University of Glasgow

Lowering Latency

Networked Systems (H)

Lecture 6

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow | This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Lecture Outline

- Key factor limiting performance is **latency**
  - Connection establishment latency → **previous lectures**
    - Reduce number of RTTs to improve connection setup time
    - Reduce number of connections needed
    - TCP + TLS 1.3 → QUIC
      - Overlapping connection establishment and security handshake
      - 0-RTT connection re-establishment with idempotent data
      - Stream multiplexing within a single connection
  - Data transfer latency
    - Impact of TCP congestion control
    - Impact of delay-based congestion control
    - Impact of explicit congestion notification
    - Impact of the speed of light

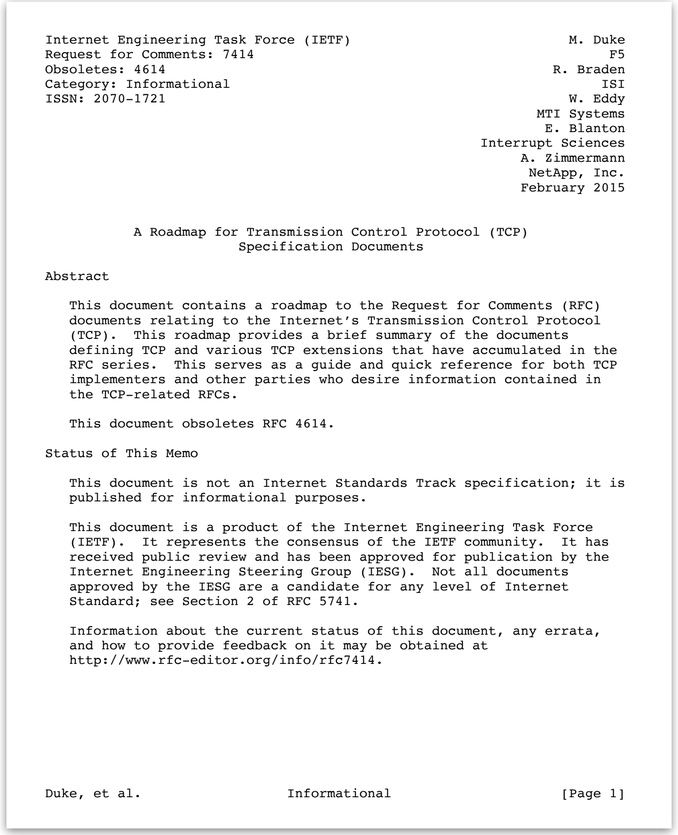
Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Congestion Control

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Congestion Control

- Congestion control principles
- Loss-based congestion control
  - How to effectively make use of network capacity
    - TCP Reno
    - TCP Cubic
- Delay-based congestion control
  - Reducing TCP-induced latency
    - TCP Vegas
    - TCP BBR
- Explicit congestion notification
- TCP is a complex, **highly optimised**, protocol
  - This lecture is a **very** simplified review of complex issues



Internet Engineering Task Force (IETF) — M. Duke
Request for Comments: 7414 — F5
Obsoletes: 4614 — R. Braden
Category: Informational — ISI
ISSN: 2070-1721 — W. Eddy
MTI Systems
E. Blanton
Interrupt Sciences
A. Zimmermann
NetApp, Inc.
February 2015

A Roadmap for Transmission Control Protocol (TCP)
Specification Documents

Abstract

This document contains a roadmap to the Request for Comments (RFC) documents relating to the Internet's Transmission Control Protocol (TCP). This roadmap provides a brief summary of the documents defining TCP and various TCP extensions that have accumulated in the RFC series. This serves as a guide and quick reference for both TCP implementers and other parties who desire information contained in the TCP-related RFCs.

This document obsoletes RFC 4614.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7414.

