‘Linux Link Bonding’ Research Assignment

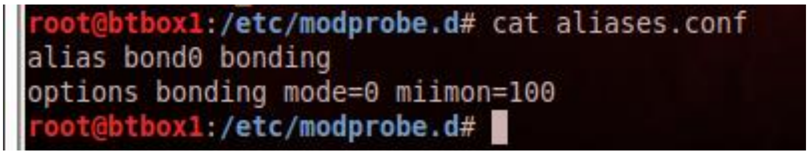
Name: David Kirwan

Course: BSc (Hons) Applied Computing

Year: 3 / 2011

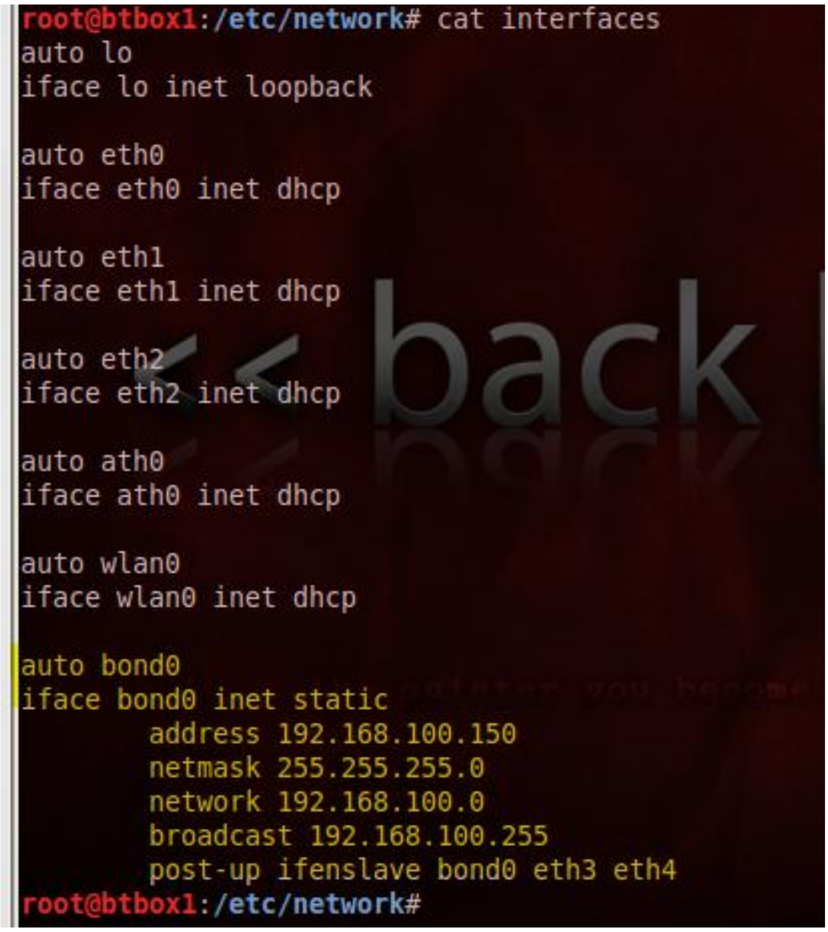
Table of Contents

Tutorial for setting up Linux Link bonding on Ubuntu (Backtrack 5):



```
root@btbox1:/etc/modprobe.d# cat aliases.conf
alias bond0 bonding
options bonding mode=0 miimon=100
root@btbox1:/etc/modprobe.d# 
```

