An Overview of SEARCH

Presented by: Jose Perez & Connor Tam

What is TCP congestion control & slow start?

TCP CONGESTION CONTROL

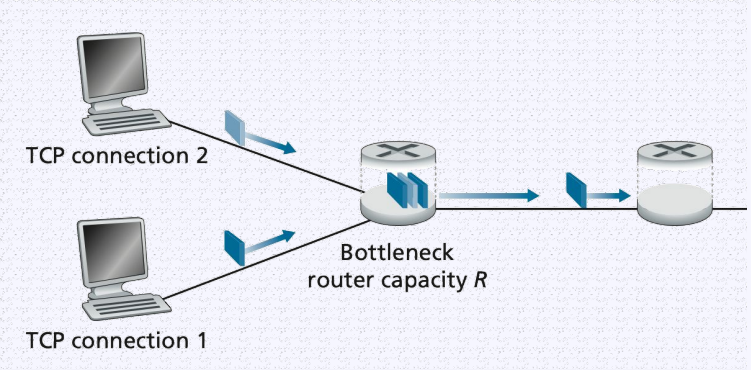
Too many packets for the network to handle leads to:

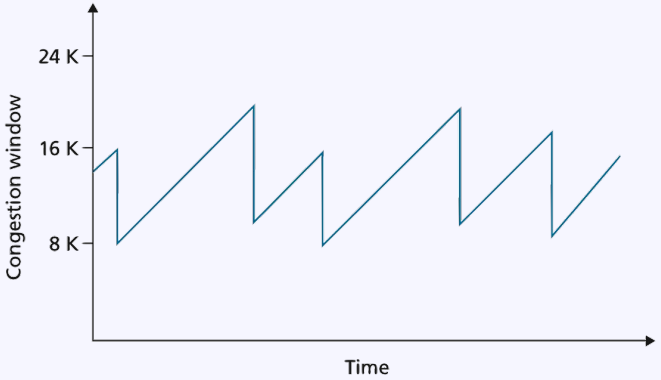
- Queue in switch buffers → long delays
- Overflow in switch buffers → packet loss

TCP end to end congestion detection: sender infers congestion from ACK's the receiver sends.

TCP congestion probing: TCP adjusts the transmission rate, that is, the number of data segments it sends (congestion window) by probing until a packet loss.

TCP AIMD: Additive increase, multiplicative decrease.



