Name 1:
Name 2:

COMPUTER NETWORKING

LAB EXERCISES (TP) 5
CONGESTION CONTROL; TCP, UDP
With Solutions

Friday, November 20, 2015

Abstract

In this lab session, you will explore in a virtual environment the effect of the congestion control mechanism of TCP and compare with a situation without congestion control. You will see what types of fairness are achieved by this congestion control mechanism. You will observe that a congestion control mechanism is also essential to avoid congestion collapse[1].

1 ORGANIZATION OF THE LAB EXERCISE

During this lab, you will use the virtual environment that you setup in the previous labs to study various aspects of congestion control. The whole lab can be done on your own machine or using the PCs in INF019.

1.1 REPORT

This document will be your report (one per group). Type the answers directly in the PDF document. Use Adobe Reader XI, as it supports saving forms. When you finish, upload the report on Moodle. Do not forget to write your names on the first page of the report. **The deadline is Wednesday, December 2nd, 23:59pm.**

• This lab is designed to be done in order. Each section uses knowledge from the previous section.

1.2 PREPARING YOUR VIRTUAL ENVIRONMENT

For this lab, we will make use of 6 virtual PCs: 3 of them will serve as regular PCs and 3 as routers. We need a few configuration steps before we start.

[1] In fact, this is the primary role of a congestion control algorithm. See *Congestion Avoidance and Control.* Van Jacobson and Michael J. Karels. ACM SIGCOMM 1988

1.2.1 Router Machines

Take one of the virtual machines from the previous labs and install two packages:
`iproute2` and `linux-sched`.
To this end, in VirtualBox, you should configure the first networking card to access the network via "NAT" (Right-click on the PC → Settings→ network →Enable network adapter 1 → Attach to: choose NAT) and start the PC from VirtualBox directly. Then, use "Package Manager" to install the required packages.

Clone[2] this machine 3 times, give the 4 machines names: Router, Router2, Router3, and PC1. When cloning, remember to reinitialize the MAC address and to remove the file
`/etc/udev/rules.d/70-persistent-net.rules`.

1.2.2 Regular PCs

On the machine PC1, you also need to download an archive that contains the programs for this lab. To do so,

1. Download the file `lab5.tgz` file from Moodle (http://moodle.epfl.ch/mod/resource/view.php?id=844753) and copy it on PC1[3]
2. Start the virtual PC and uncompress the archive:

```
# tar xvf lab5.tgz
```

   This will create a folder `lab5` that contains three folders. The folders `lab5/tcp/` and `lab5/udp/` contain tcp and udp clients and servers that we will use to measure the throughput of each flow. The folder `lab5/modules/` contains a Linux kernel module that we will use to emulate delay.
   Make the programs executable by going in the directory `lab5` (by typing `cd lab5`) and typing:

```
chmod +x tcp/tcpclient tcp/tcpserver udp/udpclient udp/udpserver
```

   Note: All folders contain binary programs as well as the source code. These binaries should work with a x86 slitaz distribution and you should not need to recompile them. If you want to (or need to) recompile them, you will probably need to install a few packages, including `gcc`, `make`, and `linux-module-headers`. Then, each program can be compiled by typing `make` in its own directory.
3. Last, we need to modify the congestion control mechanism. Slitaz, as many Linux distribution, uses CUBIC[4] as the default congestion control mechanism. In this lab, we will force TCP to use the RENO congestion control algorithm. To do so, edit the file `/etc/init.d/local.sh` and add the following line at its end:

```
echo reno > /proc/sys/net/ipv4/tcp_congestion_control
```

   (If you copy and paste this line, make sure that the "underscore" is pasted properly.) The congestion control mechanism will be set to RENO *after* the next reboot.
   Note: on a Linux machine, you can test which congestion control mechanism is used by typing in a terminal: `cat /proc/sys/net/ipv4/tcp_congestion_control`. You can also change

[2] Alternatively, instead of cloning, you can install the required packages on every machine you will use.
[3] There are two solutions to do so:

(a) download on your machine and use the shared folder that you configured in the first lab
(b) In virtualbox, configure the first networking card to access the network via "NAT" (configuration $->$ network $->$ Activate card 1 $->$ Access mode: choose NAT) and download the file directly in the virtual PC.

[4] *CUBIC: a new TCP-friendly high-speed TCP variant*. S Ha, I Rhee, L Xu. ACM SIGOPS 2008

the congestion control mechanism *until* the next reboot by typing in a terminal: `echo reno > /proc/sys/net/ipv4/tcp_congestion_control`. The script `/etc/init.d/local.sh` is executed at each boot. Therefore, this will set the congestion control to RENO at each boot.

