Link Sharing with DD-WRT

CSC 466

John Rogers

15/04/13

Problem Description

Students often deal with problems. These problems commonly fall into the 'roommate', 'technology', and 'money' categories; this paper studies a problem that lies in the intersection of these categories: internet bandwidth sharing. The question of how to fairly and efficiently addressed the needs of each member of a household is surprisingly complex. Divisions of the link must reflecting the demand, current use, and ownership of the internet link. Fortunately, all of these variables can be taken into account with the proper configuration of a DD-WRT router. This paper studies how to configure the router, what effects the configuration has on link sharing, how the router achieves these effects, and demonstrates the effects by testing an example link sharing problem among three students who wish to allocate the link speed differently between their machines.

The students have different demands of of their 10 Mbps internet connection, and so split their cable bill unequally, with Alex paying twenty percent, Betty paying thirty percent, and Carl paying fifty percent. Accordingly, Carl expects to use at least fifty percent of the connection when he needs it, Betty thirty percent, and Alex twenty percent. The students are willing to share their bandwidth when it is not in use, with the agreement that those who pay a higher bill should receive more of the unused bandwidth. So, for example if Carl is not using his link but both Betty and Alex are demanding 10 Mbps, then Betty should receive more of the free 5 Mbps bandwidth than Alex. The students also agree that guests can be allocated 0.3 Mbps, created from 0.1 Mbps taken from each share, and if there is any leftover bandwidth it should be offered to the guests first so that if any one person has a guest, he or she can reduce his or her network consumption in order increase the guest's bandwidth.

QoS and Performance

This paper uses Quality of Service as a term used to describe the average amount of bandwidth allocated to a client when communicating using DD-WRT. QoS can be set for individual clients of a DD-WRT router by using the transmission control (tc) tool. Transmission control achieves this discrimination by configuring and applying a queuing discipline program to a link. The queuing discipline, as the name suggests, queues traffic vying for link use, then, in some order that depends on the nature and configuration of the discipline, serves those queues. Link sharing using a queuing discipline does not slow down the line rate, it slows the average rate of data sent. The discipline emulates many slower link rates by rapidly switching between serving a queue at line rate and not serving it at all; each queue represents a virtual link with a rate equal to or less than the real link rate.

There are two queuing disciplines available on a DD-WRT router, Hierarchical Token Bucket (HTB) and Hierarchical Fair Service Curve (HFSC). This study focuses on HTB. A token bucket queuing discipline assigns each queue a 'bucket' that on regular intervals receives a deposit of 'tokens', each of which represent some amount of data that can be sent on the outgoing link. A 1500 byte packet can only be sent if there 1500 bytes worth of tokens in the bucket, and once sent a 1500 bytes worth of tokens are removed from the bucket. Buckets are different sizes and once full cannot accept deposits of tokens.

The hierarchical aspect comes into play when multiple queues exist within the discipline. To start, a queuing discipline is attached to an interface, then queues may be added as 'children' of that discipline, and more queues can be added as children of those queues, and so on. This results in a root-like structure, where traffic arrives at the tips of the root and travels upward to the outgoing link. The amount of traffic that can travel up a stem of the root is determined by the number of tokens that traffic's queue has. Tokens can be shared among queues that share a parent. This makes the bandwidth sharing efficient and fair because one queue's unused tokens are not wasted, rather they are 'borrowed' by a sibling queue with higher demands. This causes the outgoing link to always send while there is enqueued traffic. The sharing of unused bandwidths is determined by comparing priorities of the children, higher priority children receive all bandwidth, or, if there is a priority tie, 'quantums' are used to distribute bandwidth. By default a queue's quantum is one tenth of its allocated bandwidth, and each child gets a fraction of the bandwidth equal to the fraction of its quantum divided by the sum of all competing children's quantums.

Methodology

HTB was configured on a DD-WRT router to reflect the needs of Alex, Betty, Carl and their guest. The flows from the four computers were simulated from a laptop connected to one home LAN network, while another laptop, running on another home LAN, acted as the destination server. Three different situations were simulated. First, all of the users attempt to use the link at link rate. Second, only Alex is accessing the server at link rate, then a guest joins him, also requesting traffic at link rate. Finally, Betty and Alex use the link at link rate and then Carl joins; Carl uses half of his allocated bandwidth for a period, and then attempts to use the link at link rate. The link division in these situations will reflect effectiveness of using HTB to provide differing QoS.

