AODV tests

For creating the results we had 3 Zaurus PDA's and they all had names:

1. Zaurus 1 called ahbogdan

2. Zaurus 2 called ahleon
3. Zaurus 3 called ahsiska

```
bash-2.05# tcpdump
tcpdump: listening on eth0
01:14:47.213462 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:48.223317 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:49.223368 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:50.233312 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:51.233369 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:52.243312 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:14:53.243369 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
```

Illustration 12: tcpdump directly after the AODV is started

These tests where done while using the *kernel_aodv* kernel module for the Linux system that is present on the Zaurus. As soon as the kernel module has been loaded we see that it is starting to broadcast his RREQ packet, and hoping a neighbor will respond on it to create a path with it. Like mentioned earlier each packet will have a TTL (Time To Live) with a value of 1. This way a packet will be received by one host and won't be thrown in to the network to find new nodes. This AODV protocol uses UDP diagrams to broadcast it's packets. Unlike TCP the UDP diagrams never give a response back if a packet has been received by the other side, so you just have to hope that they received it. Like illustration 3 shows us, the OADV broadcast uses the 654 port to broadcast it's packet over the network. Also we see that all packets will receive a TTL of 1 like it should be.

This will also show us that the route table isn't being changed (Illustration 13). This is explained by the fact that there wasn't an other Zaurus broadcasting AODV packages within his range.

```
bash-2.05# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.255.255.0   U     0      0        0 eth0
bash-2.05#
```

Illustration 13: Route table

But what happens if we put an extra Zaurus broadcasting within the network. We should see 2 systems broadcasting it's AODV packages, but we should also see a change in the route table. So we should see that there is a route possible between the 2 Zauruses.

```
bash-2.05# tcpdump
tcpdump: listening on eth0
01:17:46.767477 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:47.113322 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:47.767049 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:48.113322 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:48.767004 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:49.113417 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:49.767372 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:50.123323 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:50.767319 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:51.123323 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:51.767121 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
01:17:52.123421 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
```

Illustration 14: Two Zauruses broadcasting the AODV protocol

As we see in Illustration 14, we see that this Zaurus is now also receiving packages from a neighbor. Which means that they are able to communicate with each other. This same thing would happen if we put a 3rd Zaurus next to these 2. The route table change can be seen in the image below here, illustration 15, this shows us that the Zaurus ahbogdan and ahleon will be able to send information to each other.

```
bash-2.05# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
ahleon          ahleon          255.255.255.255 UGH   0      0        0 eth0
ahbogdan        ahbogdan        255.255.255.255 UGH   0      0        0 eth0
10.0.0.0        *               255.255.255.0   U     0      0        0 eth0
```

Illustration 15: Route table change after 2 Zauruses broadcasting

So we can say that the routing protocol AODV in this case does it's job by altering the routing table to it's environment. When ahleon was takes away again the routing table was back to the situation like in illustration 13. The result of this test shows us how the AODV routing protocol does its work and dynamically alters the routing table. But there is also the principle of multihop within these networks. To try and test the multihop there was thought of the following. Take the PDA's ahleon and ahbogdan and put them as far away from each other till they can't see each other anymore. If this is done place a 3rd PDA in between ahsiska and see if ahbogdan can find ahleon. So it looks like the following:

Illustration 16: Multihop

So with this setup within AODV ahsiska routes the packages between ahleon and ahbogdan so they can exchange data.

First we will show the tcp dump and route table of ahsiska, then the tcpdump and route table of ahbogdan and finally the screen shots from ahleon.

```
# tcpdump
tcpdump: listening on eth0
14:21:09.143395 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:09.151909 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:09.281607 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:10.153386 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:10.162043 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:10.292217 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:11.163389 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:11.172033 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
14:21:11.301598 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
```

Illustration 17: tcpdump ahsiska

As we see here this device picks up the AODV requests from both the devices like it was described a bit above here. So this parts works fine of the network, now we need to check if the routing table was altered and has both other devices registered in them.

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
ahleon          ahleon          255.255.255.255 UGH   0      0        0 eth0
ahbogdan        ahbogdan        255.255.255.255 UGH   0      0        0 eth0
ahsiska         ahsiska         255.255.255.255 UGH   0      0        0 eth0
192.168.129.0   *               255.255.255.0   U     0      0        0 usbd0
10.0.0.0        *               255.0.0.0       U     0      0        0 eth0
```

Illustration 18: route table ahsiska

We see in illustration 18 that both other devices ahbogdan and ahleon are listed in this routing table, this means ahsiska can communicate with both of the devices. When we take a look at the tcpdump from ahbogdan we will see that it only gets the RREQ packets from the mobile device ahsiska. Because it can't see ahleon and has to communicate through ahsiska with ahleon. This is also caused by the TTL on the packet which is 1. This means the packet only is allowed to make 1 hop within the network. After this the packet needs to be “destroyed”.

```
# tcpdump
tcpdump: listening on eth0
23:44:14.358365 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:14.891574 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:15.368459 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:15.901602 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:16.378498 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:16.911574 ahbogdan.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
23:44:17.388124 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
```

Illustration 19: tcpdump ahbogdan

This more clear to see that ahbogdan can't communicate directly with ahleon as we look at it's route table, which we can see in illustration 20. Here we only see that ahsiska is listed and that his own address is in it.

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
ahbogdan        ahbogdan        255.255.255.255 UGH   0      0        0 eth0
ahsiska         ahsiska         255.255.255.255 UGH   0      0        0 eth0
10.0.0.0        *               255.0.0.0       U     0      0        0 eth0
```

Illustration 20: route table ahbogdan

The final mobile device in this multihop setup is ahleon. We can also see here that it can't find the node ahbogdan and only finds ahsiska. The reason for this is the same as for the node ahbogdan. So ahleon needs to communicate with ahbogdan through ahsiska.

```
# tcpdump
tcpdump: listening on eth0
09:42:35.425691 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
09:42:35.449160 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
09:42:36.435690 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
09:42:36.449094 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
09:42:37.445690 ahleon.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
09:42:37.459091 ahsiska.654 > 255.255.255.255.654: udp 20 (DF) [ttl 1]
```

Illustration 21: tcpdump ahleon

This we can see in the route table like in illustration 22. It almost looks the same as the one from ahbogdan but instead of that it says ahbogdan as own address it notes ahleon.

