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

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. Accidently, I happened to have some free time, one nearly three year old Aspire in my hands and 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 the router obviously has troubles with this. What I wanted is a wireless router over which I 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 have Free RADIUS, not only because I wanted a support for WPA and be able to append eventual access points with roaming, but also to have extensibility to any user data base, from local flat files to LDAP. The hostapd has also its own integrated RADIUS, but the freedom of having FreeRADIUS had been so tempting; beside the setup with flat ASCII users file is really easy. In this configuration RADIUS is configured to use files.

b. Ipv6 and DNSSEC are here to stay and no embedded router will give what I have with Linux; I am simply interested in this field. Ipv6 is not part of this configuration, but the freedom is there if you want it.

c. I wanted to have not only a standard firewall, but the full iptables power; a simple thing like SSH tunnels for my children allowing home access from school are tricky with my router and traffic shaping is simply not there. For this reason the Firewall Builder is included in this configuration with basic rule set. I think it is as far the best firewall management solution on the market and it is free for Linux.

d. I wanted to have at least two wireless networks "different ssid", to open safely one of them and share some of my bandwidth. This I hope will make me feel not so ripped-off next time when 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 anyway.

f. There are no requirements or specific instructions to a certain 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.

2. Hardware description

My Acer Aspire One has a Model KAV10, which is one of Acer's first over 3 year old models. Since then Acer produced many new models, but the only thing that is important is which wireless adapter model is installed in it. From what I found Acer had been changing the

adapter in nearly all models. All models I checked are with different adapter from Atheros but I cannot give warranty for all. If you are thinking of buying one, check in advance what is actually in it. 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 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 turn out that your adapter is different and you need to do some additional investigation in order to be sure it supports AP mode, you will need "iw". Probably you already have it, but for the source or some documentation check here:
http://linuxwireless.org/en/users/Documentation/iw

The most informative command is:

iw list

It will give you a pretty long output. Look in it for the part that looks similar to this:

Supported interface modes:
*** IBSS**
*** managed**
*** AP**
*** AP/VLAN**
*** monitor**
*** mesh point**
Supported commands:

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

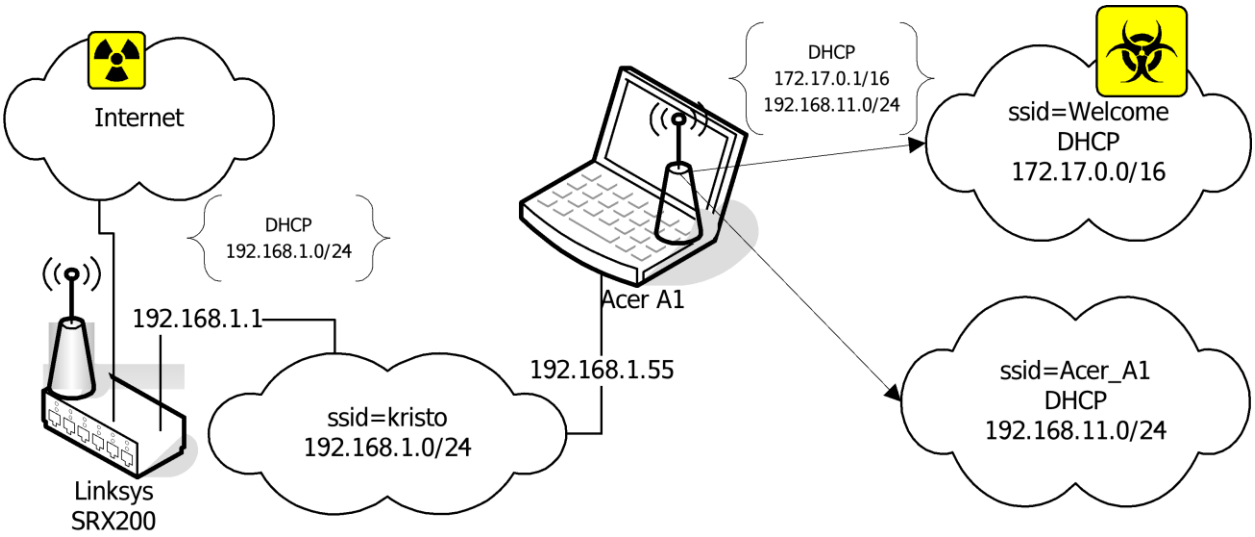
For the list of supported adapters, if it turns that your chipset is different you may check its status on the MadWifi website. MadWifi website is also as far the best source of documentation I found. When you decide to adjust to your needs, experiment or simply improve the configuration given below, this will be one of your primary sources of knowledge.