The easiest way to apply a QoS to an interface of a Linux router is by using the transmission control, 'tc', tool. On the subject DD-WRT router the interface directly connected to the internet was the

interface 'eth1'. For the test cases, the commands were issued using the script in are listed below.

```
#clear queuing discipline
tc qdisc del dev eth1 root

#place queuing discipline on interface
tc qdisc add dev eth1 root handle 1: htb default 1:13

#create root parent
tc class add dev eth1 parent 1: classid 1:1 htb rate 10mbit ceil 10mbit

#create classes for machines
#Alex's 20% class
tc class add dev eth1 parent 1:1 classid 1:10 htb rate 1.9mbit \
      ceil 10mbit prio 1
#Betty's 30% class
tc class add dev eth1 parent 1:1 classid 1:11 htb rate 2.9mbit \
      ceil 10mbit prio 1
#Carl's 50% class
tc class add dev eth1 parent 1:1 classid 1:12 htb rate 4.9mbit \
      ceil 10mbit prio 1
#Guest's 'leftovers' class
tc class add dev eth1 parent 1:1 classid 1:12 htb rate .3mbit \
      ceil 10mbit prio 0

#sort traffic to nodes
#Alex's ip is 192.168.1.110
tc filter add dev eth1 parent 1: protocol ip prio 1 u32 \
      match ip src 192.168.1.110 flowid 1:10
#Betty's ip is 192.168.1.111
tc filter add dev eth1 parent 1: protocol ip prio 1 u32 \
      match ip src 192.168.1.111 flowid 1:11
#Carl's ip is 192.168.1.112
tc filter add dev eth1 parent 1: protocol ip prio 1 u32 \
      match ip src 192.168.1.112 flowid 1:12
#Guest is any connection
tc filter add dev eth1 parent 1: protocol ip prio 0 u32 \
      match ip src 0.0.0.0/0 flowid 1:13
```

First, the queuing discipline currently attached to the interface is removed, and a new one, HTB, with the identifier 1: is added. This causes any traffic passing through the interface to be subjected to HTB queuing but, because it has no parameters or classes to queue, the command does nothing at this point. Children of a queuing discipline are not able to share unused bandwidth; in the test situation the students want to share bandwidth, therefore it was necessary to create a 'root' class who's children *can* share bandwidth. This root parent class has a rate and ceiling rate, 'ceil 10mbit' equal to the link rate that will be divided among its children. Next, the various classes for the students are added, with the appropriate rates granted to each student – the maximum amount of bandwidth a student can own is the link rate, specified by 'ceil 10mbit'. The guest class is a higher priority than the student's classes, this will cause excess bandwidth to be offered to guest traffic first. Finally, tc filter is used to filter the traffic into the created classes using an IP address as an identifier – MAC addresses can be used to identify traffic but the commands are more complex and unnecessary for this experiment.

Once the router was correctly configured, the client VM was set up with four virtual Ethernet cards, each assigned a different IP address. The program 'iperf' was used to create UDP data streams and record performance. Then the server was set up a few blocks away, and the router configured to forward requests at ports 1010 to 1013 to the server, see figure 1 for a diagram. On those ports iperf servers were set up to listen for connections. The results were obtained from the server logs.

