TCP Measurements

Jiri Hamberg

University of Helsinki

jiri.hamberg@cs.helsinki.fi

15.12.2016

Overview

Introduction

- Why should we measure TCP?
  - TCP is the backbone of the Internet
  - TCP needs to keep up with evolving Internet
  - Analytical models are hard

Metrics

- Throughput (and goodput)
- Fairness
- Latency
- Loss Rate
- Burstiness

Throughput and Goodput

Throughput: the amount of data transferred in a unit of time

Goodput: the amount of payload transferred in a unit of time

Fairness

Intuitively, fairness means that resources are distributed evenly

Jain et al. (1984)

$$f(\mathbf{x}) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}$$

- Scale independent
- Bounded
- Continuous

Other Metrics

Latency

Loss Rate

Burstiness

Measurement Platforms

Measurement platforms offer different ways to set up experiments

- Testbed
- Live Internet Test
- Simulation
- Emulation

Testbed

Testbed is an isolated network where experiments can be conducted

- Provides total control over the experiment - no interference
- Does not scale well

Live Internet Test

Use a mesh geographically distributed hosts for conducting measurements

- Can measure on $O(n^2)$ network paths with n hosts
- Inexpensive and realistic results
- Hard to control and measure the network paths

Simulation

Use a scriptable general purpose simulation framework or write your own

- Cheap, fast and scalable
- Lose some details compared to conventional measurements
- Results may require validation with real experiments

Emulation

Combination of simulation with other techniques

- Simulate part of the network that cannot easily included in the experiment or does not exist yet

Evaluating Popular Congestion Control Algorithms

The study used ns2 network simulations to evaluate goodput and fairness common TCP congestion control algorithms

Algorithms evaluated

- New Reno
- Compound
- Cubic

Wired and wireless connections were simulated

New Reno

Default congestion control implementation of some versions of OSX

TCP Reno implements the fast recovery algorithm

- 3 duplicate ACKs trigger a re-transmission

New Reno improves the fast recovery algorithm by being able to handle multiple packet drops in a single congestion window with only one window reduction

Compound

Introduced in Windows Vista and Windows Server 2008

Maintains exponentially smoothed estimate of the RTT

- Estimate is used in calculating the current window size during TCP congestion avoidance
- Congestion window consists of two components: the regular congestion window and a RTT based component
- Conjoined effect is approximately

$$win(t+1) = win(t) + \alpha * win(t)^k$$

when no packet loss or early congestion is detected, and

$$win(t+1) = win(t) * (1 - \beta)$$

otherwise

Cubic

The default congestion control implementation of Linux kernel from version 2.6.19 onwards

Cubic replaces slow start and congestion avoidance phase with a single phase where the congestion window is given by the following formula

$$W_{cubic}(t) = C(t - K)^3 + W_{max}$$

where $K = (W_{max}\beta/C)^{1/3}$, W_{max} is the largest window size before last window reduction and C and β are free parameters.

Wired Simulation: Goodput

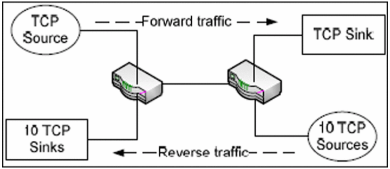


Figure : Topology of the wired ns2 simulation

	Compound	Cubic	New Reno	Rev. Traffic
0s-250s Mbps	1.99	1.99	1.98	No
250s-500s Mbps	1.79	1.96	1.71	Yes
500s-750s Mbps	2.00	2.00	1.99	No
750s-1000s Mbps	1.80	1.96	1.76	Yes

Table : Goodput achieved by each TCP variant with and without reverse traffic at different time intervals.