When you are done with PC1, clone it two times. When cloning, remember to reinitialize the MAC address and to remove the file
`/etc/udev/rules.d/70-persistent-net.rules`.
Give the 2 new machines names: PC2 and PC3. Alternatively, instead of cloning, you can install the required packages on every machine you will use.

2 TCP VS UDP FLOWS

In GNS3, create a topology with 3 PCs and one router and wire them as in Figure 1.

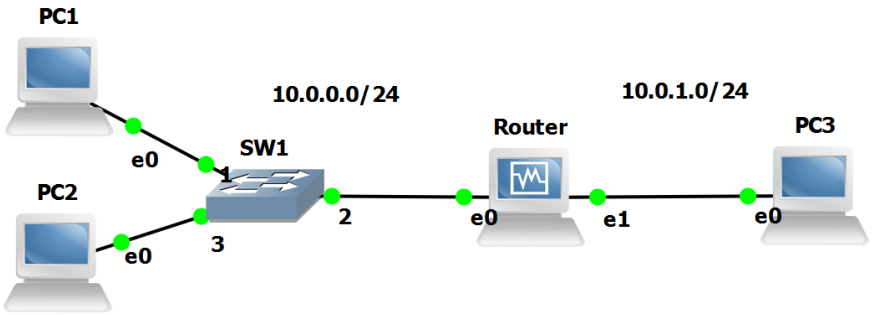


Figure 1: Initial configuration with 3 PCs and one router.

Configure the routing tables and the IP addresses according to the following addressing scheme:

- The subnet of PC1, PC2 and the router is 10.0.0.0/24. The addresses of PC1, PC2 and the router are 10.0.0.1, 10.0.0.2 and 10.0.0.10.
- The subnet of PC3 and the router is 10.0.1.0/24. The addresses of PC3 and the router are 10.0.1.3 and 10.0.1.10.

For the router, you can use `ifconfig/ip` commands or `quagga`. Test your configuration by pinging different PCs.

2.1 TESTING THE CONNECTIVITY WITH BASIC UDP AND TCP CLIENTS/SERVERS

The directory `lab5/udp` contains two programs: `udpserver` and `udpclient`. Their usage is:

```
# ./udpserver PORT
# ./udpclient IP_SERVER PORT RATE
```

For the server, `PORT` is the port number on which the server listens. For the client `IP_SERVER` and `PORT` are the IP address and port of the machine to which the packets are sent and `RATE` is the rate at which the client sends data (in kilobits per seconds). The client sends packets of size 125 bytes if the rate is lower than 50kbps and of size 1000 bytes otherwise.

The output of the UDP client has the following format:

```
4.0s - sent:  503 pkts, 1000.0 kbits/s
5.0s - sent:  629 pkts, 1000.6 kbits/s
```

5.0 is the number of seconds since the launching time of the client, 629 is the total number of packets sent by the client and 1000.6 is sending rate during the last second (in kilobits per second).

The output of the UDP server has the following format:

```
169.5s - received:  723/ sent:  741 pkts (loss 2.429%), 959.6 kbit/s
170.5s - received:  843/ sent:  867 pkts (loss 2.768%), 957.7 kbit/s
```

170.5 is the time since the launching of the server, 843 and 867 are the total number of packets sent by the client and received by the server, 2.768 is the percentage of packets that were lost and 957.7 is the rate at which packets were received during the last second.

Start a UDP server on PC3 that listens on port 10000.

QQQ1/ Launch a UDP client on PC1 that sends data to this server at rate 100kbps. What are the loss probability and the throughput that you observe on PC3?

QQQ2/ Repeat the operation with 1Mbps, 10Mbps, 100Mbps and 1Gbps. What are the throughputs and loss probabilities? At which rate the loss probability is greater than 1%? Explain the results.

The directory `lab5/tcp` contains two programs: `tcpserver` and `tcpclient`. Their usage is similar to `udpserver` and `tcpserver`, except that we do not specify a rate to the client: the client has an unlimited amount of data to send and uses TCP congestion control algorithm to control at which rate it sends the data to the server.

```
# ./tcpserver PORT
# ./tcpclient IP_SERVER PORT
```

The output of a TCP client looks like this:

```
6.3:  854.0kbps avg ( 944.5[inst], 926.5[mov.avg]) cwnd  9 rtt 83.9ms
7.3:  862.4kbps avg ( 914.6[inst], 925.3[mov.avg]) cwnd  9 rtt 86.8ms
```

7.3 is the time, 862.4 is the average rate of the client: the total amount of data that was successfully transfered by the client divided by the total time. 914.6 is the instananeous rate (approximately over the last second) and 925.3 is a moving average of this value. The value 9 is the congestion window of the TCP connection and 86.8 is the RTT measured by the TCP congestion control algorithm.