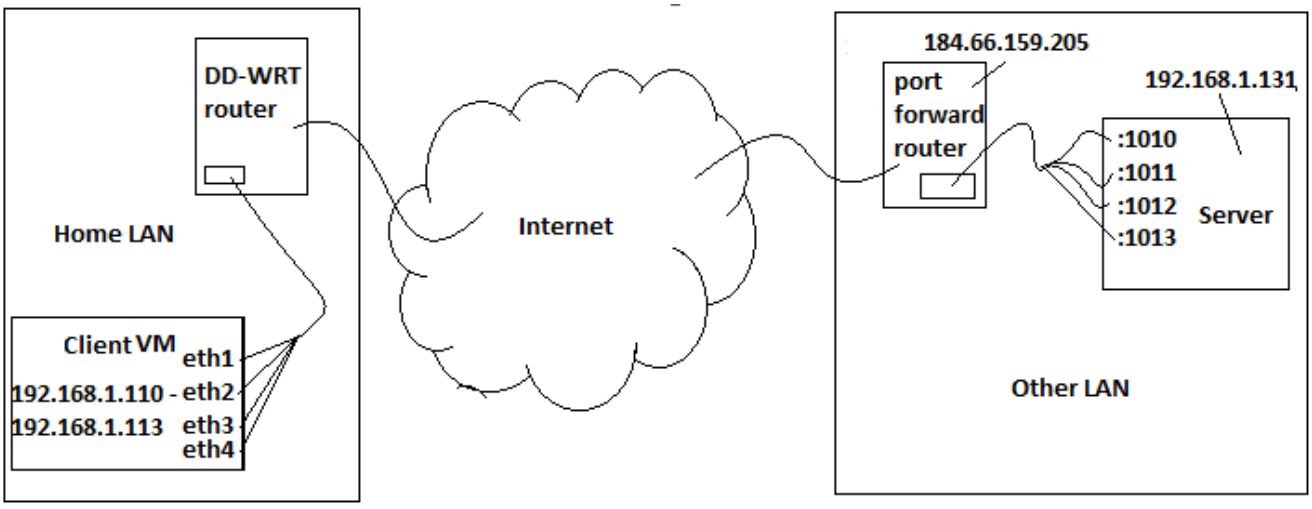


Figure 1: Diagram of the testing setup

The below commands create iperf servers that listen for UDP streams on the internet facing ip address, on different ports, to give reports every second, and to output data in megabits per second format.

```
iperf -u -s -B 192.168.1.131 -p 1010 -i 1 -f m > alex_srv &
iperf -u -s -B 192.168.1.131 -p 1011 -i 1 -f m > betty_srv &
iperf -u -s -B 192.168.1.131 -p 1012 -i 1 -f m > carl_srv &
iperf -u -s -B 192.168.1.131 -p 1013 -i 1 -f m > guest_srv &
```

UDP client streams were also generated by using the iperf tool. For the first test the streams were created near-simultaneously - each to a different interface and destination, and all sending at 10 Mbps for 10 seconds. For the second test, Alex's stream from eth1 had no competition for the link, and sends at 10 Mbps for 20 seconds. After 10 seconds in a guest stream from eth4 joins and also sends at 10 Mbps for 10 seconds. In the final case, Alex and Betty's streams from eth1 and eth2 send at 10 Mbps for 15 seconds, after 5 seconds they are joined by Carl's stream from eth3 that sends at 2.5 Mbps for 5 seconds, then at 10 Mbps for another 5 seconds. All of the commands used to generate these streams are listed below.