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

The model of "Linksys SRX 200" is not important. You may use any wireless router if you have any or avoid using it at all if you decide to dedicate permanently the Aspire as Wireless router.

3. Some possible network configurations:

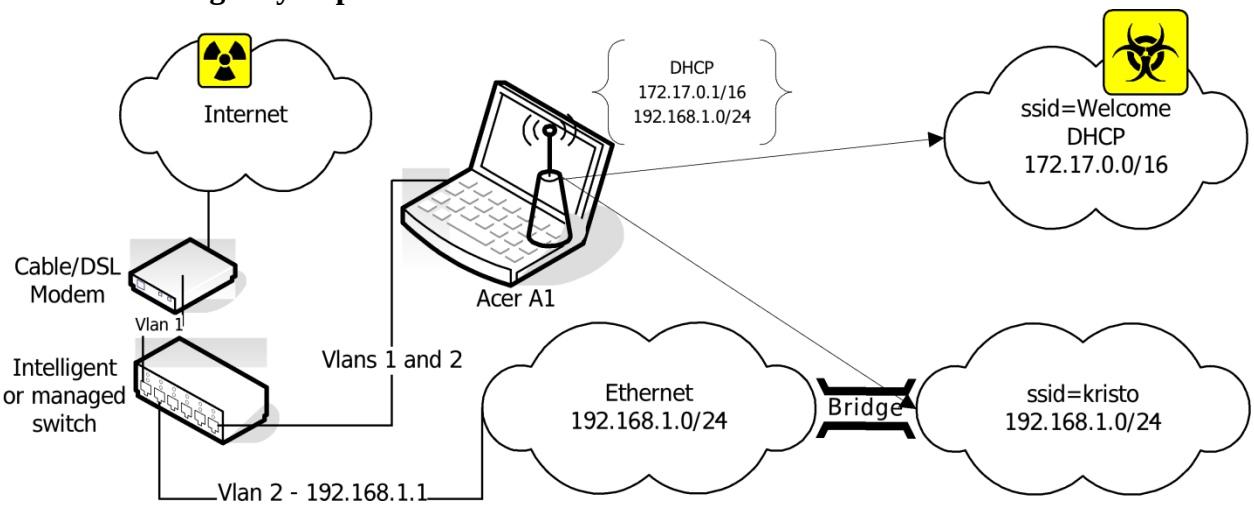
a. Keep old router and append Aspire inside, providing two additional wireless networks.



In this configuration the Ethernet port of the Aspire is connected directly to "SRX 200". This solves the problem that Aspire has only one Ethernet port; there has to be two Ethernet ports, one for the link to the internet the other for internal switch to provide internet to Ethernet connected computers. The two wireless networks are NAT'ed to 192.168.1.55 IP address. The reason is not only to put ssid "Welcome" in separate network to simplify firewalling but to resolve some NAT and routing problems. First the devices in 192.168.1.0/24 have to have the route to 192.168.11.0/24. I had no problem with Linux and Solaris but my network printer simply has no such thing as routing table in its web interface. Second appending the route in "SRX 200" is not a problem, but it refuses to NAT any other network then 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. 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 be easily reverted to any other.

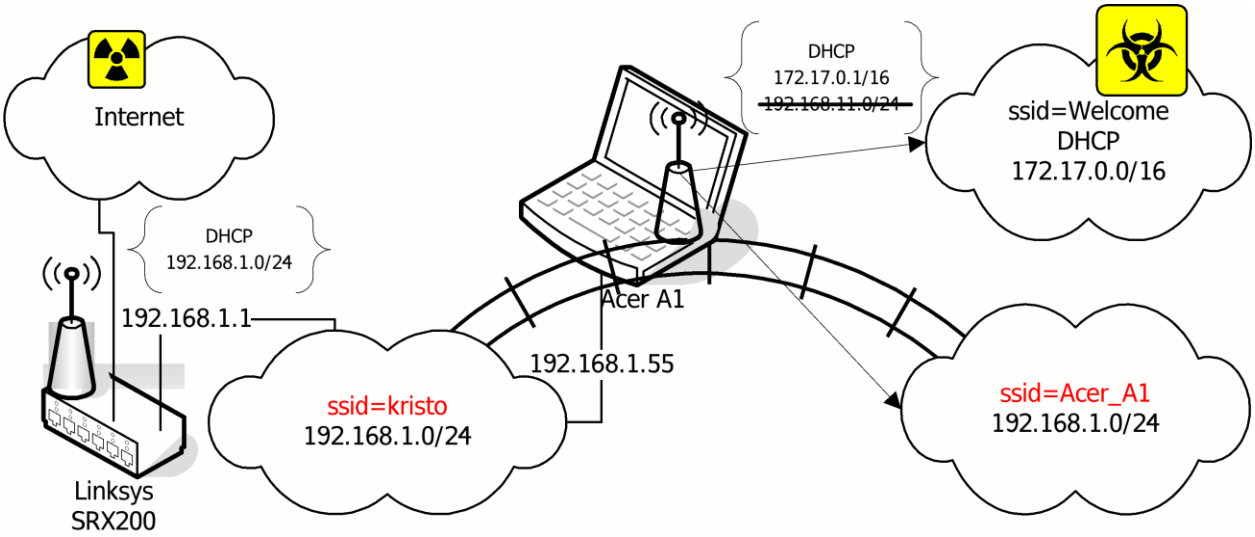
b. Using only Aspire as AP.