Start a TCP server on PC3 that listens on port 10001.

QQQ3/ Launch a TCP client on PC1 that sends data to this server. What is the throughput of the connection?

2.1.1 REMARKS ON THE PROGRAMS UDPCLIENT, TCPCLIENT, UDPSERVER AND TCPSERVER

The folders `lab5/tcp/` and `lab5/udp/` contain the executable and the source code of the programs. They are written in `C` and if you look at their source code, you will understand why we encourage `python` and not `C` for Lab2. Here are a few remarks on how they work:

•**For each UDP flow, you need one UDP client and one UDP server.** Explanation: each packet sent by a client contains its sequence number (the first packet contains the label "1", the second "2",...) and a lot of "0" to reach a size of 1000 bytes or 125 bytes. The loss probability printed by the server is one minus the ratio between the number of packet received divided by the largest sequence number received. Because of this implementation, the loss probability printed by the server is wrong if two clients talk to the same server.

- **For TCP, one server can handle mutiple clients**. The server creates one thread per accepted connection.
- **Before each experiment, kill all clients (TCP and UDP)** (you can do that by pressing "Control-C" in the terminal of the client.) This will reset the average values printed by the clients. In theory, you can keep the server running but killing them and relaunching them will not harm.
- **The printed rates correspond to application data.** They count the amount of data that was transfered by the TCP/UDP client to the TCP/UDP server. They do not take into account headers.
- **For all experiments, you have to wait until the printed values stabilize**. This is particularly important for TCP. The rate at which TCP sends packets depends on the losses that occur at random. Thus, to obtain deterministic values, you should wait for the average rate to be stable (2 minutes is probably OK for most scenarios). We also encourage you to run the experiments multiple times.
- **Throughput**. The throughput of a flow is the rate of *application data* that is successfully transmited. For the theoretical questions, you should take into account that the packets also contain header.
- **Units**. In all your answers, indicate in which unit your result is expressed (Mbps, kbps, %, ...).

2.2 Artificial limitation of the bandwidth of the router

In order to reproduce experiments where the performance is limited by the network capacities, we will limit the bandwidth of some interfaces. To do so, we will use linux tools provided by the `iproute2` and `linux-sched` packages to create virtual queues in which packets that match a given pattern will be stored. We will be using *class based queueing* (CBQ) scheduler for this purpose. We recommend to read some tutorials to understand how the CBQ works, in addition to reading the basic instructions below. One recommended source to start with is: http://lartc.org/howto/lartc.ratelimit.single.html.

On the router machine, run the following commands:

```
# tc qdisc add dev eth1 root handle 1:  cbq avpkt 1000 bandwidth 60mbit
# tc class add dev eth1 parent 1:  classid 1:1 cbq rate 2mbit allot
1500 prio 5 bounded isolated
# tc filter add dev eth1 protocol ip parent 1:  prio 16 u32 match ip
src 0/0 flowid 1:1
```

The first line installs a class based queue on `eth1`, and tells the kernel that for calculations, it can be assumed to be a 60Mbit interface. The exact value is less important as we will be limiting the rate below 60Mbit per second. The second line creates a class, with ID $1:1$, which has 2Mbit per second limit on the bandwidth allocated to it (and some other default parameters). The third line tells which traffic should go to the shaped class. In particular, it says that all packets going out of `eth1` will be stored in the class $1:1$. For more information see: http://linux.die.net/man/8/tc-cbq or the tutorial in http://lartc.org/howto/lartc.qdisc.classful.html.

In case you need to change the configuration of the CBQ, you first need to delete the previous configuration. Do so by using the command `tc qdisc del dev eth1 root`.

2.2.1 UDP TEST

When the RATE at which the UDP client sends data is greater that 50kbps, the client sends packets that contain 1000 bytes of data each.

QQQ4/ What is the size (in bytes) of Ethernet frames that you expect will be used to send the data of the UDP client? Verify with wireshark at the server side.

Solution:
UDP 8 bytes, IP 20 bytes, Ethernet 14 => total 1042 bytes.

QQQ5/ The router is the bottleneck and has a limit of 2Mbps. What is the maximum theoretical aggregate application data throughput that can be achieved? Explain how you compute.