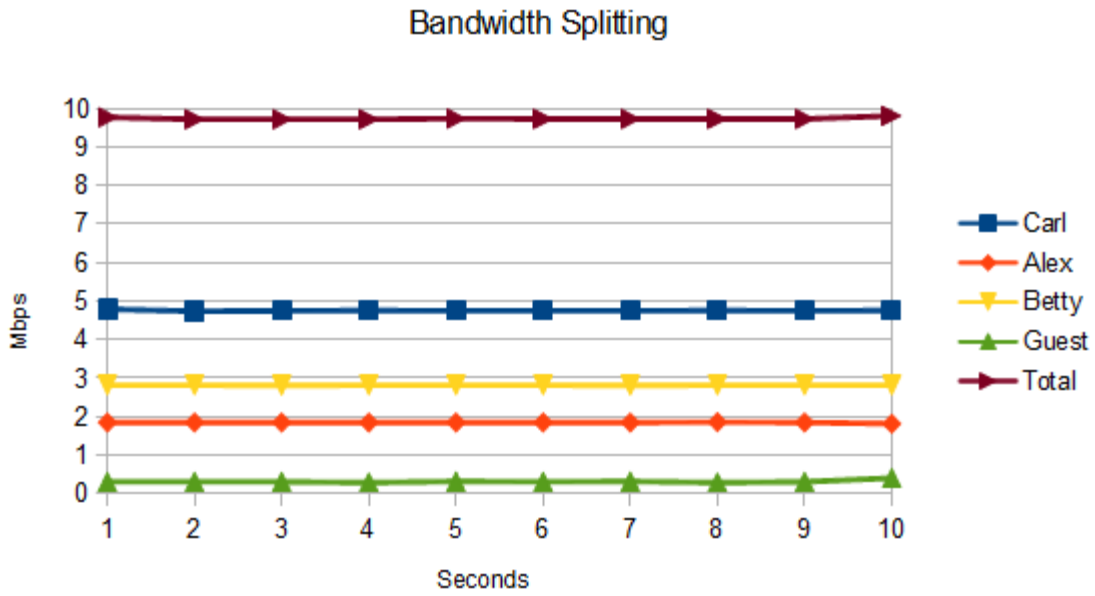
```
#test case 1
iperf -u -c 184.66.159.205 -B 192.168.1.110 -b 10000000 -p 1010 \
      --time 10 > alex_cli &
iperf -u -c 184.66.159.205 -B 192.168.1.111 -b 10000000 -p 1011 \
      --time 10 > betty_cli &
iperf -u -c 184.66.159.205 -B 192.168.1.112 -b 10000000 -p 1012 \
      --time 10 > carl_cli &
iperf -u -c 184.66.159.205 -B 192.168.1.113 -b 10000000 -p 1013 \
      --time 10 > guest_cli &

#test case 2
iperf -u -c 184.66.159.205 -B 192.168.1.110 -b 10000000 -p 1010 \
      --time 20 > alex_cli &
sleep 10
iperf -u -c 184.66.159.205 -B 192.168.1.113 -b 10000000 -p 1013 \
      --time 10 > guest_cli &

#test case 4
iperf -u -c 184.66.159.205 -B 192.168.1.110 -b 10000000 -p 1010 \
      --time 15 > alex_cli &
iperf -u -c 184.66.159.205 -B 192.168.1.111 -b 10000000 -p 1011 \
      --time 15 > betty_cli &
sleep 5
iperf -u -c 184.66.159.205 -B 192.168.1.112 -b 2500000 -p 1012 \
      --time 5 > carl_cli &
sleep 5
iperf -u -c 184.66.159.205 -B 192.168.1.113 -b 10000000 -p 1012  \
      --time 5 > carl_cli &
```

Results

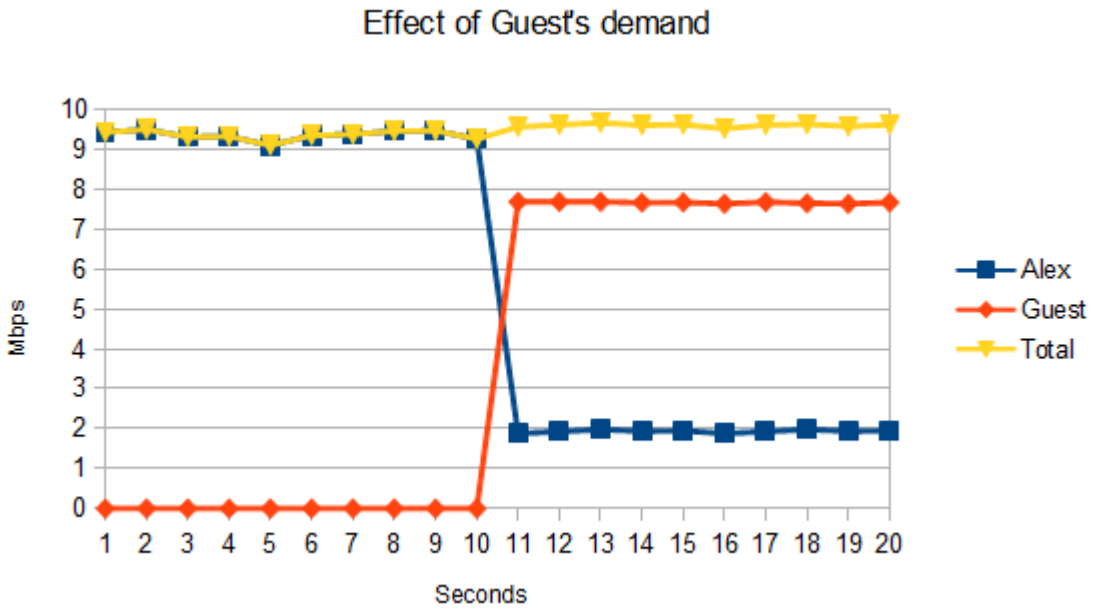
Test 1 – High Competition



Graph 1: Test case demonstrating the division of bandwidth among clients.