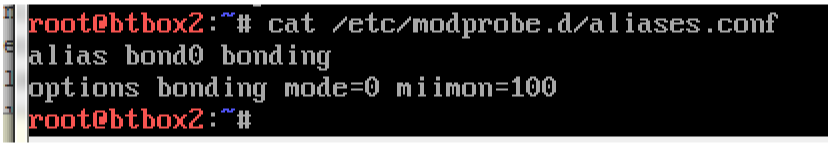
When creating a link bond on Ubuntu, the first step is to create the information above in the /etc/modprobe.d/aliases.conf file. This informs the kernel that this bonding interface bond0 will be using the round-robin mode for load balancing and that the miimon or the MII link monitoring frequency will be 100 milliseconds. I added this information into the correct file on host: btbox1



```
root@btbox1:/etc/network# cat interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet dhcp

auto eth2
iface eth2 inet dhcp

auto ath0
iface ath0 inet dhcp

auto wlan0
iface wlan0 inet dhcp

auto bond0
iface bond0 inet static
        address 192.168.100.150
        netmask 255.255.255.0
        network 192.168.100.0
        broadcast 192.168.100.255
        post-up ifenslave bond0 eth3 eth4
root@btbox1:/etc/network#
```

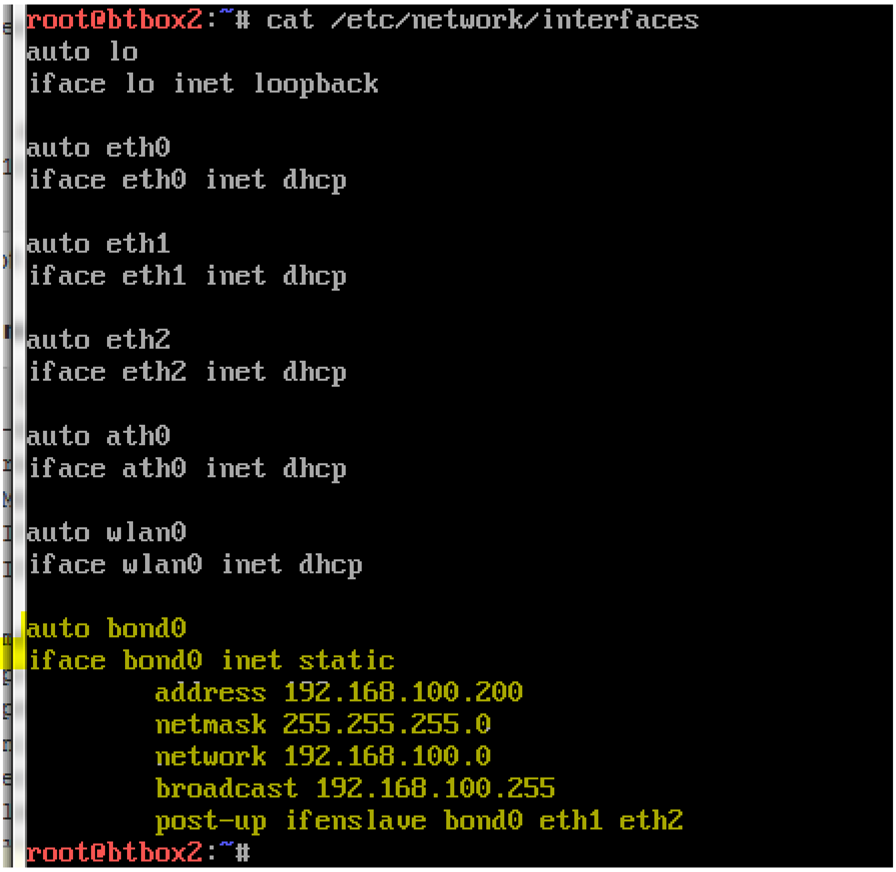
Next we add the information in yellow above, to the bottom of the /etc/network/interfaces config file. This informs the kernel that at boot time, bring the bond0 interface up automatically and assign it the static IP address 192.168.100.150 / 24 and informs the kernel that the slave interfaces eth3 and eth4 will be members of this bonded link.

I carried out the same steps on the 2nd Host: btbox2.