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
ahleon          ahleon          255.255.255.255 UGH   0      0        0 eth0
ahsiska         ahsiska         255.255.255.255 UGH   0      0        0 eth0
10.0.0.0        *               255.0.0.0       U     0      0        0 eth0
```

Illustration 22: route table ahleon

So with this we get a bit of a view how the multihop will be setup. If more nodes will join the network route tables can grow even bigger. It really depends on what other neighbor nodes will be found.

4.4 Developed applications for this network

Over the years some people at the University of Delft have been trying to develop all kind of application that will run on this kind of network. All these applications have a different point of view on how it should be done and what it should do. Here I will list a few of these applications and will tell a bit about them. The shortlist is the following:

1. ISME by Paul Schooneman
2. Lingua by Siska Fitrianie
3. PDA Assistant by Dragos Dactu

In all these cases there is a back end blackboard structure which makes it easy to distribute all the information to other mobile devices that are connected with the network. The blackboard that is used in all cases except for the PDA Assistant is a Lime blackboard.

4.4.1 ISME

The ISME program[9] is an icon based system written in Java which is included in the TICS application. This system can be used by emergency services to keep each other up to date about what is going on in a particular area. This will be done by placing icons on a map that is uploaded from the server to the PDA. With placing these icons we will get a structured wold model from the crisis that is going on at the present time.

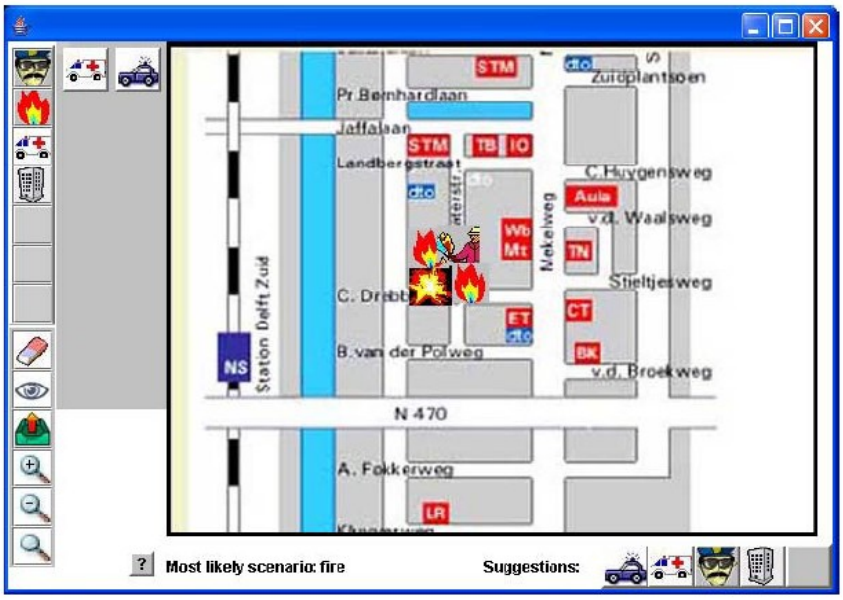


Illustration 23: ISME Client

The reason that this icon approach was chosen is because icons are easy to recognize and that way it can very easy represent something. Look around you there are icons every where, icons that tell you where to find the toilet, icons that say

you where the emergency exit is, even traffic signs are icons.

The graphical user interface from ISME is also based on icons. These icons are also used in the other applications that have been developed. Like said earlier these icons can be placed on a map and these changes will be send over the network to the other PDA's that are within the Ad hoc Network. The user interface looks like in the image below.

This application also has an intelligent module to suggest a certain icon to be placed in a specific situation. The back end of the application is that it creates an XML structure that will be send to the blackboard that is available and from there the other mobile devices will receive this data that has been send.

With this application the user just puts the icons on the map and in places how he/she is judging the crisis situation that is happening at that time. The created map gives a complete overview of the situation on what is going on at that time.

This application is also used for the TICS test to see how the Zaurus will react when using this application.

4.4.2 Lingua

This application is created by Siska Fitrianie[11] . Lingua is an other type of application then ISME. Here the user places icons on a screen and with these icons a text is formed. This text will tell what is happening at that moment. As we look to illustration 24 we see that a firefighter is searching for victims in a building. This is a whole other kind of approach then just placing icons on a map. So instead of creating an visual picture from the complete situation this application is based on sending strings that are created by icons. These strings will be the sentences that are created by the icons. So the application also contains some intelligents on creating correct sentences. So it knows how to follow the grammar rules and create correct speakable sentence.

Illustration 24: Lingua application

We also see that the same suggestion icons are also available within this application. This is the same intelligents we saw with the ISME application. That is will suggest what icon is will probably expected to be add to the situation.

The icons speak a sort of universal language, which is also the case for the other programs, and this way it will be easy to talk with other rescue workers that are at the scene of a crisis. The only point is the application was never explorered to and used to get a good view on this application.

4.4.3 PDA Assistant

This application is written by Dragos Dactu and has a whole different map approach then the ISME application. With this application you can select a special area on the map , within this area you can select a little part in the form of a circle or square. And within these subselections on the map it is possible to place icons that are based on the situation. These icons are placed in an icon bar and then will be send to the central server that spread them around to the other PDA's within the network.

Illustration 25: PDA Assistant application

The icons are the same as in the other 2 applications, simply because they are easy to understand and the ease of use with this application is also a bit better then when looking at the ISME client. When tapping the icon the icon appears in the icon bar we see on the right in illustration 25. This application also has a little advantage above some other applications, with this application it is possible to send text and photos around which isn't possible with for example the ISME client. These pictures can be taken with any mobile device that is running the application and then can be send over the network to the other mobile devices and the center so they get some sort of real life view of what is happening.

So this application got the best of both of the earlier discussed application. The way to visualize the environment you are in a certain way. Not as good as ISME but you can see in a certain way what is going on, and there is this opportunity to send over text messages. But these can just be typed instead of created with icons. Finally the extra thing that none of the other applications got is the picture sending option. This option will give the best view of what is going on at the moment of a crisis.

Illustration 26: Picture that can be send/recieved with the application

So this application has a little better approach on this point. Also when we look at back end this appication does not use the Lime blackboard and Java rmi registry. This make this application work more real time then the ISME application. This application uses it's own created communication protocol which is faster then when using the rmi registry. The only thing that still needs to be developed is a blackboard structure which probably won't be a Lime blackboard but some other own written blackboard. At this time this is still not completely clear to me.

5. Performance of the Zaurus

This chapter will cover the results from the performance tests. And will show how things react based upon the earlier produced test report. All tables and graphs can be found in Appendix A. So in some cases when a graph is not shown it can be found there.

5.1 The battery

The main thing of a mobile device is that you can take it anywhere you want, whenever you want. This means that it has to be powered while you are taking it. So a battery for a mobile device is an essential part. Otherwise you always need to walk around with a very long cable to some kind of power supply, and sometimes there isn't even a power supply in the neighborhood.

Illustration 27: Battery

Generic information
Before we could even start the tests the battery needed to be checked and rechecked. Because the batteries had been lying around for a few months they weren't as powerful as they should be. This caused some sort of trial and error tests to find out if the batteries where really broken. After a lot of trying different things and still had the same result that recharging took too long and that they were losing power too fast, it was finally concluded that they were broken. So a new set had to be ordered. The old batteries where 1700mA, the new ordered ones were 1800mA, this means that these batteries are a bit more powerful then the old ones and should last a bit longer. After a little first tries with these new batteries it was a real difference. Not only was charging them faster, also the lifetime was longer. The APM tool that is delivered with the Zaurus isn't that advanced as acpi, but the running kernel on the Zaurus has no support for this last method. So all values about how much power the battery has are recorded in the following steps: 100%, 75%, 50%, 25% and 5%. There will be some more info about this power management in a later.

5.2 Processor usage

The usage of the processor is divided in 2 ways. There is the user environment. And the system environment. The usage of the user environment will rise at the moment a processes will be started as a user. This will also be noticed when the tests will be started. The system usage on the other had will rise off course when there are system tasks that will be performed by standard routines or procedures. This happens occasionally on the system and within a Unix like environment this always happens.

Generic information
The biggest point was to look and find what kind of standard procedures that could be detected during the tests. To name a few the processes for the file system, the process for swapping (kswapd), the idle daemon, the init process, the process for the file system things (bdflush, mtdblockd,jffs2_gcd_mtd3)) the battery charge on/off processes, and a lot more of these processes.

5.3 System load

The system load is caused by the programs that run on the system. For single CPU systems that are CPU-bound (where there are on average no processes in uninterruptible sleep), one can think of load average as a percentage of system utilization during the respective time period. For systems with multiple CPUs, the number needs to be divided by the number of CPUs in order to get a percentage. So for example if there is a reading of 4.65 of load we must multiply it with 100 to see the CPU being overloaded by 465%. So it means that if it had 4.65 CPU's it could have handled the work much faster.

It was noticed that in idle ways the system load was always around 1.00 and never below it. After some searching the reason to this was found and will be mentioned later on.

5.4 Memory and Swap usage

The memory of the system is an important part of the system. It is an electronic form of storage for the system. When a CPU constantly access the hard drive to retrieve every piece of data you need , the system would be operating very slow. So when the information is kept in memory the CPU can access this data much faster and there for this is a temporarily storage option. In the figure next to here we see the higher it gets the faster it can be accessed for, cache is even faster then the RAM memory. And hard disks are lower in the pyramid. This means that these devices are slower in accessing for data. So to be sure that you can access your data faster the CPU will set some parts of the data in memory. But in some cases memory isn't enough. To extend the memory a bit you can add a swap partition to the drive or a swap file. In this case when the memory gets full some parts of the memory will be written to the swap storage till is it needed again and then it will be set back in RAM memory. So eventually swap is a slow memory storage because it is on a hard disk. And because of this it is also a bit CPU intensive.

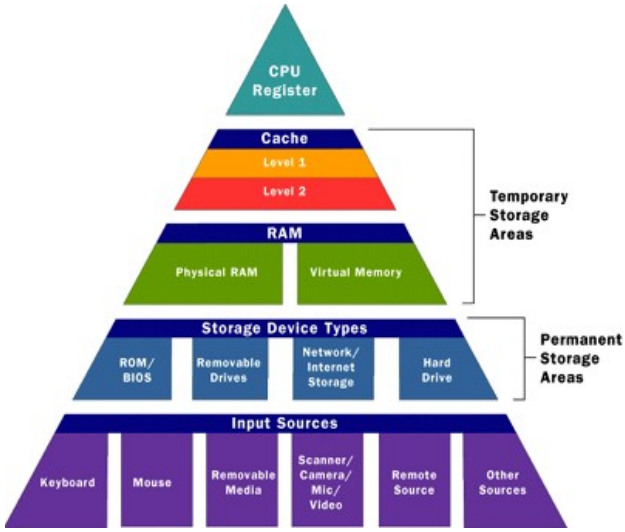


Illustration 28: Data storage pyramid in speed order

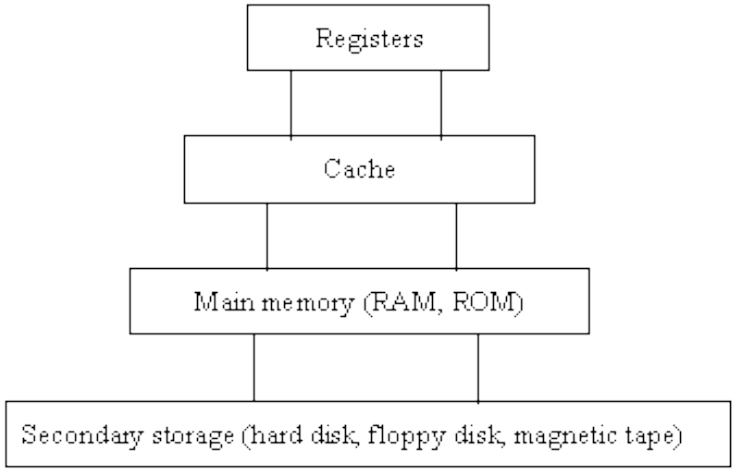


Illustration 29: The memory management simple version

Generic information