Solution:
Max total rate = 2Mbps
*Max data tp (goodput) = 2*1000/1042 Mbps = 1919.38 kbit/sec*

To see if your configuration works, enable PF with the above configuration file. Then, start a UDP server on PC3 that listens on port 10000.

QQQ6/ Start a UDP client on PC1 that sends data at rate 100 kbps. What are the loss probability and the throughput observed on PC3?

QQQ7/ Repeat the operation with 2Mbps and 10 Mbps. What are the throughputs and loss probabilities? What do you observe (explain)? Compare to the theoretical throughput computed above.

2.2.2 TCP TEST

QQQ8/ What is the size (in bytes) of Ethernet frames that you expect will be used to send the data by the TCP connection? Verify with wireshark at the server side.

Solution:
Due to the MTU, data = 1448 bytes. ehternet frame size including overheads = 1514 bytes.

QQQ9/ What is then the maximum theoretical aggregate application data throughput that can be achieved? Explain how you compute.

Solution:
Max total rate = 2Mbps
*Max data tp (goodput) = 2*1448/1514 Mbps = 1912.8 kbit/sec*

Start a TCP server on PC3 that listens on port 10001.

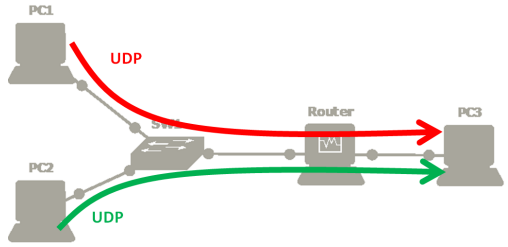
QQQ10/ Launch a TCP client on PC1 that sends data to this server. What is the throughput of the connection? Compare to the theoretical throughput computed above.

Remark: the limit of 2Mb/s is only for the packets that go out of the interface `eth1` of the router. Therefore, the packets sent from PC3 to PC1 do not interfere with the 2Mb/s limit.

2.3 COMPETING UDP FLOWS

We now want to explore what happens when two UDP flows are competing for the same bottleneck. The router has a capacity 2Mb/s and is the bottleneck. We consider the following scenarios:

- PC1 is streaming real-time data (*e.g.*, coming from a PMU) at rate 0.6Mb/s to PC3 using UDP.
- PC2 is streaming a video to PC3 using UDP. Depending on the quality, PC2 sends at rate 0.6Mb/s, 1.4Mb/s or 3Mb/s.



Scenario	PC1 (UDP)	PC2 (UDP)
A1	0.6 Mb/s	0.6 Mb/s
A2	0.6 Mb/s	1.4 Mb/s
A3	0.6 Mb/s	3 Mb/s

Before doing measurement, we want to predict the amount of data that will be sent and received in the three scenarios (denoted A1, A2 and A3).

QQQ11/ Based on a theoretical analysis, what should be the throughputs and loss probabilities of PC1 and PC2 in the scenarios A1, A2 and A3. Explain your method below and fill in the table.

Solution:
X = rate of PC1, Y = rate of PC2
*TP1 = [X*2/(X + Y)]*[1000/*1042], TP2 = [Y*2/(X + Y)]*[1000/*1042]*

	Throughput PC1		Loss probability PC1		Throughput PC2		Loss probability PC2	
Q11a/ (A1)	[11a(i)]		[11a(ii)]		[11a(iii)]		[11a(iv)]	
Q11b/ (A2)	[11b(i)]		[11b(ii)]		[11b(iii)]		[11b(iv)]	
Q11c/ (A3)	[11c(i)]		[11c(ii)]		[11c(iii)]		[11c(iv)]	

We now want to verify our analysis by emulation. As before, we use `PF` to limit the bandwidth of interface `eth1` of the router to 2Mb/s. For each scenario, start two UDP servers on PC3 that listen on ports 10001 and 10002. Then, run a UDP client on PC1 that sends data to PC3 at 0.6Mb/s and a UDP client on PC2 that sends data to PC3 at rate 0.6, 1.4 or 3Mb/s.

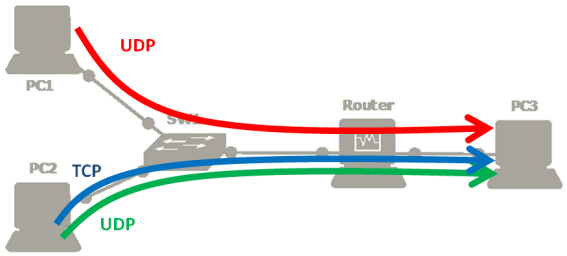
Q12/ What are the measured throughputs and loss probabilities in scenarios A1, A2 and A3?