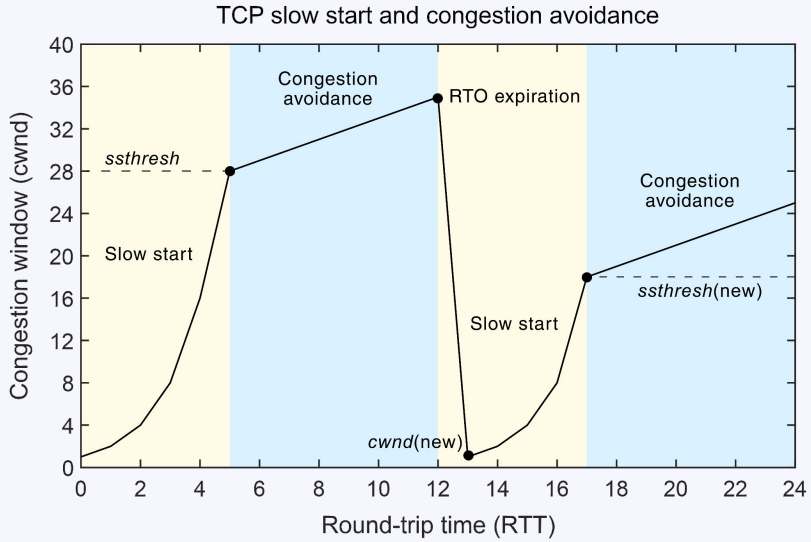


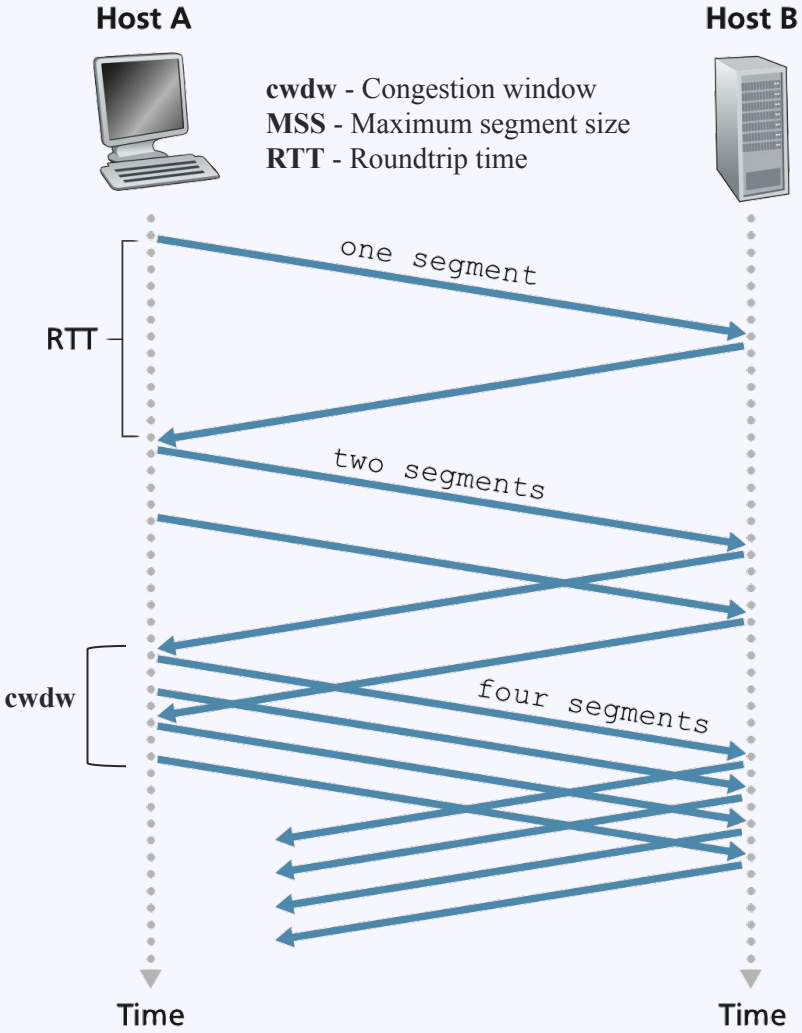
TCP CONGESTION STATE MACHINE

	SLOW START	CONGESTION AVOIDANCE	FAST RECOVERY
PURPOSE	Quickly discover network capacity and ramp up the congestion window size.	Gradually increase the congestion window to probe for available bandwidth without causing congestion.	Recover from packet loss without reducing the congestion window size too drastically.
MECHANISM	Exponentially increases the congestion window (cwnd) size for each ACK received until it reaches the slow start threshold (ssthresh).	Increases cwnd linearly, typically by one Maximum Segment Size (MSS) for each round-trip time (RTT) to avoid creating congestion. CUBIC*	Temporarily reduces cwnd upon detecting packet loss via triple duplicate ACKs, then inflates it for each duplicate ACK received during the recovery phase
TRANSITION TRIGGER	Transition to Congestion Avoidance when cwnd reaches ssthresh or on packet loss.	Remains in this state until packet loss is detected, indicating potential network congestion.	Exits this state and enters Congestion Avoidance upon receiving a non-duplicate ACK for the outstanding data.

STANDARD SLOW START

- Objective: Find the maximum bandwidth of the network quickly
- Initially: cwdw = 1 MSS
- Each ACK: cwdw = cwdw + 1 MSS
- Exponential transmission rate



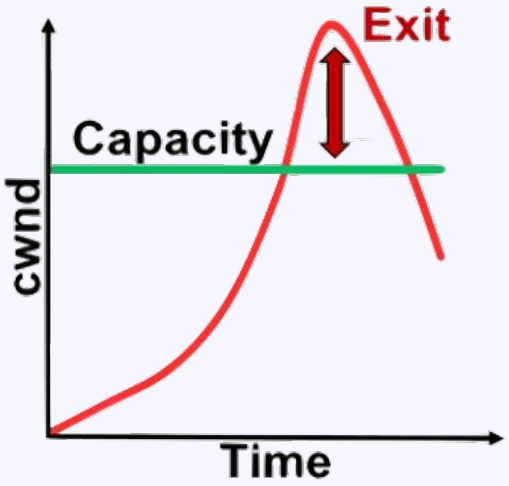


What’s wrong with the default slow start?