Wired Simulation: Fairness

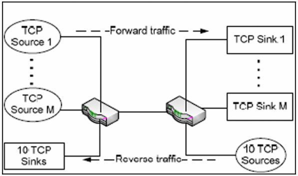


Figure : Network topology in measuring fairness

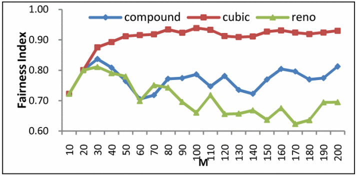


Figure : Fairness of TCP variants as a function of concurrent senders

Wireless Simulation

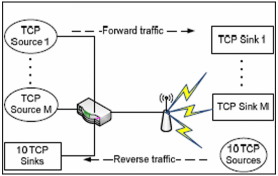
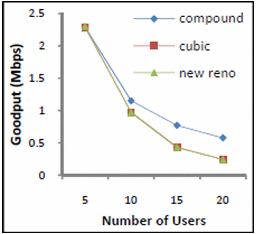
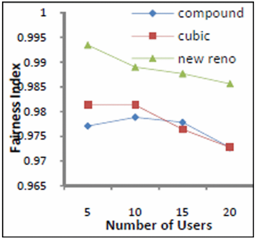


Figure : Topology of the wireless ns2 simulation



(a) Goodput



(b) Fairness

Improving TCP Startup Performance Using Active Measurements

Study introduces a novel TCP startup algorithm, *paced start*

- Estimates the available bandwidth using *Packet Transmission Rate* (PTR) method

Paced start is compared with the *slow start*

- Simulations with ns2
- Measurements on the Emulab testbed

Paced Start

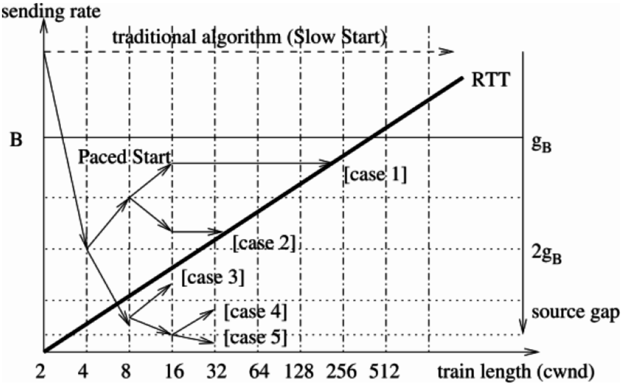


Figure : Slow start algorithm moves along the X-axis until a suitably large cwnd is found (or ssthresh is hit). Paced start also considers the Y-axis, doing a binary search to find the suitable sending rate

Paced Start and Slow Start

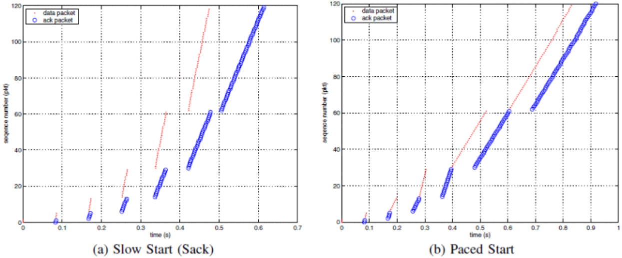


Figure : Packet trains of source and target machines plotted against time for both slow start and paced start algorithm (simulation)

Evaluation Using Simulation

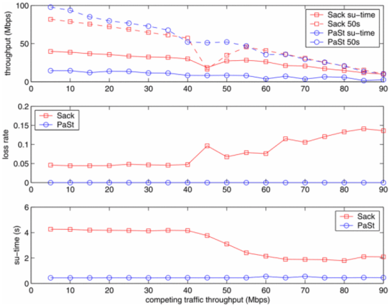


Figure : The effect of competing traffic on the performance of slow start (Sack) and paced start (PaSt) algorithms. ns2 network simulation.

Evaluation UsingTestbed

TCP Measurements

Jiri Hamberg

Introduction

Metrics

Measurement Platforms

Evaluating Congestion Control Algorithms

Improving TCP Startup Performance Using Active Measurements

Conclusion

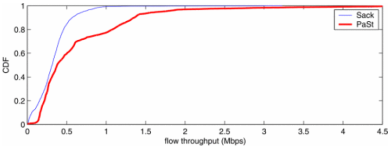


Figure : Cumulative distribution functions of flow throughput during a 500 second experiment conducted on a Emulab testbed

Conclusion

- Measurements are useful for evaluating and improving the TCP protocol
- Useful metrics include goodput, fairness and latency
- Various ways to set up an experiment, each with pros and cons

The End

Thank you!
Questions?