Before the tests could be started there were a lot of troubles with the memory. The total amount of RAM that is inside the Zaurus is 64 Megabytes. Because the Zaurus only has 64 Megabytes of RAM we can use before programs we got an out of memory error. So after some little trying I figured out that the best way as a work around would be to add a swap file. This way we were able to extend the life time of the machine a bit more. The extra add swap file was created on the SD Card and was about 30 Megabytes big. This solved the out of memory problem. So on all Zaurus I installed a new little init script that first checks if there is a swap file on the system, if not it will be created and activated. If the swap file is available it will only be activated.

With this script it was also very easy to disable the use of the swap file. Because some of the tests like mentioned were performed without this swap file.

The shell script can be found in Appendix C. The script is located in the /etc/rc.d directory and there is a symlink in the /etc/rc.d/rc.5 so it will be started on startup.

The second part was to find out what kind of memory was used within this machine. After some searching it was discovered that the RAM that was being used was SD RAM running on 100MHz. SDRAM stands for Synchronous Dynamic Random Access Memory. This type of memory can be run on clock speeds up to 133Mhz. SDRAM has a synchronous, clocked interface that is faster then any other kind of asynchronous type of DRAM module which does not have a clock input.

5.5 Design of tests

To start the performance tests there where a lot of situations that could be looked at, but it is only useful to look at those situations that will occur more often. So at first it was just trying some things if they where possible and also get a little hint of what might be happening. After a lot of these little tests where done a little list was created. This list contains all tests that are useful to look at. The tests where then divided in performed idle or performed as stress test. This created the following lists.

Idle tests
- Idle without extra hardware components
- Idle with only the NIC
- Idle with only the SD Card
- Idle with both hardware components NIC and SD Card

With these idle tests we wanted to know how long the Zaurus will stay up on a full battery and also how different things will be going on on the Zaurus

The stress test list looked like this:

Stress tests
- Copy a file over the NIC to the NAND (Flash) and continue this over and over
- Copy a file over the NIC to the SD Card and same as above over and over
- Copy a file between NAND and SD Card
- Use the Zaurus full with SD Card mounted but without NIC
- Use the Zaurus full with SD Card mounted but this time with the NIC

With these 5 stress tests we also wanted to see how long the Zaurus can stay up in these different extreme situations and if it takes much power away from the battery. These stress tests are mainly based on the idea that these are the most frequent forms that will be happening on the Zaurus.

As a final test we ran the TICS application from Paul Klapwijk, and see how the Zaurus would react on this application that was created at the University.

The interval that is being used to check all the data is every 30 minutes. The only test that isn't being measured every 30 minutes but every 10 minutes it the application TICS. The main key things that will be measured are:
1. System load
2. Processor usage (user/system processes)
3. Memory/Swap usage
4. Battery usage.

5.5.1 Idle test: without any extra hardware components

With this experiment we will let the Zaurus running till the battery will runs out of power. This way we know what the battery will be capable of when absolutely doing nothing at all with this system.

The expectation of this experiment is that this test will defiantly take the most time of all. Because it will do absolutely nothing with no extra hardware that is using battery power.

Result
This test ran for more then 9 hours. This can also be seen when we look at illustration 30 that shows the battery power level. It is strange to see that the longer the test runs the

smaller the steps are getting to go down an other level on the apm indicator. I did expected the steps to be more even from 100 to 25% because this are chucks of the same size, so that it would go down 25% every 170 minutes. But instead it stayed at 100% for about 4 hours before it went to 75% and this is almost half way it's uptime during this test.

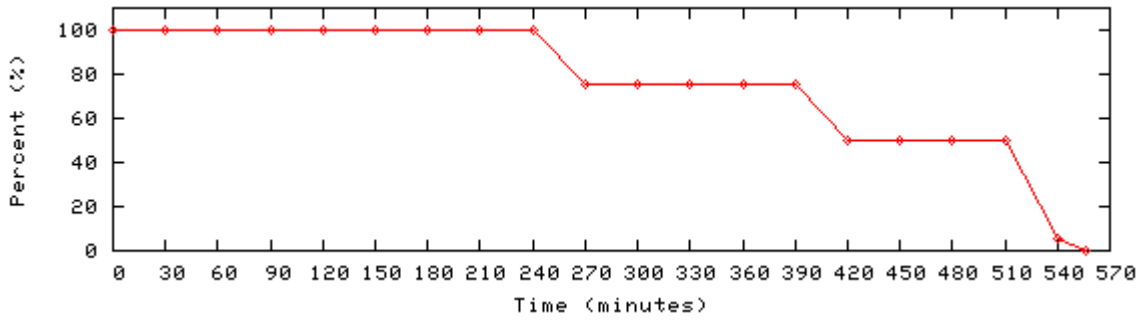


Illustration 30: Idle without SD and NIC - Battery usage

The memory on the other hand stays the same throughout this whole test. Only in the beginning it was a bit higher and after an hour there was probably some kind of procedure running on the background and freed some of the used memory. After this memory freeing it is just a horizontal line. This was expected because the machine it self isn't doing anything but standing and let the battery get down to 0%.

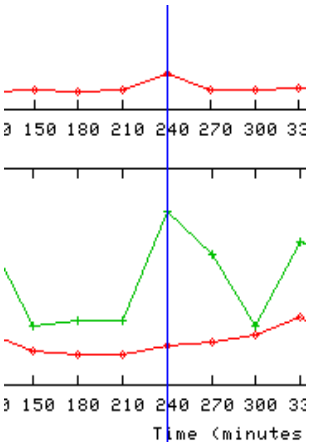


Illustration 31: Peak idle test half way

The system load throughout this whole test is also almost a straight line. We see that this line is always above 1.00, this has a particular reason that will be explained a bit later on. There is a little peak at the beginning of the graph and some where in the middle. This peak at the beginning is easy to explain because at that point some of the initialization processes weren't completed yet. So this explains that little peak around 2.00. On the other hand the little peak in the middle of this graph is caused by some other reason. The picture next to the text is a little shot from the complete graphs. The green line of the below graph and in that same graph the red line is the user process usage. The red line from the upper graph (illustration 31) shows us the system load. The blue vertical line is used to show that it is exactly in line. The green line shows us the system process usage. And there we see that the system suddenly has some task to do. So at that point the system load automatically rises also. The other peaks of the system processes had no big effect on the system load and this can be seen throughout the complete system load graph.

The user and system processor usage are just fluctuating a bit like it should and I was expecting some kind if pattern on this kind of graph. Simply because of the tasks the system has to perform and the procedures and system calls that are being send to parts of the system like for example for the application launcher which is a user process or the kapm-idled daemon which is a system process.

5.5.2 Idle test: without SD Card and with NIC

When performing this test there will also be a minimum amount of network traffic over the NIC. This will be in the form of a simple ping from an other Zaurus or system within the network. Just like in the previous test the measurements will be done every 30 minutes.

My expectation about this test will be that the battery will run out of power faster then without the NIC. Simply because there is an extra hardware component add to the device and this hardware device needs power. The problem will be like discussed earlier in Chapter 3.3 about the NIC that it can't be set in a sleep mode when it is idle due to a driver problem within the Linux kernel module, this will mean that the NIC will always be turned on.

Result
When we look at the graphs from this test and especially the battery results first. We can see that the uptime of the Zaurus is shortened to a bit more then the 4 hour mark. The reason for this is described in chapter 3.3 that the NIC can't be put into a power saving mode. This causes the NIC to be turned on the whole time. So this means that the NIC is using power throughout this whole test period. And shortens the uptime to about 50% of the idle uptime without any additional hardware components. What we do see is that the steps from 100% to 75% and from 75% to 50% have the same size, so here it is shown what was expected in the previous test. Only here the 2 parts are both 90 minutes big. After these 2 steps, the other steps are shorter again.

If we look at the system load, we can notice that the line constantly is around the 1.00 or a bit above it, but not below it. So idle really is idle and means that the system is doing nothing.

The memory usage with this test is just almost a straight horizontal line. It started a bit higher, then some memory was freed again and then it was rising again a bit. This behavior seems to be caused the the loaded kernel module for the NIC. But there is nothing really fancy to note about this memory graph. We can only say that the memory management of the kernel is doing what it should do.

Also looking at the user and system processor usage there is nothing happening here that can be pointed out like something special is happening. Most of the peaks we see at the system processes we see are just the normal routines that are performed and those are sometimes more CPU consuming then other times. In the mean time the user processes aren't doing anything special either. So this is just doing what is will always do when the machine is being idle like we also saw at the results from 5.5.1.

5.5.3 Idle test: with SD Card and without NIC

With this test only the SD Card will me mounted into the Zaurus. But the swap will be turned on to see if it will be used during this idle test. And the NIC will be taken out. The rest of the test will be performed the same way as in 5.5.1 and 5.5.2.

The expectation will be that this test will reach a higher uptime then standing idle with the

NIC alone. This is because the SD Card will likely be only initialized in the beginning and after that it won't be used anymore, because the Zaurus isn't doing a thing so it won't have to swap either. This means that there is no power that will be used further in the process of this idle test.

Result
Looking at the results from this little experiment we see that the uptime is a little bit shorter then from the idle test without any extra hardware components. It is a bit more then 8 hours uptime. So here we see that a little battery power got lost to some of the things that had to be done with the SD card. This was mounting the SD Card to the system directory */mnt/card/* and also initialize the swap file that was located on the SD Card.

The system load is like the previous 2 results still around 1.00 or a bit higher, it never got below the 1.00.

When looking at the memory and the swap we see that the memory it self a round 2 hours freed some of the unused memory. While 1,5 hours later there was some job on the background that allocated some memory again. What has caused this allocation of memory is unknown. It was only noticed when writing down all the values. What was noticeable and expected, is that the swap file was left unused. Because the memory it self wasn't going to its maximum it wasn't needed to swap parts of the memory to this swap file.

Also looking at the processor usage with the user and system processes we see that there is nothing special happening when looking at the results. There are no special high peaks that are related with other things on the moments of measuring. Only the standard peaks of system tasks that are performed.

5.5.4 Idle test: with both hardware components

This final idle test will be done with both hardware components mounted. The test will be done the same way as the others.

The expectation with this test will show that this test will be the shortest of all idle tests. The thing that will cause this are the 2 extra hardware components that are being add to the device. And information from the 2 idle tests all ready showed us that the hardware components use some of the battery power. Also there might not be any use of the swap space not be used and many memory fluctuations. Also system and user processes might show a same kind of pattern as in the other idle tests.