This configuration is setting you free; of course, there is the price. As the Aspire have only one Ethernet adapter you have to append second. There are two solutions. The first one "shown on the picture" is to use intelligent or managed switch and VLAN the eth0. The second one is to use USB to Ethernet adapter, to convert one of USB ports to Ethernet. The drawback of the solution with switch is that it is much more expensive, but it has the advantage of speed, stability and simplicity. The USB to Ethernet adapter is much cheaper, but it comes with questions about support of the adapter on Linux, speed and reliability.

There is one more small detail to mention: Depending what kind of link to internet you have, there will be different setup for the uplink adapter. If you are on the cable then it has simply to be on DHCP. In case of ADSL "like me" you will have to configure PPPoE. On Slackware you simply have to run pppoe-setup script.

c. Bridging between the two private networks and NATing only "Welcome" public network.



In this configuration the interfaces eth0 and wlan0 ere bridged. The network 192.168.1.0/24 can be accessed both through "kristo" and "Acer_A1" ssid. The DHCP server on the Aspire is bind only to the wlan0_0 interface. NAT to 192.168.1.55 is done only 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". There will have been other solution if the DHCP server on "SRX 200" had been more manageable; instead of bridging 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: First it avoids both routing problems of the solution (a.), and consequent NAT'ing of the "Acer_A1". Second it gives the ability to turn the Aspire off and still provide some networking with "SRX 200". If you can't afford to dedicate your Aspire as AP, this is the best configuration. You will have stable network when you do not need the Aspire and you can have the Aspire back when you need it.

4. Configuration (a.) - with installation instructions for all necessary software for any configuration. It is actually the initial configuration.

All programs I installed from source in /usr/local. Some configuration files I left in /usr/local/etc and some moved in /etc/. There is Slackware packages on SlackBuilds.org or you may make your own if you decide it is worth the effort, but installation from source is the easiest.

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 make a mirror; simply get the script "mirror-slackware-current.sh" from Alien Pastures and run it. It will put the mirror by default in /home/ftp which is exactly where you need it for the last step.

Second put the 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 it. Here is how to do it: http://www.slackbook.org/html/installation.html it is very easy. Use network cable to connect "makes things easier", and 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 with recompiling your kernel. Click here on my .config file and download it in /usr/src/linux. This is not well optimized version, only the processor is set to Intel Atom and some obviously unnecessary stuff is removed. I did not want to put here a version that is too customized to my needs. Using –j N option helps in making bzImage and modules. Seems –j 8 is around the optimum, but on first compilation you will not have this advantage. Anyway it takes forever to compile. It is important not to forget to reinstall 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 seems 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 how to do it http://alien.slackbook.org/blog/running-x-window-on-ms-windows/. If you have CygWin already installed you may not worry for X-Server, but simply run "xwin –query Aspire.IP.address" from CygWin terminal.

d. For consideration:

FreeRadius http://freeradius.org/ - Formally RADIUS support is necessary only if you want to have WPA Enterprise authentication, append more access points and to have authentication against external user data bases like LDAP or Novell eDirectory. Beside, you have the choice between standalone RADIUS server and the integrated in hostapd RADIUS support. With so many choices I thought it is good idea to give my arguments for choosing FreeRadius. First WEP in its 128 bit version is not so bad for security for a home, but it is so easy to configure that it takes the whole fun from the task. In its basic configuration FreeRadius is extremely easy to install and configure, which means WPA does not require so much effort. Certainly it is not easy "requires a lot of patience", and it may take days to setup RADIUS as a DAP gateway, but in the most simple scenario as in the example here with flat ASCII files, it is five minutes work. Using the integrated in hostapd RADIUS, probably has a lower CPU load then a separate RADIUS server and it has to be considered for small embedded router, but FreeRadius has so little requirements that even on Aspire the additional load is barely detectible.