Duke, et al. — Informational — [Page 1]

M. Duke, R. Braden, W. Eddy, E. Blanton, and A. Zimmermann, "A Roadmap for TCP Specification Documents", IETF, RFC 7414, February 2015 (https://tools.ietf.org/html/rfc7414)

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

QUIC Congestion Control

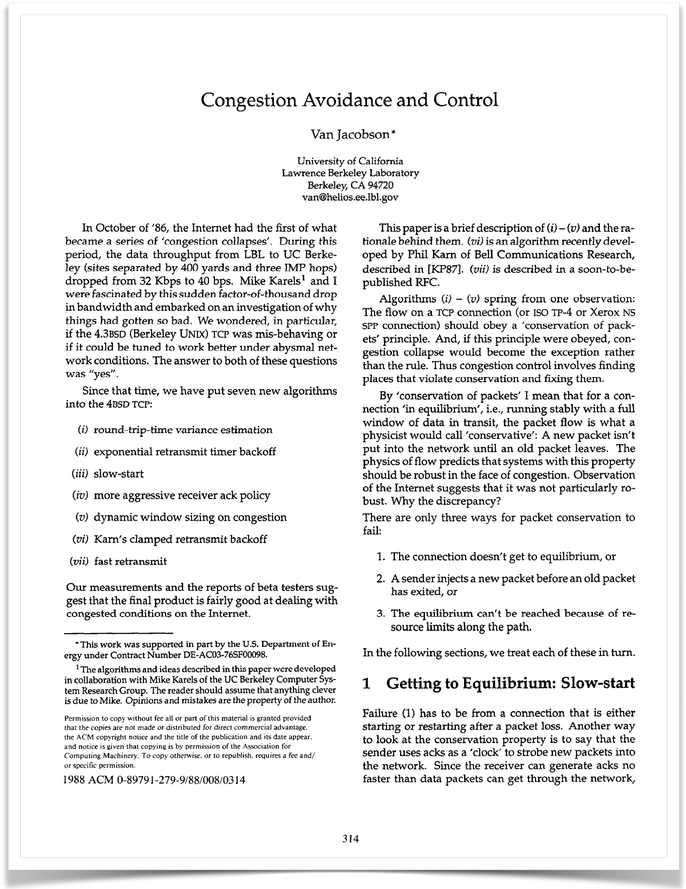
- What about QUIC?



- The QUIC v1 standard adopts congestion control as used in TCP Reno
- QUIC implementations tend to use Cubic or BBR congestion control in practice

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Congestion Control Principles



Congestion Avoidance and Control

Van Jacobson*

University of California
Lawrence Berkeley Laboratory
Berkeley, CA 94720
van@helios.ee.lbl.gov

In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and three IMP hops) dropped from 32 Kbps to 40 bps. Mike Karels[1] and I were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. We wondered, in particular, if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

Since that time, we have put seven new algorithms into the 4BSD TCP:

(i) round-trip-time variance estimation
(ii) exponential retransmit timer backoff
(iii) slow-start
(iv) more aggressive receiver ack policy
(v) dynamic window sizing on congestion
(vi) Karn's clamped retransmit backoff
(vii) fast retransmit

Our measurements and the reports of beta testers suggest that the final product is fairly good at dealing with congested conditions on the Internet.

This paper is a brief description of (i) – (v) and the rationale behind them. (vi) is an algorithm recently developed by Phil Karn of Bell Communications Research, described in [KP87]. (vii) is described in a soon-to-be-published RFC.

Algorithms (i) – (v) spring from one observation: The flow on a TCP connection (or ISO TP-4 or Xerox NS SPP connection) should obey a 'conservation of packets' principle. And, if this principle were obeyed, congestion collapse would become the exception rather than the rule. Thus congestion control involves finding places that violate conservation and fixing them.

By 'conservation of packets' I mean that for a connection 'in equilibrium', i.e., running stably with a full window of data in transit, the packet flow is what a physicist would call 'conservative': A new packet isn't put into the network until an old packet leaves. The physics of flow predicts that systems with this property should be robust in the face of congestion. Observation of the Internet suggests that it was not particularly robust. Why the discrepancy?

There are only three ways for packet conservation to fail:

1. The connection doesn't get to equilibrium, or
2. A sender injects a new packet before an old packet has exited, or
3. The equilibrium can't be reached because of resource limits along the path.

In the following sections, we treat each of these in turn.