Result
For the battery this idle test was the shortest of all 4. The reason for this is a combination of the 2 hardware components from the tests from 5.5.2 and 5.5.3. At the test from 5.5.2 we only had the NIC mounted on the device and this caused the uptime to go down by almost 50%, but we also now have the SD Card mounted at the same time now and this also caused some power loss described in 5.5.3. So those 2 things are the key factors of draining the most power from the device. And there for it was the shortest.

Also here we see that the system load doesn't do anything special. We see that also here it stays around the 1.00. This number that we see must be multiplied with 100 so we get the complete utilization of the system load. Because the system isn't doing anything special that will cause the load to rise on the system. This can be seen in illustration 32

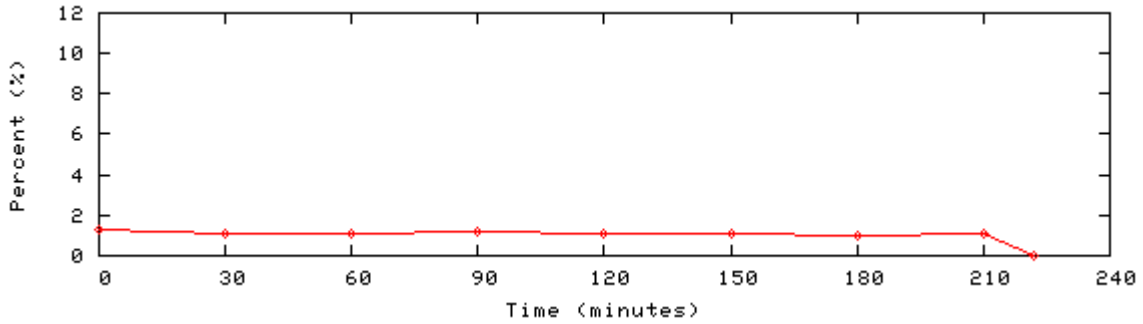


Illustration 32: Idle with SD and NIC - Processor load

The memory in this test when we compare it with the one from the previous test where we had the swap enabled for the first time doesn't have any point where there will be some memory freed. The swap also isn't being used with this test. This is logical and has the same explanation as was given in the previous test result from 5.5.3.

Also the user and system processor usage don't have anything special like in the other idle tests. So in all idle cases the system is just doing what it is doing all the time. Performing its standard procedures and routines.

5.5.5 Stress test: copy a file over the NIC to the NAND

Because the Zaurus at this point doesn't have that much internal memory to store files this test will be performed with a small file of Approx. 3 Megabytes. The SD Card will not be mounted during this test, to make sure there will be no I/O used for the SD Card.

It will be expected that this test won't take too long, because of the many things that is being done withing a returning period. The NIC will use up power from the battery, also the system load will probably rise during this test. Also there might be a high user load here, because of the user process will be used to copy/remove the big file that is used for the network copy procedure.

Result
With this test one of the expectations did can out and that was that the battery was going down faster. This took a bit more then 2,5 hours before the battery got to the 0%. This off course is caused by a few things. One of them is the hardware component the NIC. This interface is a good power consuming component within this whole setup and probably also a big one. Also an other thing is at this point the CPU load. Because the CPU load is higher then normal the system has to work harder. This causes the CPU to consume more power. These 2 things are at this point the 2 key factors for a shorter uptime of the device.

When we look at the system load we see that the system load is rising and keeps hanging

around a system load of 4. This is simply caused by the written shell script that keeps running in the background and performs this copy sequence in an endless loop. This loop ends on the moment the power of the battery is down to 0%. This higher system load is caused by the secure copy process that runs under user privileges. This is also what we can see in illustration 33.

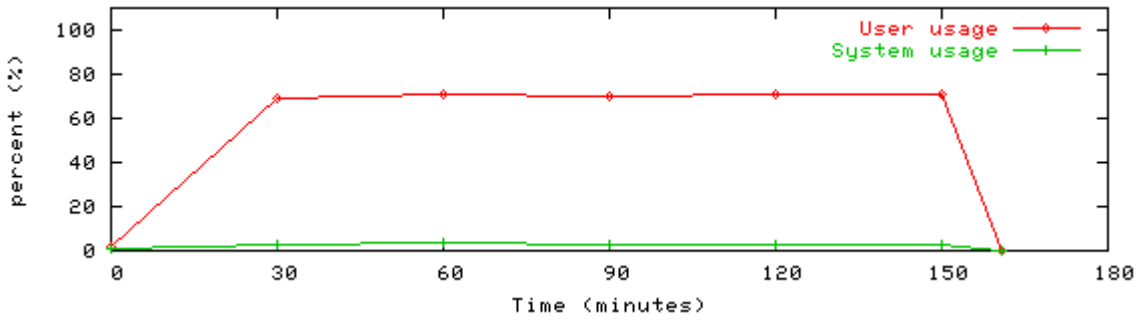


Illustration 33: Copy over NIC to NAND - User/System processor usage

What we see here is that the system processes require a lower and less CPU time while the secure copy sequence takes a lot of the CPU time to copy/write a file to the system. The media where this is written to is a slow data storage type. But also one of the key things is that the binary that is needed runs on the same medium where the file needs to be copied to. This causes the the point that the CPU gets more Input/Output on the NAND and that it is all harder to complete the tasks.

The memory on the other hand rises slowly towards the maximum, but then it gets to that point it is staying the same. So the memory management isn't doing any thing special when we look to this. Only that the process that runs for the whole copy sequence took some memory but this isn't that much.

5.5.6 Stress test: copy a file over the NIC to the SD Card

To make no separation in the traffic tests over the NIC we will use the same kind of file and method for this test. Like in the section above the swap will not be activated so all the read/write cycles will all be used for this test.

It can be expected that almost the same kind of values will appear like from the test 5.5.5. At least some same kind of patters. Because the same applications will be used as in this previous test we shall probably see that the the user will use the most CPU time.

Result
When we look at the battery usage during this whole sequence it was notices that it was noticeable that it took almost the same amount of time to go down. This test took a little bit longer. All the things where the same but the system load was a little bit lower then in the test from 5.5.5. This system load that was a little bit lower caused this test to run a little bit longer then the previous one (section 5.5.5). It was almost near the 3 hour mark.

During this test we saw and mentioned earlier that the system load during this test was

lower then when we copied a file to the NAND. Because the I/O that was needed to write the file wasn't needed for writing to the NAND this time but for writing to the SD, there was less Input/Output for the NAND. The same binary was still on the NAND that was used but the there was no extra data for writing to it. This part was now used on the SD Card. So the whole time the system load was a bit below 3. This was also seen back in the processor usage figures. The user environment had a lower number on the measurements during this test then when performing the test from 5.5.5. This time it wasn't around the 80% but about 10% less. The system environment on the other hand was a little bit higher.

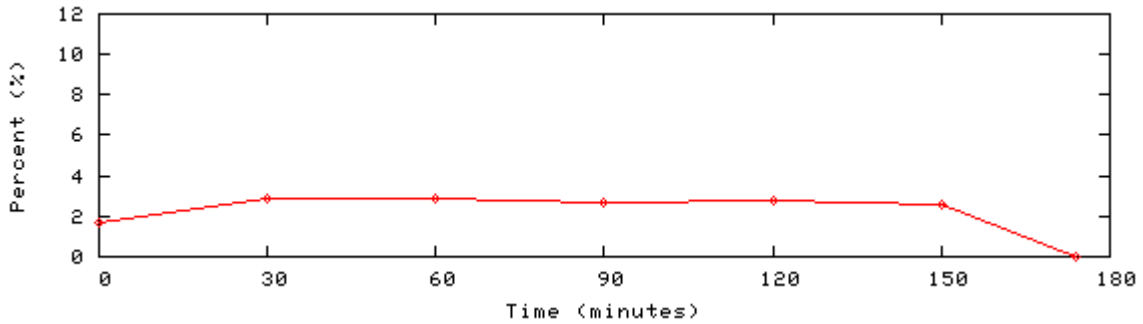


Illustration 34: Copy over NIC to SD – Processor load

The system environment was instead of under the 3.0% all above the 3.0%. The SD Card also needs its Input/Output, so to control the SD Card the system needs to address the file system it self. Cause every file system has its own process on the system. So when writing to a file system the usage goes up. And in this case there where 2 file systems in use. The NAND for reading from it for the binary and the file system of the SD Card for writing to it.

The memory also has a strange figure. During the whole test the memory is about staying the same and suddenly as we get near the end the memory is starting to rise. It is totally unknown where this sudden rising of memory is coming from.

5.5.7 Stress test: copy a file between NAND and SD Card

Just like as in chapter 5.5.5 and 5.5.6 we will use a file of Approx. 3 Megabytes because the internal memory of the Zaurus is to small to store a big file like as we used in the 2 earlier chapters (5.5.5/5.5.6). The SD card will be mounted but the swap will be turned off.

The expectation will be that this test will run longer then the copy tests with the NIC. Simply because the NIC uses up battery power more battery power, and now this factor has been pulled away. The processor load will show some similarities with test 5.5.6 because of the reason that is mentioned here why there was a higher processor load. Also some there might be a higher percentage of processor usage for the user and maybe the system also a bit, put this last case will not be that high. And last but not least, the battery usage will probably look like the old well known pattern.

Result
The NIC isn't mounted during this test and we see that the test already lasts longer then the 2 copy tests when we were using the NIC. So at this point we can conclude for a bit already that the NIC is really power consuming. In this case the battery lasted for almost 5 hours, this is a big difference if we compare this with the copy tests from 5.5.5 and 5.5.6

were the longest test was approx. 3 hours. This behavior was expected before starting this test.

The tool that is being used to test out the behavior between the NAND and the SD Card also runs in a user environment. But because of again the 2 file systems that are being used for the copy process, we see for the same reason mentioned in 5.5.6 that the system load it self is getting higher, even higher then in this test. The system processor usage is around an average of 5.5% or a bit lower with a peak to 8.2% usage. This high usage is caused like said by the 2 file system processes that are being used at that moment, but also the reading from the NAND for the binary that is being used to copy the file and also the Input/Output for writing the file to the NAND at some point. So here we see a whole bunch of factors causing the system processor usage to rise. But still the user processor usage is the highest caused by the copy tool that runs under user privileges. The funny thing is though at this if we do “system usage”+”user usage”=99.9%. And at all measurements this is the case. So the processor is constantly busy doing things.

Because the system is constantly busy we see that the system load in this case is rising to around 4, sometimes a bit above it sometime a bit below it. So this is really fluctuating, with some remarkable things, if the user processor usage is getting higher we see that the system load is also rising, when the user processor usage is decreasing the system load is also decreasing.

Finally the memory usage is rising bit by bit every measurement, but at one point I think there was some kind of routine going on that freed some of the memory that caused it to decrease a bit, because after this the memory was increasing a bit faster like shown in illustration 35 below here.

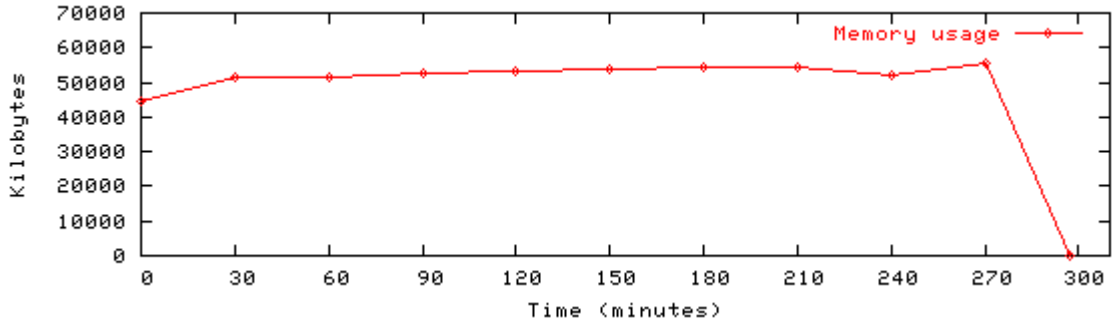


Illustration 35: Copy NAND to SD - Memory usage

5.5.8 Stress test: full use of the Zaurus with SD Card mounted but without NIC

The Zaurus will be used as a normal object like the average user would use it only we try to speed things up and use it all at the same time. Things like word processor, e-mail, spreadsheet etc. etc. will be used all at the same time. In this test case the SD Card will be used for it's swap file and to save and delete a file that will be created with the voice recorder.

We will probably see some real high CPU loads here, these will be the result of many running applications at the same time. This will stress out the processor a lot so it will get more instructions that it can handle. This will result in a slow working PDA. With this high load the use of swap will also increase at every point that it will be measured, and the use of RAM will stay at his highest point. There will be a constant swapping process going on, which will be power consuming. And will take down the battery pretty fast.

Result
When we performed this test we were able to systematically increase the work load on the Zaurus, we started slow and from there on the work pressure increased every time. So this test caused the Zaurus to reach an uptime of a bit more then 4 hours. When the pressure was low on the device we see that there is period of about 90 minutes that the battery indicator still gives us 100%, but as soon we start pushing the device more we see a sudden drop in battery power. And we see that the more we keep on pushing the faster it runs out of power.

This pushing is clear when we look at the load graph as we see in illustration 36.

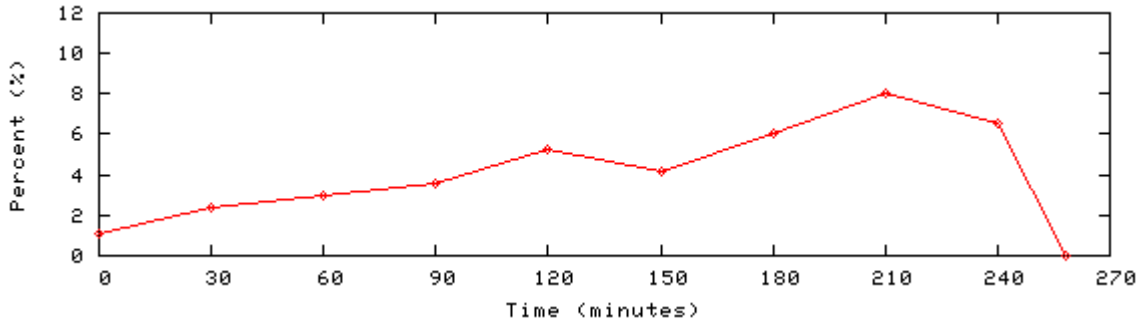


Illustration 36: Full use without NIC - Processor load