If you opt for FreeRadius, download the latest version from http://freeradius.org/download.html. I used version freeradius-server-2.1.11. The installation is as simple as **./configure, make, make install**. For the configuration here you have to do only what is described in the Basic Configuration http://wiki.freeradius.org/Basic-configuration-HOWTO. The only difference if you did ./configure without customizing is that "users" file is in "**/usr/local/etc/raddb"**.

First, create some users, simply by appending at the end of "user" file something like –

User1 Cleartext-Password := "password1"

Second, change the "secret" statement in the clients.conf; all communication in this configuration is going through looback address, which is configured by default.

Third, copy rc.radiusd script from the freeradius-server-2.1.11/scripts to /etc/rc.d/. During the first tests of your newly installed RADIUS server run it with "**radiusd –X**", but 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, if you have problems with RADIUS you will look in **/usr/local/var/log/radius**. Also in **/usr/local/var/log/radius/radacct/127.0.0.1** you will find a lot of information related to the authentication.

Fourth - optional: 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 then you may generate yours. All certificates are in the RADIUS subdirectory "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** as there is a remark about hostapd in README-WPS, that hostapd needs to know where the libraries of MadWifi are. All instructions for installing MadWifi are in the INSTALL file in the source. Look in README file in the source for the necessary kernel configuration. Make the eventual necessary changes in your kernel or simply use my kernel .config. The only thing that is not mentioned but I thought necessary is to do "make install" at the end, simply to be sure all is on place.

Second I installed **hostapd**. Here is the most sophisticated part of all installation. In the source directory "whatever place you extracted it", there is a subdirectory hostapd. In order to compile the file you need a .config file. You may copy the file defconfig to .config in same directory and then edit it. The changes you make in it depend from the configuration that you want to build and types of authentication that you plan to support. Here is the place that you have settings for example about what kind of RADIUS support you want, and many more. These are the changes in 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 the confusion with kernel .config. It is with the settings for the configuration described here. You may simply copy it in hostapd subdirectory and rename it .config; then make, make install. You will have to change the path to MadWifi libraries:

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

to wherever you installed them in previous steps. Even if you decide to go simply by using my configuration file, I strongly recommend you to read the file and also README and README-WPS that are in the same directory. This will not only give you better understanding but also may spark ideas for interesting experiments. If you decide to dig deeper, check the dependences between the variables in the Makefile "ifdef constructions". Follow the advice of the "Matrix" movie "Go to the source".

Create the directory /etc/hostapd/ and copy in it at least hostapd.conf from the source directory "do not confuse it with my hostapd.config it is copy of my .config for hostapd". 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 have 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 network configuration (a.) You may use 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 terminal, like this:

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

You may start directly with my file or simply at first with the example file from source. The example file will create one open network with ssid=test, and give you some confidence that you are doing well. It is a good idea to begin with this until setting up the DHCP server and eventual masquerading firewall. This will help you to pinpoint where are the problems that have to be fixed. If you start two or more encrypted ssid, DHCP, DNS and firewall at once it will be more difficult to identify the source of the eventual problems. It will be good also if you have for the test wireless client something different from Windows; even a simple iPod is better. Making windows work with self-signed keys from RADIUS for WPA is a bit tricky and it is difficult to pinpoint what gives you a 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 preferred to leave this for configuration (c.), because this is the configuration I will keep until I can afford to dedicate my Aspire to configuration (b.)”.

After you get bored looking on the hostapd in a terminal and running it manually, you may get the rc.hostapd from http://slackbuilds.org/repository/13.0/network/hostapd/, put it in /etc/rc.d, 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, something 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 commands "**/sbin/ifconfig wlan0 up ; /sbin/iwconfig wlan0 channel auto**" in rc.local should not be necessary, but if you do not give them you will have 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 necessary, 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 and you will have to do some NAT with iptables anyway. Of course you may do it manually but I strongly recommend you the Firewall Builder. From my point of view it is as far the best firewall management solution on the market "it is free for linux". It is my hobby to optimize firewalls and surely given enough time I will write a more optimized script "humans always do better than the computer :-)", but I will never find the time to write a better structured and easy human readable script. Here is e script acerap.fw generated with it for the configuration a., as an example of the quality. On Slackware, simply download the source and compile it; run ldconfig after "make install".

