Lab – End-to-end transport

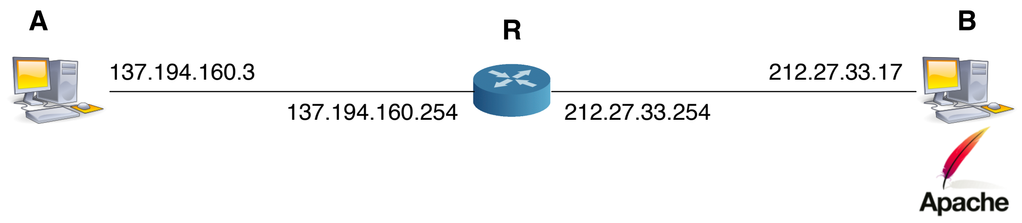
RES 841

October 2013

Introduction

This lab concerns end-to-end transport. You will experiment the behavior of transport protocols on a simple network. You will change a link characteristics and will notice the impact that different parameters have on the performance of a file transfer.

Launch the virtualized environment and get in the `TP_2` directory before launching the virtual network. The following network starts up:



1 TCP behavior

To evaluate TCP behavior, you will download on machine A a file located on a web server, hosted on machine B. You will capture the traffic on A with `tcpdump` and you will be able to view various TCP parameters with a tool called `tcptrace` from the host machine, the same way you used `wireshark` to view tcpdump output in the introduction lab.

1. Start capturing traffic on A using `tcpdump` and save the capture file, for example in `/hostlab/` or in `/hosthome/`.
2. From A, you will download, using HTTP the file `test.pdf` from machine B by typing :
   ```
   wget http://B/test.pdf.
   ```
   `wget` is a command line tool that mimics a browser behavior, uses HTTP and saves the file on the local disk. It will in addition display the average throughput when the transfer is finished.
3. Once the file transfer is complete, halt `tcpdump`, go back on the host machine to analyze the capture file. Wireshark gives you interesting tools to follow a TCP stream, from the statistics menu. Select "Statistics ¿ Flow Graph" to follow your file transfer (you can select TCP flow in the dialog box to de-clutter the output). You can note TCP's sequence numbers evolution in both directions and see control packets. Can you explain why it seems like only acknowledgments are transmitted? Try to see (generally) how the congestion window evolves from the number of successive packets sent in both directions.
4. Complementary to Wireshark, there is a command line tool called `tcptrace` which analyzes the TCP headers in detail. `tcptrace` usage is quite straightforward: `tcptrace <options> capture.pcap`. We recommend using the following options :

- Option `-n` de-activates name resolution (DNS) and displays raw IP addresses, which accelerates the display. Otherwise, tcptrace tries to replace IP addresses in the output by names, which can take some time (and yield weird results, the virtual IP addresses are fake).
- Option `-l` will display a very detailed report on the captured TCP connections. You may find a description of all the statistics available at the following address: : `http://www.tcptrace.org/manual/node8_mn.html`
- Option `-r` adds to the report statistics on the RTT (round-trip-time), used to compute TCP congestion window.
- Option `-W`, when used with `-l` allows to evaluate the congestion window value **explain why this value is an estimate and imagine what tcptrace does to estimate it**
- Option `-G` produces a set of files (XPL) that can be passed to a visualization application.

5. After you have tested the different outputs and explored the different statistics, generate the graphic outputs using the -G option. The files you obtain can be visualized with `xplot.org`. Be careful, you will have to use the `xplot.org` application and not `xplot`, this version being too limited for the `tcptrace` color output. This elementary tool allows you to zoom on a part of the graph (selecting the box you want) and to dezoom by double-clicking; You can examine the following graphics:
   - `b2a_tput.xpl` displays the throughput of the TCP flow from B to A (bytes/sec).
   - `b2a_owin.xpl` displays the estimated evolution of the congestion window from B to A.
   - `b2a_rtt.xpl` displays the evolution of the round-trip-time in milliseconds, from B to A.
   - `b2a_ssize.xpl` displays the evolution of the TCP segments size between B and A.
   - `b2a_tsg.xpl` dispays the sequence numbers evolution from B to A.
6. The virtual network you have been working with is not realistic. It has a very large bandwidth, a very low latency and a zero loss rate. Let us now examine TCP's behavior when these parameters evolve. To compare scenarios, save your results (capture file,s graphs, ...) using different names.

   You will have to modify interface `eth1` characteristics on router R (i.e. on the link between R and B). For this you will use a tool, called `tc` which is able to modify the traffic to introduce delay, to provoke losses, to reorder packets and to limit bandwidth on a link. This tool (or its adaptation to the virtualized environment which is called `orig-tc`) is used on R by typing the following command: `orig-tc qdisc add dev eth1 root netem <parameters>`. [1] You can play with the following parameters:
   - `orig-tc qdisc add dev eth1 root netem delay 10ms`: adds a 10 ms delay when passing through the interface
   - `orig-tc qdisc add dev eth1 root netem loss 5%`: provokes a 5 % frame loss rate when passing through the interface
   - `orig-tc qdisc add dev eth1 root netem duplicate 5%`: duplicates 5 % of the packets when passing through the interface
   - `orig-tc qdisc add dev eth1 root netem gap 5 delay 10ms`: introduces a 10 ms for one packet every 5 packets, therefore modifying the order the packets are emitted on the interface.
7. Test various values for the above parameters and observe the effects on TCP graphs. Between two tests, you will have to re-initialize the queueing discipline by typing `orig-tc qdisc del dev eth1 root`.Examine closely the RTT, the congestion window and the throughput and the relationship between these parameters.
8. What can you observe on the order of magnitude of network metrics? Is a 1 s delay important? And a 5 % loss rate?
9. How would UDP behave in these scenarios in your opinion?

[1]Typing this command, you indeed tell the system to add a queueing discipline (`qdisc`) to interface `eth1` (`dev eth1`). This discipline type is *network emulation* (`netem`). The `root` parameter refers to the tree-like organization of the queueing disciplines associated to an interface.

@