When we are pushing the processor we reach a system load which is almost up to 8. This means that the processor has so much to do that it really is slowing down the system. And during this test that was also very clear to notice, when typing something it sometimes took up to 1 minute before the letter even appeared on the screen. So when pushing the load this high we are sure that there will be a lot of power consuming also by the processor it self.

Also when we look at the processor usage for user and system environment we see that in the beginning both values aren't that high but when we keep on pushing the device we clearly see that both start to rise also. One of the things is that some applications run in a user environment but also use a part of the system environment so it needs both of the environments. The launcher for example runs in this user environment, but when a program is started it starts a bit in the system environment. So it needs a bit of both, also when recording a sound file it starts the recorder as a user but writes the data as a user. And we see that when we do that user+system formula again we see that in a lot of those cases there is again a 99,9% processor usage.

When this is all happening we also see that the memory that is rising to the maximum that is possible. When this is starting we see that the system is starting to swap. And this swapping of memory is starting pretty fast in this case. After 30 minutes we see the first

bytes of swap that are being used. And throughout this test we see that the used swap is increasing at every measurement. So it means that the system is swapping a lot and that there is also a lot of activity going on with the SD Card. So with this happening we can relate this to also a factor for power loss in this case. The only problem is that when we didn't had the swap file included in this test we would have got a memory error after about 90 minutes.

5.5.9 Stress test: full use of the Zaurus with SD Card and NIC mounted

This test will be the same as the previous one, only in this case there will also be network traffic besides the applications that will be used. This network traffic can be done with a file copy from the previous test and a massive ping.

We will see a lot of things return from section 5.5.8, the full use stress test without NIC, but now we will use the NIC. This will likely give out some of the same results, the only factor will be that with the constant powering of the NIC, the battery will go down faster then in section 5.5.8. This will be the only big difference. The rest will probably show same pattern types and no crazy things.

Result
This was the shortest test of all, after 2,5 hours the Zaurus was completely down to 0% battery power. Reason for this is that the NIC again was mounted, also this NIC was doing some work like receiving constant ping requests and also some file transferring. And like the test from 5.5.8 we use the same program sequence and pushing. Although like the slower rising of the load the load this time starts to rise very fast. And while we are pushing everything we see that the battery power goes down in almost a complete straight line (illustration 37). So the system was really stressing at this point.

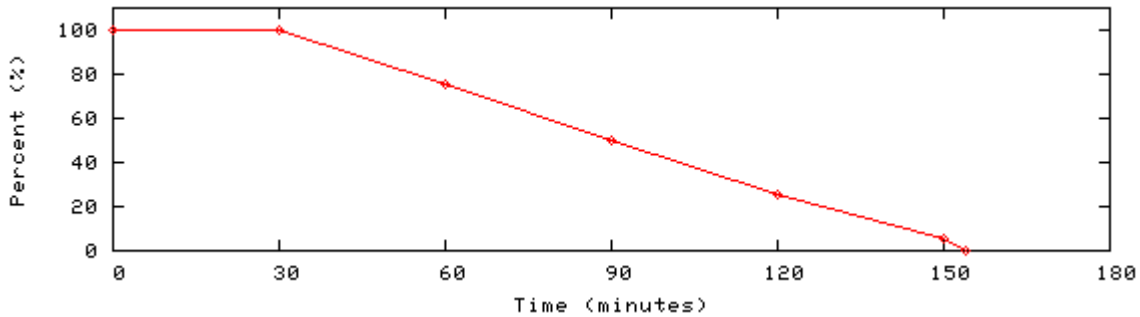


Illustration 37: Full use with NIC - Battery usage

When we push the load We see that the load is rising during the measurements to a peak of 10.48 So at this point the processor has to do so much it was really unbearable to work on. It wasn't reacting in a lot of cases and after waiting some times 1 to at most 3 minutes suddenly things where happening. So the machine was under a little local Denial Of Service (DOS). A lot of the explanation from 5.5.8 can be used to explain the high load on this test. This is the same for the user and system processor usage and the memory/swap usage. It are almost the same tests only in this case there is an extra hardware device add

in the form of a NIC. And this NIC is copying a file and being pinged at the same time.

5.5.10 The TICS application

This final section will show all figures and numbers that are related to the TICS application of Paul Klapwijk. The application will be used in a normal way like it should be used. Also the swap file will be used on the SD Card cause of the otherwise constant returning low memory warning. And final also the NIC will be used to send the information to the other Zauruses.

We will likely see that there will be some similarities between this test and the full use stress test with NIC. Only the steps are taken a bit smaller. Here they are 10 minutes instead of 30. There will be a less higher CPU load in this case but probably a more constant use of System and User processes. Also there will be a lot of swap usage when the application will be running. And this will result in a almost full use of RAM, This will result in a constant swapping and probably an increasing use of swap. There will also likely be a a lot of CPU usage as well for user ans system processes. The Java application TICS is mainly the cause of this. Because the lack of floating point co processor all the graphical points within the application need to be calculated. This will mean that the CPU needs extra cycles to calculate these larger values. So there will be some switching between these 2 process types.

Result

One of the major problems during this test was the freezing of the application. The TICS application was tested 3 times and in all 3 times the application seemed to freeze, all at different points.

With the first try I was just using the system to try out and monitor at the same time when suddenly there was this big warning on the screen saying the machine was out of memory. First thing to check was to see if the swap file was enabled, this was the case. The swap file wasn't even fully used only for 50%, the memory it self was already performing on it's maximum.

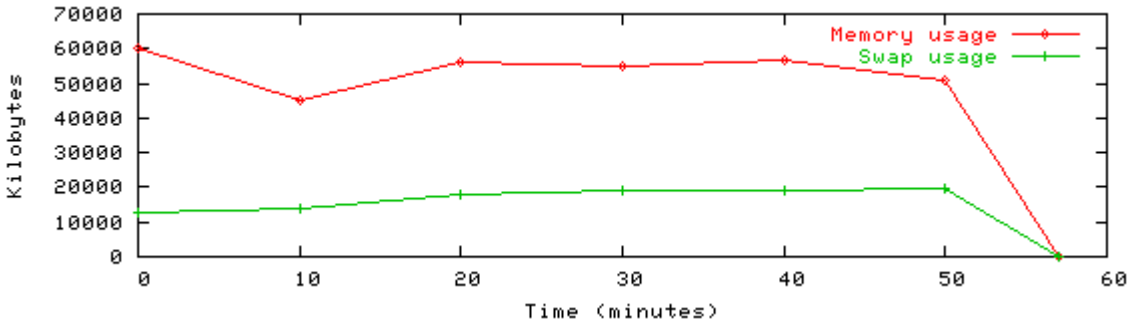


Illustration 38: TICS try 1 - Memory/Swap usage

So after 53 minutes this test was stopped. Within this test it was clear that the application

was being pushed to a load off around 4 with a little peak just a bit above 5. and there was a lot of fluctuating on the user and system processor use. This was mainly caused by the Java runtime environment in which the user starts the Java process and a lot of child processes (more Java instances) are loaded and these are mostly done by the system.

The 2nd try with this application it just froze after sending information over the network, the first time I was sending over the information everything went fine, but the second time it just stopped working. The application did totally nothing and after checking the on screen logging and switching back the application window stayed gray, the only way to get out of it was to totally shutdown the PDA and reload the battery for one more try. During the second try we saw the same kind of graph as from the 1st try. So this was more or less the pattern that it would look like also again the load around 4. the switching between user and system processes.

The final and 3rd try was noting different from the 2nd try only it lasted a lot shorter where the 2nd try has an uptime of 47 minutes before crashing this try laster for 38 minutes. So these test weren't completed the way it shouldn't but gave a little picture of what was happening at the moments that it was up.

The reason why the application crashed is a bit unknown. The freezing of the application happened again after trying to send the information to the world model server that contains the blackboard. Only this time we where able to send data over only the first time while the second time it had the same thing as what happened at the 2nd try. So somewhere in this communication process something went wrong I guess. I haven't completely figured it out. What I do need to note is that sending over the icons really was slow. Sometimes it took up to 3,5 minutes. While I heard that it worked faster. So where it is going wrong in this part of the application might be the communication layer. But this might be nice to check out in the future.

6. Conclusion

6.1 Battery

One of the tests that took the most amount of time before it ran out of power was the idle test without any extra kind of hardware. This took about 9 hours (illustration 30) before it was completed. The shortest test except the TICS application, was the stress test with the NIC. This took a bit more then 2,5 hours. If we look at the graph of this test it is also very clear to see, simply because the battery line is almost a straight line to zero. If we compare this with the line from the idle test we see that this is a major difference.

When we look at all the idle test results we see that the time that it jumps from 100% to 75% takes a while in all cases. But when we turn up the heat for the system we see that this step will take much shorter to be completed. The same goes for the rest of the steps. But even looking at the idle tests in the end the time will get shorter every time when we take a step down. So it isn't evenly spread in either case. But we can see how long a Zaurus in a particular situation can stay up.

So depending on the situation the Zaurus is used in it is crucial that the battery must you don't run out of power fast. If we take the picture of the battery from the full use stress test with NIC in thew we see that the battery runs down in 2,5 hours. We don't want these kind of things happening when we are in a crisis situations. Because you don't know how big the crisis will be in the first place we must make sure that we have enough power with us. Because losing a PDA while you are exploring can mean that the battery can be down or that the PDA has been dropped to the ground and broke while a rescue worker was for example hit by a falling object. So make sure we have enough power with us, there can be several solutions, but what is the best solution needs to be found out. One thing that might be draining power can be the NIC. So one of the things that can be done for example is look at the kernel module and see if the NIC can be set into a sort of sleep mode when it is doing nothing.

6.2 Processor usage

When looking at the idle tests we see that the numbers show almost no strange spikes, they all seem to stay low. This means the system and user almost aren't doing a thing. When we do the copy test, the user calls up a command in the network copy test it was the sshd and scp processes for secure copy and for the NAND<->SD test it was the copy process. With these tests we see that the user percentage of processes is getting higher but the system stays low. On the final full use stress test we also made the system processes work hard, because now we saw when we combined the user and system percentage it was almost 99,9% every time. So the system was really working at these points. one thing that will follow out of this is the rising in CPU load discussed earlier.

So it really depends on what is going on on the system, that requires on how much work the processor needs to do. In some cases the user can be really demanding, most of the time it is the user doing it, but when some processes also go to the background the system can also demand a lot of processor power, this mainly happens when the system is

starting to swap or needs to calculate big things, like the graphical user interface from the TICS application. At this moment the processor is starting to emulate a co processor so the system will demand a lot of processor usage for it self instead of giving it to the user. So a lot of scheduling is happening on these moments.