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

It is not necessary 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. The only recommendation put at least "./configure --enable-threads" if no other option; it improves the performance and it will remain 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 on the Cisco Catalyst it did not want to detect the SpeedStream. At the end I gave up and used one small 5 port TrendNet TE100-SS/CA switch in between them. As SpeedStream 5360 are nearly despaired nowadays, you probably will not have such a problem. Most of the now a days DSL modems are actually routers and have integrated PPPoE support and for this configuration it is only necessary to VLAN the switch and eth0. I used Cisco Catalyst, only because this is what I managed to borrow which is actually not so bad, but if you are thinking of buying one, 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 setup it as they will depend of whatever model switch is available.

From the Linux side it is really easy; simply give the commands:

```
ifconfig eth0 0.0.0.0
```

to remove the IP address from eth0, then:

```
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 "vlan id NN" will have to be replaces with yours. In my case the new IP are set back like this:

ifconfig eth0.1 0.0.0.0
ifconfig eth0.2 192.168.1.55 netmask 255.255.255.0

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

In my case I had to setup PPPoE by running pppoe-setup script. For most people this will not be necessary but if it happens it is easy; you should only pay attention to the last question and answer it depending on the firewall management you choose. Only in the case of Bell Sympatico you may hit on additional problems with MTU auto discovery. If it turn out that you are able to ping external machines, but browsing barely work, if at all, you will have to use some commands like next one in your firewall script:

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

In the case you decide to use firewall builder, it is only 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 did not make the setup with USB to Ethernet convertor, instead of manageable switch; I have no such device. I think that the biggest problem in it will be to make the device actually work. What I do not like about this solution is that, I have difficulties to believe in the performance of this configuration, even with the advertisements of the all manufacturers for unbelievable speeds of the USB to Ethernet convertors.

6. Configuration (c.) - bridging

This is the configuration I chose to stay on for now, as it will allow me to pull time to time my Aspire from the network and still have a network "I use the Aspire as GPS with USB satellite antenna when I travel". Here is a /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 about bridging is in bold, it is I think self-explanatory; the bridging on Linux is really easy and never gave me any troubles. Spanning tree should be off as it is by default; put it on only if you really know what you are doing.

The dhcpd is bound only to wlan0 to serve 172.17.0.0/16 addresses to Welcome network. Acer_A1 network is getting its IP addresses from "Linksys SRX 200" DHCP server trough the bridge "it transfers broadcasts transparently".

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

Of course you need a firewall too; here is the created with FWbuilder project acerap_br.fwb and the generated from it script acerap_br.fw; really basic, but a good start point.

After I used this configuration for around a month I decided to switch to WPA2. It require only to change **wpa=1** in /etc/hostapd.conf to **wpa=2** and restart hostapd. I was worried about the amount of work required to reconfigure all clients, but it turn that there is only some small changes for Windows clients.

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, simply install it. Most other collections seem to be using NetworkManager, but anyway there are no problems.

b. Mac OS X 10.7.2

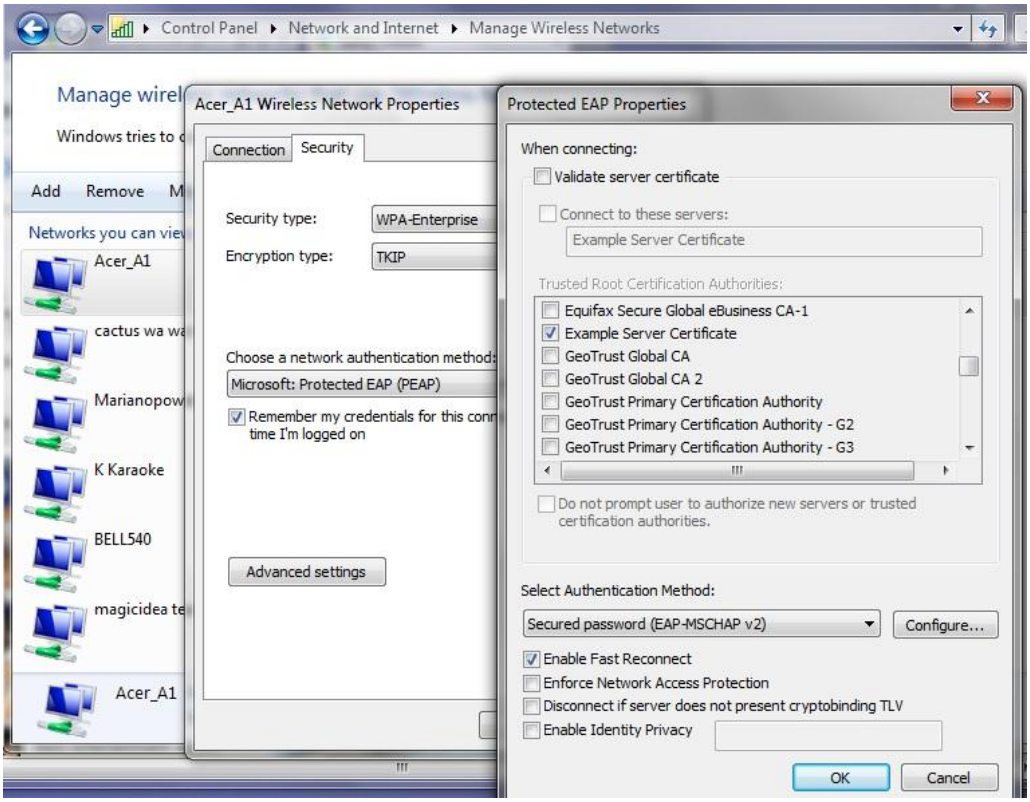