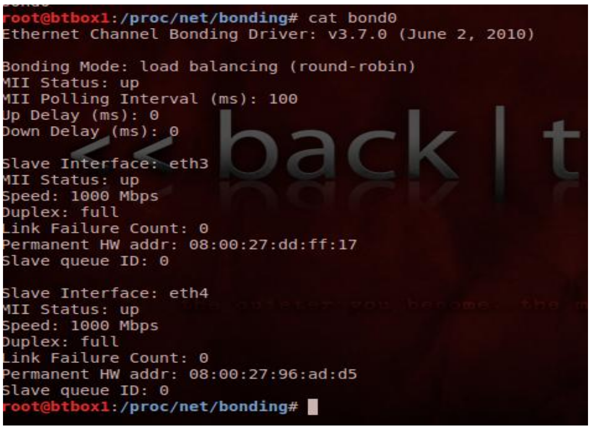
```
root@btbox2:~# cat /etc/modprobe.d/aliases.conf
alias bond0 bonding
options bonding mode=0 miimon=100
root@btbox2:~#
```

And



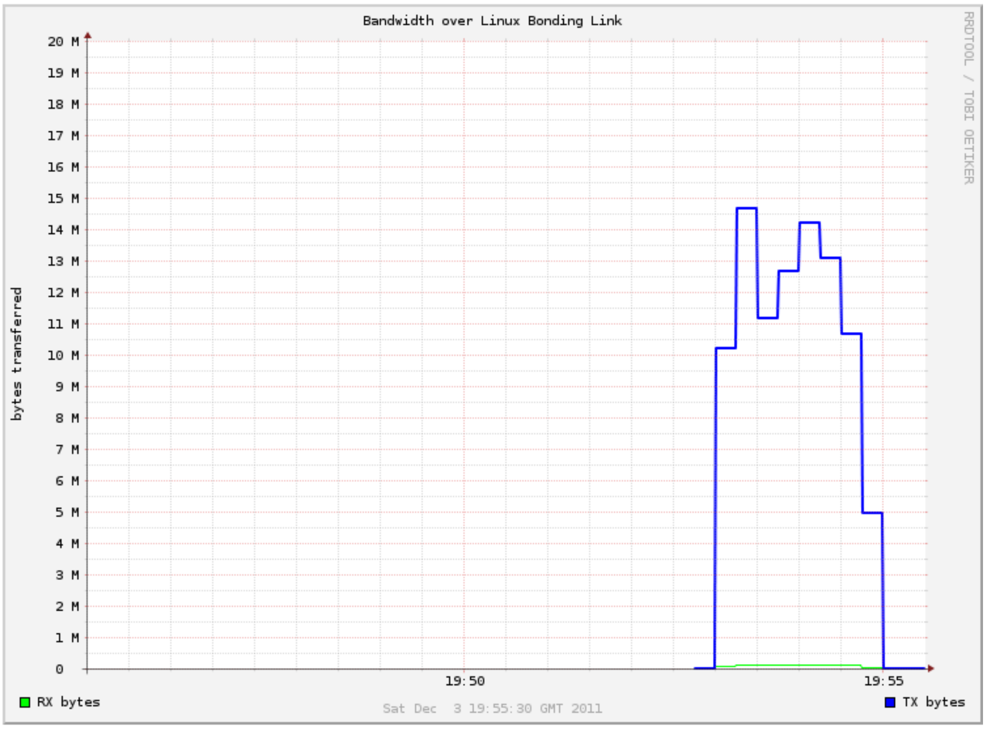
```
root@btbox2:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet dhcp

auto eth2
iface eth2 inet dhcp

auto ath0
iface ath0 inet dhcp

auto wlan0
iface wlan0 inet dhcp

auto bond0
iface bond0 inet static
        address 192.168.100.200
        netmask 255.255.255.0
        network 192.168.100.0
        broadcast 192.168.100.255
        post-up ifenslave bond0 eth1 eth2
root@btbox2:~#
```

Lastly I rebooted both machines, once they were back up, I examined the contents of the file /proc/net/bonding/bond0, this shows that the bond0 interface is up, and using the 2 slave interfaces eth4 and eth3, and is also using the round-robin load balancing mechanism for choosing which of the slave interfaces to transmit the packets out.



```
root@btbox1:/proc/net/bonding# cat bond0
Ethernet Channel Bonding Driver: v3.7.0 (June 2, 2010)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth3
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:dd:ff:17
Slave queue ID: 0

Slave Interface: eth4
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 08:00:27:96:ad:d5
Slave queue ID: 0
root@btbox1:/proc/net/bonding#
```

