Cleveland State University
Department of Electrical and Computer Engineering

Experimental Study of an Ad-hoc Positioning System based on IEEE 802.11-based PDA Network

Electrical Engineering Technical Report
TR – 04 - 103

Srinivas Tera, Amit Panchal, Abhijit Kadam, Lubo Song, and Chansu Yu

October 29, 2004

2121 Euclid Avenue, Stilwell Hall 332
Cleveland, Ohio 44115-2425
http://www.csuohio.edu/electrical_engineering/

Experimental Study of an Ad-hoc Positioning System based on IEEE 802.11-based PDA Network[1]

Srinivas Tera, Amit Panchal, Abhijit Kadam, Lubo Song, and Chansu Yu

Department of Electrical and Computer Engineering
Cleveland State University
2121 Euclid Avenue, SH 332, Cleveland, OH 44115

INTRODUCTION

Ad Hoc Wireless networks are primarily composed of stations that have high mobility and high power. But sensors cannot meet the particular requirement because of the limitations of the power and mobility. The limitations can be overcome if the routing in the network can be done if the nodes do not have to store the routing tables. The self-positioning of nodes in the network can be used as an alternative to the routing tables. The position and orientation information of the nodes in the network can be calculated using algorithms such as Cartesian routing, geo cast routing. When deploying an ad hoc network that uses positions the important decisions to be made are with respect to node density, availability of landmarks, and node capabilities versus the desired quality of the positions. One-hop solutions are desirable for the computational simplicity but the limitations on the number of landmarks and line of sight problems make this option unsuitable. Thus multi hop solutions are more appropriate but at the cost of computational complexity and problems with error propagation and provisioning.

Based on the tradeoff between the capabilities of nodes (Angle and range measurement hardware), the density of the network, the placement of landmarks, and the quality of positions obtained we choose any of the following four multi hop algorithms

1. Angle and Range measurement
2. Angle Measurement
3. Range Measurement
4. Angle/Range free (connectivity based)

In the experiment performed we have tried to do range measurements using the signal strength parameter. We use the Range measurement algorithm to calculate the position. If the quality of error can be compromised for the particular application in comparison to the infrastructure angle/range free algorithm can be used

Ad-hoc Positioning System (APS)

Ad hoc positioning algorithms can be used to determine the positional information of all the nodes in a given network. The network consist a fraction of nodes that have self-positioning capability; these nodes are called "Landmarks". The other nodes rely on this fraction of landmarks to determine the positional information. The nodes measure the distance, range and angle to the landmarks that

[1] This study was in part supported by Center for Environmental Science, Technology and Policy (CESTP) of CSU.

are often erroneous due to noise. The algorithm addresses the problem of constructing the position information of all the nodes in the network given that the fraction of landmarks is very less and the measurements made by the remaining are subject to errors.

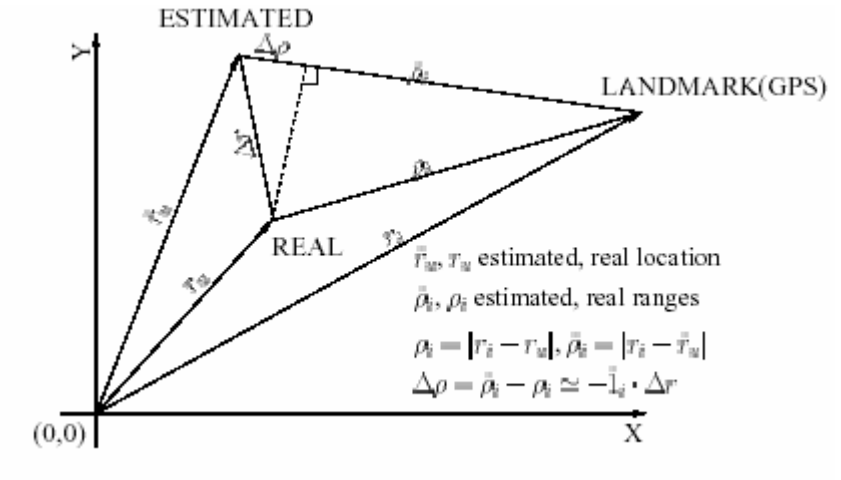
Ad hoc positioning system is a hybrid between the

1. Triangulation technique used in Global Positioning System (GPS).
2. Correction propagation using AODV where the position information is propagated originating from the landmarks to the other nodes.

1. Triangulation

The mathematical concept at the crux of the GPS is triangulation. The triangulation uses ranges to at least 4 known satellites (out of the 24 GPS satellites in the atmosphere) to find the coordinates of the receiver, and the clock bias of the receiver.