1 Getting to Equilibrium: Slow-start

Failure (1) has to be from a connection that is either starting or restarting after a packet loss. Another way to look at the conservation property is to say that the sender uses acks as a 'clock' to strobe new packets into the network. Since the receiver can generate acks no faster than data packets can get through the network,

* This work was supported in part by the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

[1] The algorithms and ideas described in this paper were developed in collaboration with Mike Karels of the UC Berkeley Computer System Research Group. The reader should assume that anything clever is due to Mike. Opinions and mistakes are the property of the author.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

1988 ACM 0-89791-279-9/88/008/0314

V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM Conference, Stanford, CA, USA, August 1988.. http://dx.doi.org/10.1145/52324.52356

- Key congestion control principles:
  - Packet loss as a congestion signal
  - Conservation of packets in flight
  - Additive increase/multiplicative decrease



Source: PARC

Van Jacobson

- Practical TCP congestion control:
  - Robust IETF standards for TCP
  - Embodies congestion control principles
  - **Extremely high performance**



Source: Sally Floyd

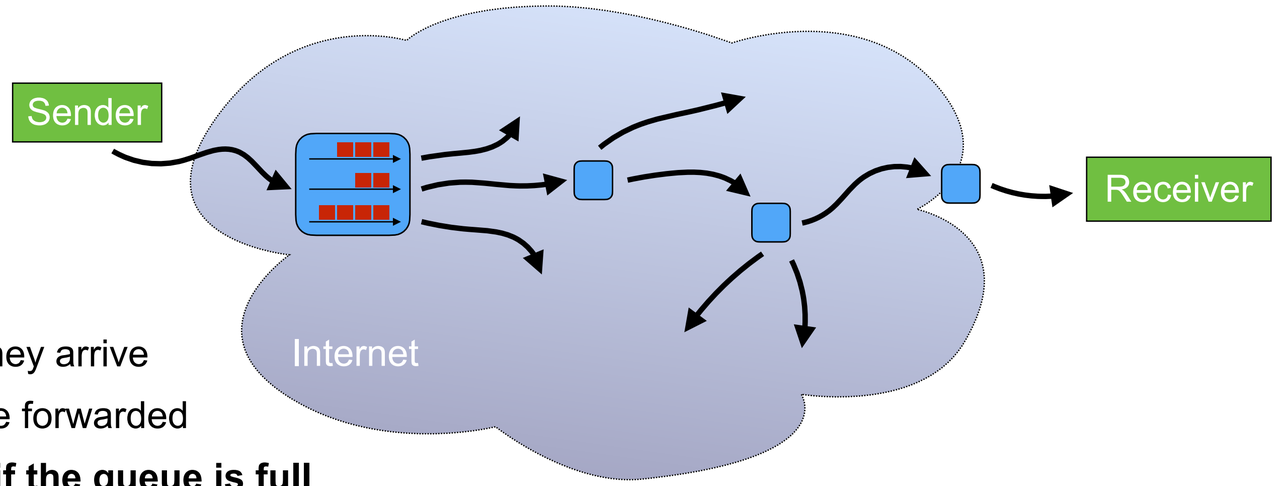
Sally Floyd

https://www.nytimes.com/2019/09/04/science/sally-floyd-dead.html

CC BY ND

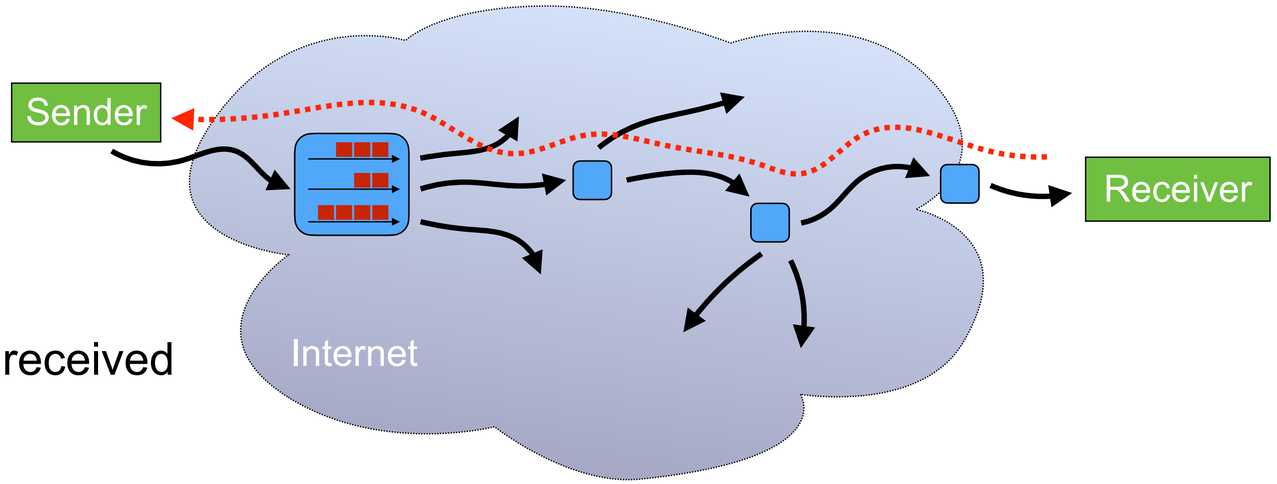
Key Principles: Packet Loss as a Congestion Signal

