

How to turn the Acer Aspire One into a wireless access point

Krastyo Komsalov

<kkomsalov@gmail.com>

October 9, 2011

Table of Contents

1. [Introduction](#)
2. [Hardware description](#)
3. [Some possible network configurations:](#)
 - a. [Keep old router and append Aspire inside, providing two additional wireless networks.](#)
 - b. [Using only Aspire as AP.](#)
 - c. [Bridging between the two private networks and NATing only "Welcome" public network.](#)
4. [Configuration \(a.\) - with installation instructions for all necessary software for any configuration. It is actually the initial configuration.](#)
 - a. [The easiest way of installing Slackware on Acer Aspire One](#)
 - b. [Kernel configuration](#)
 - c. [Remote access – XDMCP](#)
 - d. [For consideration:](#)
 - e. [Necessary:](#)
 - f. [Optional programs:](#)
5. [Configuration \(b.\) - VLAN's and switches](#)
6. [Configuration \(c.\) - bridging](#)
7. [Clients setup – WPA and WPA2 with self-signed certificates.](#)
 - a. [Linux.](#)
 - b. [Mac OS X 10.7.2](#)
 - c. [Ipod.](#)
 - d. [Windows 7](#)
8. [Additional administrative tasks you may consider necessary.](#)
 - a. [Limit bad clients - bit torrent](#)
 - b. [Traffic shaping](#)
 - c. [Cache DNS server.](#)
 - d. [Log configuration.](#)
9. [Some final words.](#)
10. [Copyright](#)

Introduction

The main reason for writing this document is to share my surprise of how easy it is to convert the Aspire One into a wireless access point on Slackware and how good the Aspire One

hardware is for this. Accidentally, I happened to have some free time and one three year old Aspire in my hands so I decided to do something about my growing dissatisfaction with my home router. I live in a crowded Wi-Fi area with over 30 access points coming from the apartments around me and my router obviously has troubles with this. What I wanted was a wireless router over which I will have full control of all settings: log levels control, ability to install additional software for traffic analysis, a decent iptables firewall, RADIUS; in short a wireless router with full Linux installed on it.

a. I chose to use Free RADIUS, since I wanted not only support for WPA and the ability to append eventual access points with roaming, but also the extensibility to any user data base, from local flat files to LDAP. Hostapd has its own integrated RADIUS, but the freedom of having FreeRADIUS was so tempting; besides the setup with flat ASCII users file is really easy. In this configuration RADIUS is set up to use files.

b. Ipv6 and DNSSEC are here to stay and no embedded router has all the functionality which I have with Linux. Ipv6 and DNSSEC configuration is not included in this HOWTO guide, but the freedom to configure them is there.

c. I wanted to have not only a standard firewall, but the full power of iptables. A simple functionality like SSH tunnels that allows home access from school for my kids is tricky with my router and traffic shaping is simply not available. For this reason the Firewall Builder is included in this configuration with a basic rule set. I think it is by far the best firewall management solution on the market and it is free for Linux users.

d. I wanted to have at least two wireless networks “different ssid”, to open safely one of them and share some of my bandwidth with my neighbours. This I hope will make me feel less ripped-off next time I pay my internet bill.

e. The other solution OpenWrt had two disadvantages: my router is too weak to support OpenWrt and any router that is powerful enough for everything that I want will cost nearly as much or more then the Aspire; which I already have.

f. There are no requirements or specific instructions for any specific Linux collection in this configuration. I chose Slackware because I love it, I can’t put it in better words then it is in “[Ten reasons for giving Slackware Linux a go](#)” by Jack Wallen.

Hardware description

My Acer Aspire One has a Model KAV10, which is one of Acer’s oldest models. Since then Acer has produced many new models, but the only important part for this configuration is the model of the wireless adapter within it. From what I found Acer has been changing the adapter in nearly all newer models of Aspire. All the models I checked come with a different adapter made by Atheros (although its important to verify the producer). If you are thinking of buying the laptop, check in advance its wireless adapter. For mine, lspci and dmesg are giving this:

```
bash-4.1# lspci
```

```
01:00.0 Ethernet controller: Atheros Communications Inc. AR5001 Wireless Network Adapter (rev 01)
```

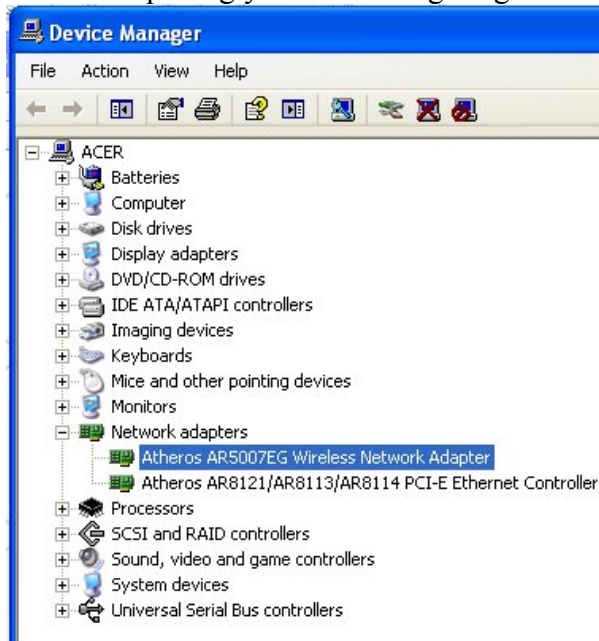
```
03:00.0 Ethernet controller: Atheros Communications AR8121/AR8113/AR8114 Gigabit or Fast Ethernet (rev b0)
```

```
bash-4.1# dmesg |grep Atheros
```

[10.367156] ath5k phy0: Atheros AR2425 chip found (MAC: 0xe2, PHY: 0x70)

This is really good news as it seems that Atheros is one of the best supported adapters on Linux (the people from The [MadWifi project](#) are doing excellent work).

Surprisingly Windows is giving different information:



If it turns out that your adapter is different from mine, you will have to investigate further in order to be sure it supports AP mode. To accomplish this you will need the “iw” command. You probably have it already, but for the source and some documentation go to: <http://linuxwireless.org/en/users/Documentation/iw>. The most informative syntax is:

iw list

It will give you a pretty long output. In it look for the part that is similar to the following:

Supported interface modes:

- * **IBSS**
- * **managed**
- * **AP**
- * **AP/VLAN**
- * **monitor**
- * **mesh point**

Supported commands:

If there is a line “* **AP**” it is good news, you have the necessary AP support for hostapd.

If it turns out that your chipset is different from mine you can check if it is supported on the MadWifi website. The MadWifi website is also by far the best source of documentation I have found. This will be one of your primary sources of knowledge when you decide to adjust to your needs, experiment or simply improve the configuration given below.