Experimentation with the rrdtool to show the current throughput on the bonded link:

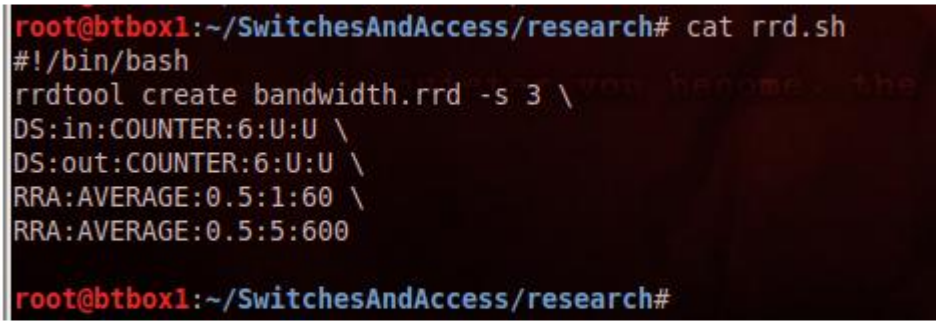


Graph generated by rrdtool depicting the number of bytes being transmitted on the bond0 interface on host1.

Rrdtool is an opensource framework for logging data suitable for graphing at a later stage, and is freely available from http://oss.oetiker.ch/rrdtool/download.en.html

Rrdtool came installed as standard in the Backtrack 5 Linux distribution.

In order to have rrdtool log data and produce a graph three steps must first be carried out. An rrd database needs to be constructed.



```
root@btbox1:~/SwitchesAndAccess/research# cat rrd.sh
#!/bin/bash
rrdtool create bandwidth.rrd -s 3 \
DS:in:COUNTER:6:U:U \
DS:out:COUNTER:6:U:U \
RRA:AVERAGE:0.5:1:60 \
RRA:AVERAGE:0.5:5:600

root@btbox1:~/SwitchesAndAccess/research#
```

The screenshot above contains the information stored inside the script I used to generate the rrdtool database.

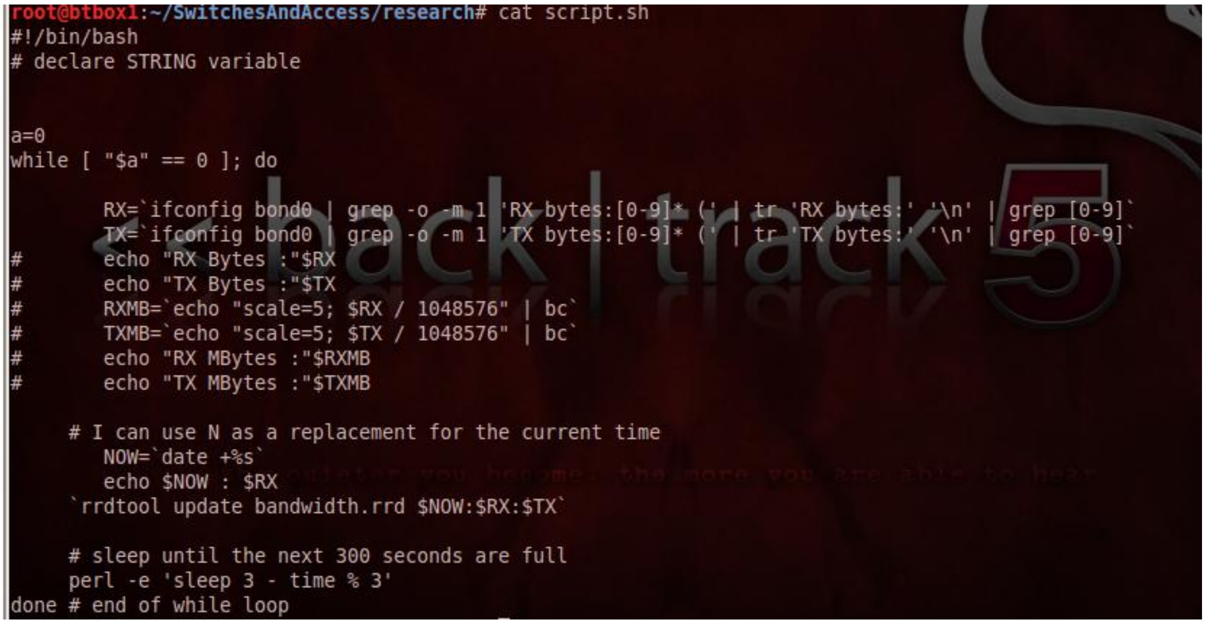
Bandwidth.rrd is the name of the database