In the first test all users request 10 Mbps of bandwidth. Alex, Betty, Carl, and the guest receive, on average, 1.8 Mbps, 2.8 Mbps, 4.8 Mbps, and .31 Mbps of bandwidth respectively, for a total average transmission rate of 9.7 Mbps. The guest receives .01 more than its allocated bandwidth, and Alex, Betty, and Carl receive .1 Mbps less than their allocated rates. The total link usage is 0.3 less than supposed ceiling rate of 10 Mbps. The average rates allocated at each second are shown in Graph 1.

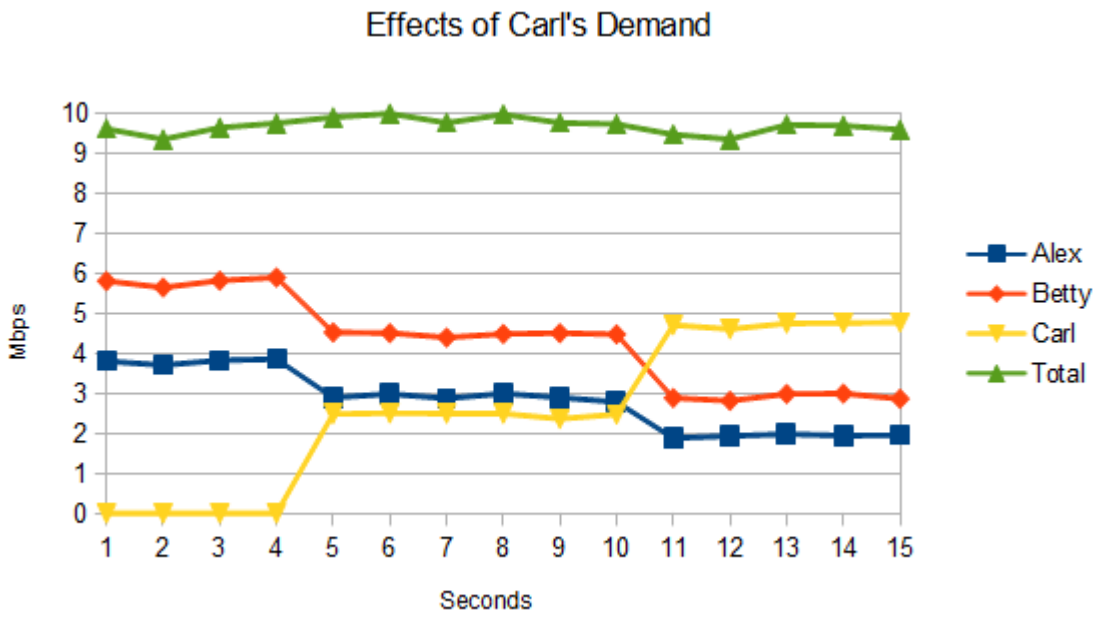
Test 2 – Link Sharing Using Priorities



Graph 2: Test demonstrating the effect of priority in bandwidth sharing

In the second test Alex enjoys unfettered access to the link between seconds 1 and 9; she requests 10 Mbps of bandwidths and receives on average 9.7 Mbps of bandwidth. When the guest joins at second 10 with a requested 10 Mbps of service, Alex's bandwidth is cut to an average 1.9 Mbps for the duration of the test, while the guest receives 7.7 Mbps on average. The total average link rate in the second half of the test is 9.6 Mbps. The average rates allocated at each second are shown in Graph 2.