When I first tried to connect, it showed a message stating that the certificate is not from a known authority and offered me a check a box to accept it permanently, then it asked for username and password and then it simply worked. There is a key management program in Utilities called Key Chain Access. Here I marked the certificate as trusted and it became green. I do not think this was necessary but simply wanted to be on a safe side. The instruction I found, actually recommended to install the certificate manually in advance, but it turned out that Mac will do it for you. When you decide to use WPA2 there will be no need even to touch it. It detects the change of the access point, changes the stings accordingly and even reuses the username and password from the previous configuration

c. Ipod.

I have only second hand experience, as I asked my son to do it; did not want to look for my glasses and break my fingers with the toy. It was the same as the Mac, asked to accept the certificate and for username and password.

d. Windows 7

Nightmare – Do not expect to get a question about the certificate at this state - you will have to install the certificate in advance by yourself. Get it from **/usr/local/etc/raddb/certs /**; I did it with USB stick. The instructions from internet are to create connection manually and install the certificate when the wizard reaches this point; it does not help and Windows continues to complain about the certificate. I installed it through control panel, internet options; did not help either. Played with the check boxes "the best approach to Windows isn't it", and did some googleing. Suddenly it worked – when I tried to connect it asked to accept the certificate "the same one which I installed and I marked it trusted in all the places possible in advance!!!" Then username and password and it worked. This is the picture with the options for the certificate.



Simply because this is not enough you will have to do the same for all users on each laptop separately; even on the same laptop the procedure for each user will be similar but different.
The only advice I can give is "be persistent" and it will work. If you google the problem you will find that some people simply got around by buying certificates "obviously it comes cheaper for big number of laptops", but maybe quitting Windows is better.

If you decide to use WPA2, it will work but it will not detect the change automatically. The simple change in the properties "the picture above" will work. The encryption has to remain TKIP. It will ask for username and password and then it will 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 a 30 of my neighbors to start using the open "Welcome" network. Most of them turned to be with a very modest requirement, doing mostly mail and some surfing, but two or three fans of bittorrent turned out to be a problem. If you decide to provide some Internet for your neighbors you certainly should do something about it.

You have at least two options, l7-filter from http://l7-filter.clearfoundation.com/ and ipp2p from http://www.ipp2p.org/. One time or another I used both of them and the results are relatively the same. Still I prefer ipp2p with the ungrounded idea that it is less CPU consuming. On the project web page it shows that the project is discontinued. This is not exactly true; it is discontinued in the form of separate project for external module. It has been moved to the patch-o-matic and continued to be supported. Nowadays when netfilter.org discontinues patch-o-matic, it is moved to xtables-addons and netfilter.org is continuing to support it.

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

Then you may need something like this:

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

Most of these commands are unnecessary, even putting simply:

```
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 the best results.

Anyway, do not expect too much from it, or from l7-filter for that matter. They will slow down the bittorrent a lot, but boot have problems with recognizing encrypted connections. At least for sure bittorrent will not be able to kill all other connections anymore. If you are not satisfied with the results you should combine it with traffic shaping – next paragraph.

b. Traffic shaping

Decision to spend some time to setup and fine tune traffic shaping depend on the type of Internet connection, 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 symmetric connection you have nothing to worry, but if you are on something like ADSL and your provider have wired behavior, then taking your queue to your machine makes a real difference. You may read about the idea here The "Ultimate Traffic Conditioner".