- Triangulation



(For typing convenience I have used p for ρ and " ′ " for " ^ ")

$p_i' = |r_i - r_u'| + \epsilon_i$
$p_i = |r_i - r_u| + \epsilon_i$

The correction range Δp is approximated to accommodate a linear system if
$1_i'$ is the unit vector of p_i'

$$1_i' = r_i - r_u' / |r_i - r_u'|$$

and

$$\Delta r = r_u' - r_u$$

the approximate correction

$$\Delta p = p_i' - p_i \approx -1_i'. \Delta r + \Delta\epsilon$$

Performing the above approximation leads us to a linear system in which the unknown is the location correction $\Delta r = [\Delta x \ \Delta y]$

$$\begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \dots \\ \Delta\rho_n \end{bmatrix} = \begin{bmatrix} \hat{1}_{1x} & \hat{1}_{1y} \\ \hat{1}_{2x} & \hat{1}_{2y} \\ \hat{1}_{3x} & \hat{1}_{3y} \\ \dots & \dots \\ \hat{1}_{nx} & \hat{1}_{ny} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

After iteration the corrections Δx and Δy are applied to the current position estimate. The iteration process goes on till the corrections are below a given threshold. Then the linear system is solved using any least square method.

2. Correction Propagation

There are 3 algorithms from which we can choose based on the capabilities of the network system.

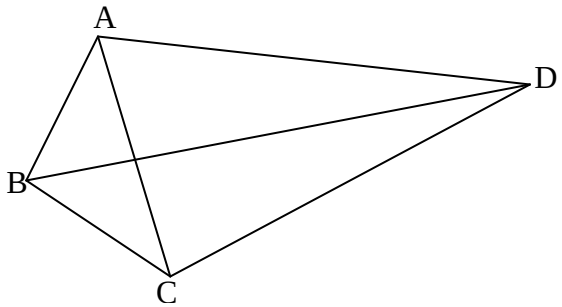
1. *Hop Propagation method*: This scheme relies only on the connectivity of the network.
   - Using the AODV routing all the nodes get shortest paths to the landmarks in hops. Each node maintains a table $\{X_i, Y_i, h_i\}$ and exchange this information with its immediate neighbour. X_i, Y_i are the coordinates of the landmark 'i' and h_i is the distance to the landmark in hops
   - Each landmark calculates the average distance of one hop based on its distances to the other nodes and landmarks. This information is propagated as a Correction to all the nodes in the neighborhood.

     Correction $c_i = \Sigma\ p_j / \Sigma\ h_j$, $i \neq j$, all landmarks j heard by i
     h_j is the shortest distance, in hops from node I to node j and p_j is the straight line distance between i and j

   - The nodes calculate the distances to the landmarks finally based on the corrections received.

This technique is easy to use and relies only on connectivity but can be applied only to networks that have uniform physical characteristics in all directions (isotropic).

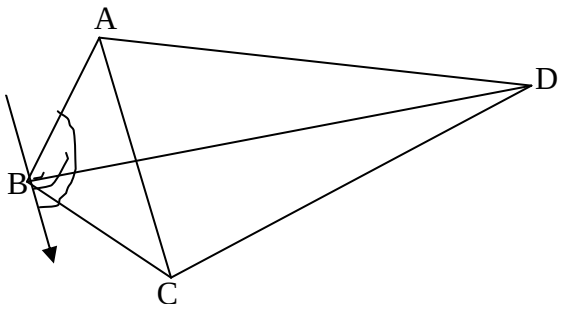
2. *EUCLIDEAN PROPAGATION:*



The Range measurements can be done using this scheme. The algorithm makes use of the range measurements to the neighbors. The algorithm can be explained with the help of the above diagrammatic scenario.

It is necessary that any node B has 2 neighbors A and C which know the range measurements to landmark D. B knows the range measurements to A and C and also know the distance between A and C. So given the quadrilateral ABCD and known values
AB, BC, AC and AD, CD we can calculate the 2nd diagonal BD i.e. the range measurements between the node B and D. Once the ranges have been measured the node A can use trilateration technique to find its location with respect to the landmark D

3. RADIAL TECHNIQUE



Angle measurements of B can be calculated with respect to landmark D if we know all the angles in the triangles BAC and triangle ACD. Thus the knowledge of the radial information of A and C with respect to the landmark D we get the information of their neighbor B. In the case of radial technique similar to the Euclidean method we use GPS but here we use the method of triangulation. Thus the propagation of radial information coupled with triangulation helps us to find the position of the node B.

4. POSITION SCHEME:

Once we make the range and angle measurements we can use both the measurements to exactly pinpoint the locations of the nodes in the network subject to accurate calculations. They can be applied in the case of any kind of network atmosphere isotropic or anisotropic. But the expense we need to bear for the hardware maybe a drawback.

It can be easily observed with more sophisticated measurement hardware the accuracy of the position of the nodes can be greatly improved.

EXPERIMENTS

Objective: The main objective of all the experiments is to obtain experimental values that enable us to implement the Ad Hoc Positioning System Algorithms.

