15-712 Project Report: An Active Middlebox for Classifying Congestion Control

Sruti Bhagavatula (sbhagava), Anson Kahng (akahng), Ranysha Ware (rware)

December 13, 2017

Introduction

In 2006, Senator Ted Stevens infamously described the Internet as a "series of tubes", where "filled" tubes delayed his email. While Ted was largely criticized for his simplistic view of the Internet, his characterization of the Internet was not wrong:

> "It's a series of tubes. And if you don't understand, those tubes can be filled and if they are filled, when you put your message in it, it gets in line and it's going to be delayed by anyone that puts into that tube enormous amounts of material, enormous amounts of material."

The more technical term for what Ted is referring to is congestion. Logical "tubes" are created between two processes across a network by transport protocols. These tubes run over physical links between these two processes that may become filled by competing traffic. When the network becomes congested, packets are delayed and throughput degrades. If congestion goes unchecked, the network can become so slow it grinds to a halt, a condition called congestion collapse.

In the late 1980s, Van Jacobson saved the Internet from congestion collapse by adding congestion control to the de-facto reliable transport protocol, TCP [1]. While TCP has significantly evolved over the years, Jasobson's additive increase multiplicative increase (AIMD) algorithm is still a core part of TCP's congestion control mechanism.

AIMD effectively limits congestion and converges to an equal share of bandwidth between competing TCP flows with equal RTT [2], but it has several limitations. New approaches to congestion control have been proposed and deployed in the wide area to overcome TCP's limitations or for special use cases. These include TCP CUBIC in Linux [3], Google's BBR [4], Microsoft's Compound-TCP [5], LEDBAT in BitTorrent [6], and others.

The fairness properties of competing TCP flows using an AIMD congestion control algorithm are well-understood, but we know little about how this alphabet soup of alternatives all interact in wide area networking. Indeed, very little data exists today regarding which protocols are in use and with what popularity at all.

As a first step to understanding competition between heterogeous congestion control algorithms, we designed and implemented a testbed to instantiate TCP connections with remote servers and identify the running congestion control algorithm. However, doing so is non-trivial because congestion control algorithms don't tell you what they're running — they all look like TCP. So, we must indirectly identify these algorithms based on rate and reaction to drops. In what follows, we describe the design and implementation of our testbed and discuss the challenges we faced in its implementation: controlling drops at the bottleneck link (including diagnosing a bug in the BESS scheduler), determining the running congestion control algorithm based on its queueing behavior, and handling noise due to cross-traffic in the wide area. The latter challenge is the topic of our current work: while our testbed can accurately identify TCP algorithms in controlled circumstances, it still fails to identify congestion control algorithmss correctly in wide-area experiments across the live Internet.