Without Hystart

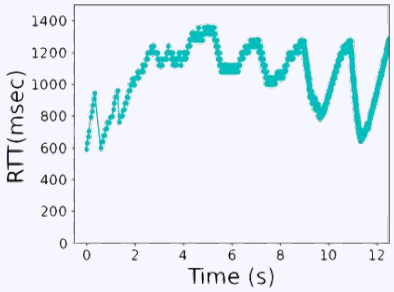
- Roundtrip segments double until... **first dropped acknowledgement**

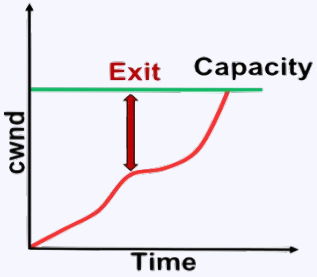
- Exits too late – packets need to be retransmitted
  - High RTT means more wasted data during retransmission (Satellites)



With Hystart

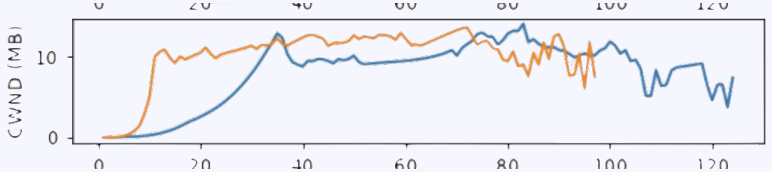
- Roundtrip segments double until…
  “Delay Increase Algorithm”
  - Reliant on RTT variations to determine congestion





- “Conservative Slow Start”
  - ccwd grows at a slower rate

- Exits too early – bandwidth not used to its full extent until later

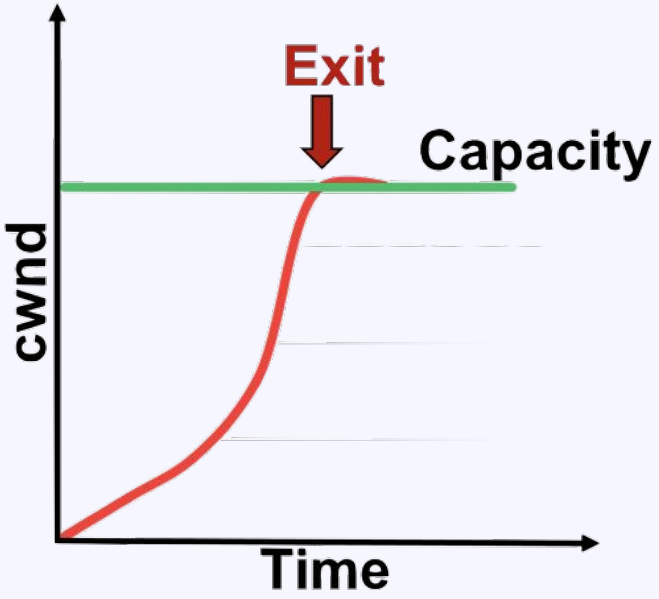


A SOLUTION: SEARCH (Exit on time)

- Instead of probing until a packet loss, predict when the first packet loss will occur

- Exit right before first loss and enter congestion avoidance

- Avoid loss

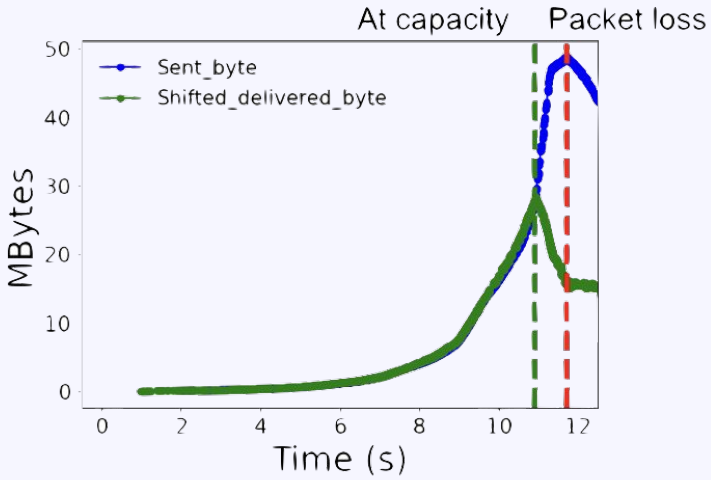


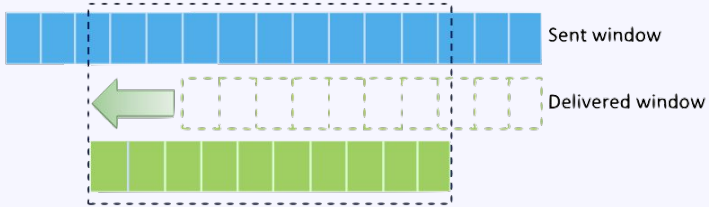
How does SEARCH work?