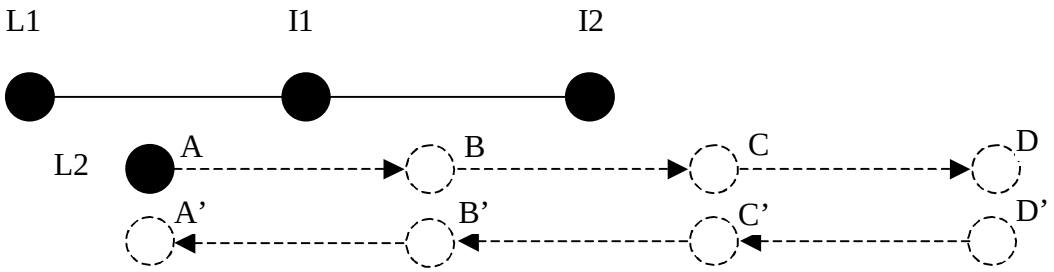
1. Multi-hop Routing Experiments

- Experimental objective

  The objective of this experiment is to investigate if aodv-uu really implements muti-hop routing and if aodv-uu supports dynamical routing path optimization.

- Experimental scenario

  In this experiment, two Toshiba satellite laptops (named L1 and L2) and two Compaq iPAQs (named I1 and I2) are used. The node topology is shown in the following figure.



  L1, I1, and I2 are located on a line. L1 can directly communicate with I1, but can not directly communicate with I2. I1 can directly communicate with both L1 and I2. Now place L2 at position A, then move L2 away and back along the dotted line (A→B→C→D→D′→C′→B′→A′). During the movement, on L2 run command "ping L1 -R" to see if L2 can reach L1 and what routing path is used.

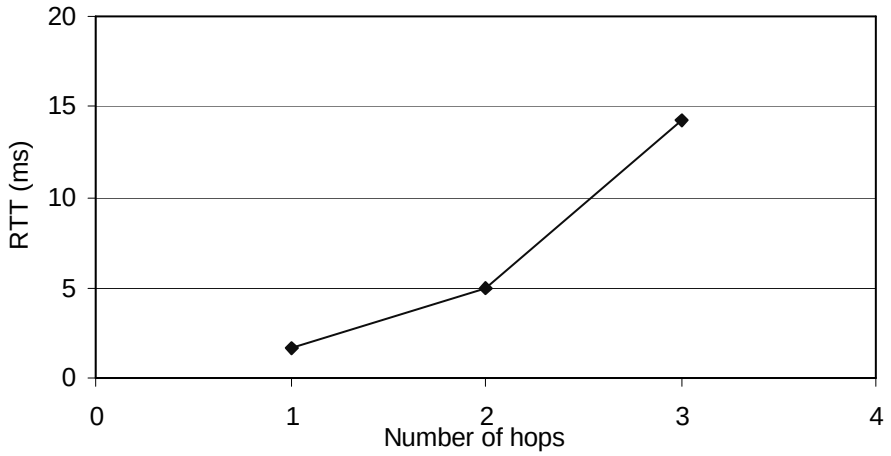
- Experimental results

  1) During the movement of L2, the changes of routing path were observed. The results are shown in the following table.

Location of L2	Routing path from L2 to L1
A	L2→L1
B	L2→I1→L1
C	L2→I2→I1→L1
D	Routing failure
D′	Routing failure
C′	L2→I2→I1→L1
B′	L2→I1→L1

A'	L2→L1

2) The ping command records the RTT (Round Trip Time). The results of RTT vs number of routing hops are shown as follows.



- Experimental conclusions

1) Aodv-uu implements multi-hop routing.
2) Aodv-uu supports dynamical routing path optimization.
3) RTT is proportional to the number of routing hops.

2. Signal Strength Experiments

- Experimental objective

The range measurements for APS can be done based the theoretical relation between signal strength and the distance.

Signal strength α 1/ (distance) ^{2}

The objective of this experiment is to investigate practical feasibility of the theoretical relation between the received signal strength and the distance in the present scenario.

