CS 536 Lab Answers 6

Prashant Ravi — ravi18@purdue.edu

December 7, 2016

Problem 1

On analyzing the traffic generation application, there was not significant difference between one-hop or multi-hop . This is owed to the proximity of the machines that the overlay routers are set up on. However, by bombarding with multiple client requests , supposing a router, had a dense fan-out it increased the latency of the flow.

Problem 2

1 Overview of solution

To overcome the issue of repeated file I/O, I used an in-memory solution mmap that reads the entire file to memory on the server process. And a similar mmap is created on the client end. During the setup phase, the file size is communicated to the client via positive acknowledgment so it can know how many sequence numbers it can expect and initialize its NACK data structure accordingly, such as the NACK array and the mmap. Basically, I defer writing to disk until we are absolutely sure that the file has been fully received, owing to very high speeds for files such as 20MB.Obviously, my solution will suffer from super sized file sizes because there won't be enough heap space.

The design of the NACKs involves only sending them out after all the packets that the server wanted to send have been sent, and any holes in the NACK array are later requested to be filled from the server, cumulatively. Meaning I send out the whole NACK array to the server after it has completed its first pass through file and it responds. This goes on until the client has empty NACK array, at which point the client signals an ACK of the entire file.

I'm confident the utilization of mmap has significantly improved the speed of my file transfer program. I hope it holds a candle to the other designs.

2 Performance sample

Completion Time : 0.601000 seconds
File received

Filesize: 10485760
Recv size: 10485760
Total Nack packets: 5

Bonus Problem

My solution is inspired from Operating Systems where we set priorities to different flows and change these priorities based on how much bandwidth has already been used up, giving opportunity for other flows to catch up, aka prevent starvation. An algorithm such as the multi-level feedback queue would work well here. A good estimate of congestion can be made single transfer packet delay instead of packet dropping(which is used in TCP) to signify potential congestion in the system, so that the congestion window size can be halved, in case self congestion does take place.