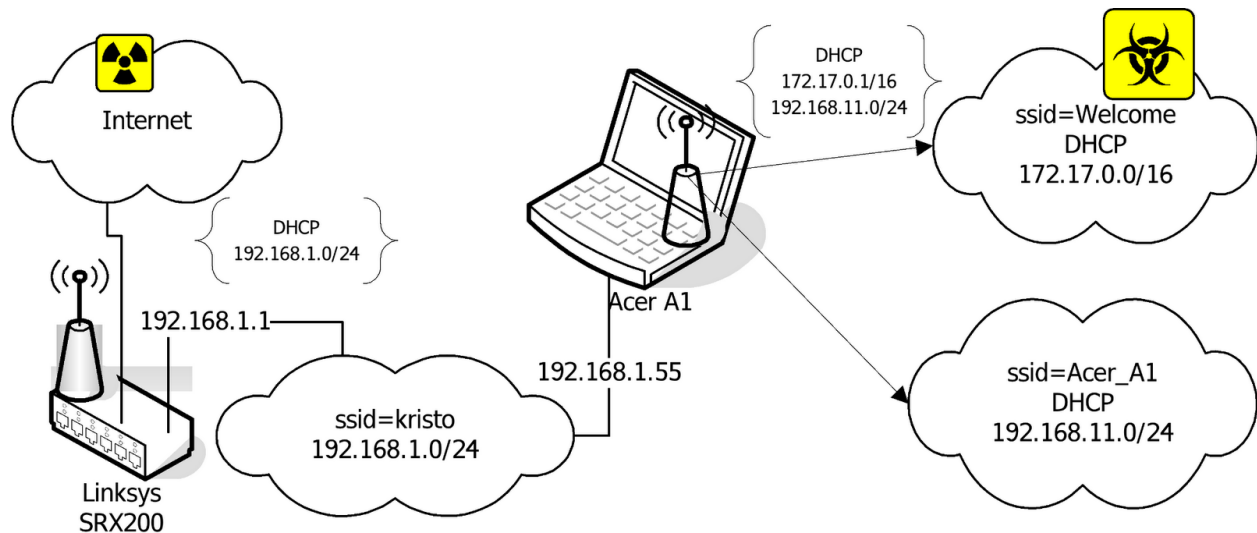
If you do not have Linux already installed, you can boot it from Slackware or SystemRescueCd USB stick and do some investigation on your Aspire.

The model of my old router “Linksys SRX 200” shown as part of two of the three configurations is not important. You may use any wireless router if you have any or avoid using

it at all if you decide to permanently dedicate the Aspire as your Wireless router.

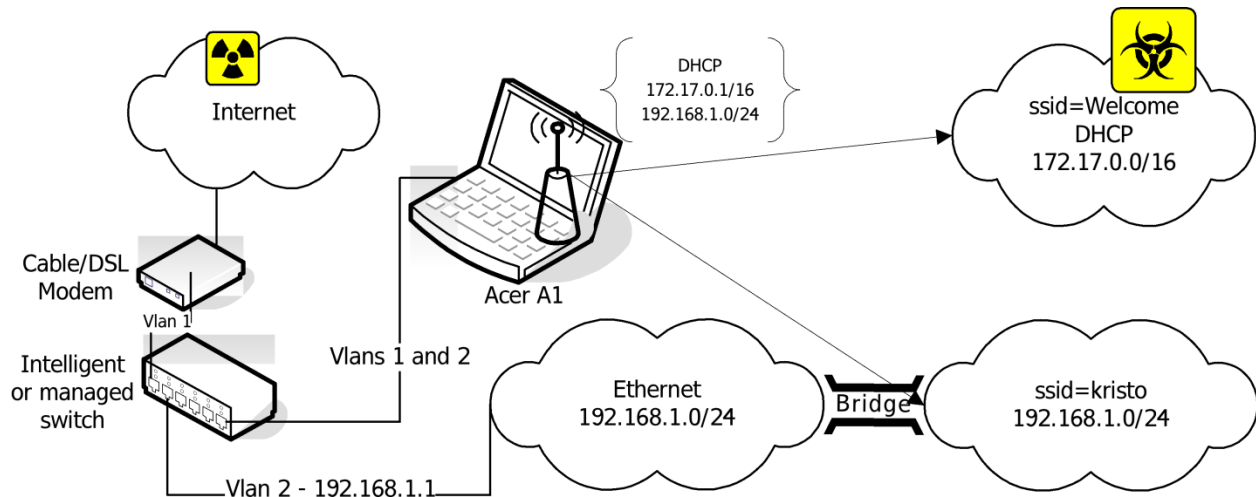
3. Some possible network configurations:

- a. **Keep your old router and append the Aspire inside, providing two additional wireless networks.**



In this configuration the Ethernet port of the Aspire is connected directly to “SRX 200”. This solves the problem of Aspire having only one Ethernet port. Two Ethernet ports required are one for the Internet link the other for the internal switch to provide Internet access to Ethernet connected computers. The two wireless networks are NAT’ed to the 192.168.1.55 IP address. The reason for this is not only to put ssid “Welcome” in a separate network and simplify firewalling, but also to resolve some NAT and routing problems. First the devices in 192.168.1.0/24 must have a route to 192.168.11.0/24. I had no problem adding routes within Linux and Solaris, but my network printer simply has no such thing as a routing table in its web interface. Second, appending the route in “SRX 200” is not a problem, but “SRX 200” refuses to NAT any other network than the one connected to its interface. This is probably solvable by sub-networking its network, but I think the next configurations (b.) and (c.) are better solutions. Even with all its disadvantages, I think this configuration is the best starting point as it will not cause any disruptions or changes in your current setup until all configurations on Aspire are done and tested; then it can easily be converted to any other.

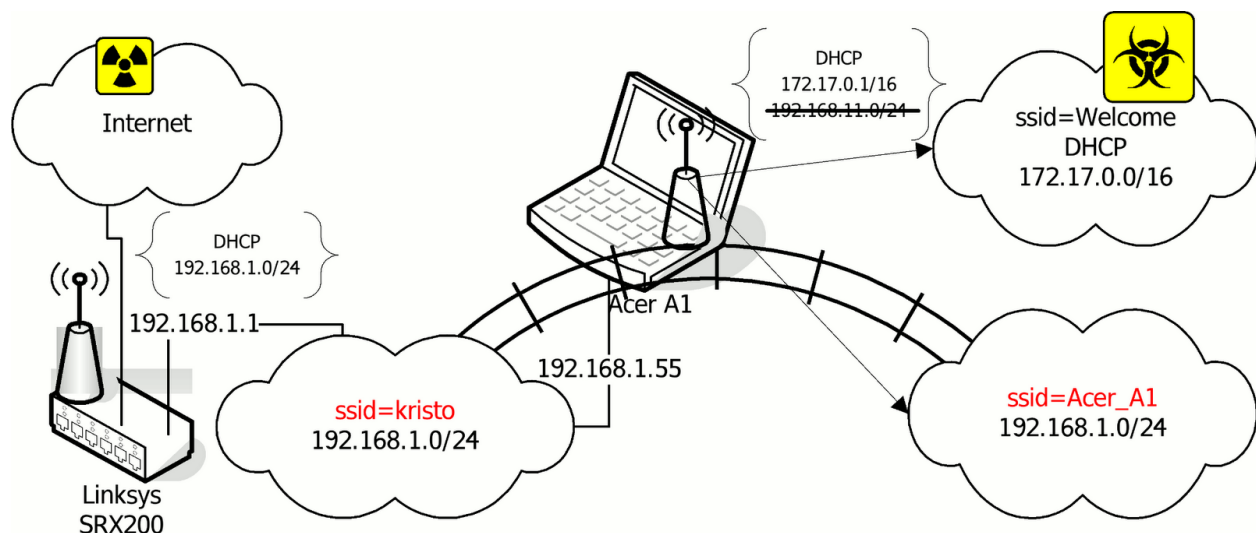
- b. **Using only Aspire as AP.**



This configuration is setting you free from any later worries and is the optimal variant, but there is a price to pay. Since the Aspire has only one Ethernet adapter you have to append a second one. There are two solutions. The first one “shown on the picture” is to use an intelligent or managed switch to VLAN the eth0. The second one is to use a USB to Ethernet adapter, to convert one of the USB ports to Ethernet. The drawback of the switch solution is that it is much more expensive, though it has the advantage of speed, stability and simplicity. The USB to Ethernet adapter is much cheaper, but it comes with a doubtful Linux driver support and uncertain speed and reliability.