DS:in is a dataset called “in” it is of type COUNTER and has a timeout counter of 6 seconds before the database enters a 0 value for this tick. The U:U are the min/max values, we set them as unlimited.

The DS:out is the dataset where the TX values are to be stored, like DS:in the other values are identical.

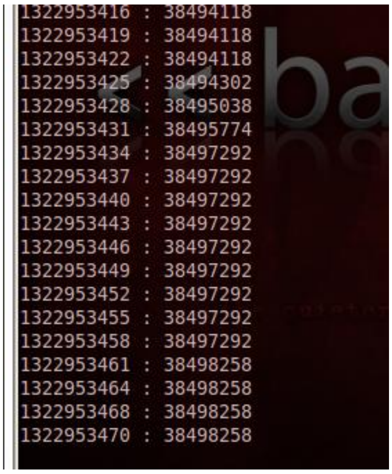
Running that script generates the rrd database.

Next we need to populate the database with data, and I achieved this with a second bash script which I leave running as long as data logging is required. I decided against adding a cron job as this is a temporary use for this Linux VM.



```
root@btbox1:~/SwitchesAndAccess/research# cat script.sh
#!/bin/bash
# declare STRING variable


a=0
while [ "$a" == 0 ]; do

        RX=`ifconfig bond0 | grep -o -m 1 'RX bytes:[0-9]* (' | tr 'RX bytes:' '\n' | grep [0-9]`
        TX=`ifconfig bond0 | grep -o -m 1 'TX bytes:[0-9]* (' | tr 'TX bytes:' '\n' | grep [0-9]`
#       echo "RX Bytes :"$RX
#       echo "TX Bytes :"$TX
#       RXMB=`echo "scale=5; $RX / 1048576" | bc`
#       TXMB=`echo "scale=5; $TX / 1048576" | bc`
#       echo "RX MBytes :"$RXMB
#       echo "TX MBytes :"$TXMB

     # I can use N as a replacement for the current time
        NOW=`date +%s`
        echo $NOW : $RX
     `rrdtool update bandwidth.rrd $NOW:$RX:$TX`

     # sleep until the next 300 seconds are full
     perl -e 'sleep 3 - time % 3'
done # end of while loop
```