- Experimental scenario

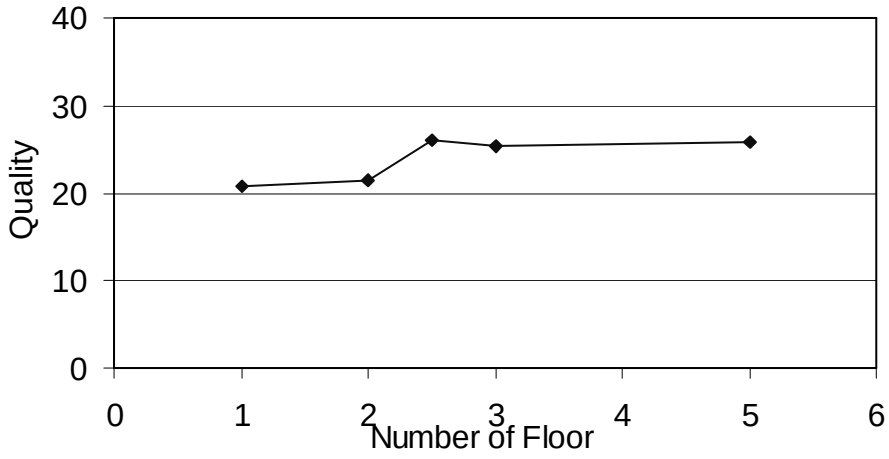
This experiment was done in the University Center of Cleveland State University. In this experiment, one Toshiba satellite laptop (named L1) and one Compaq iPAQ (named I1) are used.
At first, both L1 and I1 are placed on the first floor. Then keep I1 there and move L1 upstairs though Floor 1 → Floor 2B (inside) → Floor 2B (outside) → Floor 3 → Floor 5. During the movement, on L1 run ping command record the

information of the received (this information is available in the file /proc/net/wireless) and we obtain it using the command *iwspy*.

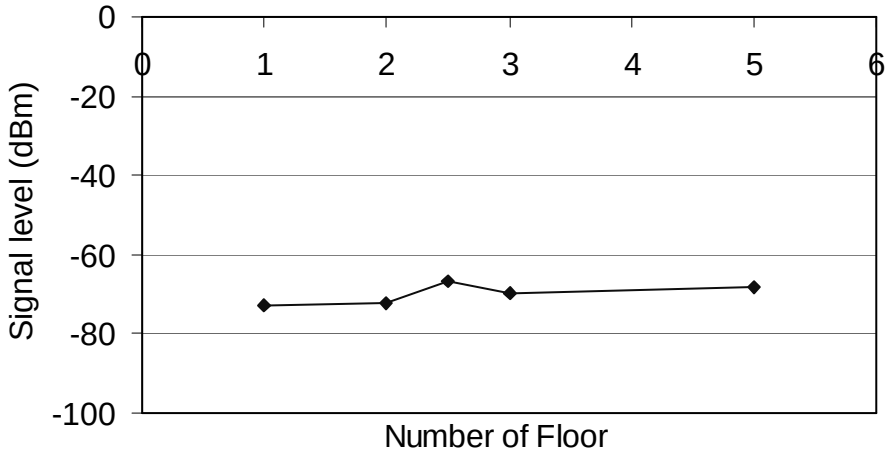
The user interface is composed of 3 programs and a /proc entry. The main goal of the Wireless Extension development effort wasn't the user interface, so this interface is quite basic.

- Experimental results

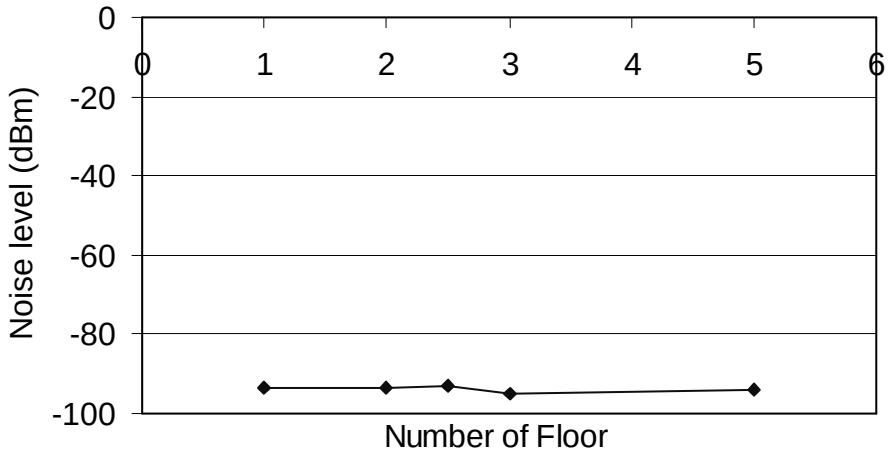
  1) Signal quality vs Distance



  2) Signal level vs Distance



3) Noise level vs Distance



- Experimental conclusions

1) We found that $Signal\ quality = \frac{Signal\ level}{Noise\ level}$

2) The results do not show that the signal strength decreases if the distance is larger. In reverse, the signal strength received on Floor 3 and 5 is stronger that that received on Floor 1. These unexpected results might be due to the following reason: There are some obstacles on Floor 1. The communication environments on Floor 3 and 5 are more like Open area. So doing the experiment in open area may be necessary.

CONCLUSION:

The AODV-UU version of the routing protocol effectively has been implemented on all the nodes we have experimented. The routing of packets has been simulated in various scenarios like single hop and multihop scenarios and the signal strengths have been calculated to make the range measurements. The experimental results show a lot of anomaly in the signal strength variations. This might necessitate the provisioning of more precise measuring devices. The location chosen for the experiment "Cleveland State University – University Center" being a very crowded place influence the signal strength values. Thus repeating the same experiment in a more open place may give better experimental results.