Test 3 – Link Sharing Using Quantums



Graph 3: Test demonstrating the effect of quantums on bandwidth sharing

In the third test Alex and Betty are the only people using the link between seconds 1 and 4, and both request 10 Mbps of service. Alex receives an average bandwidth of 3.8 Mbps and Betty receives an average bandwidth of 5.8 Mbps. Carl joins at second 5, with bandwidth demand of 2.5 Mbps and receives an average bandwidth of 2.5 Mbps, while Alex and Betty's average bandwidth drops to 2.9 and 4.5 Mbps. At second 11 Carl's demand jumps to 10 Mbps, and he is allocated 4.7 Mbps of bandwidth on average, while Alex and Betty's average bandwidth allocation drops again, to 2 Mbps and 2.9 Mbps of bandwidth each. The average rates allocated at each second are shown in Graph 3.

Conclusions

All of the test results show some anomalies. The anomalies may be due to inaccurate readings from the iperf server, or unpredicted behavior of the iperf clients; future tests should use another tool that provides real-time reports on connection status. The most obvious anomaly is no test has a total average transmission rate equal to the maximum specified rate of 10 Mbps; the first test comes closest with a 9.7 Mbps average rate, and the furthest is test 2 with a 9.6 Mbps average rate. Some of this wastage would likely be caused by packet loss since UDP traffic isn't guaranteed, but a loss of 4% of packets is unlikely. It is an important result that HTB never allows the outgoing flow to exceed the ceiling rate; this is key if HTB is being used to adhere to a service contract. Another anomaly is in test

1 where the guest was able to send at an average rate of .31 Mbps, which is .01 Mbps higher than its allocated bandwidth. These two anomalies indicate that HTB is not totally reliable under these test conditions, but performs well enough to be trusted to not exceed the maximum allocated bandwidth.

The second test shows that nearly all of the link rate can be consumed by Alex when no others are contenting for bandwidth. It also shows how the excess bandwidth, the extra 8.1 Mbps that Alex was using beyond his given rate, is allocated to the guest once the guest starts transmitting. This happens because the guest has a higher priority than Alex. When the guest joins, the available bandwidth decreases from 8.1 to 7.8, and all of it is allocated to the guest.

The final test shows how a user's quantum determines how much of the free bandwidth he or she is allocated. Prior to Carl joining, Alex and Betty are competing for Carl and the guest's unused 5.2 Mbps share of the bandwidth. To split the excess bandwidth between them, their quantums are first added: 0.19 + 0.29 = 0.48, then divided by the sum, (0.19/0.48) = 0.4, (0.29/0.48) = .6, and multiplied by the available bandwidth, therefore Alex gets 0.4*5.2 Mbps, or 2.08 Mbps, for a total of 3.98 Mbps while Betty gets 0.6*5.2 Mbps, or 3.12 Mbps, for a total of 6.02; and indeed, Alex gets an average of 3.8 Mbps while Betty gets an average of 5.8 Mpbs. After second 5, Alex and Betty are competing for 2.7 Mbps of available bandwidth, Alex gets an extra 0.4*2.7 Mbps, or 1.08 Mbps, for a total of 2.98 Mbps and Betty gets an extra .6 * 2.7, or 1.62 Mbps for a total of 4.52 Mbps; in the test Alex gets an average of 2.9 Mbps and Betty gets 4.5 Mbps. At second 11 Carl's demand jumps up to 10 Mbps, and we see sharing similar to in the first test, but not totally the same since the 0.3 Mbps allocated to the guest is unused and split between the students.

Learning how to use Linux at this level was a challenge, as was the process of discovering how QoS is provided in Linux machines, but the most time consuming task was searching for a program that monitors ports and generates graphs using real time statistics. Tools exist that use SNMP to grab statistics from the router, but the time required to overcome the learning curve and trouble shoot problems proved too large. It seems that HTB works as specified, with a few imperfections such as not using all of the available bandwidth and infrequently over allocating small amounts of bandwidth to a class. Importantly, HTB never attempts to send at higher rate than the specified ceiling rate of the root parent class, and distributes available bandwidth correctly according to the rules of priority and sharing with respect to a class's quantum. The tool could be used to satisfaction for bandwidth sharing in a residential setting, but more comprehensive tests should be conducted before it is used in a commercial environment. Future tests should use a monitoring tool that reports bandwidth consumption statistics in real time and include TCP transmissions in their test cases.