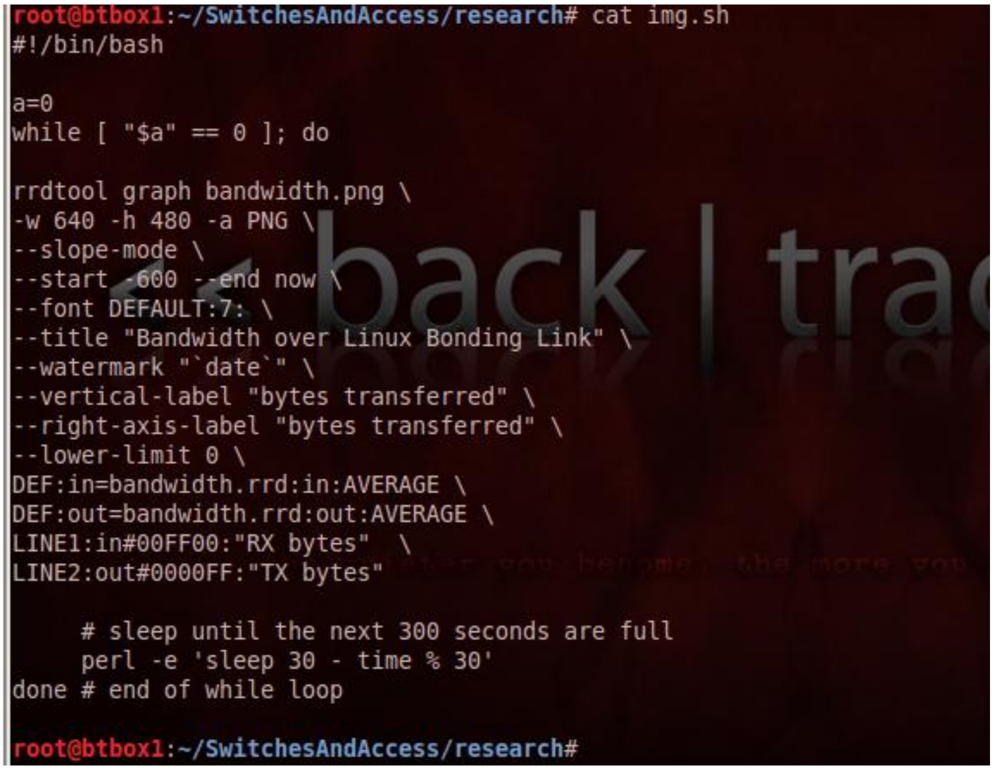
In a loop which runs every 3 seconds, I grab the RX and TX byte values from the bond0 interface, and add them to the rrd database.



```
1322953416 : 38494118
1322953419 : 38494118
1322953422 : 38494118
1322953425 : 38494302
1322953428 : 38495038
1322953431 : 38495774
1322953434 : 38497292
1322953437 : 38497292
1322953440 : 38497292
1322953443 : 38497292
1322953446 : 38497292
1322953449 : 38497292
1322953452 : 38497292
1322953455 : 38497292
1322953458 : 38497292
1322953461 : 38498258
1322953464 : 38498258
1322953468 : 38498258
1322953470 : 38498258
```

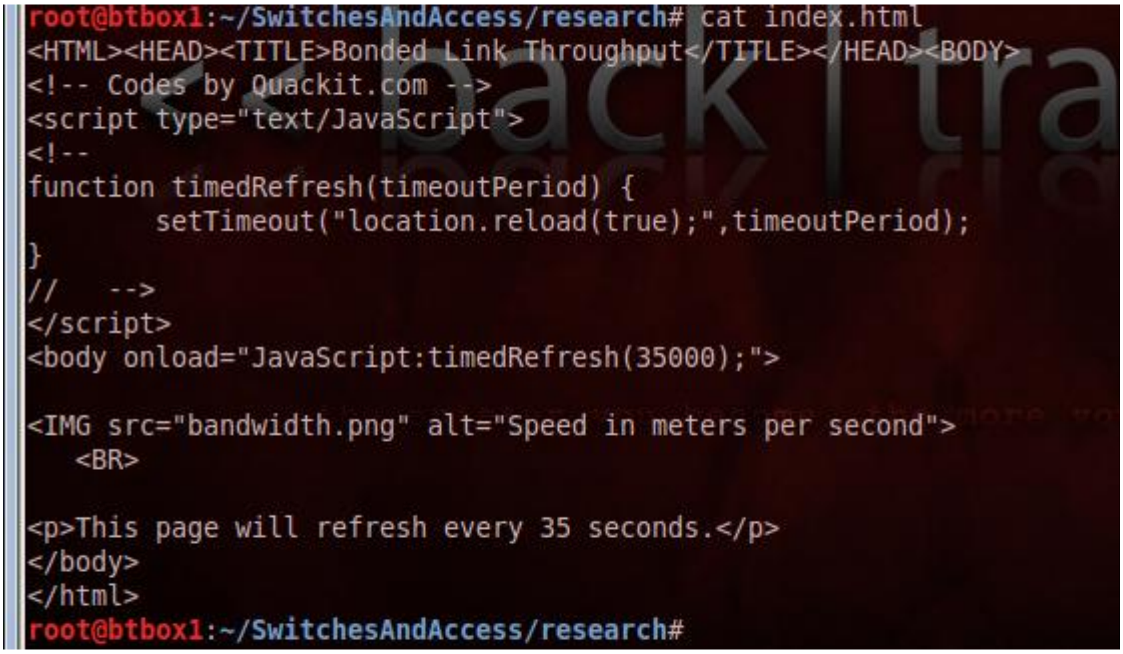
This script runs away every 3 seconds adding the data to the database as is visible above.