	Throughput PC1		Loss probability PC1		Throughput PC2		Loss probability PC2	
Q12a/ (A1)	[12a(i)]		[12a(ii)]		[12a(iii)]		[12a(iv)]	
Q12b/ (A2)	[12b(i)]		[12b(ii)]		[12b(iii)]		[12b(iv)]	
Q12c/ (A3)	[12c(i)]		[12c(ii)]		[12c(iii)]		[12c(iv)]	

Q12.d/ Do you see a difference between your theoretical analysis and the experiment? If yes, comment on the difference, and try to explain the possible sources. 3em

2.4 TCP FLOWS COMPETING WITH UDP FLOWS

We consider a similar scenario as in the previous case. PC1 streams PMU data at rate 0.6Mb/s to PC3 and PC2 streams video data to PC3 at rate 0.6Mb/s, 1.4Mb/s or 3Mb/s. In addition to this traffic, PC2 is also using a TCP connection to send a software update to PC3.



Scenario	PC1 (UDP)	PC2 (UDP)
B1	0.6 Mb/s	0.6 Mb/s
B2	0.6 Mb/s	1.4 Mb/s
B3	0.6 Mb/s	3 Mb/s

QQQ13/ Based on a theoretical analysis, what should be the throughputs of the UDP flow of PC1, the TCP flow of PC2 and the UDP flow PC2 in the scenarios B1, B2 and B3 (explain bellow and fill in the table)?

Solution:

X = rate of UDP1, Y = rate of UDP2, Z = rate of TCP2, Z = 2 - X - Y.

*TP-UDP1 = X*1000/1042, TP-UDP2 = Y*1000/1042, TP-TCP = Z*1448/1514*

	UDP flow of PC1		UDP flow of PC2		TCP flow (PC2)	
Q13a/ (B1)	[13a(i)]		[13a(ii)]		[13a(iii)]	
Q13b/ (B2)	[13b(i)]		[13b(ii)]		[13b(iii)]	
Q13c/ (B3)	[13c(i)]		[13c(ii)]		[13c(iii)]	

We now want to verify our analysis by emulation. As in the previous case, use `PF` to limit the bandwidth to 2Mb/s and start two UDP servers on PC3. Start also a TCP server on PC3.

Q14/ What are the measured throughputs of the three flows in scenarios B1, B2 and B3?

	UDP flow of PC1		UDP flow of PC2		TCP flow (PC2)	
Q14a/ (B1)	[14a(i)]		[14a(ii)]		[14a(iii)]	
Q14b/ (B2)	[14b(i)]		[14b(ii)]		[14b(iii)]	
Q14c/ (B3)	[14c(i)]		[14c(ii)]		[14c(iii)]	

Q14.d/ Do you see a difference between your theoretical analysis and the experiment? If yes, comment on the difference, and try to explain the possible sources. 3em

3 TCP: FAIRNESS AND INFLUENCE OF RTT

The congestion control algorithm of TCP guarantees that the network resources are shared among the different connections. In this part, we will explore how the RENO algorithm shares the bandwidth when one or multiple bottlenecks are present in the network. In particular, we will investigate two characteristics: RENO provides a fairness *per flow* and is sensitive to delay.

For Sections 3.1 and 3.2, we will reuse the GNS3 setting of Figure 1. Also, `PF` should be set to limit the bandwidth of the router to 2Mbps (use the same configuration file as in Section 2.2).

In this part in particular, it is important to wait until the printed throughputs stabilize. To speedup the convergence, it is **very** recommended to close all the unnecessary programs on your computer. Especially, you should close the programs that may perform things on background (such as web-browsers, Dropbox synchronization, other virtual machines, etc). In any case, you should wait around 5 minutes to see the stable results.

3.1 ADDING DELAY TO AN INTERFACE

To obtain more realistic and more reproducible experiments, we will add delay in the network. To do so, we will use the module `netem` of the software *traffic control* that exists in Linux. To activate `netem`, we first need to load the corresponding kernel module `sch_netem.ko`. Then, we can use the command `tc` to add a rule in order to delay packets on this interface (see `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem` for more information about `tc` and `netem`).