References:

1. *Ad Hoc Positioning System (APS)* by Dragos Niculescu and Badri Nath Computer Science Department, Rutgers University at GLOBECOM 2001, San Antonio, Nov 2001
2. *Error characteristics of ad hoc positioning systems (aps)* by Dragos Niculescu and Badri Nath Computer Science Department, Rutgers University at International Symposium on Mobile Ad Hoc Networking & Computing.
3. *AODV-UU* website from the University of Uppsala http://user.it.uu.se/~henrikl/aodv/

APPENDIX 1

Aodv-uu Installation and Configuration on Laptop

1. Aodv Installation

AODV is an on demand routing protocol used by MANETS. A version of AODV "AODV-UU" was implemented at the Uppsala University and the open source code is available at following URL for download

1) Download the aodv-uu.0.8.1.tar.gz file
2) Untar the file using the command
   tar zxvf aodv-uu.0.8.1.tar.gz
3) the contents will be extracted into the folder aodv-uu0.8.1
4) enter the directory and type the following command *make install*
5) Verify that the location of the header files is matched with the MAKEFILE
6) Once the compilation is complete, kaodv.o and aodvd files are generated

Notes

The compilation gave problems in the earlier stages

The most common reason for this happens to be messing up the kernel header files or installing kernel over already present kernel.
A precaution that avoids this most of the times is to try logging in as a separate user and use the root login only if necessary.

**Careful*

IPTables -F

We need to flush out the firewalls on all the nodes for the communication to take place. This is done using the command

"iptables –F "

2. Wifi Configuration

1) Open Network configuration from the startup menu > system settings > network or use the command "neat" in a new terminal.
   # neat
   This takes us to the "Network Configuration" dialog box.
2) In the network configuration dialog box choose the sub dialog box for "Devices" and from the menu bar choose "New".
   This takes us to an "Add new Device Type" dialog box.
3) Choose the device type as "Wireless Connection" and click *"forward"*
   This takes us to a "Select Wireless Device" dialog box.

4) We get a dialog box for selection of a wireless device if the WiFi card is already present in the dialog box chooses the device else choose "Other wireless card" and perform the following steps.
   - You get a "Select Ethernet Adapter" dialog box
   - Select the *Adapter* (Wifi card) from a drop down menu bar.
   - Choose the *Device* (Ethernet interface) i.e. eth1 or eth0
   - Click forward

   This takes us to the Configure Wireless Connection
5) Choose the
   - *Mode* as "Ad-Hoc" Mode.
   - Choose a common *Network Name (SSID)* for all ad hoc devices you choose to network by choosing Specified and type in a common network name (we have chosen the name as "aodvd").
   - Choose a common "Channel" between the numbers 1 to 14
   - Choose a transmission rate from the options (11 Mbps)
     Click forward

   This takes us to a "Configure Network Settings "
6) Choose the dhcp for internet configuration
   For Ad Hoc configuration statically assign the IP address
   Our experimental values were (192.168.10.10, 192.168.10.20, 192.168.10.30, 192.168.10.40, 192.168.10.50)
   This takes us to a Create Wireless Device dialog box which is nothing but a confirmation dialog box.
   Click Apply and we are good to go!.