One other processes is the kapm-idled [6] process. This process is used to save battery when the machine stands idle. This process will always run in the background. When the screen of the Zaurus is touched or a key is being pressed this process will stop working, but when the machine is going to an idle state a gain this process will return to save battery power. There is a little bug around this process. If the process is running we see that most of the time this process uses about 97% of CPU power. This is not true, this process takes no CPU power, so this can fool a user by thinking the CPU is very busy at that present time. So to get a better read on what the Zaurus was going took the complete usage and took of the percentage that the kapm-idled was using, so the formula in this case was: total usage - kapm-idled = effective usage. So don't be fooled by the CPU utilization that you read in the first place.

6.3 System Load

In earlier sections I described that the machine was idle although the load was above 1.00. There is a little glitch with this load. Although the system indicates it as 1.00 it probably is lower. This is caused by the idle daemon which is controlled by the Linux kernel. This daemon is sending HLT messages to the processor to keep the processor in power saving mode. This way the system won't be using more battery power then needed. Because of this constant sending the load on the system will be affected. This is is caused by the kapm-idled daemon that is described in the previous result section.

We see that in all tests the load is above 1, This is simply caused by the situation the Zaurus is in. When we look at the idle tests we see that the load is above 1.00 most of the time. The reason for this is the HLT call from the kapm-idled to the processor that is being send all the time to save battery power. So at these times the load seems above 1.00 but probably is just 0.00 or maybe 0.01. So reading out the load of the system is very hard in this case and can only be guest when the kapm-idled is running. If this process in not running you will see the real system load, but because the power saving is disabled you will lose your battery power faster.

When we take a quick peek at the full use stress tests again we can say that we are putting the device in a Denial Of Service (DoS). This means that the system can't do a lot and is starting to react very slow, with this very slow reaction of the device it is almost impossible to work with. So you must be carefull enough that you don't overload the processor.

6.4 Memory/Swap

As expected the memory management worked fine. In all idle tests the memory stays the same throughout the tests. Which should have been the case because on those moments the system isn't doing a thing. So everything here with all the idle tests the line is almost horizontal.

When we look at some of the stress tests we see that the memory in most cases started to rise to a certain maximum. When this maximum was reached the memory management started to write parts of the memory to the created swap file. If this swap file wasn't created, an out of memory error would be appearing on the screen. So at this point the machine also started to swap. We only see this at the stress tests where the swap file was enabled. And these are the full use stress tests, and the tests where we try to run the TICS application. So when the swapping starts this will cause extra load on the system. This extra load will have it's effect on the battery, and this way the battery will be losing power faster.

The copying stress tests on the other hand where the swap file was deactivated, here we see the memory rise to a certain level and does not get an out of memory error or anything that goes within that direction.

So in an overall view of how the memory performs in the selected situations we do see that the freeing and paging of memory works fine. Also when the swap in enabled we see that when it's getting to full it starts to write parts of the memory to this swap file by the kswapd process running on the device. Also getting back certain parts of the memory from the swap file are included in this procedure. So we can say the the Linux kernel memory management does its job like it should do.

6.5 The Network

When looking at the network we see that it uses the IEEE 802.11b standard. This is a wireless type of network, The network it self is very easy to setup. First you must make sure that all essid names are the same for this wireless network, if this is not the case you won't be able to ping the other machine, so there will be no way to communicate with it.

When we got 2 Zauruses next to each other and we let the tcpdump run like done in chapter 4 we quickly notice that there is totally no network traffic between these 2 machines. When switching over to the AODV protocol for out Ad hoc network we will see the first network traffic passing by. These are the AODV RREQ packets. So it is searching for other nodes. From the test result we can see that this AODV protocol works fine and also can we say that there is no strange data being send around the network while we are just using the protocol to scan off the network. The only thing noticed it that it constantly is sending requests and waiting for replies and when they found each other every thing works like it should work according to the RFC3561.

Also in multihop mode the AODV module does all its work like it should be doing so there is nothing wrong on this network setup approach. In fact when there is no other way to communicate this is the best way to setup a quick network because it really initializes it self.

So this AODV protocol is almost RFC compliant. The AODV version that is being used is a bit older then the one that is described. For example new is it RERR packet that is used as error packet. The year that this AODV module has been developed is a bit older then the last RFC change which was in 2003.

The only thing that is really sending a lot information over the network is when we use the TICS application. This application works with the RMI registry from Java and is constantly

sending packets over the network to other hosts, and in this case I'm not talking a bout a few packets but really about a whole lot. But the network is strong enough to hold this because the performance of the network is also fine.

So for this kind of situations this kind of network is fine, the only thing that should be really looked at is also the security point, at this moment all data is easy to intercept and can be altered by an evil person if they want, so some sort of encryption is advised to be used when people will develop any further on these networks. Overall there isn't much overhead within this network at the present time.

6.6 Recommendations

The processor is a based on a RISC structure which means that it is a cheap to fabricate processor. This also means that some functionality things have been stripped from the CPU in comparison with a normal CPU we are used from personal computers. So for big calculations, where it misses a FPU, it is very slow and sometimes take a lot of the time. This is because the CPU needs to emulate this FPU. It is very likely that when for example the ISME client is started it takes some time to get the working GUI because a lot of the calculations that it makes need to be emulated. When this same GUI is loaded on a normal computer with an enabled FPU the GUI is there within a second. We can see the difference with and without FPU back in chapter 3 where there was performed in a a benchmark.

Also if we take with us the system load, only looking at TICS we pushed the system load over 4.00 sometimes, this means that the processor is doing the work of 4 processors of it's type. This is pretty absurd because he can't fix all the work that has been planned in the amount of time that it will take 4 processors. This is also mainly the thing why at this point the application isn't working real time. If things can be speed ed up on an other system the application will likely turn into a more real time application.

A stronger processor will also mean that the user and system processes will not have a big impact anymore, because tasks can be performed faster. This will mean that a lot of the spikes we saw from the illustrations probably will disappear cause of the speed that can be used for certain tasks. So this will also cause the system load to decrease a lot. But in this case we want the application to be able to run on the PDA. So it might be smarter to have a look at the Qtopia development kit and develop this kind of application for Qtopia. This will likely be a lot quicker then the Java environment that is running at the present time. One of the things that slows it a bit down is the Java virtual machine that has to be started. Because when the application is started, in this case TICS, we will see in the process list that a lot of Java process instances will appear. All these processes are slowing down the system. Probably when you develop the application in a Qtopia environment the system load won't rise as high then when we use the same application in Java. It might be nice to try and figure something like this out in the future, is it really faster if I use this kind of application in Qtopia environment then in a Java environment. The only thing with Qtopia is that such an application needs to be cross compiled. And this is an advantage for Java, there is no cross compile needed, because it is multiplatform.

An other thing can be to see if there is a way of porting a newer Java version to the device. The latest version is likely to have a major list of bug fixes if we compare this with the

present running version of Java on the PDA. So if people really like to stay on Java it might be worth a shot to try and port the runtime environment of this never Java version.

When looking at the NIC we have a little problem with the driver for it. The NIC can't be put into a sleep mode. This way the radio transmitter will stay on all the time, even when the Zaurus is in an idle state. The kapm-idled can't put it in a sleep mode like it does for the Zaurus it self when the keyboard or touch screen isn't used. The problem for this probably lies within the kernel module, it looks like module does not have the support for it. If this is the case then this issue might be resolved in a later version of the kernel, probably the 2.6.x series . So an update of the complete kernel will be necessary in that case. There are also some Zaurus on the market with the NIC build in and probably also a good working driver to put it the NIC in sleep mode. So this can be an option to find this out if this is true and if it is true does it save a lot of power in comparison with how it is now.

Also spend some more time on the network security this might also become a big issue, you don't want people to be able to read all data that is being transferred to one and an other. If people can intercept the data and can alter them, if you are evil, you can put the lives of people and rescue workers at stake and we don't want such a thing to happen. So a thing to think about is network security, how to realize this and how to implement this.

Besides the network security that should be looked after, it might be also smart to update the AODV protocol, the version that has been used is a few years old and in the mean time there is a newer version released that is more RFC compliant. Cause like in the network setup the error package is described but in the running version this isn't implemented because the RFC is newer then the developed source code.

Finally the applications, I think the PDA assistant application is one of the better applications I've seen from the 3 described ones. I have seen a little bit of Lingua but the version I saw was some ow not 100% completed. The ISME application was working but the network communication was very very slow. Some times it took up to 3,5 minutes before data was received on an other device. This will mean that if some one is reporting a fire and by the time the fire was reported with other the building will be almost burned down. The problem with this probably lies within the communication layer. But to discover this you need to start disable things from the application and start finding it out by a trial and error principle. So I think it might the best thing to continue up on the PDA assistant application, this application had a more real time approach on the communication layer and also because it has the option for sending pictures and text.