There is one more small detail to mention: depending what kind of Internet connection you have there will be different setups for the uplink adapter. If you use a cable connection than it simply has to be on DHCP. In the case of ADSL (my case) you will need to configure a PPPoE. On Slackware you simply have to run a pppoe-setup script.

c. **Bridging between the two private networks and NATing only “Welcome” public network.**



In this configuration the interfaces eth0 and wlan0 are bridged. The network 192.168.1.0/

24 can be accessed either through “kristo” or “Acer_A1” ssid. The DHCP server on the Aspire is bind only to the wlan0_0 interface. NAT to 192.168.1.55 is only done for 172.17.0.0/16. The computers assessing the 192.168.1.0/24 network through ssid “Acer_A1” are getting IP addresses from the DHCP server on “SRX 200”. Other solutions will be available if the DHCP server on “SRX 200” was more manageable. For example, instead of bridging the two parts of 192.168.1.0/24, it will be more elegant to subnet 192.168.1.0/24 and setup a DHCP helper for the part in ssid “Acer_A1”.

This configuration has two advantages. The first is that it avoids both, the routing problem of the solution (a.) and the consequent NAT’ing of the “Acer_A1”. Second, it allows the Aspire to be turned off while networking remains through your old router. If you can’t afford to dedicate your Aspire as AP, this is the best configuration. It provides a stable network when you do not need the Aspire and allows you to disconnect the Aspire from the network for personal use, while preserving a functional network.

4. Initial Configuration (a.) - installation instructions for all the necessary software for all configurations.

I installed all the necessary programs from the source in /usr/local. I left some configuration files in /usr/local/etc and moved some in /etc/. There are Slackware packages on SlackBuilds.org or you can make your own if you decide it is worth the effort, considering that installing it from source is easier.

a. The easiest way of installing Slackware on Acer Aspire One

This chapter is probably unnecessary, but I love to preach about Slackware.

You need a Linux FTP server, to host a Slackware and a USB stick.

First you have to create a Slackware mirror by getting the script “[mirror-slackware-current.sh](#)“ from [Alien Pastures](#) and running it. The script will put the mirror by default in the /home/ftp directory, which is exactly where you need it for the last step.

After this is done insert a USB stick, go to the directory /home/ftp/pub/Linux/Slackware/slackware-current/usb-and-pxe-installers, unmount the USB if it is mounted and run the script to make a startup USB.

```
pwd
/home/ftp/pub/Linux/Slackware/slackware-current/usb-and-pxe-installers
dmesg |grep sd
[86504.700524] sdb: sdb1
[86504.708517] sd 6:0:0:0: [sdb] Assuming drive cache: write through
umount /dev/sdb1
sh usbimg2disk.sh -i usbboot.img -o /dev/sdb
```

Boot from the USB and install the Slackware. Here is the how to do it: <http://www.slackbook.org/html/installation.html>. Use a network cable (makes things easier) to connect the laptop to the network, you will need it anyway to access the machine during the configuration of the access point, and after as your uplink.

b. Kernel configuration

It is a good idea to start by recompiling your kernel. Click on [.config](#) to download my configuration file in /usr/src/linux. This is not a fully optimized version and only the processor is set to Intel Atom and some obviously unnecessary stuff is removed. I chose not to put here a version that is too customized to my needs. I used the `-j 8` option since it makes bzImage and modules faster. It seems `-j 8` gives the best results, but on the first compilation you will not have this advantage. Anyway it will take forever to compile even with the `-j 8` option. It is important not to forget to reinstall the MadWifi drivers, if someday you decide to optimize your kernel.

In case you want to keep Windows and resize its partition, the best solution is [SystemRescueCd](#). Follow the instructions for installing it on a USB stick from here http://www.sysresccd.org/Sysresccd-manual-en_How_to_install_SystemRescueCd_on_an_USB-stick. It is a good idea to archive the partitions of your Aspire, in case you decide to return it back to the current state someday; if you can afford the space to keep the images.

c. Remote access – XDMCP

Depending on how comfortable you feel with the small keyboard and monitor of Aspire, you may consider enabling XDMCP. Here is good guide of how to do it: <http://alien.slackbook.org/blog/running-x-window-on-ms-windows/>. If you have CygWin already installed you would not need to install X-Server, simply run “`xwin -query Aspire.IP.address`” from a CygWin terminal.

d. For consideration:

[FreeRadius http://freeradius.org/](http://freeradius.org/) - Formally RADIUS support is necessary only if you want the following capabilities: having WPA Enterprise authentication, being able to append more access points or authentication against external user data bases like LDAP or Novell eDirectory. It is also important to consider the choice between the standalone RADIUS server and the hostapd integrated RADIUS support. With so many choices, I thought it is a good idea to explain my arguments for choosing FreeRadius. First of all WEP in its 128 bit version is acceptable for home security, but it is so easy to configure that it takes away all the fun from the task. In its most basic configuration FreeRadius is really easy to install and configure, which means that obtaining WPA requires only a modest amount of effort. Configuring RADIUS is certainly not easy (it requires a lot of patience) and it may take days to set it up as a DAP gateway, but it is a five minutes work in the simplest scenario as in the example here with flat ASCII files. I suspect that using the hostapd integrated RADIUS allows for a lower CPU load than a separate RADIUS server and this has to be considered for small embedded routers.

FreeRadius though has low processing requirements so the dedicated Aspire can easily run it.

If you opt for FreeRadius, you can download the latest version from <http://freeradius.org/download.html>. I used the freeradius-server-2.1.11 version. The installation is as simple as typing the command **./configure, make, make install**. The following instructions can be used to configure RADIUS <http://wiki.freeradius.org/Basic-configuration-HOWTO>. If you used **./configure** without additional options the “users” file will be created in “**/usr/local/etc/raddb**”.

First, create some users, simply by appending at the end of the “user” file something like:

```
User1 Cleartext-Password := "password1"
```

Second, change the “**secret=12345:-)**” statement in the clients.conf. No further actions are needed since all communications in this configuration are going through the loopback address that is configured by default.