- Data flows from sender to receiver through a series of **IP routers**
- Routers perform two functions:
  - **Routing:** receive packets, determine appropriate route to destination
  - **Forwarding**: enqueue packets on outgoing link for delivery
    - Queues shrink if packets are forwarded faster than they arrive
    - Queues grow if packets arrive faster than they can be forwarded
    - Queues have maximum size → **packets discarded if the queue is full**
    - Congestion control can use this packet loss as a congestion signal



Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

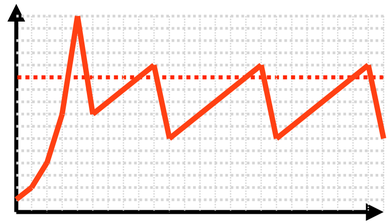
Key Principles: Conservation of Packets

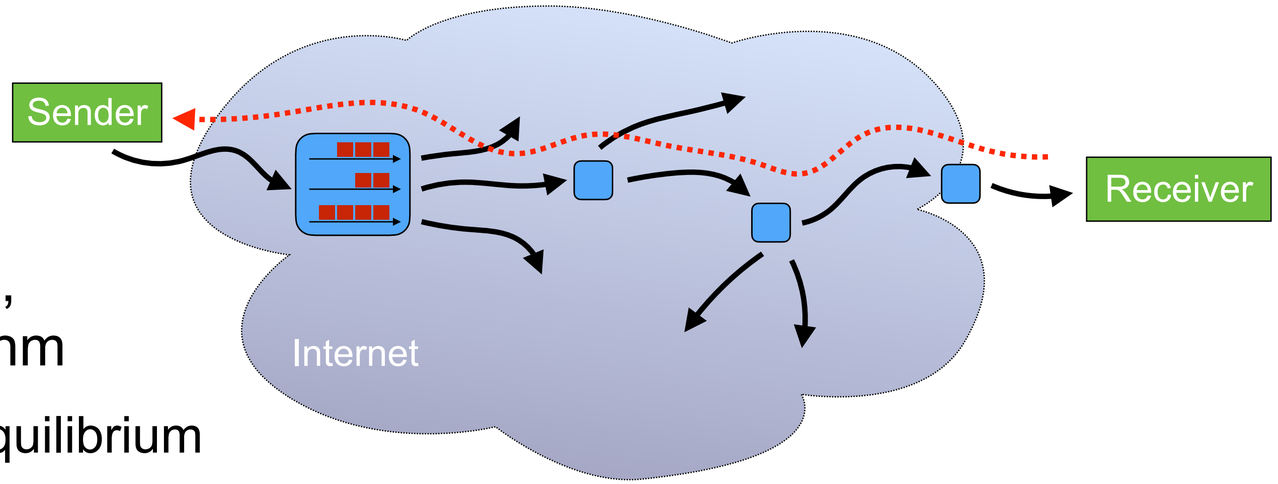


- Acknowledgements returned by receiver
  - Send one packet for each acknowledgement received
  - Total number of packets in transit is constant
    - System is in equilibrium; queues neither increase nor decrease in size
    - **ACK clocking** – each acknowledgement “clocks out” the next packet
- Automatically reduces sending rate if network gets congested and delivers packets more slowly

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Key Principles: AIMD Algorithms





- Sending rate follows additive increase, multiplicative decrease (AIMD) algorithm
  - Start slowly, increase gradually to find equilibrium
    - Add a small amount to the sending speed each time interval without loss
    - For a window-based algorithm $w_i = w_{i-1} + \alpha$ each RTT, where $\alpha = 1$ typically
  - Respond to congestion rapidly
    - Multiply sending window by some factor $\beta < 1$ each interval loss seen
    - For a window-based algorithm $w_i = w_{i-1} \times \beta$ each RTT, where $\beta = 1/2$ typically
  - Faster reduction than increase → stability; TCP Reno

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

TCP Congestion Control

- Packet loss as congestion signal
- Conservation of packets in equilibrium
- AIMD

BY ND