For example, the following command adds 200ms of delay to all packets going out of the interface `eth0`:

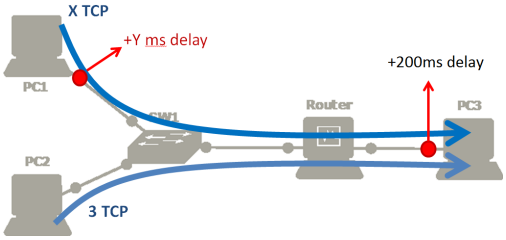
```
# insmod lab5/modules/sch_netem.ko
# tc qdisc add dev eth0 root netem delay 200ms
```

This rule can be changed by typing `tc qdisc change dev eth0 root netem delay 300ms` or deleted by typing `tc qdisc del dev eth0 root`.

QQQ15/ Add 200ms of delay to the interface `eth0` of PC3. Ping PC1 from PC3. What is the observed RTT?

Remark: if you flush the ARP table (for example, by doing `ifconfig eth0 down && ifconfig eth0 up`) and you reconfigure `netem` to add 200ms, the RTT of the first packet should be larger than the RTT of the second packet, because of the ARP request.

3.2 FAIRNESS BETWEEN TCP CONNECTIONS AND DELAY



Scenario	X (# conn. on PC1)	Delay Y
D1	1	0ms
D2	3	100ms
D3	3	200ms

TCP provides a fair sharing of the bandwidth at the flow level. Therefore, a machine that opens several TCP connections will obtain more bandwidth. To verify that, we will use the setup D1:

- There is an additional delay of 200ms on the interface `eth0` of PC3 but none on PC1 or PC2.
- PC1 opens one TCP connection to PC3 and PC2 opens three TCP connections to PC3.

QQQ16/ Using a theoretical analysis, what is the total throughput that PC1 and PC2 will get in scenario D1?

Solution:
PC1: 478.2, PC2: 1434.6
(if completely symmetric without retransmissions)

Run the 3 TCP clients on PC2 and one TCP client on PC1. Wait until the rates stabilize.

QQQ17/ What are the aggregate throughputs obtained by PC1 and by PC2 in scenario D1?

Q17.a/ Does this correspond to your theoretical analysis and if there is a difference, can you explain why? 3em

Q17.b/ Can you tell if there is any queuing delay?3em

We now explore scenario D2 and D3, where PC1 and PC2 both open 3 TCP connections to PC3.

Assume that the RTT is 200ms for the connections coming from PC2 and $(200 + Y)$ms for the connections coming from PC1.

QQQ18/ In theory, what is the throughput that PC1 and PC2 will get as a function of Y?

Solution:
*TP2 = (200 + Y)*TP1/200, TP1 + TP2 = 1448*2/1514*

Numerically, what are these values when:

	Total throughput for PC1		Total throughput for PC2	
Q18a/ (D1) $Y = 100ms$	[18a(i)]		[18a(ii)]	
Q18b/ (D2) $Y = 200ms$	[18b(i)]		[18b(ii)]	

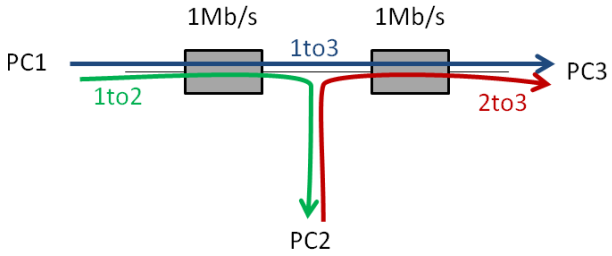
Now, run the simulation corresponding to scenarios D2 and D3:

Q19/ What are the measured aggregate throughputs obtained by PC1 and by PC2.

	Total throughput for PC1		Total throughput for PC2	
Q19a/ (D1)	[19a(i)]		[19a(ii)]	
Q19b/ (D2)	[19b(i)]		[19b(ii)]	

Q19.c/ Does this correspond to your theoretical analysis? If there is a difference, can you explain why? 3em

3.3 FAIRNESS OF TCP CONNECTIONS TRAVERSING MULTIPLE BOTTLENECKS



In this part, your goal is to study how the available bandwidth is shared when one TCP connection traverses two bottlenecks that are shared with two TCP connections that each share one of the bottlenecks.

The notion of fairness is difficult. A rate allocation is always a trade-off between maximizing the total rates sent by the connection or trying to equalize the rates of all users. For example, in this scenario, the flow *1to3* uses twice more resources than the flows *1to2* and *2to3*. Thus, the bigger the traffic *1to3* is, the lower the aggregate throughput can be.

3.3.1 THEORETICAL ANALYSIS

We first perform a theoretical analysis to compute two *fair* allocations corresponding to this network.

Q20/ Using a theoretical analysis:

Q20.a/ What is the max-min fair allocation that corresponds to this network (explain)?3em

Solution:
0.5 to all, by waterfilling.