Third, copy the rc.radiusd script from the freeradius-server-2.1.11/scripts to /etc/rc.d/. Run the first tests of your newly installed RADIUS server with “**radiusd -X**”. Once you are satisfied, insert the line “**/etc/rc.d/rc.radiusd start**” in the “**/etc/rc.d/rc.local**” file. Create the file “**/etc/rc.d/rc.local_shutdown**”, make it executable and put the corresponding “**/etc/rc.d/rc.radiusd stop**” in it. From now on, if there are problems with RADIUS you will look in **/usr/local/var/log/radius**. There is also a lot of authentication related information in **/usr/local/var/log/radius/radacct/127.0.0.1**.

Finally, the default self-signed certificates generated during the installation in “**/usr/local/etc/raddb/certs**” are good, but if you want your self-signed certificates to show something different you can generate your own. All of the certificates are located in the RADIUS sub-directory “**certs**”.

e. **Necessary:**

MadWifi project

<http://madwifi-project.org/>

hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator

<http://hostap.epitest.fi/hostapd/>. I used hostapd-0.7.3 and madwifi-0.9.4-r4167-20110827.

First, I installed **MadWifi** since there is a remark about hostapd in README-WPS

mentioning that hostapd needs specifications about the location of the MadWifi libraries. All the instructions for installing MadWifi are found within the INSTALL file of the source. Look in the README file in the source for the necessary kernel configuration. Make the necessary changes in your kernel or simply use my kernel [.config](#). The only thing not mentioned that I thought necessary to do is to enter “make install” at the end, simply to be sure all is on place.

Second I installed **hostapd**. Here comes the most sophisticated part of the whole installation. Within the source directory (wherever you extracted it), there is a subdirectory hostapd. You need a `.config` file in order to compile the file. Copy the `defconfig` file to the `.config` file in the same directory and then edit the `.config` file. Its editing depends on the configuration that you want to build and types of authentication that you plan to support. There are many important options that need consideration when editing the `.config` file. For example, do you want RADIUS and if yes with what kind of support etc. These are the changes I made to my `.config`:

```
CONFIG_DRIVER_MADWIFI=y
CFLAGS += -I/tmp/2/madwifi-0.9.4-r4167-20110827 # change to the madwifi source directory
CONFIG_DRIVER_NL80211=y
CONFIG_WPS=y
CONFIG_WPS_UPNP=y
CONFIG_RADIUS_SERVER=y
CONFIG_IEEE80211R=y
CONFIG_DRIVER_RADIUS_ACL=y
CONFIG_IEEE80211N=y
```

And here is the link to get my `.config` which I called “[hostapd.config](#)” to avoid confusion with the kernel `.config`. Copy it in the hostapd subdirectory and rename it `.config` and then “make”, “make install”. Change the path to the MadWifi libraries depending on your installation location choice.

```
CFLAGS += -I/tmp/2/madwifi-0.9.4-r4167-20110827
```

I strongly recommend reading my file and also the README and README-WPS that are in the same directory even if you simply decide to use my configuration file. This will give you better understanding and may also spark ideas for interesting experiments. If you decide to dig deeper, check the dependencies between the variables in the Makefile “ifdef constructions”. Follow the “Matrix” movie’s advice and “Go to the source”.

Create the `/etc/hostapd/` directory and copy in it at least the `hostapd.conf` file (do not confuse it with my `hostapd.config` it is a copy of my `.config` for hostapd) from the source directory. This is the changes for configuration (a.):

```
#driver=madwifi

#ctrl_interface_group=0

#ssid=test

ssid=Acer_A1

hw_mode=g

channel=11

ieee8021x=1

eapol_key_index_workaround=1

nas_identifier=komsalov.homelinux.org

auth_server_addr=127.0.0.1

auth_server_port=1812

auth_server_shared_secret=12345:-)

acct_server_addr=127.0.0.1

acct_server_port=1813

acct_server_shared_secret=12345:-)

wpa=1

wpa_key_mgmt=WPA-EAP

wpa_pairwise=TKIP

wpa_group_rekey=300

wpa_gmk_rekey=640

bss=wlan0_0

ssid=Welcome
```

You may need to copy some other files and eventually create some if you decide to change the configuration and of course fix the path to them in `hostapd.conf`. Here is my [hosapd.conf](#) for the network configuration (a.). It can be used as it is, the only absolutely necessary change is to put your RADIUS secret.

```
auth_server_shared_secret=12345:-)
acct_server_shared_secret=12345:-)
```

At first run `hostapd` in the terminal, like that:

```
/usr/local/bin/hostapd -dd /etc/hostapd/hostapd.conf
```

You can either start directly with my file or with the example file from the source directory first. The example file will create one open network with `ssid=test`, giving you some confidence. It is a good idea to begin like this before setting up the DHCP server and eventually masquerading with the firewall. This will help you pinpoint the problems that need to be fixed. If you start two or more encrypted `ssid`'s, DHCP, DNS and the firewall at once, it will be harder to identify the source of the eventual problems. It will also be good to test the configuration with

any wireless client, but Windows, even a simple iPod would be better. Configuring Windows to work with RADIUS self-signed keys for WPA is a bit tricky and it is hard to pinpoint what gives you the problem, the client or the AP. There are two things you may consider here: to use CCMP instead of TKIP and to switch from WPA to WPA2. I decided to leave this decision for configuration (c.), because this is the one I will keep until I can afford to dedicate my Aspire to configuration (b.).

You can get the rc.hostapd from <http://slackbuilds.org/repository/13.0/network/hostapd/>, after you get bored looking on the hostapd in a terminal and running it manually. Put the rc.hostpd in the /etc/rc.d di directory, fix the paths in it, call it from /etc/rc.d/rc.local and stop it from rc.local_shutdown.

At this state your rc.local should look like this:

```
#!/bin/sh
#
# /etc/rc.d/rc.local: Local system initialization script.
#
# Put any local startup commands in here. Also, if you have
# anything that needs to be run at shutdown time you can
# make an /etc/rc.d/rc.local_shutdown script and put those
# commands in there.

/etc/rc.d/rc.radiusd start

/sbin/ifconfig wlan0 up
/sbin/iwconfig wlan0 channel auto

route add default gw 192.168.1.1

/sbin/ifconfig wlan0 192.168.11.1

/etc/rc.d/rc.hostapd start

/sbin/ifconfig wlan0_0 172.17.0.1

/usr/sbin/dhcpd wlan0 wlan0_0

#EOF
```

And your rc.local_shutdown:

```
#!/bin/sh
#

/etc/rc.d/rc.hostapd stop

/etc/rc.d/rc.radiusd start

#EOF
```

The “/sbin/ifconfig wlan0 up ; /sbin/iwconfig wlan0 channel auto” commands in rc.local should not be necessary, but if you do not give them you will get an error when hostapd sets the channel.

