CSCI 558L FALL 2011 LAB#8

ANDREW GOODNEY - GOODNEY@USC.EDU - YOUNG CHO - YOUNGCHO@ISI.EDU

1. Improving TCP Performance Over Lossy Links

To date in this course we have examined the performance of TCP and UDP under several scenarios. In the UDP file transfer lab we developed a specific solution to transferring data over a high-latency lossy link. However, now we want to explore generic solutions to improve TCP performance so that any TCP application can take advantage our solution.

1.1. **Congestion vs. Lossy Links.** The acknowledgement and back-off mechanisms of TCP were developed to avoid congestion collapse [2]. However, generally TCP can not tell the difference between a packet lost due to congestion and a packet lost due to an unreliable link. The exponential back-off algorithm is the appropriate algorithm to deploy when TCP flows are sharing a congested link, however under lossy-link conditions, reducing the sending rate and window may not be the proper response.

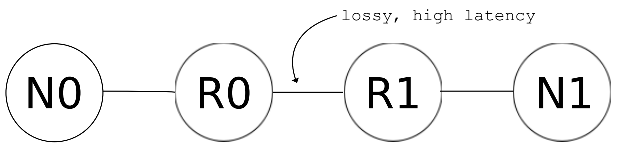

Figure 1. Lab 8 Topology

1.2. **Experimental Set-up.** The topology for this experiment is shown in figure 1. Nodes **n0** and **n1** need to exchange data using TCP. Nodes **r0** and **r1** are connected by a high latency, lossy link (think satellite, or long range wireless uplink).

2. Initial Experiments

- The goal for this lab is recognize that the packet drops in this network are from a lossy link and not from congestion, and thus improve TCP performance. The test for this lab will be to use the standard FTP program to move a 1GB (1024^3 byte) file from **n0** to **n1**
- Begin by writing a NS file for this experiment. You may use any OS you'd like, but you may need to consider how you'll go about implementing the changes (see below for more details). Set the link between the routers to 100MB/s with an initial delay of 10ms with no loss. Create a local filesystem using `mketrafs` and a 1GB file. Enable the FTP daemon on **n1** and use the standard FTP program to transfer the 1GB file. Record the performance obtained for your report.
- Explore the problem space by varying the delay and loss up to 200ms delay and 25% loss (both directions, so 400ms RTT and 50% total packet loss). What throughput can FTP achieve under these conditions? You may want to present this as a 3D graph, or one graph with multiple lines.

3. Improving TCP Performance

Now we need to consider ways to improve the performance of TCP under these conditions. First however, a note about how to go about the modifications. The easiest way to modify or implement your own TCP stack is to do it on Linux. There already exist a number of plug-able TCP implementations for Linux [3]. In order to use them you'll need to figure out how to build a new kernel on your chosen Linux distribution and include these modules. It should then be fairly simple to modify the modules to improve performance.

There are two general ideas we have as far as changes that can be made in order to improve TCP performance. You are not required to do either of these exactly, however we believe these are two areas that will be fruitful. You may combine these ideas, or come up with your own ideas by reading the related literature.

- **Removing exponential back-off** Start by reading the paper "Removing Exponential Backoff from TCP" [1]. Start with the standard TCP stack for Linux and remove the exponential backoff algorithm as described in the paper. Make sure to understand the *implicit packet conservation principal.*
- **Store and Forward at the router** Modify your user space router code to buffer the packets of the TCP connection. Keep track of the time-out value and if a packet is dropped (e.g. no ACK for that byte range is received before timeout) and resend the missing packet. At the same time, generate an ACK back to the sending host to keep the host from detecting the packet drop.

4. Evaluation

To evaluate your improvements to TCP perform the same experiment as above with FTP, except include your modified TCP stack and/or router. If you do more than one modification, make sure to test and evaluate each modification separately (and together) to see which change improves performance under what conditions. Again there is a competition aspect to this lab, extra credit will be based on the ranking of your improvement: (max. throughput with modification)/(baseline ftp speed).

References

[1] Amit Mondal and Aleksandar Kuzmanovic. Removing exponential backoff from tcp. *SIGCOMM Comput. Commun. Rev.*, 38:17–28, September 2008.

[2] John Nagle. Congestion control in ip/tcp internetworks. *SIGCOMM Comput. Commun. Rev.*, 14:11–17, October 1984.

[3] René Pfeiffer. Tcp and linux' pluggable congestion control algorithms.