Aodv-uu installation and configuration on iPAQ

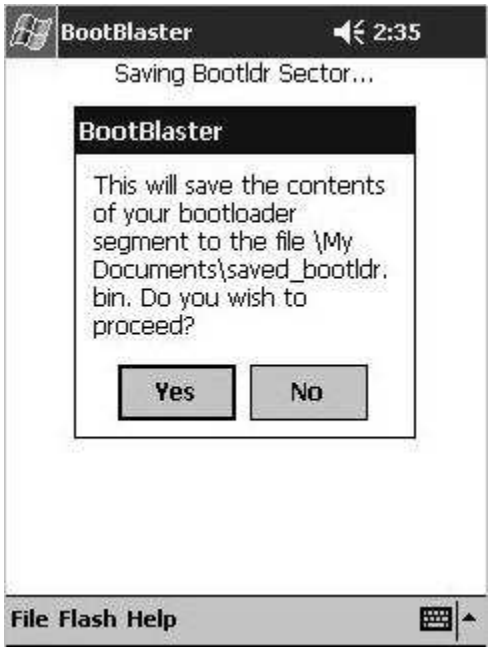
1. Install Bootloader

- Transfer BootBlaster and bootldr to your iPAQ using ActiveSync.
  1). If ActiveSync isn't already installed on your Windows PC, install it by inserting the iPAQ Pocket PC Companion CD-ROM into the PC's drive and following the ensuing instructions.
  2). Copy the BootBlaster and bootldr files that came with your Familiar Linux distribution to your Windows PC if they're not already there.
     - If you're installing on an H3900 iPAQ, the files are named BootBlaster3900-2.6.exe and bootldr-pxa-2.21.12.bin.
     - Otherwise, they're named BootBlaster_1.19.exe and bootldr-sa-2.21.12.bin.
  3). Plug the iPAQ cradle into an AC power outlet.
  4). Connect the USB connector from the cradle to the Windows PC.
  5). Slide the iPAQ into its cradle. If a "Set Up a Partnership" screen appears on the PC, choose "No" and then click Next.
  6). Copy BootBlaster_1.19.exe or BootBlaster3900-2.6.exe to the default folder on the iPAQ by clicking Explore in ActiveSync and dragging their icons there. Ignore any "may need to convert" messages.
  7). Do the same thing for bootldr-sa-2.21.12.bin or bootldr-pxa-2.21.12.bin.

- Start BootBlaster.
  1). Select "Start -> Programs" on the iPAQ touchscreen.
  2). Tap on File Explorer.

3). Tap on BootBlaster.

- Save your PocketPC image for later restoration, if desired.
  1). Execute "Flash -> Save Bootldr .gz Format" in BootBlaster to save the bootloader in file "\My Documents\saved_bootldr.gz" on the iPAQ.



Note that the Linux Bootloader will also boot PocketPC, so restoration of this file is not generally required. Right at the moment, there is a bug in the Linux Bootloader which causes PocketPC to reinitialize itself every few boots. You may indeed wish to keep and restore this bootloader if you restore PocketPC.

2). Execute "Flash -> Save Wince .gz Format" in BootBlaster to save the PocketPC image in file "\My Documents\wince_image.gz" on the iPAQ. This takes two to three minutes.

- If no backup of Pocket PC is desired, you can skip this step entirely.

- Note that this procedure saves your bootloader and Pocket PC executable image: it does not preserve any data you may have entered in your iPAQ under Pocket PC. So also synchronize your iPAQ to your host to preserve this data. Note that Familiar does not *yet* have any way to resynchronize this data to Linux (we hope/expect to have Linux<->Host synchronization in a near future release).

- Copy saved_bootldr.gz and wince_image.gz to your Windows PC.
  1). Select "View -> Refresh" in the ActiveSync Explore window on the PC. Icons for the saved_bootldr.gz and wince_image.gz files should appear.
  2). Drag the saved_bootldr.gz and wince_image.gz icons from the ActiveSync Explore window to a local folder on your PC.

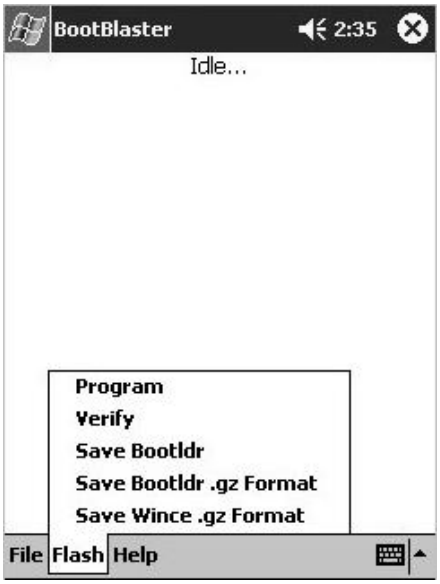
  As with any backup files, please store saved_bootldr.gz and wince_image.gz in a safe place. We highly recommend verifying the built-in gzip checksum in both files before presuming your backup is safe (see the gzip man page for details).

- Install the bootloader.

  Before continuing, be sure that the iPAQ is plugged into external power, and that the battery is charged, to protect against the small chance of power

failure during the very limited period the iPAQ is reprogramming the bootloader flash. Do NOT touch the power button or reset button on your iPAQ until you have performed the "Verify" step below.

From the "Flash" menu on BootBlaster, select "Program".



A file dialog will open allowing you to select the bootloader to use. Select bootldr.bin.gz, which may have a version number embedded in it. We use a gzip file because it has an internal checksum.

Wait patiently. It takes about 15 seconds to program the bootloader. Do not interrupt this process, or the iPAQ may be left in an unusable state.

From the "Flash" menu on BootBlaster, select "Verify".

1). If it does not say that you have a valid bootloader, do *NOT* reset your iPAQ, do *NOT* turn off your iPAQ.
2). Instead, try programming the flash again.
3). If that doesn't work, program your flash with your saved bootloader.
4). If that doesn't work, send e-mail to bootldr@handhelds.org and/or get on the IRC and ask for help. Leave the iPAQ plugged in and do *NOT* reset it or turn it off.

If everything has gone well, you have successfully installed the CRL bootldr program, which can run either Linux or PocketPC. As yet, your PocketPC image is intact and should restart normally; the next step actually installs Linux (overwriting Pocket PC).