Maybe looking for a device with a little bigger screen might be an option, because when you are full of adrenaline and the icons are very small there might be a chance that people place wrong icon on a map or are more busy trying to get the right icon tapped on the screen. Or maybe create a bit larger icons for this problem.

Bibliography

[1] TICS: A Topology based Infrastructure for Crisis Situations, Paul Klapwijk, August 2005

[2] Zaurus Quickstart guide: http://www.trisoft.de/pdf/c750qse.pdf

[3] Linksys WCF12 Specifications: http://www.wardrive.net/general/WCF12-DS.pdf

[4] Dane Elec SD Specifications: http://www.wynit.com/specs/DAN102/DASD1024PK5.pdf

[5] Zaurus review: http://www.brighthand.com/article/Sharp_SL-C760_Review

[6] Little info about kapm-idled: http://www.linuxquestions.org/questions/showthread.php?t=19501

[7] http://www.cs.umd.edu/class/fall2001/cmsc411/index.htm

[8] http://www.intel.com

[9] ISME: Icon based System for Managed Emergencies, Paul Schooneman, January 2005

[10] RFC3561, Ad hoc on Demand Distance Vector Routing, July 2003

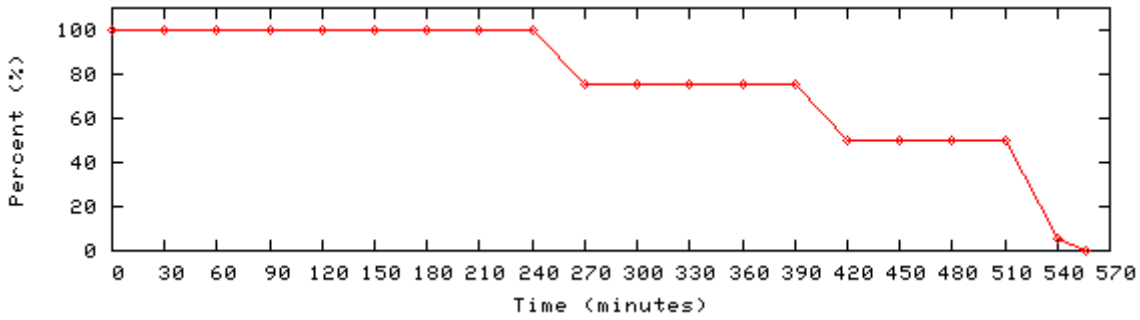
[11] An Icon-Based Communication Tool on a PDA , Siska Fitrianie and Leon J.M. Rothkrantz , April 2005

Appendix A

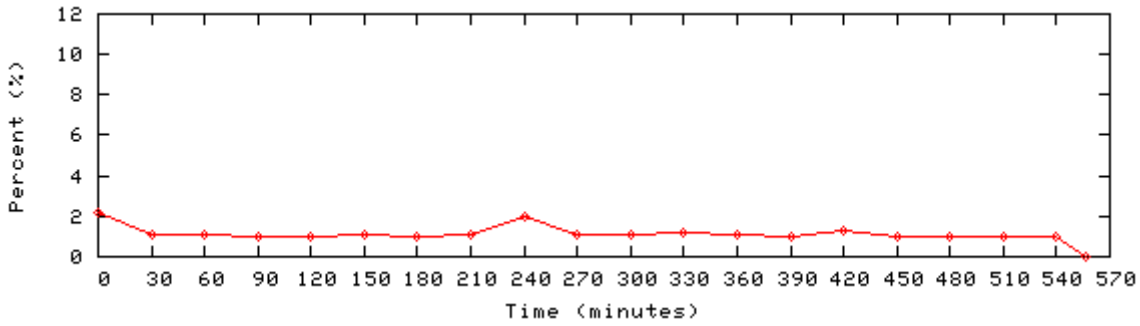
Tables and graphs of the results

Idle without any Hardware components

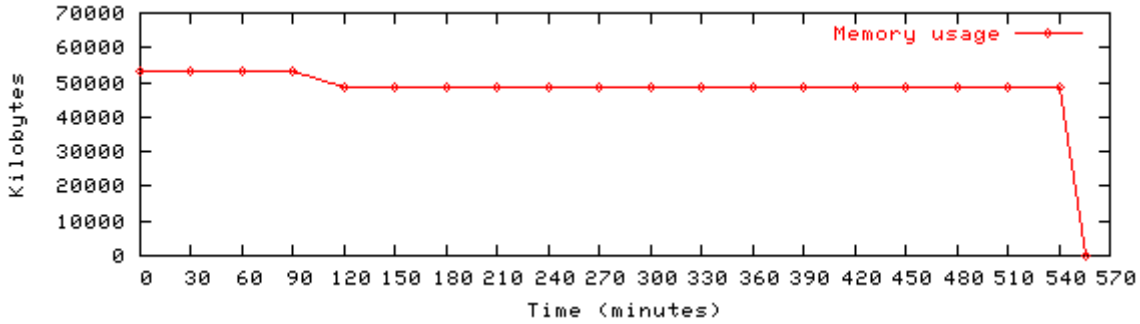
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*CPU Load*
0	100	1,4	2,7	53260	2,14
30	100	1,8	1,7	53260	1,06
60	100	1,7	3,1	53260	1,06
90	100	1,2	1,9	53260	1,02
120	100	1,7	4,8	48824	1,02
150	100	1,1	1,9	48824	1,06
180	100	1,0	2,1	48824	1,03
210	100	1,0	2,1	48824	1,06
240	100	1,3	5,6	48824	2,00
270	75	1,4	4,2	48824	1,07
300	75	1,6	1,9	48824	1,14
330	75	2,2	4,6	48824	1,15
360	75	1,3	4,1	48824	1,05
390	75	1,4	5,0	48824	1,02
420	50	1,1	1,7	48828	1,25
450	50	1,3	1,5	48828	1,01
480	50	1,2	4,8	48828	1,01
510	50	1,5	5,4	48824	1,01
540	5	1,3	4,9	48824	1,02
556	0	0	0	0	0



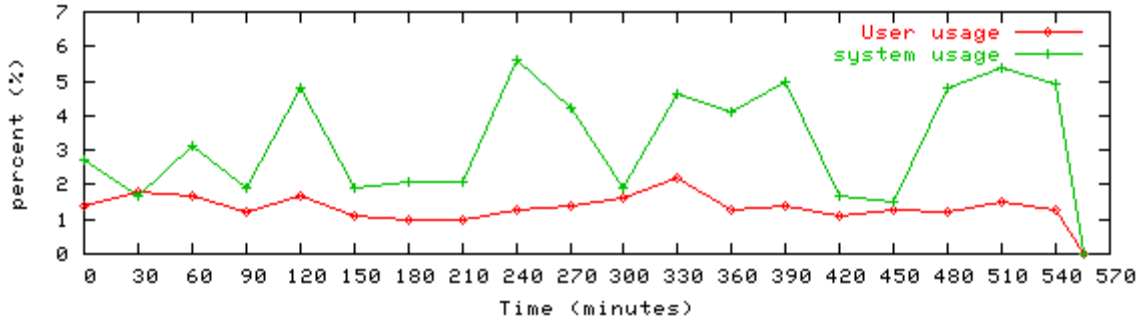
Idle without SD and NIC - Battery usage



Idle without SD and NIC - Processor load



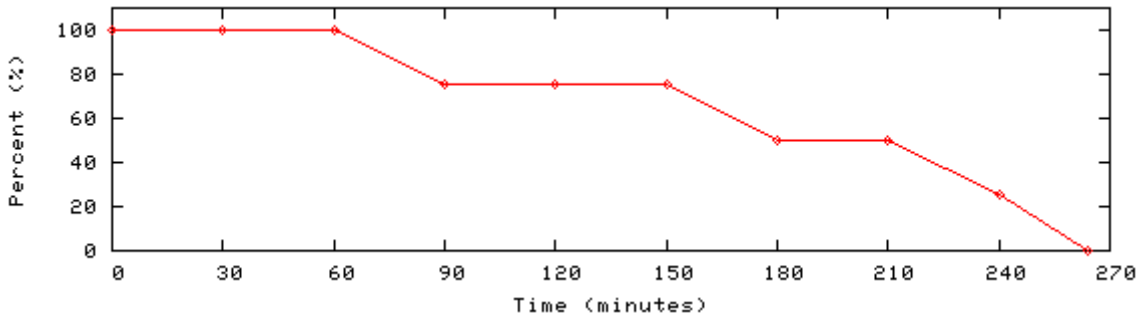
Idle without SD and NIC - Memory usage



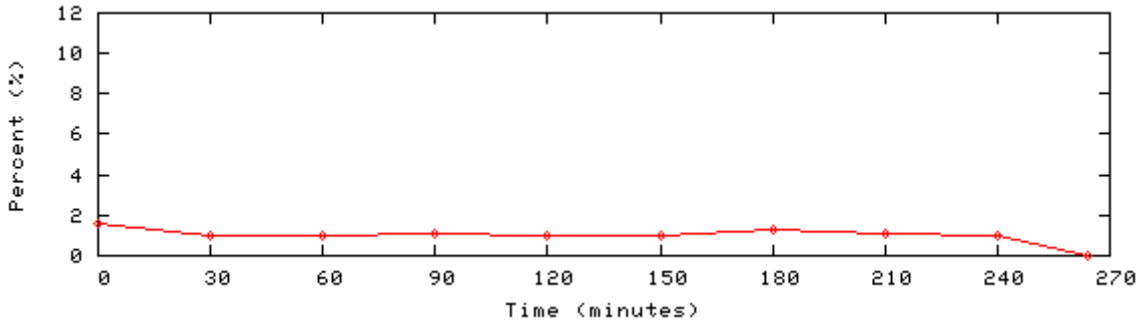
Idle without SD and NIC - User/System processor use

Idle without SD Card and with NIC

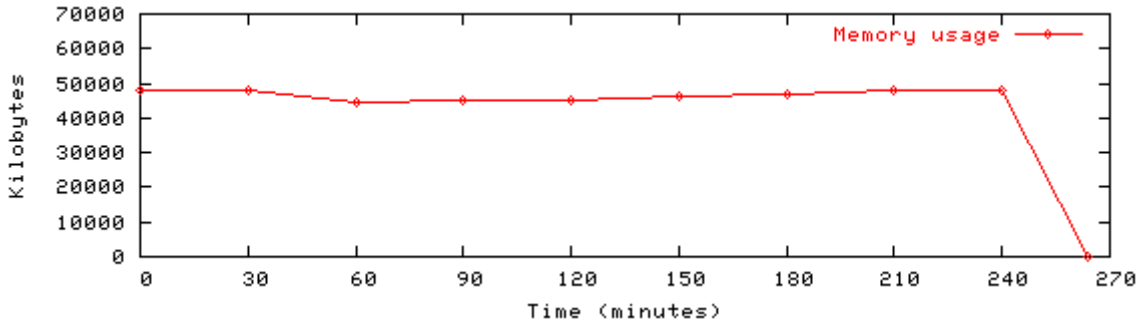
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*CPU Load*
0	100	1,5	2,7	48216	1,60
30	100	1,0	2,9	48216	1,03
60	100	1,4	5,2	44660	1,01
90	75	1,1	2,3	45400	1,08
120	75	1,6	1,7	45040	1,00
150	75	2,0	5,2	46164	1,01
180	50	2,3	4,1	46968	1,24
210	50	1,0	1,5	47776	1,06
240	25	1,3	6,1	47780	1,03
264	0	0	0	0	0



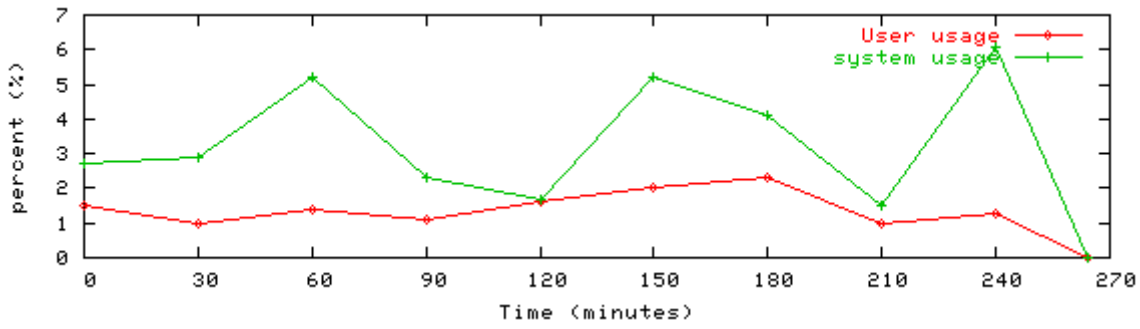
Idle without SD and with NIC - Battery usage



Idle without SD and with NIC - Processor load



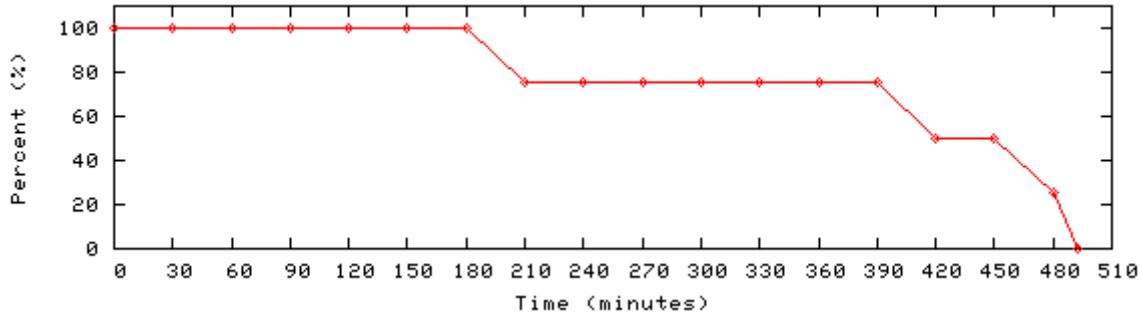
Idle without SD and with NIC - Memory usage



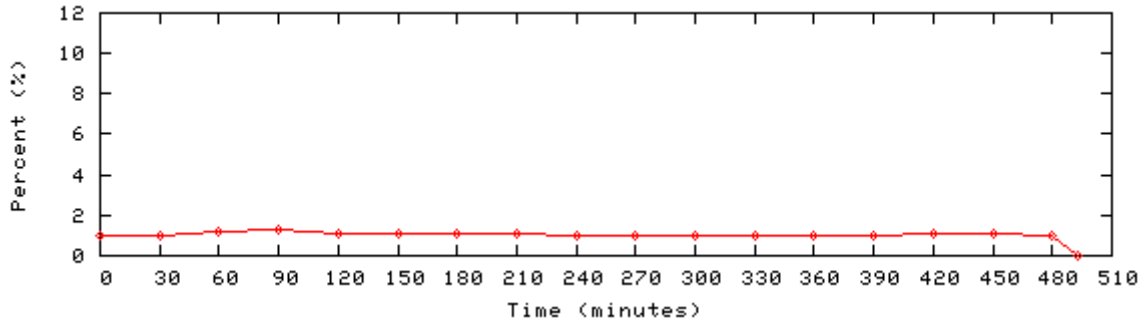
Idle without SD and with NIC - User/System processor use

Idle with SD Card and without NIC

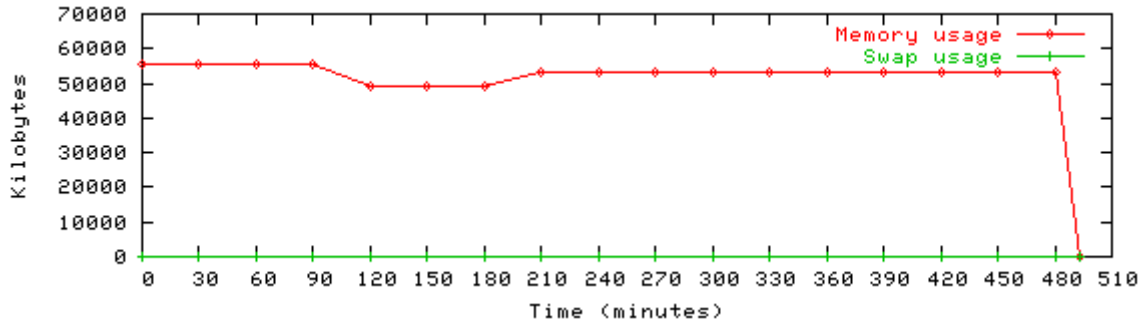
Time	*Battey (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	2,1	3,1	55304	0	1,03
30	100	1,1	4,8	55304	0	1,00
60	100	1,8	2,3	55348	0	1,21
90	100	1,6	1,7	55348	0	1,30
120	100	2,0	2,1	49216	0	1,07
150	100	1,5	2,1	49220	0	1,09
180	100	1,4	1,9	49240	0	1,09
210	75	1,7	2,3	53560	0	1,06
240	75	1,1	1,5	53560	0	1,04
270	75	1,2	5,8	53508	0	1,04
300	75	2,2	3,5	53388	0	1,00
330	75	1,2	3,1	53388	0	1,03
360	75	1,2	1,7	53388	0	1,01
390	75	1,6	2,1	53388	0	1,02
420	50	1,1	2,1	53388	0	1,06
450	50	1,2	5,8	53260	0	1,09
480	25	1,7	5,6	53272	0	1,04
493	0	0	0	0	0	0



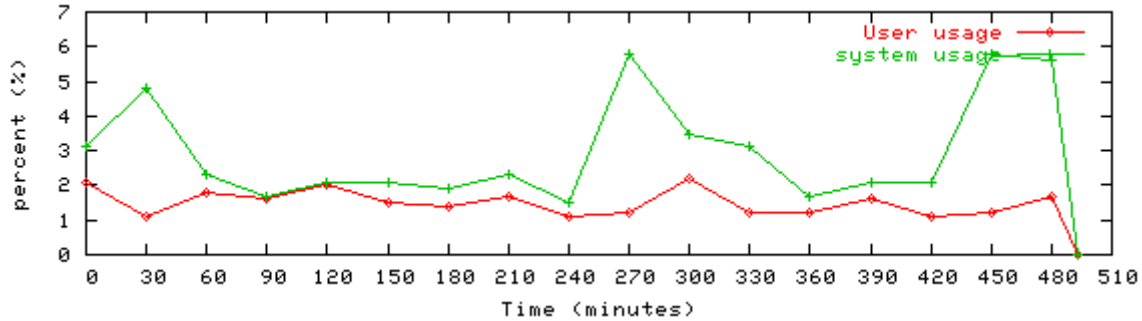
Idle with SD and without NIC - Battery usage



Idle with SD and without NIC - Processor load



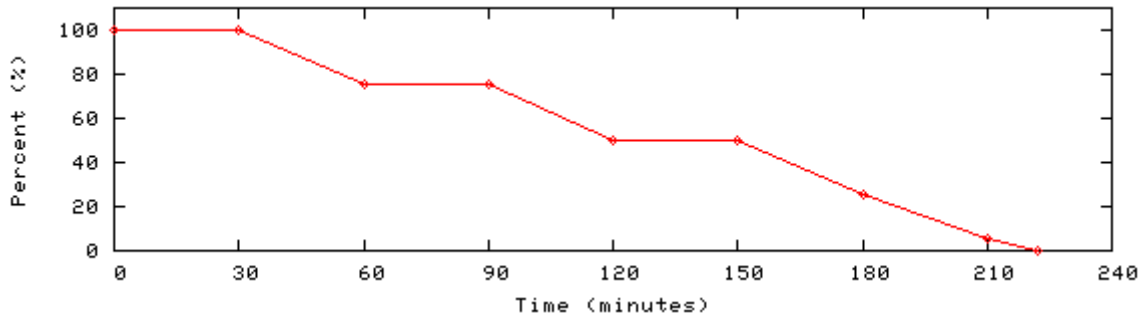
Idle with SD and without NIC - Memory-Swap usage



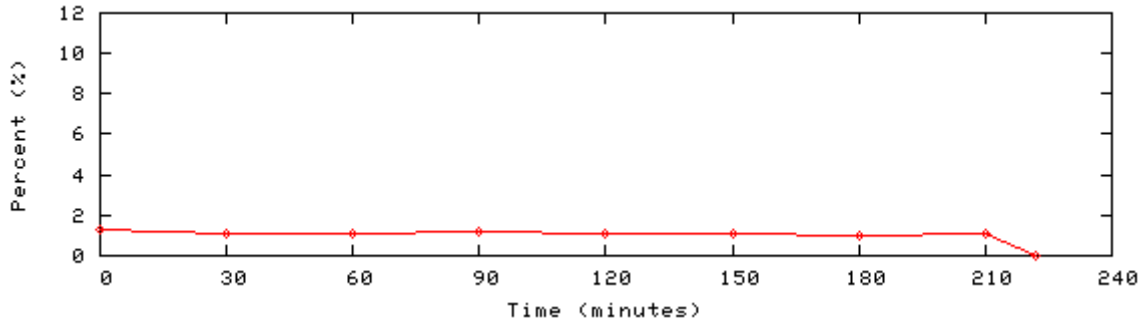
Idle with SD and without NIC - User-System processor usage

Idle with both SD Card and NIC

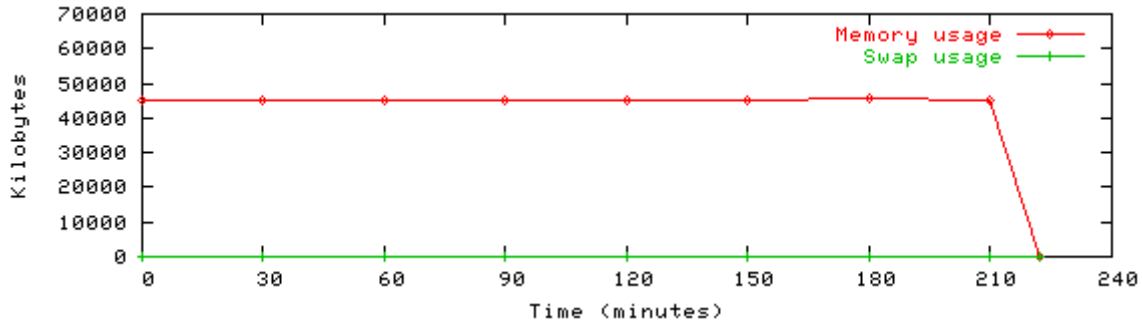
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*SWAP (KB)*	*CPU Load*
0	100	1,6	1,7	45000	0	1,26
30	100	1,2	1,3	45048	0	1,09
60	75	1,6	1,7	45400	0	1,11
90	75	2,6	0,7	45400	0	1,16
120	50	1,7	0,9	45400	0	1,09
150	50	1,5	1,5	45400	0	1,07
180	25	1,6	1,3	45416	0	1,02
210	5	1,6	0,9	45412	0	1,05
222	0	0	0	0	0	0



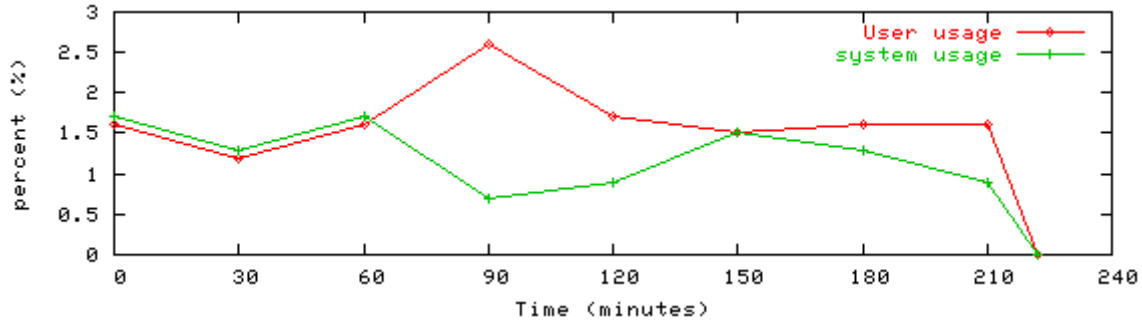
Idle with SD and NIC - Battery usage



Idle with SD and NIC - Processor load



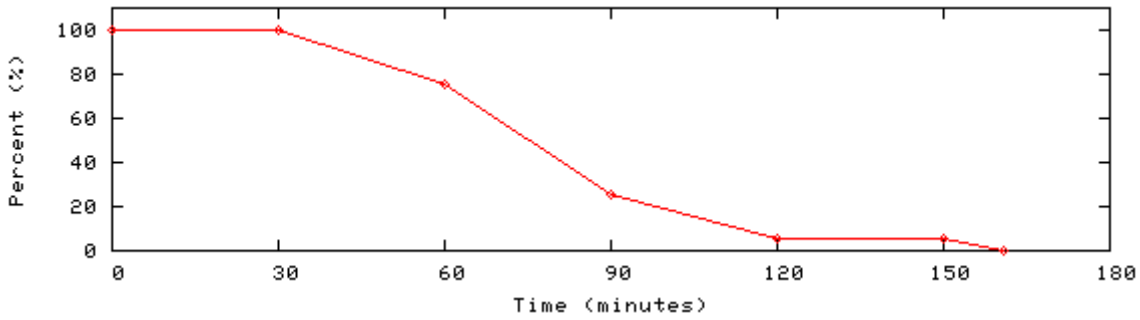
Idle with SD and NIC - Memory/Swap usage



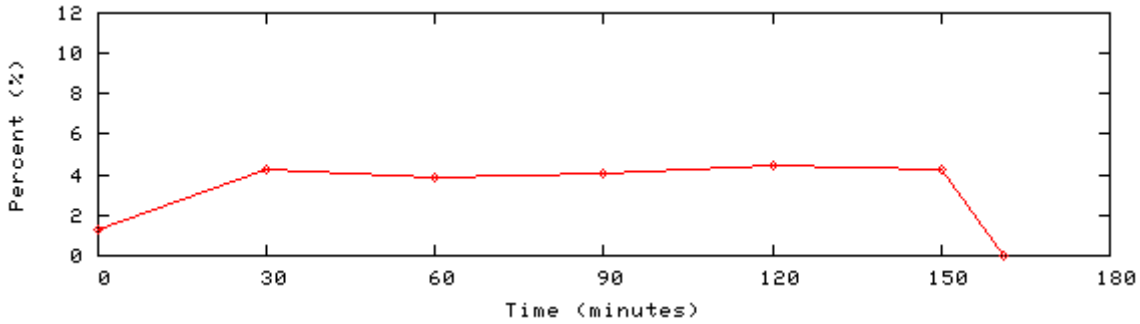
Idle with SD and NIC - User/System processor usage

Transfer a file over the NIC to NAND Flash memory

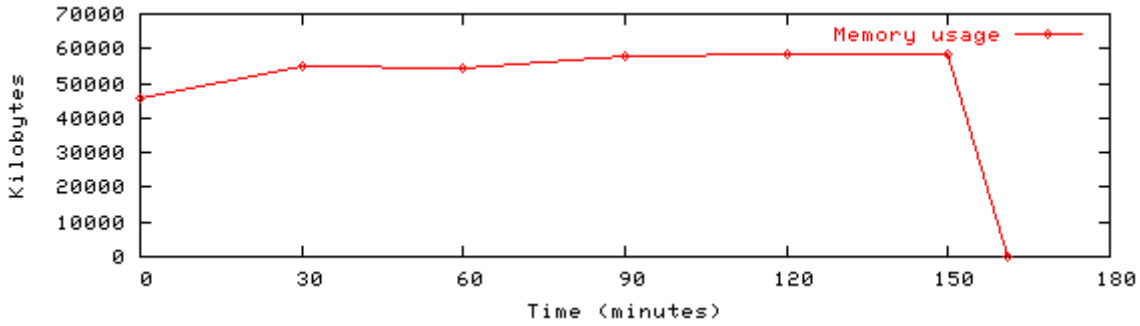
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*CPU Load*
0	100	1,6	0,9	45564	1,24
30	100	69,2	3,0	54832	4,27
60	75	70,7	3,8	54424	3,91
90	25	70,4	2,5	57576	4,09
120	5	70,9	2,5	58172	4,43
150	5	71,0	2,6	58612	4,28
161	0	0	0	0	0



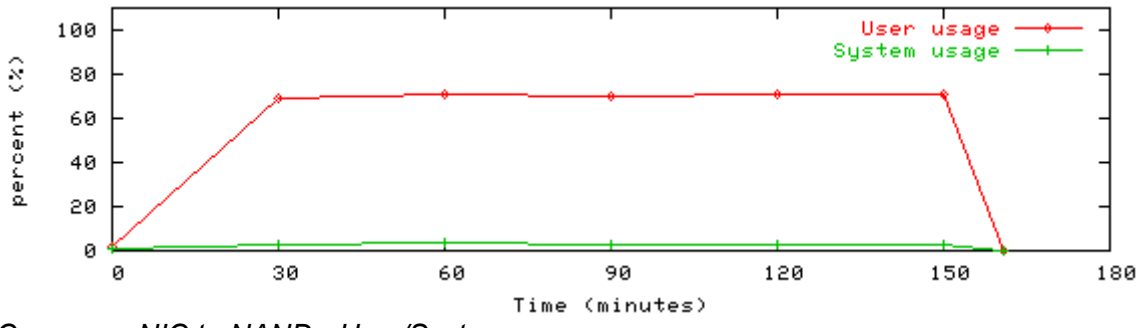
Copy over NIC to NAND - Battery usage



Copy over NIC to NAND - Processor load



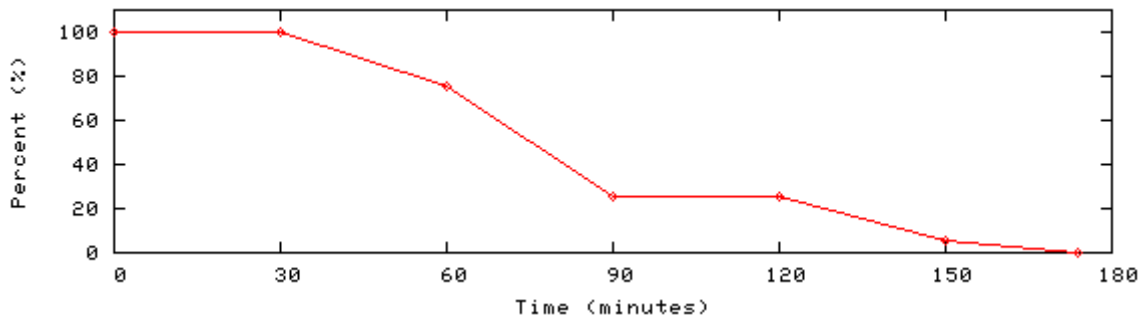
Copy over NIC to NAND - Memory usage



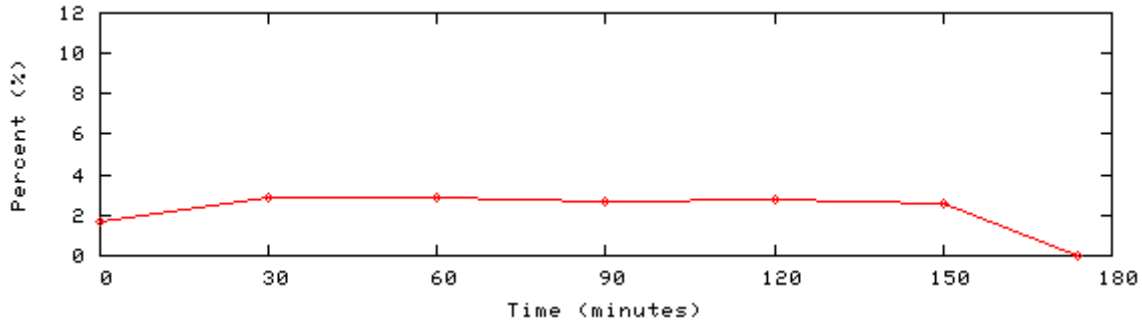
Copy over NIC to NAND - User/System processor usage

Transfer file over NIC to SD Card

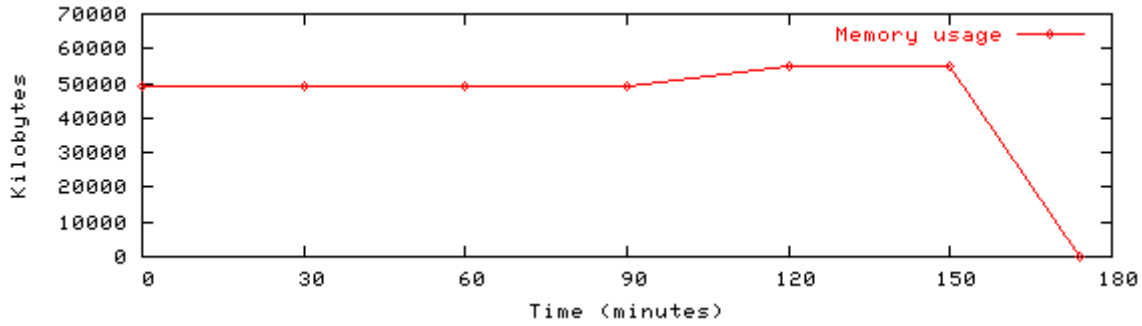
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*CPU Load*
0	100	1,7	3,5	49032	1,73
30	100	85,2	3,1	49032	2,88
60	75	76,7	3,7	49360	2,83
90	25	87,6	3,0	49448	2,64
120	25	70,1	3,2	54844	2,81
150	5	69,8	3,6	54800	2,61
174	0	0	0	0	0



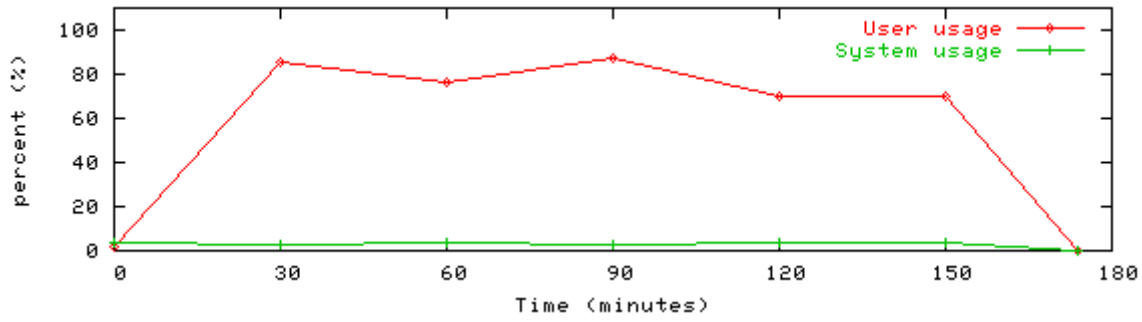
Copy over NIC to SD - Battery usage



Copy over NIC to SD – Processor load



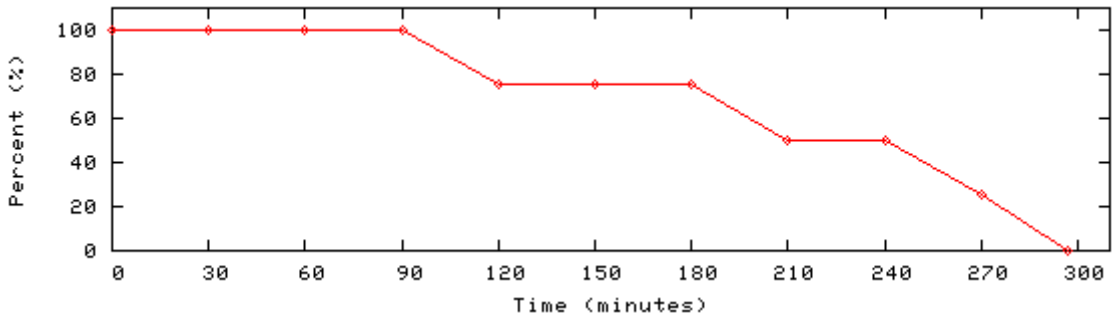
Copy over NIC to SD – Memory usage



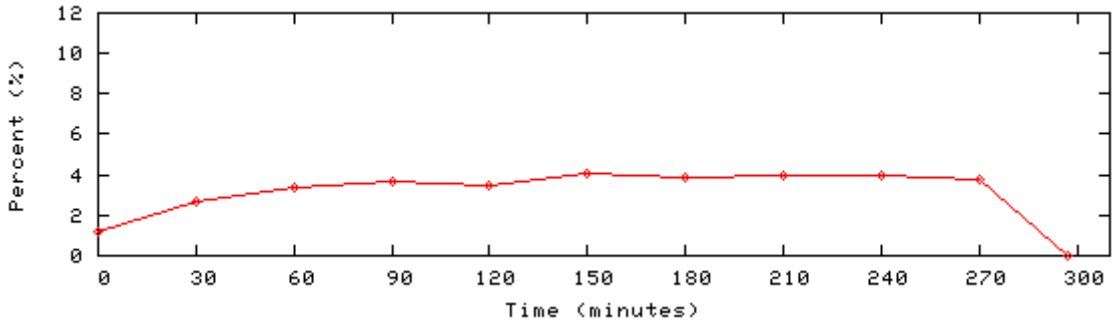
Copy over NIC to SD – User/System processor usage

Copy file from NAND flash memory to SD Card

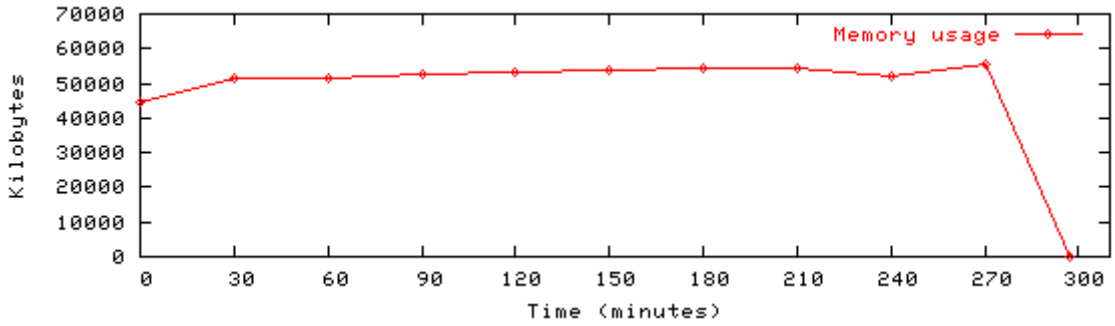
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*CPU Load*
0	100	1,4	1,7	44676	1,23
30	100	94,4	5,5	51492	2,65
60	100	93,7	6,2	51736	3,41
90	100	94,2	5,7	52408	3,68
120	75	91,7	8,2	53276	3,51
150	75	96,6	3,3	53276	4,02
180	75	93,3	6,6	53904	3,87
210	50	94,2	5,5	54500	3,92
240	50	96,3	3,6	52272	4,00
270	25	94,2	5,7	55452	3,75
293	0	0	0	0	0



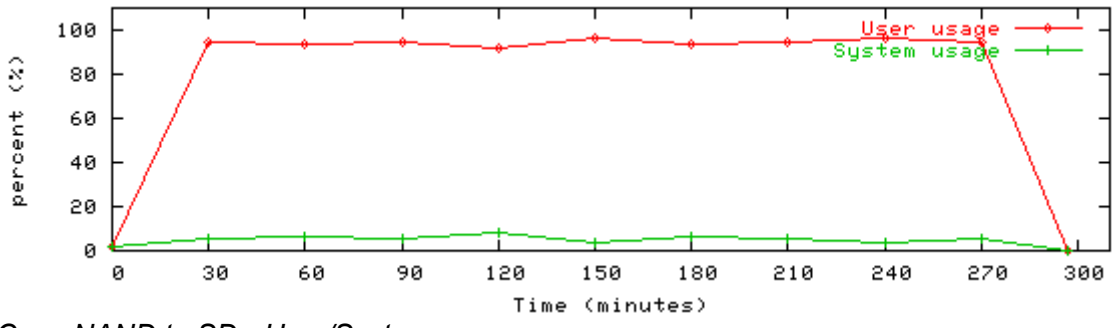
Copy NAND to SD - Battery usage



Copy NAND to SD - Processor load



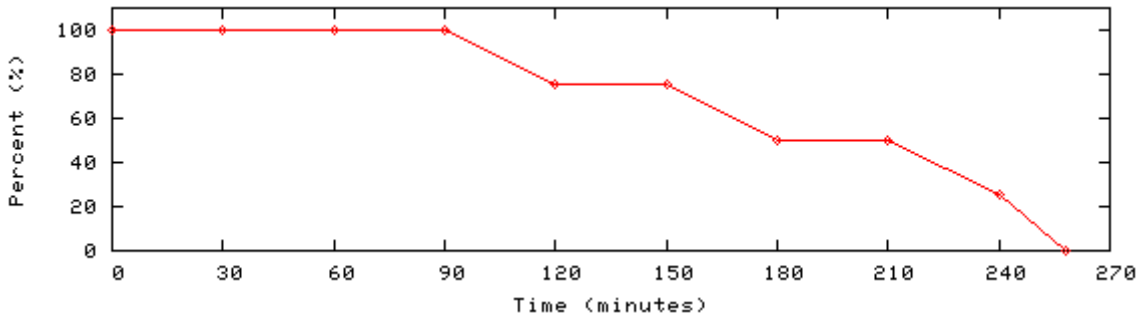
Copy NAND to SD - Memory usage



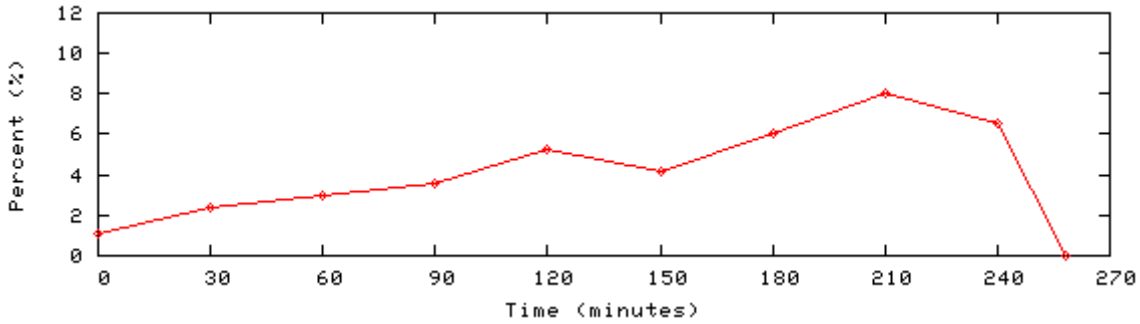
Copy NAND to SD - User/System processor usage

Use all stress test without a NIC

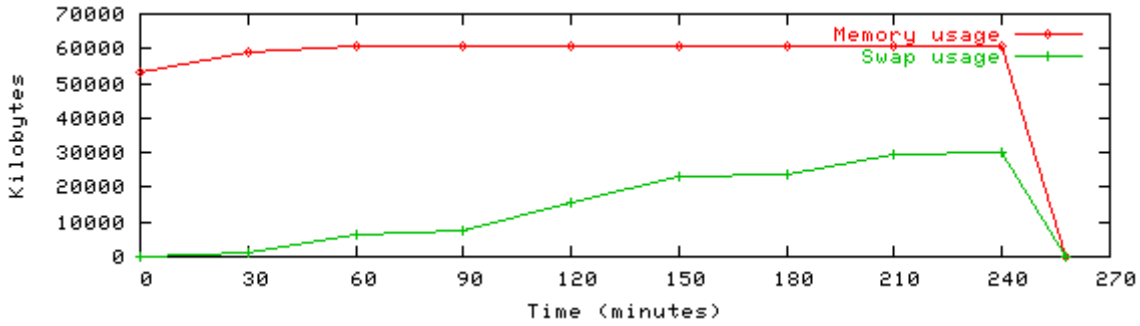
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	1,1	3,3	53388	0	1,08
30	100	2,3	3,1	58928	916	2,36
60	100	9,1	7,8	60716	6212	3,02
90	100	10,7	12,0	60832	7508	3,56
120	75	56,6	43,3	60644	15596	5,23
150	75	15,3	84,6	60746	23284	4,21
180	50	24,0	45,3	60976	23752	6,02
210	50	43,6	56,3	60756	29308	7,99
240	25	30,3	69,3	60968	29944	6,54
258	0	0	0	0	0	0



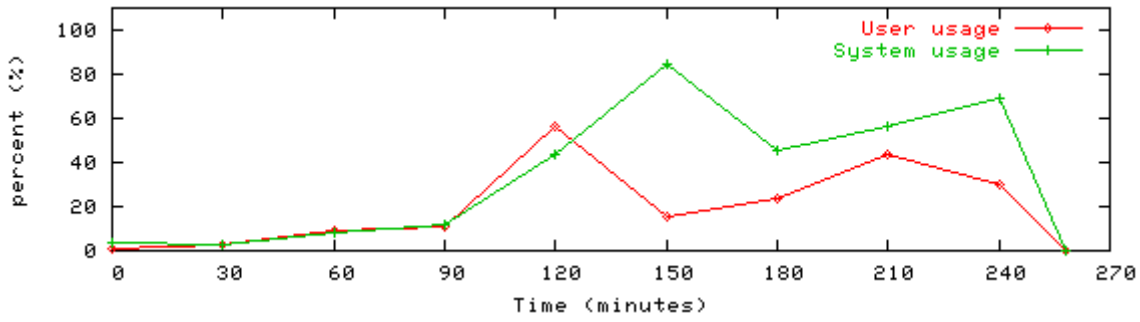
Full use without NIC - Battery usage



Full use without NIC - Processor load



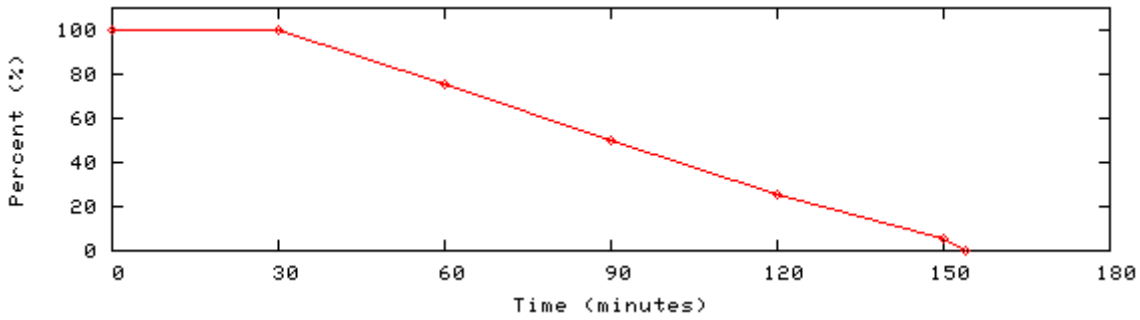
Full use without NIC - Memory/Swap usage



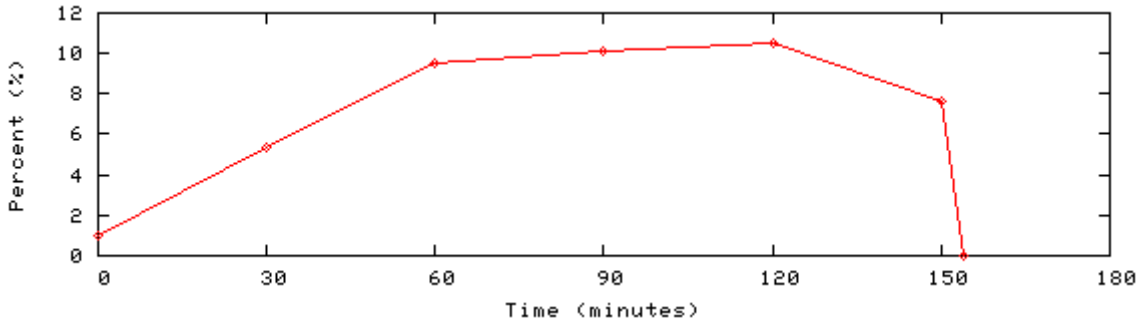
Full use without NIC - User/System usage

Use all stress test with a NIC

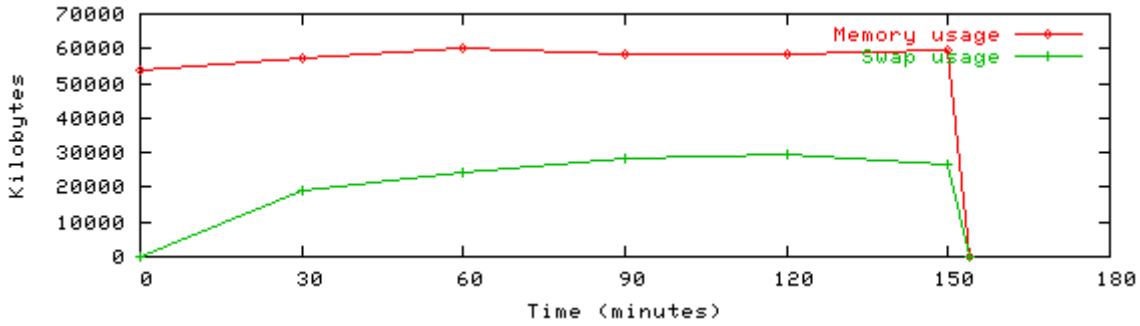
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	1,2	1,3	54084	0	1,03
30	100	4,3	8,2	57416	19268	5,36
60	75	61,6	38,3	60040	24316	9,56
90	50	66,1	33,8	58160	28232	10,07
120	25	52,0	47,9	58556	29664	10,48
150	5	72,4	27,9	59556	26712	7,62
154	0	0	0	0	0	0



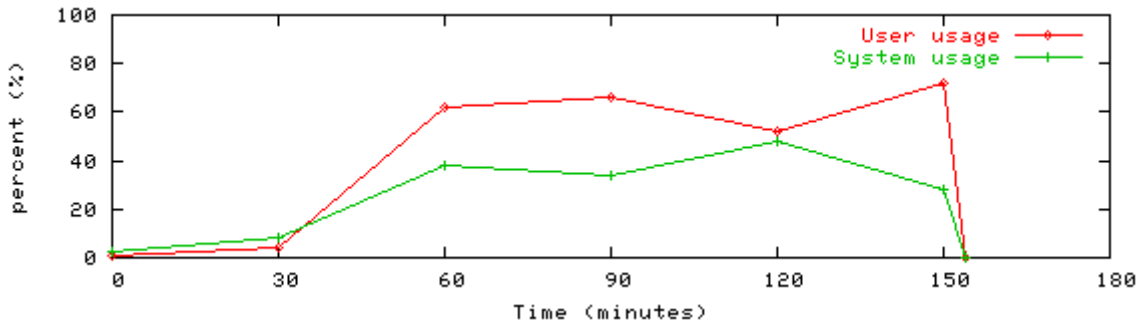
Full use with NIC - Battery usage



Full use with NIC - Processor load



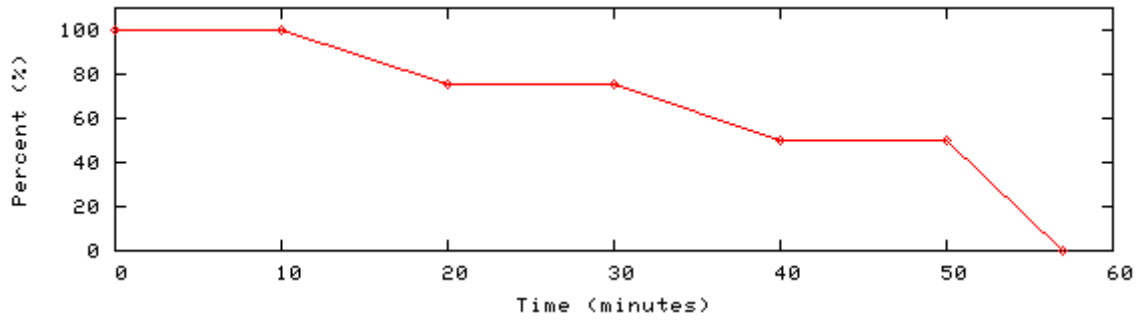
Full use with NIC - Memory/Swap usage



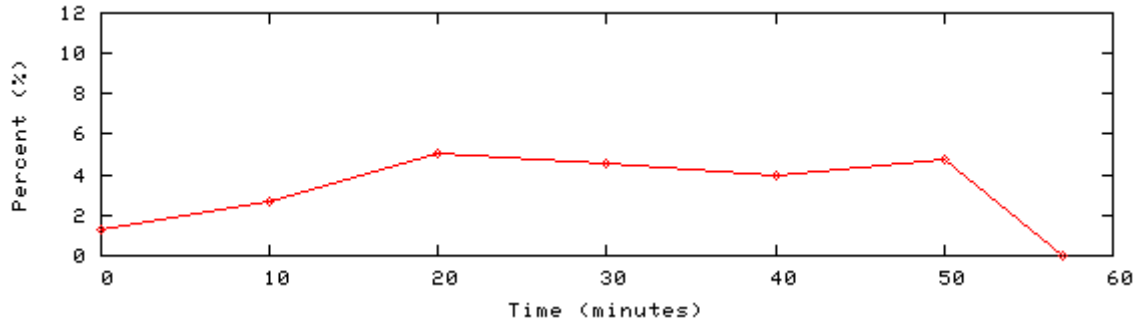
User/System processor usage

TICS try 1

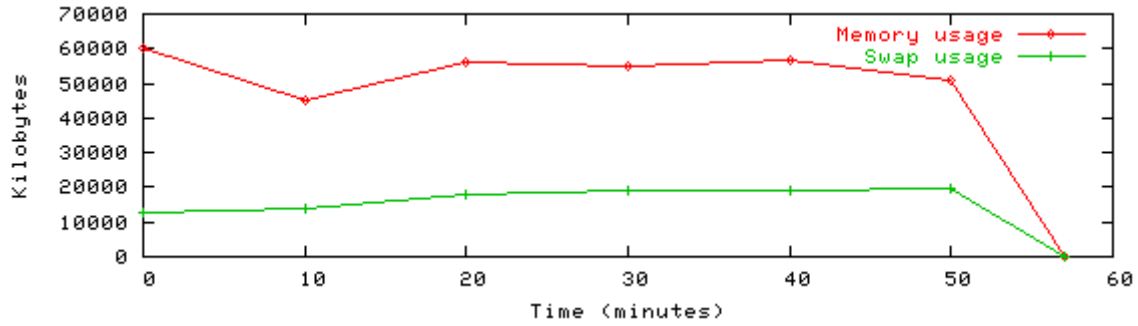
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	61	38	59948	12015	1,31
10	100	60	39	45192	13916	2,65
20	75	43	56	56128	17996	5,02
30	75	47	52	54676	19052	4,54
40	75	30	69	56524	18996	3,99
50	50	35,5	64,4	51116	19404	4,78
57	0	0	0	0	0	0



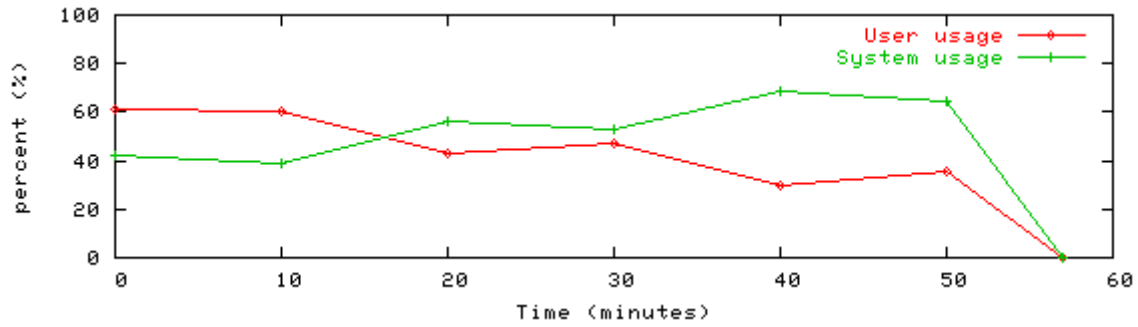
TICS try 1 - Battery usage



TICS try 1 - Processor load



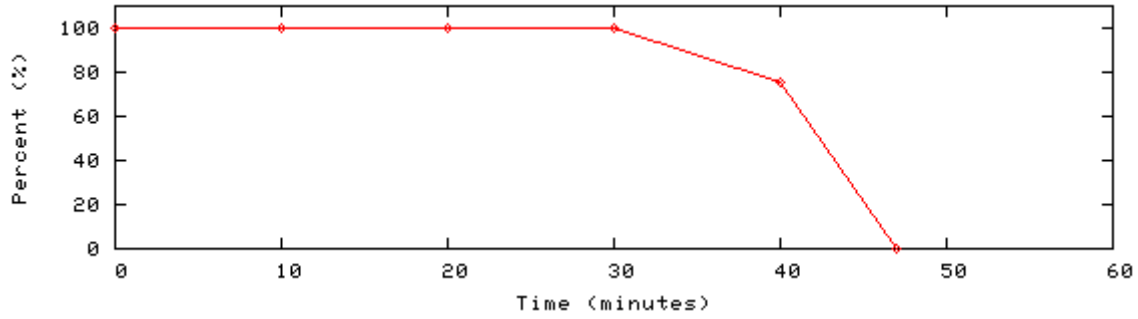
TICS try 1 - Memory/Swap usage



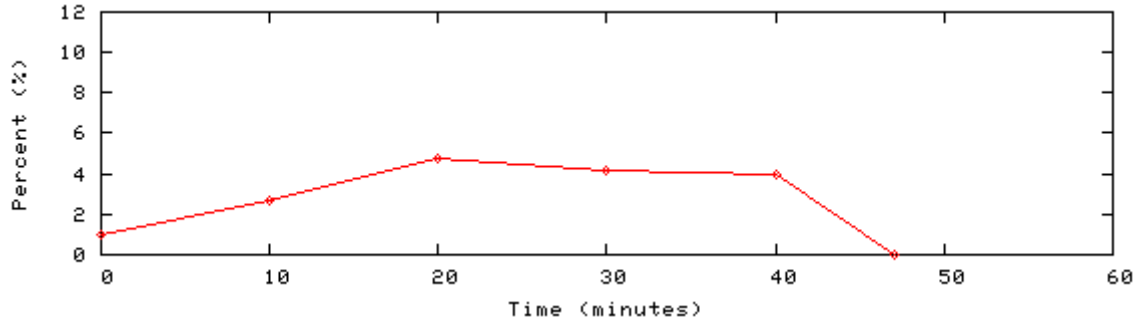
TICS try 1 - User/System processor usage

TICS try 2

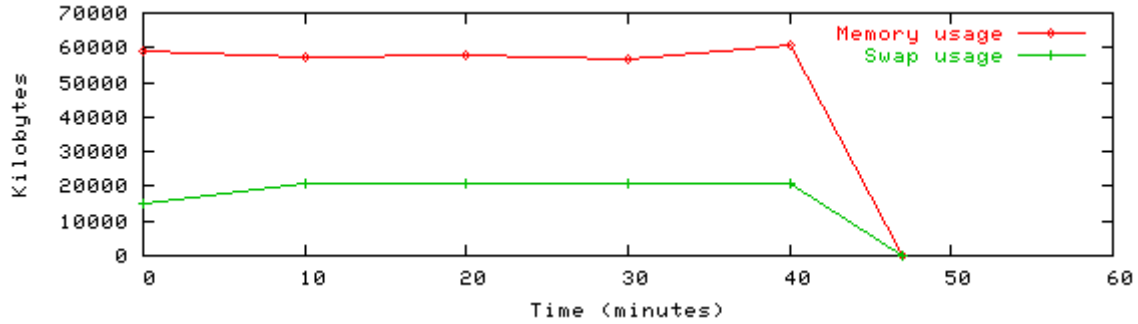
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	54,2	32,9	59000	14988	1,02
10	100	40,1	59,8	57184	20600	2,64
20	100	38,3	61,6	57768	20596	4,78
30	100	42,2	55,7	56436	21112	4,21
40	75	92,3	7,6	61032	21096	3,98
47	0	0	0	0	0	0



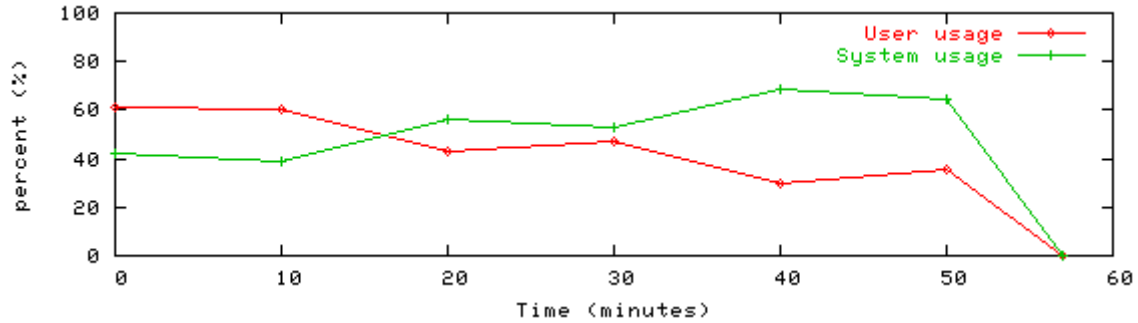
TICS try 2 - Battery usage



TICS try 2 - Processor load



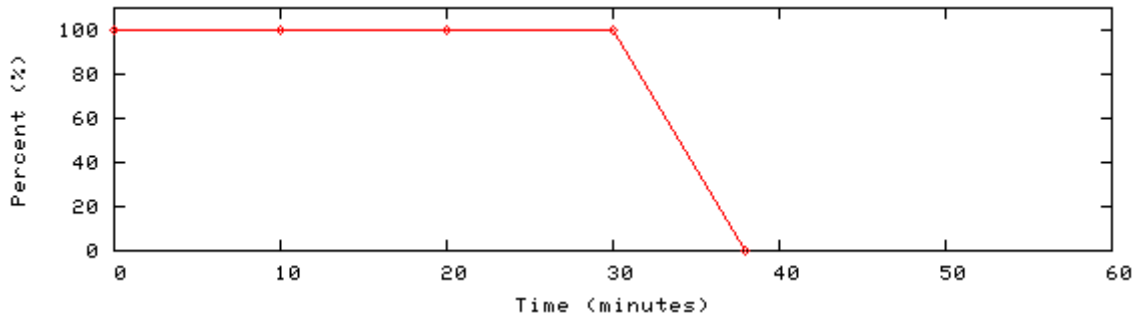
TICS try 2 - Memory/Swap usage



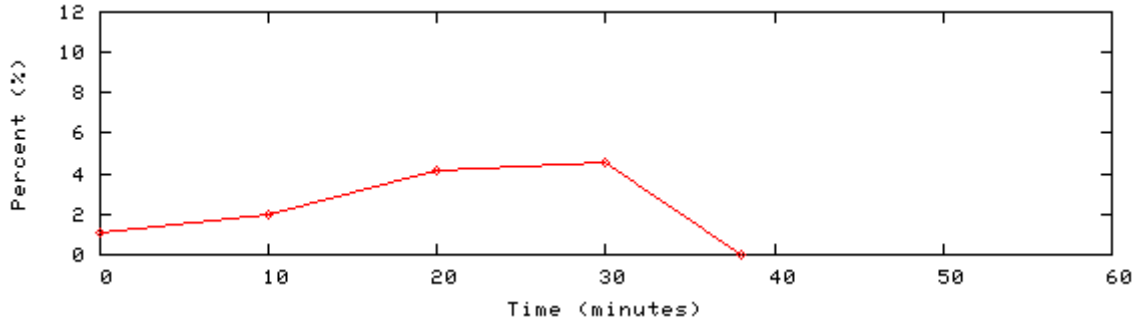
TICS try 2 - User/System processor usage

TICS try 3

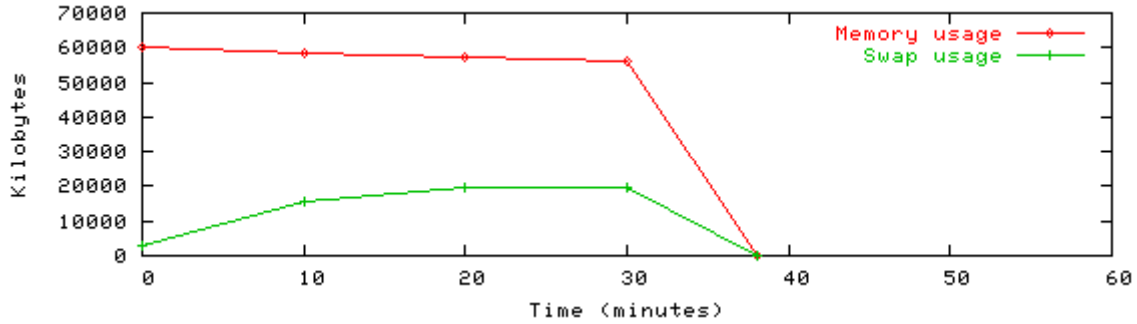
Time	*Battery (%)*	*CPU User (%)*	*CPU System (%)*	*Memory (KB)*	*Swap (KB)*	*CPU Load*
0	100	57,4	45,2	60096	3052	1,08
10	100	37,3	62,6	58160	15544	1,98
20	100	53,4	46,5	56996	19724	4,21
30	100	38,0	61,7	56324	19536	4,58
38	0	0	0	0	0	0



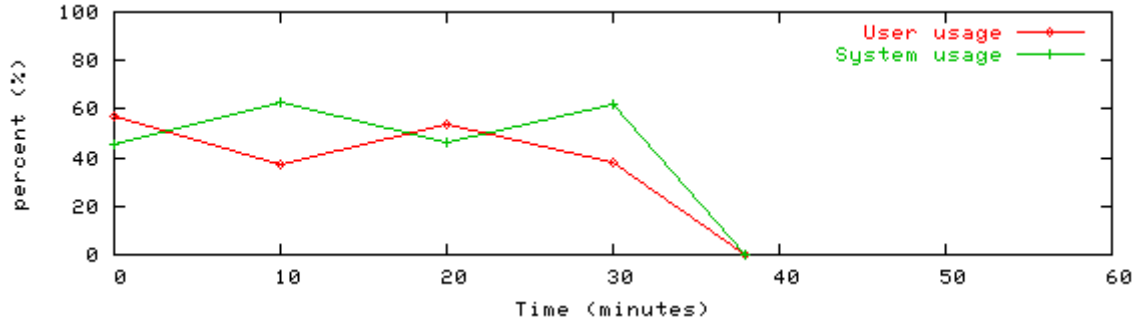
TICS try 3 - Battery usage



TICS try 3 - Processor load



TICS try 3 - Memory/Swap usage



TICS try 3 - User/System processor usage

Appendix B

Hardware specifications

Zaurus Specifications

Display	3.7" TFT CG Silicon display color LCD, 64,000 colors. Semi-transmissive, backlight. Resolution: 640 x 480.
Battery	1800mAh Lithium Ion rechargeable. Battery is user replaceable. This battery is the extended battery, Sharp also makes a smaller capacity battery.
Power consumption	2.7W
Performance	Intel PXA255 400 MHz XScale processor. 128 megs of RAM for storage, ~ 69 of which is available to the user and 64 MB program memory.
Dimension and weigth	4.25" x 3.25" x 1.0 at thickest point of hinge, .8" for the rest of the unit. Weight Approximately 9.5 ounces (120 x 83 x 23.6mm.Weight: 250 grams).
Audio	Built in speaker, and 3.5mm stereo headphone jack.
Expansion	1 SD (Secure Digital) slot. 1 CompactFlash type II slot. IR port.
Software	Linux-based operating system (OpenPDA). Calendar, Address Book, To-Do, and Memo apps, Hancom Office suite: Word processor compatible with Word docs, spreadsheet app compatible with Excel files, NetFront v.3 web browser, E-mail program supporting POP3, SMTP, IMAP4 protocols, ImagePad image viewer and editor, Video Player (MPEG4), Music Player for MP3s, Voice Recorder, Text Editor, Calculator, Clock, City Time and more. Java runtime included.

NIC Specifications

Standard	IEEE 802.11b, Type I CompactFlash
Channels	11 Channels (US) 13 Channels (Europe) 14 Channels (Japans
Transmit	15 dBm
Receive sensitivity	-84 dBm
Network Protocols	IPX/SPX, TCP/IP, NetBEUI
LED	Link
Dimensions	2.37” x 1.70” x 0.13”
Power	3.3V DC, 250mA
Certification	FCC Class B, CE Mark
Range	100m (theory)
Weigth	0,018kg

SD Card Specifications

Clockspeed	25 Mhz
Max MB in read/write mode	3MB/sec (4 data lines)
Power	2.0-3.6V SDLV (low voltage 1.6-3.6)
MTBF (Mean time between failure)	1.000.000 hours
Number if insertions	10.000 Minimum
Data retention	100.000 hours
Endurance	1.000.000 cycles (read/write)
Preventive maintainance	None
ECC Operation	Yes
Capacity	512 MB
Dimensions	24mm x 32mm x 2.1mm
Weigth	0,002 kg

Appendix C

Swap file Shell Script