Recording Sent + Received Bytes

- The client and server keep track of **sent bytes and delivered bytes**

- **Sliding Window**
  - Bins of 3.5 * RTT size
  - Delivered window shifted back 1 RTT
    - Same timeframe as sent byte



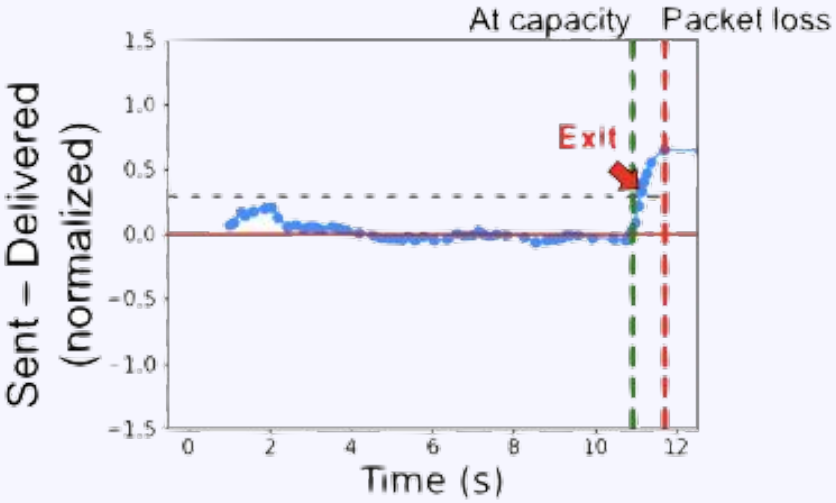


When To Exit

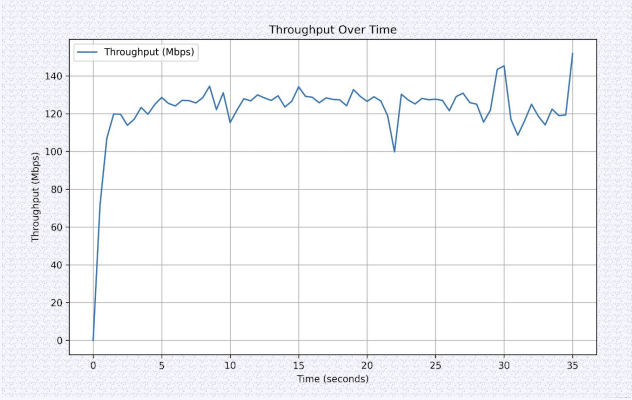
- Detect packet loss during a timeframe

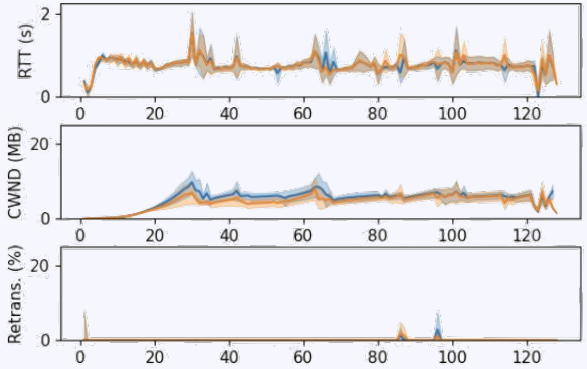
$$\text{packet loss } \% = \frac{bytes_{sent} - bytes_{received}}{bytes_{sent}}$$