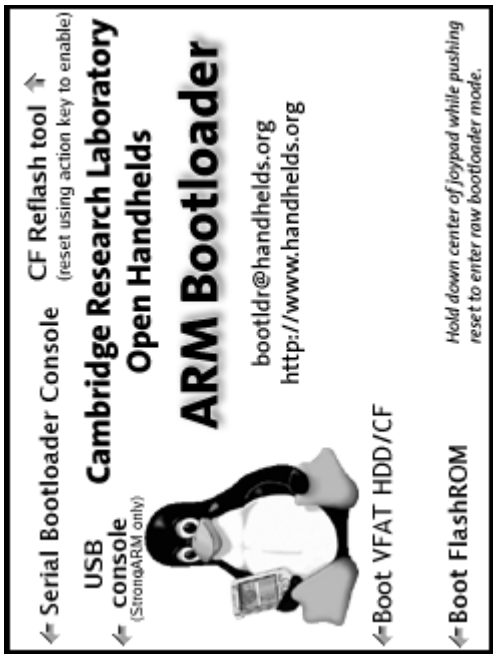
2. Install Familiar V0.7.2 with A Serial Line

You will need a serial sync cable or serial sync cradle. The dual USB/Serial cradle that comes with the H3800 and H3900 will also work. You will need to use a terminal program such as minicom, kermit, or Hyperterminal.

If you use minicom or kermit, you will need to use an external ymodem program such as sb, which is available in the Linux lrzsz package.

1) Hold down the joypad and push the reset button on the iPAQ. You will need to remove it from the cradle to access the reset button.

   - For non-H5xxx: When the bootloader splash screen appears, release the joypad.



   - For H5xxx: When the iPAQ buzzes, release the joypad. The screen will not change from whatever was previously displayed (blank, PocketPC, etc). If the iPAQ does not stop vibrating, remove the AC adapter and the battery, then reinsert the battery and the AC adapter and perform try this step again.

2) Press the calendar button on the iPAQ. This is the leftmost action button, labelled "Serial Bootldr Console" on the screen.

3) Make sure the terminal emulator is up and running, and is properly interacting with the bootloader. Proper interaction consists of being able to issue commands, and get responses (e.g. the help command should return the bootloader usage). Your terminal emulator must be set to **115200 8N1 serial configuration, no flow control, no hardware handshaking**. Failing to use these settings will lead to trouble, so double and triple check all settings.

   If you cannot interact with the bootloader, make sure your terminal settings are correct, the iPAQ is properly connected to the host computer, and the iPAQ is actually on. If everything seems fine, try restarting the host terminal emulator and resetting the iPAQ again.

   Hyperterminal is particularly ill-behaved. Sometimes it uses 100% of the CPU without allowing any interaction with the iPAQ. In that case, you will need to use the task manager to terminate Hyperterminal before you can restart it.

4) At the "boot>" prompt, issue the following command: load root
5) Proceed to send or "upload" the jffs2 file (from the tarball that you downloaded earlier) with ymodem, using the terminal emulator. If you have not used ymodem before or you have any trouble with this command, please see handhelds-faq/getting-started.html#USING-XYZMODEM.

   Note that the bootldr now expects ymodem by default, not xmodem as in earlier versions. If you are unable to use ymodem for some reason, you can revert to xmodem operation with the command set ymodem 0

   You should see something like:

```
boot> load root
loading flash region root
ready for YMODEM download..
Erasing sector 00140000
Erasing sector 00180000
Erasing sector 001C0000
Erasing sector 00200000
.
.
.
addr: 00360000 data: 781590DB
addr: 00370000 data: 642637AE
addr: 00380000 data: E0021985
addr: 00390000 data: 15DA97EC
Erasing sector 00FC0000
writing flash..
addr: 00100000 data: E0021985
addr: 00110000 data: E3BAD617
addr: 00120000 data: 0FA1F57B
addr: 00130000 data: 9343AEEB
.
.
.
```

```
addr: 00600000 data: E0021985
addr: 00610000 data: FFFFFFFF
addr: 00620000 data: FFFFFFFF
addr: 00630000 data: FFFFFFFF
verifying ... formatting ... done.
boot>
```

6) At the "boot>" prompt, issue the following command: boot

   Linux should now start booting.

Aodv-uu Cross-compilation and Installation on iPAQ

1) First download the cross-compiler, for example:
   > wget ftp://ftp.handhelds.org/pub/linux/arm/toolchain/arm-linux-toolchain-current.tar.gz
   the existing file may have a different name, before downloading check the file name please.