It is important to mention that since the time of writing of "Linux Advanced Routing & Traffic Control HOWTO", lots of things had changed, but probably the most important thing in

the field of traffic shaping is the "Intermediate Functional Block device". Lots of work had been done in the field and you have to be really careful when you are doing your own research as many of the documentation and examples in the Internet are outdated. Most examples will still work fine, but often better solutions had been developed since then.

My traffic shaping script had the following of goals:

a. Move the queue to my machine.

b. Provide fairness booth between my family clients and between guests in the "Welcome" network.

c. Give warranted advantage to my own clients, leaving the clients in "Welcome" with what is left, but in the same time to warrant them some bandwidth even in the moments of heavy load. My Internet connection is actually 99% unused anyway, but I did not want to listen to complains from my family.

d. Have a method to separate really misbehaving clients from the crowd.

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

One very important thing that has to be mentioned is that limiting the outgoing traffic from certain source, does not lead to proportional limitation of the income traffic. Most of the streaming protocols require very small amount of outgoing requests in order to get real flood of incoming video. The result is that even class 1:13 "this is the place bad people finish", may seems too restrictive with its "rate 10kbit burst 15kbit" it actually gives around 600kbits download speed. The point is – to have a precise control; you have to shape incoming connections too.

Next is the chart of outgoing traffic shaping.

The traffic goes as follow:

From Acer_A1 ->1:11

From Welcome -> 1:12

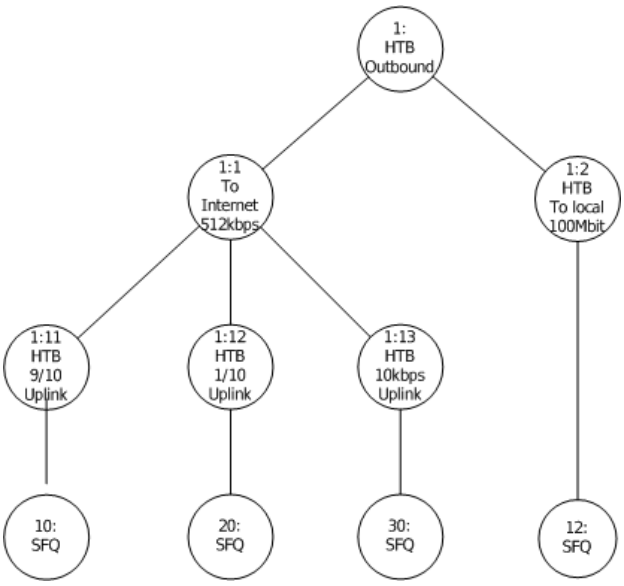
Bad clients -> 1:13

Between may cable clients and Acer_A1 -> 1:2

The traffic is classified by iptables with rules like this for example:

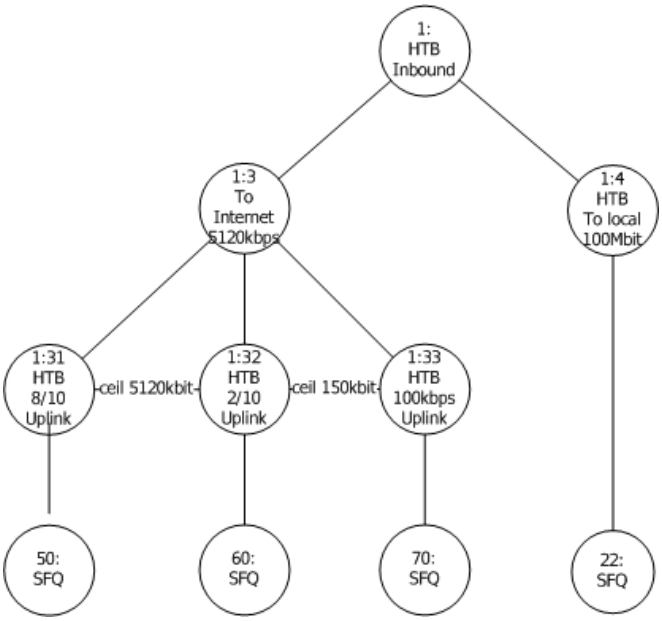
iptables -t mangle -A POSTROUTING -s 192.168.1.0/24 -d 192.168.1.0/24 -j CLASSIFY --set-class 1:2

You may see it in "Policy: Traffic_Control" in acer_br.pdf or check the details in acerap_br.fw script.