Here is my simple dhcpd.conf file:

```
authoritative;
ddns-update-style none;

default-lease-time 604800;
# 7 days 7*86400
```

```

max-lease-time 2592000;
# 30 days 30*86400

subnet 192.168.11.0 netmask 255.255.255.0 {
    range 192.168.11.10 192.168.11.100;
    range 192.168.11.150 192.168.11.200;

    option domain-name "mydomain.org";
    option broadcast-address 192.168.11.255;
    option routers 192.168.11.1;
    option domain-name-servers 192.168.11.1, 207.164.234.193, 207.164.234.129;

}

subnet 172.17.0.0 netmask 255.255.0.0 {
    range 172.17.0.10 172.17.255.250;

    option domain-name "mydomain.org";
    option broadcast-address 172.17.255.255;
    option routers 172.17.0.1;
    option domain-name-servers 172.17.0.1, 207.164.234.193, 207.164.234.129;

}

#log-facility local7;

```

I decided to have a caching DNS server on the Aspire; it is not mandatory, but it is necessary to put your DNS servers in the dhcpd.conf.

f. Optional programs:

Firewall builder by NetCitadel <http://www.fwbuilder.org/>

Having a firewall is not exactly an option, but you will have to do some NAT with iptables anyway. Of course you may do it manually but I strongly recommend Firewall Builder. It is from my point of view by far the best firewall management solution on the market and is free on linux. Here is a simple script [acerap.fw](#) generated with it for the configuration (a.), as an example. On Slackware, download the source and compile it and then run ldconfig after using “make install”.

Wireshark - <http://www.wireshark.org/>

Wireshark is unnecessary for the current configuration, but at some moment you certainly will want to know what is going on. As you are anyway in the process of downloading and compiling, install it to have it on hand when necessary. I recommend that you put at least “./configure --enable-threads” if no other option. It improves performance and the program remains stable.

5. Configuration (b.) - VLAN's and switches

For this configuration I used a Cisco Catalyst 2900 XL switch. I am on Bell Sympatico ADSL with SpeedStream 5360 Ethernet ADSL modem, which is actually only a bridge. It turned out that it does not matter how I configured the port of the Cisco Catalyst, it did not detect the SpeedStream. Finally, I gave up and used one small 5 port TrendNet TE100-SS/CA switch in between them. Since all SpeedStream 5360's are gone nowadays, you probably will not have this problem. DSL modems nowadays are actually routers and have integrated PPPoE support and for this configuration it is only necessary to VLAN the switch and eth0. I used a Cisco Catalyst (which is actually not so bad) only because this is what I managed to borrow, but if you are thinking of buying a switch look for something better.

I configured two additional VLAN's on it:

VLAN Name	Status	Ports
1 default	active	Fa0/2, Fa0/3, Fa0/4, Fa0/5, Fa0/6, Fa0/7, Fa0/8, Fa0/13, Fa0/14, Fa0/15, Fa0/16, Fa0/20, Fa0/21, Fa0/22, Fa0/23, Fa0/24
2 VLAN0002	active	Fa0/9, Fa0/10, Fa0/11, Fa0/12
3 VLAN0003	active	Fa0/17, Fa0/18, Fa0/19

Port FA01 is a tagged port. I am not giving the detailed commands to set it up, since they will depend on whatever switch model you have.

From the side of Linux it is really easy; type the commands:

```
ifconfig eth0 0.0.0.0
```

(to remove the IP address from eth0)

```
ip link add link eth0 name eth0.1 type vlan id 1
```

```
ip link add link eth0 name eth0.2 type vlan id 2
```

```
ifconfig eth0.1 up
```

```
ifconfig eth0.2 up
```

Of course in “vlan id NN” the NN will have to be replaced with your VlanID. In my case the new IP addresses are set back like this:

```
ifconfig eth0.1 0.0.0.0
```

```
ifconfig eth0.2 192.168.1.55 netmask 255.255.255.0
```

If you like you can go with something more traditional like 192.168.1.1 for your future default gateway. I used eth0.1 as the uplink. If you want your physical wireless and wired networks to be in the same network, to mimic the behavior of the commercial routers, you can bridge eth0.2 and wlan0. Check configuration (c.) below for help with bridging. The only real reason you may want this is to use Microsoft workgroup network, though in this case you should consider installing Samba as a master browser on the Aspire.

In my case I had to setup PPPoE by running the pppoe-setup script. This will not be necessary for most people, but if it is for you than pay attention to the last question (asked by the pppoe-setup script) and answer it depending on the firewall management you choose. You may encounter additional MTU auto discovery problems with Internet providers such as Bell Sympatico. If it turns out that you are able to ping external machines, but browsing barely works if at all, you will have to use some commands like the next one in your firewall script:

```
iptables -A FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

In case you decide to use the Firewall Builder, it is only a matter of checking the check box “Clamp MSS to MTU” in firewall settings. If you want to know more about this problem check “[Linux Advanced Routing & Traffic Control HOWTO](#)”.

I implemented the configuration with a manageable switch rather than the one with the USB to Ethernet converter, since to start with I have no such device. The second problem with such devices is actually making them work. Finally I have difficulties believing the advertised speeds of all USB to Ethernet converters that the manufacturers claim.

6. Configuration (c.) - bridging

I chose to stay with this configuration for now, since it allows me to pull my Aspire out of the network from time to time without loosing Internet connectivity. When travelling, I use my Aspire as a GPS device in combination with a USB connected satellite antenna.

These are the commands given in the necessary order placed inside the `/etc/rc.d/rc.local` file:

```
#!/bin/sh
#
# /etc/rc.d/rc.local: Local system initialization script.
#
# Put any local startup commands in here. Also, if you have
# anything that needs to be run at shutdown time you can
# make an /etc/rc.d/rc.local_shutdown script and put those
# commands in there.

/etc/rc.d/rc.radiusd start

/sbin/ifconfig wlan0 up
/sbin/iwconfig wlan0 channel auto

/etc/rc.d/rc.hostapd start

/sbin/ifconfig wlan0_0 172.17.0.1

/sbin/ifconfig eth0 up
/sbin/ifconfig wlan0 up

/usr/sbin/brctl addbr br0
/sbin/ifconfig br0 up
/usr/sbin/brctl addif br0 eth0
/usr/sbin/brctl addif br0 wlan0

