5/3-17
358352

Estimating Bottleneck Bandwidth using TCP

June, 1998

Mark Allman
NASA Lewis Research Center
mallman@lerc.nasa.gov
http://gigahertz.lerc.nasa.gov/~mallman

Hit our satellite with feeling
Give the people what they paid for
–The Flaming Lips

Introduction

- Why do we want TCP to estimate the bottleneck bandwidth?
  - Startup more rapidly
  - Avoid loss
- We will concentrate on estimating the bottleneck bandwidth in order to set *ssthresh* to an appropriate value and thus avoid loss.
  - "Satellite friendly" TCP often includes large windows
  - Large windows can allow a TCP to overwhelm a gateway with a larger number of segments than a small window connection
  - Therefore, estimating the bottleneck bandwidth can help TCP limit loss by slowing down before overwhelming the gateway

Background

- Making an estimate of the bottleneck bandwidth has been proposed and tested via simulation (Janey Hoe at MIT)
  - used packet-pair algorithm on the first few returning ACKs
  - the time between successive ACKs is caused by the data segments "spreading out" based on the bandwidth
  - the bandwidth estimate combined with the measured RTT can be combined to give the appropriate delay-bandwidth product of the link and therefore, the correct window size

Theory vs. Practice

- However, real networks make predicting the bottleneck bandwidth more difficult
  - delayed ACKs
    - getting successive ACKs requires more segments
  - network jitter
    - traffic from other connections getting between two successive packets
  - asymmetric networks

Possible Packet-Pair Solutions

- So, how do we get around these problems caused by real networks?
  - watch the incoming ACKs for a longer period of time
- The larger the window grows, the more important it is to avoid loss (due to the possible magnitude of the loss event)
  - As the window increases and we get more segments into the network the problem of delayed ACKs naturally fades
  - Network jitter can be averaged out if we are able to watch the ACKs for "a while"
  - Asymmetric networks?
    - Hmmm...

Preliminary Results

- We collected packet-level traces from various networks and analyzed attempted to determine the bandwidth of the bottleneck link based on the ACK stream
- ACKS were observed in the order in which they arrived only, as to attempt to simulate a TCP stack

Preliminary Results: ACTS

- We ran several FTPs over the ACTS satellite and were able to successfully estimate the bottleneck bandwidth (and therefore, the appropriate window size) within the first 40 segments (data and ACK) observed.
- This environment was free from competing traffic so it is mostly uninteresting
  - But, it shows that delayed ACKs do not make the task impossible

Preliminary Results: Internet

- We ran several FTPs between LeRC and OU to obtain traces from a dynamic environment with competing traffic
- We were able to determine a "good" window size within the first 50 segments observed
  - Our estimate is 60% higher than the window size needed to obtain the throughput we observed, on average
    - window / RTT = bandwidth
  - Our estimate is 66% lower than the window size at which loss occurred, on average
- Therefore, we hypothesize that a TCP using this algorithm would perform just as well, if not better than a TCP without bandwidth estimation (on average)

Preliminary Results: Internet (cont.)

- But, sometimes the estimate is not all that "good"
  - If the estimate is too low, we terminate slow start too soon and then depend on congestion avoidance to provide window growth
    - Slow!
  - Estimating too high is not as big a deal as we can do no worse than current implementations and possibly avoid *some* loss, even when we overestimate

Future Work

- Refine algorithms used to average the inter-ACK space
- Test in a hybrid terrestrial/satellite network
- Other mechanisms can be investigated once we have a good estimate of the bandwidth