```
#!/bin/sh
### Start swap file automatically from now on
### Made for the Zaurus ;)
### The symlink is on the rc5.d to start
###  Jacques Weewer

FILE="swapspace"
LOCATION="/mnt/card"

### Functions
start_swap()
{
        echo "Starting up swapfile"
        /sbin/swapon $LOCATION/$FILE
}

case "$1" in
        'start')
        if [ -f $LOCATION/$FILE ]; then
                start_swap  # call start_swap function
        else
                echo "No file found creating one now, please wait or get a cup of coffee ;)"
                # around 30MB swap should be enough and not on screen
                dd if=/dev/zero of=$LOCATION/$FILE bs=1024 count=30000 >/dev/null 2>&1
                mkswap $LOCATION/$FILE >/dev/null 2>&1
                start_swap   # call start_swap function
        fi
        ;;

        'stop')
        if [ -f $LOCATION/$FILE ]; then
                echo "Stop using the swapfile"
                /sbin/swapoff $LOCATION/$FILE
        fi
        ;;

        *)
                echo "Usage: $0 start|stop"
        ;;
esac
```

Glossery

ARM	Advanced RISC Machine
CPU	Abbreviation of central processing unit, and pronounced as separate letters. The CPU is the brains of the computer. Sometimes referred to simply as the processor or central processor, the CPU is where most calculations take place. In terms of computing power, the CPU is the most important element of a computer system.
FPU	Floating Point Unit, a mathematical co-processor which is used by the system to do big and heavy calculations.
Linux	Linux is a computer operating system and its kernel. It is one of the most prominent examples of free software and of open-source development: unlike proprietary operating systems such as Windows, all of its underlying source code is available to the public for anyone to freely use, modify, improve, and redistribute.
NAND	NAND Flash architecture is one of two flash technologies (the other being NOR) used in memory cards such as the CompactFlash cards. It is also used in USB Flash drives, MP3players, and provides the image storage for digital cameras. NAND is best suited to flash devices requiring high capacity data storage. NAND flash devices offer storage space up to 512MB and offers faster erase, write, and read capabilities over NOR architecture. NAND flash architecture was introduced by Toshiba in 1989.
NIC	A network card (also called network adapter, network interface card, NIC, etc.) is a piece of computer hardware designed to allow computers to communicate over a computer network.
PDA	Personal Digital Assistant
RISC	Reduced Instruction Set Computer, a microprocessor CPU design philosophy that favors a smaller and simpler set of instructions that all take about the same amount of time to execute.
SD Card	Secure Digital (SD) is a flash memory memory card format. It is used in portable device, including digital cameras and handheld computers. SD cards are based on the older Multi Media Card (MMC) format, but most (not all) are physically slightly thicker than MMC cards.
SDRAM	Synchronous Dynamic Random Access Memory