/sbin/ifconfig br0 192.168.1.55

/sbin/route add default gw 192.168.1.1

/usr/sbin/dhcpd wlan0_0

/etc/rc.d/firewall/acerap_br.fw
/etc/rc.d/rc.traffic_shaping start

/etc/rc.d/rc.bind restart

#EOF
```

The part that concerns bridging is in bold. Bridging on Linux is really easy and it should not cause you any troubles. The spanning tree should be off as it is by default. Turn it on only if you really know what you are doing.

The `dhcpd` is bound only to the `wlan0` to serve `172.17.0.0/16` addresses to the Welcome network. The network with the Acer_A1 ssid is getting its IP addresses from the “Linksys SRX 200” DHCP server through the bridge (it transfers broadcasts transparently).

The `rc.traffic_shaping` script is for traffic shaping which turned out to be necessary, because some of the clients in Welcome misbehaved (see 6. Additional administrative tasks).

Of course you will need a firewall as well, so here is the `acerap_br.fwb` file created with the FWbuilder project and the script `acerap_br.fw` it generated, really basic, but a good starting point.

I decided to switch to WPA2 after using this configuration for about one month. This required only a change of `wpa=1` in `/etc/hostapd.conf` to `wpa=2` and a restart of the `hostapd`. I was worried about the amount of work necessary to reconfigure all clients, but it turned out that only some small changes to the Windows clients are required.

7. Clients setup – WPA and WPA2 with self-signed certificates.

a. Linux.

Slackware comes with Wicd in /Slackware/slackware-current/extra/wicd directory and it works fine, so simply install it. Most other collections seem to be using NetworkManager, but anyway there are no problems.

b. Mac OS X 10.7.2

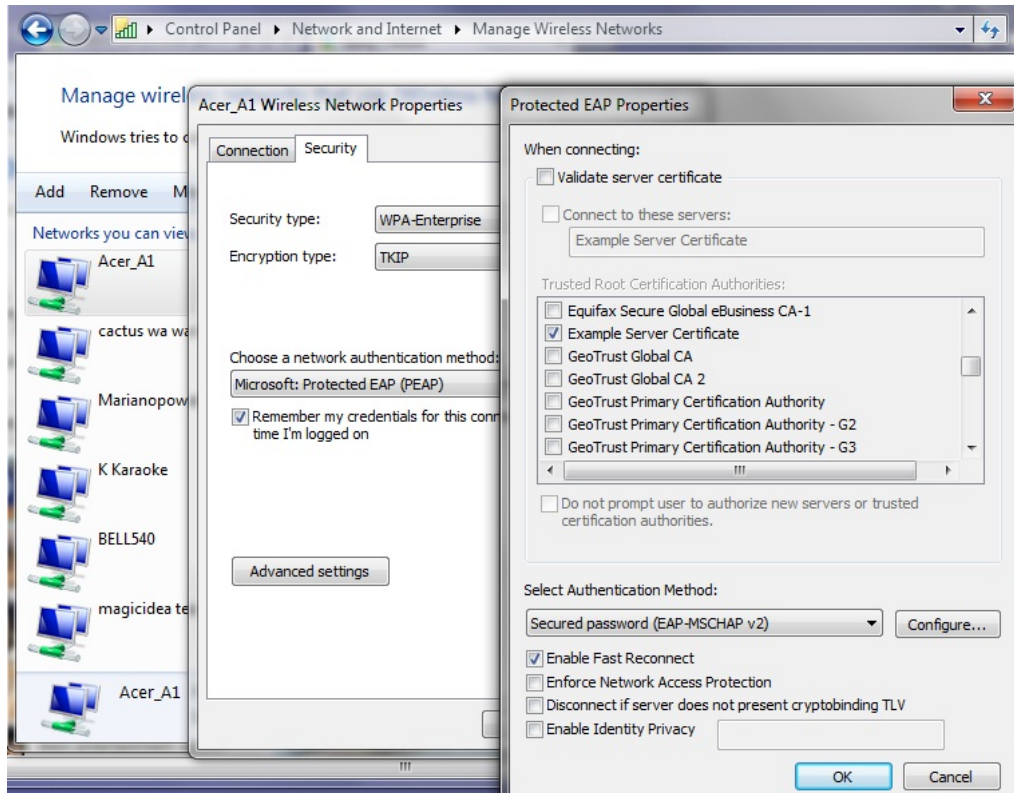
When I first tried to connect it showed a message stating that the certificate is not from a known authority and offered me a check box to accept it permanently, then it asked for the username and password and worked fine after. There is a key management program in Linux' utilities called Key Chain Access. Here I marked the certificate as trusted and it became green. I am not sure if this was necessary, but I wanted to be on the safe side. I found an instruction that recommended installing manually and in advance the certificate, but it turned out that Mac does it for you. When you decide to use WPA2 there will be no need to even touch a Mac. It detects the change in the access point and reacts accordingly by readjusting its settings and even reusing the username and password from the previous configuration.

c. Itouch.

I asked my son to do it, since I only have second hand experience with those devices. Besides I did not want to look for my glasses. It behaved the same way as the Mac, it asked me to accept the certificate and then asked me for the username and password.

d. Windows 7

Do not expect to get a question about the certificate at this point. You will have to install the certificate in advance by yourself. Go to `/usr/local/etc/raddb/certs/` to get it, I used a USB stick. The instructions, I found online, are to create the connection manually and install the certificate as soon as the network connection setuping wizard reaches the point of asking you for them. This did not help and Windows continued to complain about the certificate. I installed it through the Internet options section within the Control Panel, but this did not help either. I played with the check boxes "the best Windows approach isn't it?", and did some googling. Suddenly it worked and when I tried to connect asked me to accept the certificate (the same one which I installed and marked in advance as trusted in all the possible places!!!) This is the picture with the certificate's options.



As if this is not enough you need to do the same for all the users on each Windows machine, even with the same laptop the procedure for each user will be similar but unique.

The only advice I have is to be persistent and it will work. If you google the problem you will find that some people simply got around the problem by buying certificates obviously it comes cheaper for big number of laptops, but maybe quitting Windows is better.

If you decide to use WPA2 Windows 7 will work fine, but it will not detect the change automatically. The properties change shown on the the picture above will work. The encryption should remain TKIP. Windows 7 will ask for the username and password and then it should work fine.

8. Additional administrative tasks you may consider necessary.

a. Limit bad clients - bit torrent

It did not take more than a couple of days for around thirty of my neighbors to start using the open "Welcome" network. Most of them turned to being modest doing mostly mail and some surfing, but two or three bittorrent fans turned out to be a problem. If you decide to provide some Internet for your neighbors you certainly should do something about this problem.

You have at least two options: [17-filter](http://17-filter.clearfoundation.com/) from <http://17-filter.clearfoundation.com/> and [ipp2p](http://www.ipp2p.org/) from <http://www.ipp2p.org/>. During one time or another I used both of them and the results are relatively similar. I still prefer ipp2p as I believe it is less CPU consuming. The project web page claims that the project is discontinued. This is not exactly true, it is only discontinued as a separate project for the external module. It was moved to patch-o-matic which is today defunct. Nowadays after netfilter.org discontinued patch-o-matic, it was moved to

xtables-addons and netfilter.org is still support it.

First, do not forget to install libmnl from <http://www.netfilter.org/projects/xtables-addons/index.html> and then the xtables-addons.

Then you will need something like this:

```
SIPTABLES -N Bittorrent
SIPTABLES -t mangle -N Bittorrent
SIPTABLES -t mangle -A PREROUTING -s 172.17.0.0/16 -m ipp2p --bit -j Bittorrent
SIPTABLES -A INPUT -s 172.17.0.0/16 -m ipp2p --bit -j Bittorrent
SIPTABLES -A OUTPUT -s 172.17.0.0/16 -m ipp2p --bit -j Bittorrent
SIPTABLES -A FORWARD -s 172.17.0.0/16 -m ipp2p --bit -j Bittorrent
SIPTABLES -A Bittorrent -j LOG --log-level info --log-prefix "Bittorrent "
SIPTABLES -t mangle -A Bittorrent -j LOG --log-level info --log-prefix "Bittorrent m "
SIPTABLES -A Bittorrent -j DROP
SIPTABLES -t mangle -A Bittorrent -j DROP
```

Most of these commands are unnecessary, just doing:

```
iptables -t mangle -A PREROUTING -s 172.17.0.0/16 -m ipp2p --bit -j DROP
```

will do most of the job. You simply put it before the line