Q20.b/ What is the proportionally fair allocation (explain)?3em

Solution:
*1/3 to 1to3 and 2/3 to 1to2 and 2to3, by solving: max(log(x) + 2*log(1 - x)) s.t.* $0 < x < 1$.

3.3.2 EXPERIMENTAL SETTING

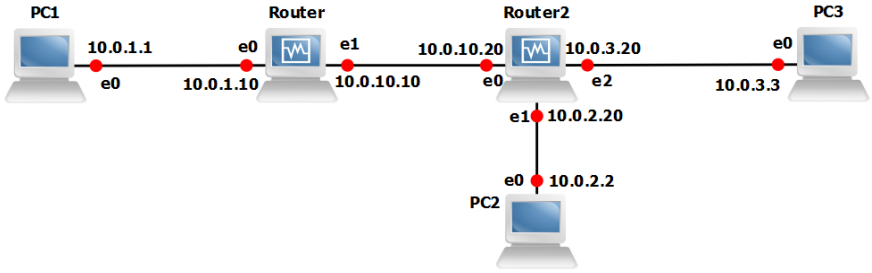
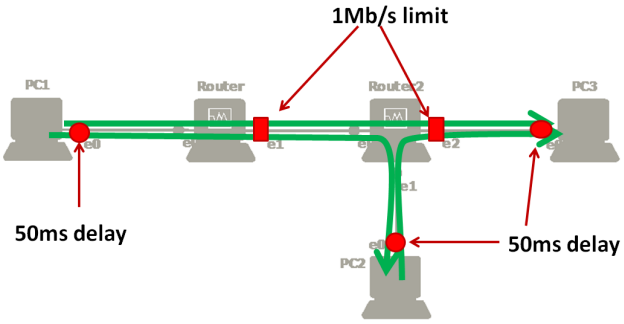


Figure 2: Fairness of TCP connections traversing multiple bottlenecks: wiring and addressing scheme.

We now want to explore what is the allocation provided by TCP.

Create a new GNS3 configuration and wire your PCs according to Figure 2 (note: as in Lab 4, router 2 needs to have 3 interfaces available). Configure the interface of each router and PCs as in Figure 2 and setup the routing tables. You can use a static configuration or quagga.

Configure the delay and bandwidth as follows:



As in Section 2.2 of this lab, use `PF` to limit the interfaces `eth1` of router1 and `eth2` of router2 to 1Mb/s.

As in Section 3.1 of this lab, use `tc` to add 50ms of delay to the outgoing packets of interfaces `eth0` of PC1, PC2 and PC3.

QQQ21/ Assume that there is no queuing delay. In theory, what should be the RTT of the connections *1to2*, *2to3* and *1to3*?

Now, start a TCP server on PC2 and on PC3. On PC1, open two TCP connections to PC2 and two TCP connections to PC3. On PC2, open two TCP connection to PC3. Wait until the rates stabilize.

Q22/ What is the measured throughput of the three connections?

[A22.a] *1to2*

[A22.b] *1to3*

[A22.c] *2to3*

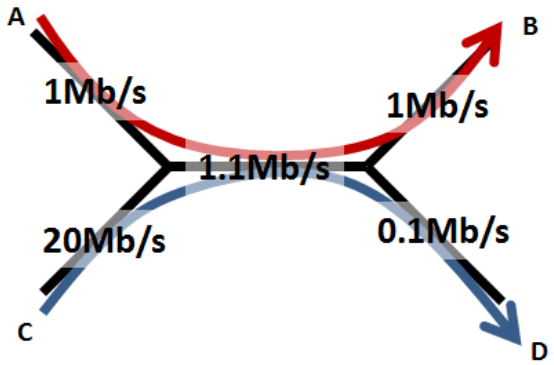
Q22.d/ Does this corresponds to your theoretical analysis?3em

Q22.e/ What is the average RTT of all three connections? Can you estimate the queuing delay on router1 and router2? 3em

Now, modify the delay on PC2 to 200ms and repeat the same experiment.

QQQ23/ Is the throughput of *1to3* bigger or smaller than the ones of *1to2* and *2to3* (comment the results)?

4 THE IMPORTANCE OF CONGESTION CONTROL



In this section, we will explore why having a congestion control mechanism is necessary. The system that we want to emulate is composed of five links depicted on the left. The capacities of the links range from 0.1Mb/s to 20Mb/s. There are two flows in this network:

- one flow that goes from A to B (in red),
- one flow that goes from C to D (in blue).