2) Unpack the cross-compiler according to instructions in ftp://ftp.handhelds.org/pub/linux/arm/toolchain/README, usually:
   > cd /; tar zxvf /path/to/arm-linux-toolchain-current.tar.gz

3) Retrieve the kernel source code matching the kernel used on the ARM Device. You may check the URL below for binary pre-compiled kernel packages, installable via ipkg. There are no guarantees that these are always available or up to date...

   http://www.docs.uu.se/docs/research/projects/ape/familiar/

   Otherwise, for the Familiar distribution, the kernel source code can be retrieved via anonymous cvs:

   > export CVSROOT=:pserver:anoncvs@cvs.handhelds.org:/cvs
   > cvs login
   Password=anoncvs

   Get the matching version with "-r":

   > cvs export -r K2-4-19-rmk6-pxa1-hh30 linux/kernel
   Note: the kernel for familiar 0.7.2 is 2.4.19-rmk6-pxa1-hh30

4) Re-link the "asm" and "linux" include directories in arm cross-compiler tree to point to those in the ARM kernel source tree:

> ln -s /path/to/arm-kernel-source/include/linux /path/to/cross-compilier/arm-linux/include/linux
> ln -s /path/to/arm-kernel-source/include/asm /path/to/cross-compilier/arm-linux/include/asm

5) Make sure the arm compiler is in the PATH and that /usr/src/linux points to the ARM kernel source.
> export PATH=$PATH:/path/to/cross-compilier/arm-linux/bin
Another way is to modify the Makefile in aodv-uu to explicitly specify the path of cross-compilier.

> ln -s /path/to/arm-kernel-source /usr/src/linux
The symbolic link may not be nessary.

6) Since the default Familiar kernel do not have the proper netfilter support for AODV-UU (CONFIG_IP_NF_QUEUE) it is necessary to compile a new kernel. Follow the instructions at
http://www.handhelds.org/handhelds-faq/development.html.
Build ipkg-packages for easy installation. Then install kernel using ipkg. It may be possible to transfer only the ip_queue.o module so that installing a new kernel can be avoided.

Actually the instructions at the above web site do not work well. What I did was using the dowanloaded kernel source code in step 3) to generate ip_queue.o module. The steps are as follows. (Build kernel to generate ip_queue.o module)

> make ipaqsa_config

> make old_config

modify .config to change the setting for CONFIG_IP_NF_QUEUE as follows.
CONFIG_IP_NF_QUEUE=m

> make dep

> make zImage

> make modules

After that the ip_queue.o module will be generated.

7) Compile AODV-UU for ARM:

> make arm

To install, copy kaodv.o and aodvd to the ARM device.

APPENDIX 2

NETWORK TOOLS

/proc/net/wireless

- *status* is the status reported by the modem.
- *Link quality* reports the quality of the modulation on the air (direct sequence spread spectrum) [max = 16].
- *Level* and *Noise* refer to the signal level and noise level [max = 64].

The *crypt discarded packet* and *misc discarded packet* counters are not implemented.

IWSPY

NAME

iwspy - Get wireless statistics from specific nodes

SYNOPSIS

iwspy *interface*
iwspy *interface* **[+]** *DNSNAME* **|** *IPADDR* **|** *HWADDR* **[...]**
iwspy *interface* **off**

DESCRIPTION

Iwspy is used to set a list of addresses in a wireless network interface and to read back quality of link information for each of those. This information is the same as the one available in */proc/net/wireless*: quality of the link, signal strength and noise level.

This information is updated each time a new packet is received, so each address of the list add some overhead in the driver.

Note that this functionality works only for node part of the current wireless cells.

PARAMETERS

You may set any number of addresses up to 8.

DNSNAME | **IPADDR**

Set an IP address, or in some cases a DNS name (using the name resolver). As the hardware work with hardware addresses, **iwspy** will translate this IP address through *ARP*. In some case, this address might not be in the ARP cache and **iwspy** will fail. In that case, *ping*(8) this name/address and retry.

HWADDR

Set a hardware (MAC) address (this address is not translated & checked like the IP one). The address must contain a colon (**:**) to be recognised as a hardware address.

+

Add the new set of addresses at the end of the current list instead of replacing it. The address list is unique for each device, so each user should use this option to avoid conflicts.

off

Remove the current list of addresses and disable the spy function

We use the iwspy to make the range measurements in the experiment.

NET FILTER AND IPTABLES

Netfilter and iptables are building blocks of a framework inside the Linux 2.4.x and 2.6.x kernel. This framework enables packet filtering, network address [and port] translation (NA[P]T) and other packet mangling.

Netfilter is a set of hooks inside the Linux kernel that allows kernel modules to register callback functions with the network stack. A registered callback function is then called back for every packet that traverses the respective hook within the network stack.

Iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists out of a number of classifiers (iptables matches) and one connected action (iptables target).

Netfilter, iptables and the connection tracking as well as the NAT subsystem together build the whole framework.

All the nodes we need for the experiment must be netfilter enabled and we need to flush out the iptables before performing the networking experiments.

The command to flush out the firewalls is

#iptables -F