Next I wrote a third bash script to generate the image graph, it is detailed below:



```
root@btbox1:~/SwitchesAndAccess/research# cat img.sh
#!/bin/bash

a=0
while [ "$a" == 0 ]; do

rrdtool graph bandwidth.png \
-w 640 -h 480 -a PNG \
--slope-mode \
--start -600 --end now \
--font DEFAULT:7: \
--title "Bandwidth over Linux Bonding Link" \
--watermark "`date`" \
--vertical-label "bytes transferred" \
--right-axis-label "bytes transferred" \
--lower-limit 0 \
DEF:in=bandwidth.rrd:in:AVERAGE \
DEF:out=bandwidth.rrd:out:AVERAGE \
LINE1:in#00FF00:"RX bytes"  \
LINE2:out#0000FF:"TX bytes"

     # sleep until the next 300 seconds are full
     perl -e 'sleep 30 - time % 30'
done # end of while loop

root@btbox1:~/SwitchesAndAccess/research#
```

As with the script which populates the rrd database, this image generator script also runs in a loop, every 30 seconds, this script generates a bandwidth.png image which I have embedded on a HTML webpage for easy viewing. I also added a simple javascript function to automatically refresh the page every 35 seconds, this is now suitable for perhaps displaying on a monitor attached to a server to easily view the current throughput on a bonded link (quackit.com).



```
root@btbox1:~/SwitchesAndAccess/research# cat index.html
<HTML><HEAD><TITLE>Bonded Link Throughput</TITLE></HEAD><BODY>
<!-- Codes by Quackit.com -->
<script type="text/JavaScript">
<!--
function timedRefresh(timeoutPeriod) {
        setTimeout("location.reload(true);",timeoutPeriod);
}
//   -->
</script>
<body onload="JavaScript:timedRefresh(35000);">

<IMG src="bandwidth.png" alt="Speed in meters per second">
   <BR>

<p>This page will refresh every 35 seconds.</p>
</body>
</html>
root@btbox1:~/SwitchesAndAccess/research#
```

Bibliography

Gite, V. (2007, September 25). */etc/network/interfaces example for Ubuntu Linux*. Retrieved December 3, 2011, from http://www.cyberciti.biz: http://www.cyberciti.biz/faq/setting-up-an-network-interfaces-file/

Linux IP. (n.d.). *Link Aggregation and High Availability with Bonding*. Retrieved November 16, 2011, from Linux IP: http://linux-ip.net/html/ether-bonding.html

linuxhorizon.ro. (n.d.). *Bonding (Port Trunking)*. Retrieved December 3, 2011, from www.linuxhorizon.ro: http://www.linuxhorizon.ro/bonding.html

Oetiker, T. (2009, October 28th). *RRDTool*. Retrieved November 16th, 2011, from RRDTool Graphing and Logging: http://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html

quackit.com. (n.d.). *Javascript Refresh Page*. Retrieved December 3, 2011, from quackit.com: http://www.quackit.com/javascript/javascript_refresh_page.cfm

The Linux Foundation. (2009, November 19). *Link Bonding*. Retrieved November 16th, 2011, from The Linux Foundation: http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding

Vosburg, J. (2005, June 21). *Linux Ethernet Bonding Driver HOWTO*. Retrieved December 3, 2011, from cyberciti.biz: http://www.cyberciti.biz/howto/question/static/linux-ethernet-bonding-driver-howto.php