The next picture is 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



Here, first you have to get ifb0 "Intermediate Functional Block device" device working. It turned out that the module does not load automatically. It is loaded in the script rc.traffic_shaping by:

/sbin/modprobe ifb

ifconfig ifb0 up

The next problem is really interesting – see the part of the script rc.traffic_shaping 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, probably there is better solution, but this is the only one I found to work.

You will need the next commands, to investigate and adjust the script to your 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 the cache DNS server was a great advantage in the time when we thought that 28’800 modem is lighting fast. With nowadays speed the percentage of economized bandwidth is close to zero, but it is so easy to install, and beside old habits die hard. Simply make /etc/rc.d/rc.bind executable. Slackware has /etc/named.conf pre-ready. It is good idea to setup regular updates of named.root, let’s say simply by creating the script /etc/cron.monthly/named.root and put 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 empty dhcpd.log by:

:> /var/log/dhcpd.log

Of course dhcpd and syslogd have to be restarted.

iptables log. If you use Firewall Builder or simply like to have extensive log from your firewall it is a tempting idea to move iptables log in separate file. The complication here comes from the limited choice of "--log-level X" you have. As a result that iptables, actually the kernel is doing all the filtering, all logs come in log facility "kern.*", your choice is limited between all levels **"0 emerg, 1 alert, 2 crit, 3 err, 4 warning, 5 notice, 6 info, 7 debug"**. Beside "**crit**" is the default level for klogd to send messages to the console; whatever goes on this level inevitably goes on the console as well. You may experiment with other levels or try to change "klogd -c 3" to something else.

After these difficult choices everything else is simple.

First either change log level setting in Firewall Builder, or if you wrote your own script set something like "**-j LOG --log-level warn --log-prefix "my log text**"".

Second 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 it 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 to the following line:

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

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

If you want see what else goes in /var/log/fwbuilder.log together with iptables, next command will help you:

cat /var/log/fwbuilder.log |grep RULE –v

9. Some final words.

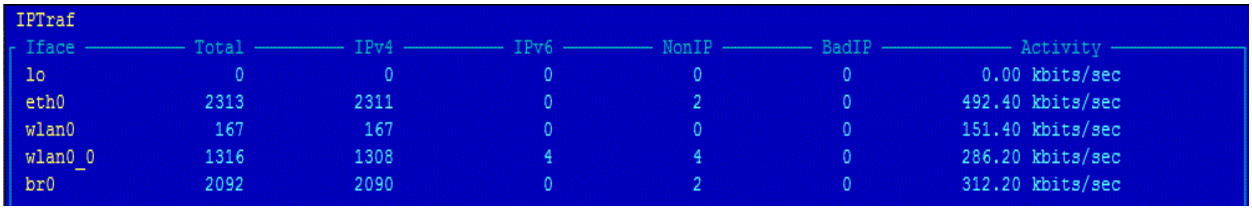
It passed over a month since I started writing this document, creating the access point took me three days, but writing did not go as fast as I wanted it.

For this time the AP did not drop one connection and provided nearly 100m coverage. No common error messages common for "Linksys SRX 200".

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

At the moment:

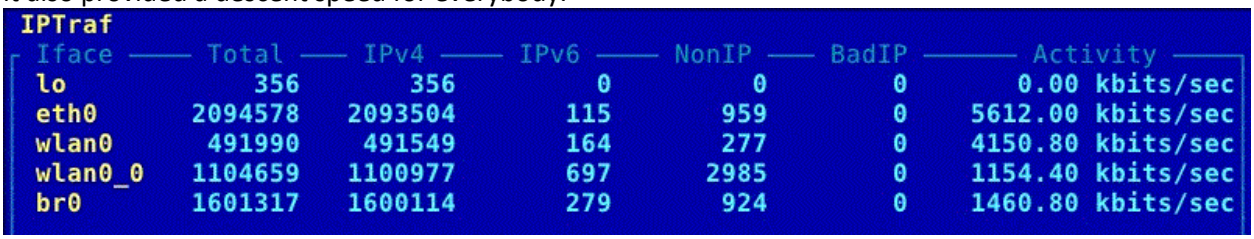
root@acer:/var/log# cat /proc/net/ip_conntrack |grep tcp -c
565
root@acer:/var/log# cat /proc/net/ip_conntrack |grep udp -c
76



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

But even at the moment of a heavy load, like below, not only the network remained stable, but it also provided a descent speed for everybody.



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 descent 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 is to the best of all expectations.

10.Copyright