```
iptables -A XXXXX -m state --state ESTABLISHED,RELATED -j ACCEPT
```

to get it working.

Anyway, do not expect too much from it, or from I7-filter for that matter. They will slow down bittorrent clients significantly, but both have problems recognizing encrypted connections. At least the bittorrent clients for sure will not be able to kill anymore all other connections. If you are not satisfied with the results of the solution just described you should combine it with traffic shaping (next paragraph).

b. Traffic shaping

The decision to spend time to setup and fine tuning traffic shaping depends on: the type of Internet connection used, the number of clients you have, their behavior and most important, will you provide some Internet for your neighbors.

If you have a relatively fast and symmetric connection you have nothing to worry about, but if you are on something like ADSL and your provider has an illicit behavior than moving the queue to your machine makes a real difference. You can read about the reasons for getting control over your queue here "[The Ultimate Traffic Conditioner](#)".

It is important to mention that since "[Linux Advanced Routing & Traffic Control HOWTO](#)" was written, lots of things have changed, though probably the most important new thing in the field of traffic shaping is the "[Intermediate Functional Block device](#)". A lot of work has been done in the field and you have to be really careful when you are doing your own research since many of the online documentations and examples are outdated. Most examples will still work fine, but often better solutions have been developed.

My traffic shaping script had the following goals:

- a. Move the queue to my machine.
- b. Provide fairness between both my family clients and guests in the "Welcome" network.
- c. Give a warranted advantage to my own clients, leaving the clients in "Welcome" with what is left, while at the same time warranting some bandwidth for Welcome even in moments of heavy load. My Internet connection is actually 99% unused anyway, but I did not want to listen to complaints from my family.
- d. Have a method to separate clients that misbehave from the crowd.

Here is the resulting script [rc.traffic_shaping](#). It does what I wanted it to, but is certainly not perfect and will require additional fine-tuning. Anyway you will have to readjust it to your conditions.

One important thing that needs to be mentioned is that limiting the outgoing traffic from a specific source, does not lead to proportional limitation to the incoming traffic. Most streaming protocols require small amounts of outgoing requests in order to get real floods of incoming video. As a result even class 1:13 (this is where baddies go), can seem too restrictive with its “rate 10kbit burst 15kbit”, but it actually gives them around 600kbits of download speed. This demonstrates that in order to have precise control you need to shape incoming connections as well.

Next is the chart of outgoing traffic shaping.

The traffic goes as follow:

From Acer_A1 ->1:11

From Welcome -> 1:12

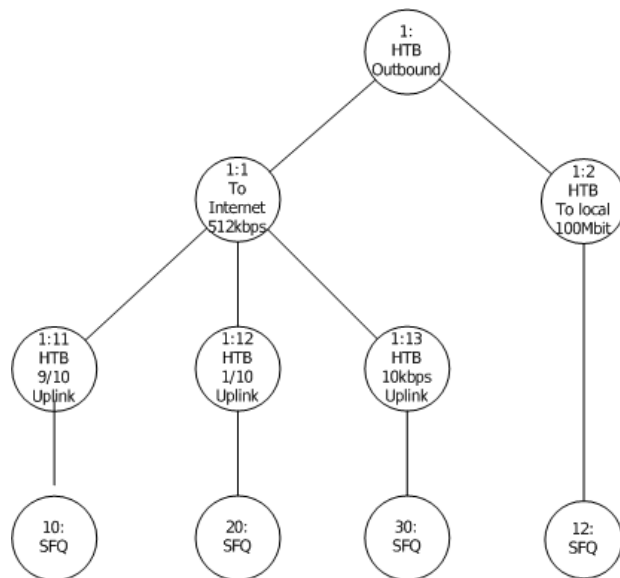
Bad clients -> 1:13

Between may cable clients and Acer_A1 -> 1:2

For example the traffic is classified by iptables with rules like this:

```
iptables -t mangle -A POSTROUTING -s 192.168.1.0/24 -d 192.168.1.0/24 -j CLASSIFY --set-class 1:2
```

You can see how I set the classes in “Policy: Traffic_Control” in [acer_br.pdf](#) or check the detailed syntax inside the [acerap_br.fw](#) script.



The next picture represents the chart of incoming traffic shaping.

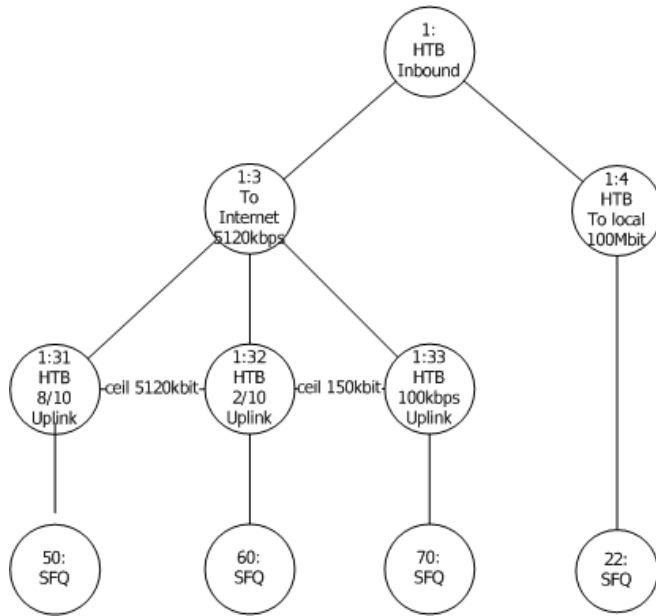
The traffic goes as follow:

To Acer_A1 ->1:31

From Welcome -> 1:32

Bad clients -> 1:33 – nobody is there yet, but it is ready:-).

Between may cable clients and Acer_A1 -> 1:4



The first step in shaping the outgoing traffic is to get the `ifb0` “Intermediate Functional Block device” working. It turned out that the module does not load automatically, but I rather loaded it in the `rc.traffic_shaping` script by:

```
/sbin/modprobe ifb
ifconfig ifb0 up
```

The next problem is really interesting look at the part of the `rc.traffic_shaping` script pasted below:

```
#####
# It is necessary to mirror both eth0 and br0 to ifb0 in order to have both traffics
# with destinations 172.17.0.0/16 and 192.168.1.0/24,
# because each of them sees only one destination as outgoing.
# You may check it by remarking one of the mirrors and the running WireShark on ifb0.

tc filter add dev $DEV parent ffff: protocol ip prio 10 u32 \
  match ip dst 0.0.0.0/0 flowid 1: \
  action mirred egress redirect dev ifb0

tc filter add dev br0 parent ffff: protocol ip prio 10 u32 \
  match ip dst 0.0.0.0/0 flowid 1: \
  action mirred egress redirect dev ifb0
#####
```

The `$DEV=eth0` is set at the beginning of the script.

There is probably a better way of directing traffic to `ifb0`, but this is the only way that works for me.

You will need the following commands, to investigate and adjust the script to your own needs:

```
tc class ls dev eth0
```

```
tc class ls dev ifb0
```

```
tc -s -d qdisc show dev eth0
```

```
tc -s -d qdisc show dev ifb0
```

```
tc -s class show dev eth0
```

c. Cache DNS server.

Having a cache DNS server was a great advantage in the time when everyone thought that a 28'800 modem is lightning fast. With today's speed the percentage of economized bandwidth is close to zero, but it is so easy to install, and besides old habits die hard. Just make `/etc/rc.d/rc.bind` executable. Slackware has a `/etc/named.conf` pre-ready. It is a good idea to setup regular updates of `named.root` by simply creating the script `/etc/cron.monthly/named.root` and putting the following two commands in it:

```
#!/bin/sh
#
/usr/bin/wget --user=ftp --password=ftp http://www.internic.net/zones/named.root -O /var/named/
caching-example/named.root
/etc/rc.d/rc.bind restart
```

d. Log configuration.

The `dhcpd` log can be moved to separate files by three simple steps:

Putting the next line at the end of the `dhcpd.conf`

```
log-facility local7;
```

Append at the end of `/etc/syslog.conf` the line

```
local7.* -/var/log/dhcpd.log
```

Create an empty `dhcpd.log` by:

```
:> /var/log/dhcpd.log
```

Of course `dhcpd` and `syslogd` need to be restarted.

iptables log. It is a tempting idea to move the `iptables` log in a separate file if you use Firewall Builder or just enjoy having extensive logs from your firewall. The complication here comes from the limited choice of “`--log-level X`” available. As a result, the kernel (and not the `iptables`) is in reality doing all the filtering thus all logs go in log facility “`kern.*`”. The choice for `*` is limited between those levels “**0 emerg, 1 alert, 2 crit, 3 err, 4 warning, 5 notice, 6 info, 7 debug**”. Besides “`crit`” is the default level for `klogd` to send messages to the console so whatever goes on this level inevitably goes on the console as well. You may experiment with other levels or try changing “`klogd -c 3`” to something else.

Everything else is simple after these difficult choices are made.

First either change the log level setting in Firewall Builder, or if you wrote your own script set it to something like “`-j LOG --log-level warn --log-prefix “my log text”`”.

After this is done, append at the end of `/etc/syslog.conf` the line:

```
kern.=warn -/var/log/fwbuilder.log
```

and exclude it from

```
*.warn;kern.!=warn;\
```

```
authpriv.none;cron.none;mail.none;news.none -/var/log/syslog
```

If you decide to experiment with other levels, for example “`notice`”, change the line like this:

```
kern.=notice /var/log/fwbuilder.log
```

But in the case of “`notice`” you will also have to exclude “`kern.notice`” from `/var/log/messages` by

editing the related line in syslog.conf in a similar way yielding the line:

```
*.info;*.!warn;kern.!=notice;\n      authpriv.none;cron.none;mail.none;news.none    -/var/log/messages
```

There is no perfect choice and some of your boot messages will always go to fwbuilder.log instead of going in to the messages or syslog files. The biggest problem are the eventual error messages generated during the normal course of work, which will be buried in the fwbuilder.log.

If you want see what else goes on in the /var/log/fwbuilder.log and iptables logs, the next command will help you:

```
cat /var/log/fwbuilder.log |grep RULE -v
```

9. Some final words.

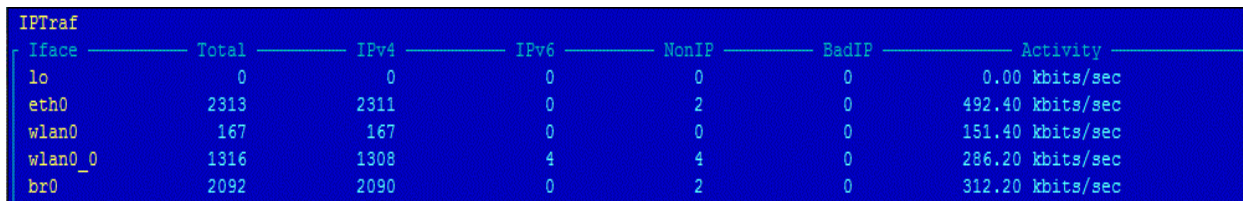
Over a month passed, since I started writing this document, while creating the access point took me only three days.

During the time being the AP did not drop one connection, while providing coverage over almost a 100 meter diameter. No timeout error messages occurred, which was so common for the “Linksys SRX 200” router.

Over 30 of my neighbors started using it more or less heavily and some really heavily.

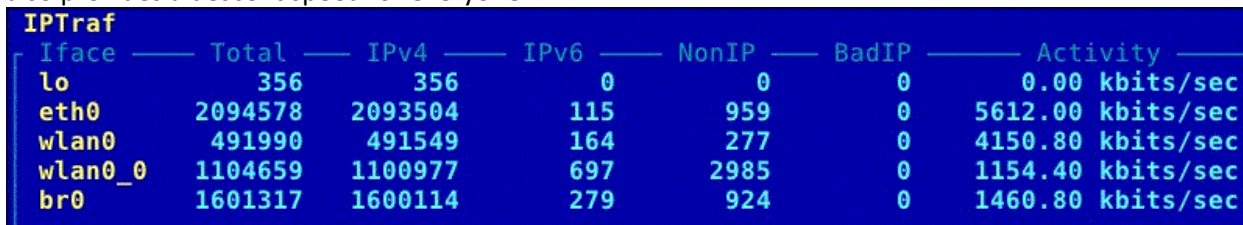
For example, while writing this:

```
root@acer:/var/log# cat /proc/net/ip_conntrack |grep tcp -c\n565\nroot@acer:/var/log# cat /proc/net/ip_conntrack |grep udp -c\n76
```



IPTraf							
IFace	Total	IPv4	IPv6	NonIP	BadIP	Activity	
lo	0	0	0	0	0	0.00 kbits/sec	
eth0	2313	2311	0	2	0	492.40 kbits/sec	
wlan0	167	167	0	0	0	151.40 kbits/sec	
wlan0_0	1316	1308	4	4	0	286.20 kbits/sec	
br0	2092	2090	0	2	0	312.20 kbits/sec	

And even in a moment of heavy load, like below, not only does the network remain stable, but it also provides a descent speed for everyone.



IPTraf							
IFace	Total	IPv4	IPv6	NonIP	BadIP	Activity	
lo	356	356	0	0	0	0.00 kbits/sec	
eth0	2094578	2093504	115	959	0	5612.00 kbits/sec	
wlan0	491990	491549	164	277	0	4150.80 kbits/sec	
wlan0_0	1104659	1100977	697	2985	0	1154.40 kbits/sec	
br0	1601317	1600114	279	924	0	1460.80 kbits/sec	

Now I have comprehensive log files like:

Nov 21 21:29:49 acer kernel: [420579.216945] RULE 3 -- CONTINUE IN=wlan0_0 OUT=br0 SRC=172.17.128.154 DST=173.194.31.138
LEN=40 TOS=0x00 PREC=0x00 TTL=63 ID=58669 DF PROTO=TCP SPT=49604 DPT=80 WINDOW=0 RES=0x00 RST URGP=0

All in all I am satisfied with the outcome. It was worth the effort, and the result surpassed the best of all my expectations.

Copyright

Copyright (C) 2011 Krastyo Komsalov.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available from <http://www.fsf.org/licenses/fdl.html>.