

DYNAMIC RIGHT-SIZING

Research & Development in Advanced Network Technology (RADIANT) Team

Los Alamos National Laboratory

<http://www.lanl.gov/radiant/>

HPDC01: August 7-9, 2001

Motivation:

TCP (particularly the ‘Reno’ variant) is the Internet’s ubiquitous transport protocol. Unfortunately, the design of its flow and congestion control mechanisms, leads to abysmal performance over high-bandwidth or high-delay links. This means that the performance of application infrastructures built on TCP, such as computational grids or high-volume web servers, are crippled. Dynamic right-sizing addresses the first of these problems, while other work addresses the second.

Today, acceptable performance is attained only via manual optimization of buffer sizes. While such tuning can increase throughput by an order of magnitude, it requires tedious point-specific kernel configuration changes on each host and any firewalls between those hosts; changes that can easily back-fire. We propose a straightforward modification to TCP that automatically and transparently addresses the above problems while maintaining connection semantics and the ubiquitously deployed features of TCP.

Algorithm:

TCP’s effective transmission window ($ewnd$) is the minimum of the flow control window ($fwnd$) and congestion control ($cwnd$). While the $cwnd$ varies dynamically as network state changes, $fwnd$ has always been static. Ideally, $fwnd$ should vary with the bandwidth-delay product of the network and $ewnd$ should be limited only by network congestion. This is true when $fwnd \geq cwnd \rightarrow ewnd = cwnd$, and memory utilization is most efficient $fwnd = cwnd$. This approach removes the $fwnd$ limitation currently detracting from performance.

- Let
 - $awnd \equiv$ receiver’s advertised window. This is used to set $fwnd$ at the sender.
 - $cwnd \equiv$ sender’s congestion window.
 - $maxwnd \equiv$ maximum amount of memory that any one connection can utilize.
- At the source (sender) set:
 - $fwnd = \min(cwnd, awnd, maxwnd)$.
 - This is *actual* memory, and can be reduced after packets are acknowledged.
- At the destination (receiver) set:
 - $awnd \geq 2 \times cwnd$ if sender in slow start,
 - $awnd \geq cwnd + 1$ if sender in additive increase.
 - This is *potential* memory, and generally cannot be reduced after being advertised.

COMPARISON

	Stock	Tuned	Dynamic
Flow-Control Buffer	64KB	Arbitrary, Fixed	Arbitrary, Dynamic
Configuration	Minimal	Significant	Minimal
Point-Specific	Implicitly	Yes	No
Response to Changing Network Conditions	None	None	Optimizes <i>fund</i>
Bandwidth Achievable	Poor	Poor to Excellent	Good to Very Good

Flow-Control Buffer sizes change as required given network conditions with Dynamic Right-Sizing (DRS), while Stock TCP is limited to 64KB, and Tuned TCP is limited to some predetermined value.

Configuration of Stock and DRS connections is minimal (usually limited to turning on window scaling and turning off Nagel's algorithm). Tuning TCP connections requires much more effort and coordination between administrative domains.

Point-Specific and Unresponsive solutions are provided by Stock and Tuned TCP connections, that are optimized for a specific pair of hosts and a specific set of network conditions. Connections between other pairs of hosts or under slightly different network conditions do not benefit. DRS flows do not suffer from these problems.

Bandwidth Achieved by Stock TCP is poor, due to its many limitations in high-bandwidth or high-delay networks. Tuned TCP connections perform no better than stock connections in most circumstances, but can achieve nearly 100% utilization of a network given an omniscient administrator and sufficient time to perform configuration. DRS connections perform better than Stock connections or Tuned connections in the general case, giving better bandwidth with lower memory utilization. We recognize that properly tuned connections can achieve better performance in limited circumstances— thus we leave the ability to tune connections in implementations of DRS for these cases.

PERFORMANCE

Values assume a 100ms round-trip time, and are even more marked with larger RTTs.

- **Stock TCP**

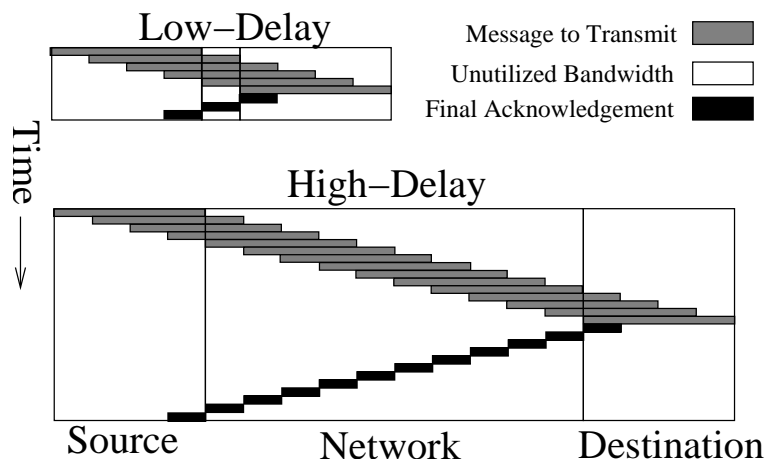
- 64 KB buffer → only 64KB extant per round-trip time.
- 100 ms round-trip time
- Implies *maximal* bandwidth of $64KB/100ms = 5.24Mbps$:

- **Manually Tuned or Dynamically Right-Sized TCP**

- Arbitrary buffer; set large enough to fully utilize network.
- $100ms \times 1Gbps \rightarrow 11.92MB$ required.
- Check via stock utilization: $64KB/11.92MB \approx 0.524\%$ of $1000Mb = 5.24Mb$

Thus tuned or DRS flows can theoretically fully utilize this high-delay link if they allocate 12MB buffers. Yet they do not– why?

- Application-level traffic patterns:



- Reno's AIMD congestion control:

- Adds one packet per RTT, cuts to half on a loss.
- A cut in half requires over 7 minutes to reconverge to optimal.
- Multiple loss more likely with large windows– leading to massive cuts.

- Linux's 2.2 features:

- TCP heuristics tuned for local area networks.
- No zero-copy stack.

- Hardware issues

- Not using jumbo packets (9K versus 1.5K packets)
- Not fully utilizing NIC hardware (dual on-board processor, memory)

- Our implementation

- Methodology sound, implementation sometimes insufficiently aggressive.
- Usually 5-7 times better than stock TCP, but not as good as tuned TCP.