Improving TCP Start-up Behavior in High-Speed Networks

Ren Wang, Giovanni Pau, M.Y. Sanadidi, and Mario Gerla
Computer Science Department, University of California, Los Angeles
Los Angeles, CA 90095, USA
{renwang, gpau,medy,gerla}@cs.ucla.edu

ABSTRACT

TCP is a reliable data transfer protocol used widely over the Internet for numerous applications, from FTP to HTTP. The current implementation of TCP Reno/NewReno mainly includes two phases: Slow-start and Congestion-avoidance. In Slow-start phase, the sender opens the congestion window (*cwnd*) exponentially, doubling *cwnd* every Round-trip Time (RTT) until it reaches the Slow-start Threshold (*ssthresh*). Then the connection switches to Congestion-avoidance phase, where *cwnd* grows linearly, 1 segment every RTT. The initial *ssthresh* is set to be an arbitrary value, ranging from 4K to 64K Bytes, depending on the O/S implementation.

Next generation high-speed networks with large bandwidth and long propagation delays pose a major challenge to TCP performance, especially during the startup period. By setting the initial *ssthresh* to an arbitrary value, TCP may suffer from two problems:

(a) If it is set too high compared to the path Bandwidth Delay Product (BDP), the exponential increase generates too many packets too fast, causing multiple losses at the bottleneck router resulting in a coarse Timeout;

(b) If the initial *ssthresh* is too low, the connection exits Slow-start and switches to linear *cwnd* increase prematurely, resulting in poor start-up utilization especially when the Bandwidth Delay Product (BDP) is large.

Recent studies reveal that a majority of TCP connections are short-lived (mice), while a smaller number of long-lived connections carry most of the Internet traffic (elephants). A short-lived connection usually terminates even before it reaches 'steady state'. That is, before window size grows to make good utilization of the path bandwidth. Thus, the start-up stage can significantly affect the delay performance of the mice. In a large bandwidth delay network, with the current Slow-start scheme, it takes many RTTs for a TCP connection to reach the ideal window (=BDP). For example, in a current Reno implementation with initial *ssthresh* set to 32 Kbytes, a TCP connection takes about 100 sec to reach the ideal window over a path with a bottleneck bandwidth of 100 Mbps and RTT of 100ms. The utilization in the first 10 sec is a meager 5.97%. Thus, the rapid development of the high-speed networks and their ever-growing BDP, demand a better and more efficient Slow-start mechanism to achieve good link-utilization.

In this presentation, we briefly review previous work on initial *ssthresh* setting. We evaluate via simulation experiments the performance of three current TCP implementations: (1) Reno/NewReno, (2) NewReno with Hoe's modification, and (3) Vegas; all in high-speed networks scenarios. We show that both Reno and Vegas are unable to achieve high utilization in such networks due to premature slow-start termination. Hoe's modification estimates the bottleneck bandwidth and sets the initial *ssthresh* accordingly, yielding good utilization in ideal network conditions. However, when the bottleneck buffer size is relatively small comparing to BDP, or other traffic joins in during the slow-start phase, Hoe's modification may cause multiple losses and very low utilization.

We propose a sender-side only modification, called Adaptive Start (Astart), to improve TCP startup performance. AStart takes advantage of the Eligible Rate Estimation (ERE) mechanism proposed in TCP Westwood (TCPW). ERE relies on an adaptive estimation technique applied to the ACK stream. ERE has been shown to provide high total link utilization while maintaining fairness and friendliness. When a connection initially begins or re-starts after a coarse timeout, Astart adaptively and repeatedly resets *ssthresh* when ERE indicates that there is more available capacity. By adapting to network conditions in the start-up phase, the sender is able to grow the congestion window (*cwnd*) **fast** without incurring the risk of buffer overflow and **multiple losses**.

Ns-2 simulation experiments show that AStart is able to significantly improve the link utilization under various bandwidth, buffer size and round-trip time conditions, avoiding both under-utilization due to premature termination of the Slow-start, and multiple losses due to setting the *ssthresh* too high initially, or increasing *cwnd* too fast.

Keywords — Congestion control, Slow-start, Eligible rate estimation, Large bandwidth delay networks