We will show evidence of a phenomenon called *congestion collapse*: the more aggressive C is, the smaller the total throughput will be.

4.1 THEORETICAL ANALYSIS

We first assume that there is no congestion control. The two senders, A and C, send data using UDP.

QQQ24/ If both sender A and sender C try to send data at maximum speed (i.e. 1Mb/s and 20Mb/s), what are the throughputs received by B and D? What are the loss probabilities of these two links?

Solution:
rateA + rateC = 1.1, rateA/rateC = 0.05 =¿ rateA = 0.05, rateC = 1.05
*rateAB = 0.05, rateCD = 0.1, TP = rate*1000/1042*

We now assume that sender A and sender C use a congestion control mechanism.

Q25/ What is the rate at which A and B will send data if we use

Q25.a/ a max-min fair allocation? 3em
Solution:
1, 0.1 by waterfilling.

Q25.b/ a proportionally fair allocation?3em
Solution:
same as max-min, e.g., by solving the optimization.

4.2 EXPERIMENTAL SETTING

We now want to verify these results in the virtual environment.

In GNS3, create a new topology according to Figure 3.

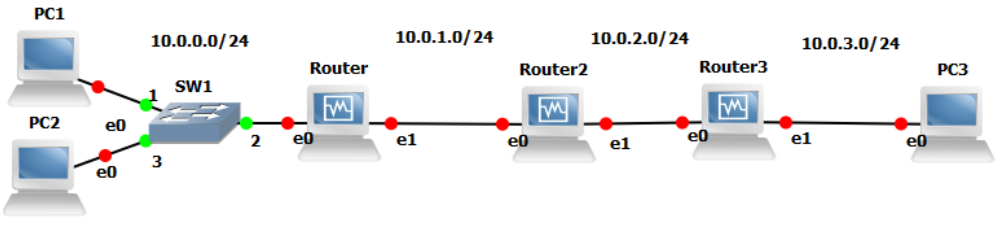


Figure 3: The GNS3 setting for scenario E.

The addressing scheme is as follows:

- The addresses of PC1, PC2 and PC3 end with 1, 2 and 3.
- The addresses of the routers 1, 2 and 3 end with 10, 20 and 30.

Configure the IP addresses of the PC and the routers. Configure the default gateways of the PCs and the routing tables of the router (for the routers, you can use RIP or a static routing table).

Try to ping PC1 from PC3 to verify that it works.

4.2.1 BANDWIDTH

To emulate the scenario, we will use the CBQ schedular to create:

- two queues on router1 that have bandwidths 1Mb/s and 10Mb/s and that filter packets from PC1 and PC2;
- one queue on router2 that has a bandwidth 1.1Mb/s.
- two queues on router3 that have bandwidths 1Mb/s and 0.1Mb/s.

An example of configuration for router1 is as follows:

```
# tc qdisc add dev eth1 root handle 1:  cbq avpkt 1000 bandwidth 60mbit
# tc class add dev eth1 parent 1:  classid 1:1 cbq rate 1mbit allot
1500 prio 5 bounded isolated
# tc class add dev eth1 parent 1:  classid 1:2 cbq rate 20mbit allot
1500 prio 5 bounded isolated
# tc filter add dev eth1 protocol ip parent 1:  prio 16 u32 match ip
src 10.0.0.1 flowid 1:1
# tc filter add dev eth1 protocol ip parent 1:  prio 16 u32 match ip
src 10.0.0.2 flowid 1:2
```

The configuration for router3 is similar and the configuration for router2 is similar to the one of Section 2.2.

Before applying a new configuration, don't forget to delete the previous one by using `tc qdisc del dev eth1 root`.

4.2.2 UDP

Launch two udp servers on PC3 that listen on port 10001 and 10002. Launch two UDP clients, one on PC1 and one on PC2 that send data at rate 1Mb/s on PC1 and 20Mb/s on PC2.

QQQ26/ What are the throughputs of the two flows? What are the loss probabilities?

Q26.a/ Do these values match your theoretical analysis (explain)? 3em

Now, launch two UDP clients, one on PC1 and one on PC2, that send data according to the max-min fair allocation that you computed before.

QQQ27/ What are the throughputs of the two flows? What are the loss probabilities?

Q27.a/ Do these values match your theoretical analysis (explain)? 3em

4.2.3 TCP

Repeat the same process using TCP connections instead of UDP.

QQQ28/ What are the throughputs of the two connections? What are the loss probabilities?

Q28.a/ Do these values match your theoretical analysis (explain)? 3em

QQQ29/ Can you conclude on what are the advantages of having a congestion control mechanism?