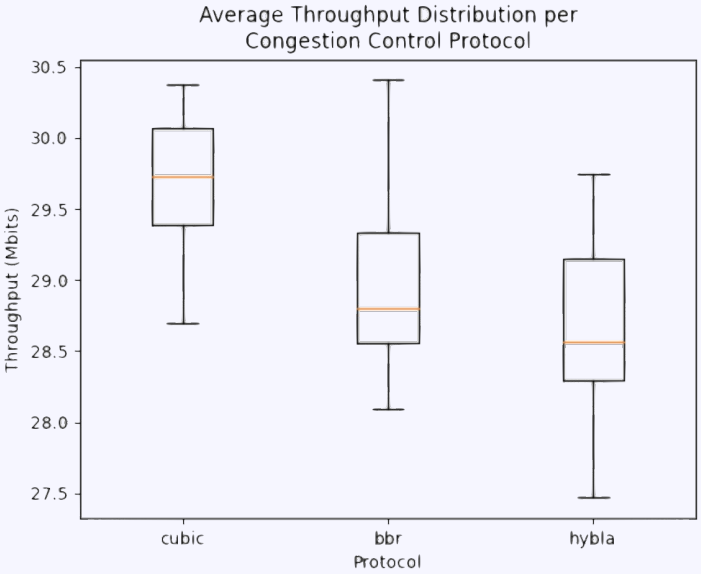
- If above threshold (25%), exit



EXPERIMENTS TO TEST SEARCH







SEARCH

Hi-Start

Cubic

TCP Congestion Control Algorithms over Satellite Network

Presented by: Katy Stuparu

https://web.cs.wpi.edu/~claypool/papers/tcp-compete-22/paper.pdf

Congestion Control Algorithms

TCP Cubic

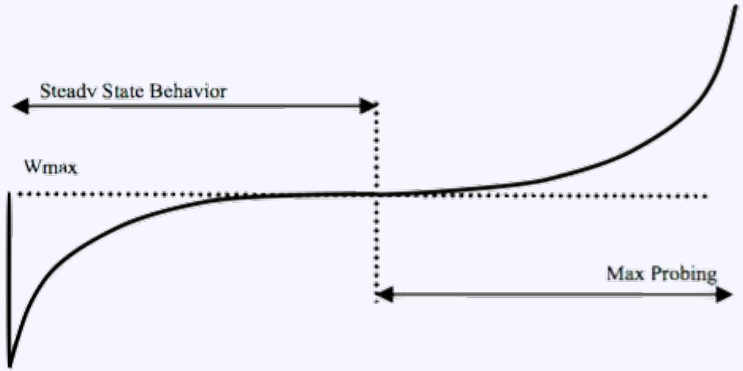
Default for most operating systems

Other Algorithms

TCP Hybla

TCP BBR

TCP PCC



CUBIC: A New TCP-Friendly High-Speed TCP Variant
https://dl.acm.org/doi/epdf/10.1145/1400097.1400105

Satellite Internet Networks

Can connect to remote areas where wired networks aren't available

Used in case of emergencies

Important part of modern network: 67% increase in # of satellites 2014-2019

High latency, minimum 550 milliseconds

https://www.iot-now.com/2022/10/13/124572-what-can-a-virtual-satellite-network-do-for-you/

How effective are congestion control algorithms when used with Cubic over satellite?

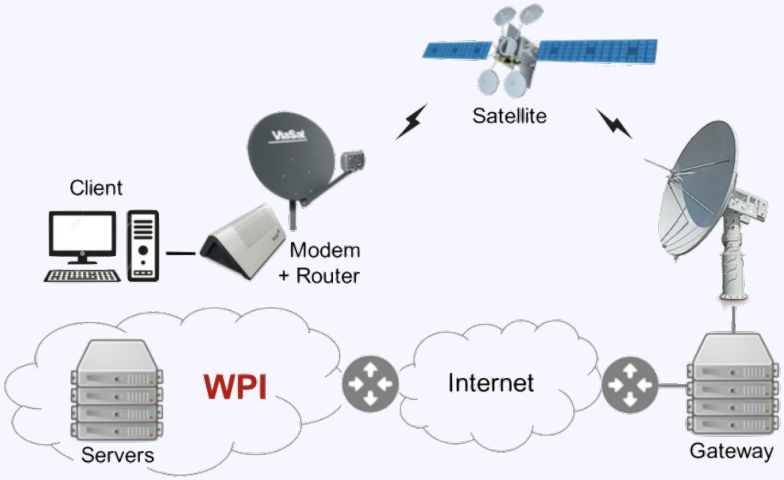
Methodology

One server with Cubic

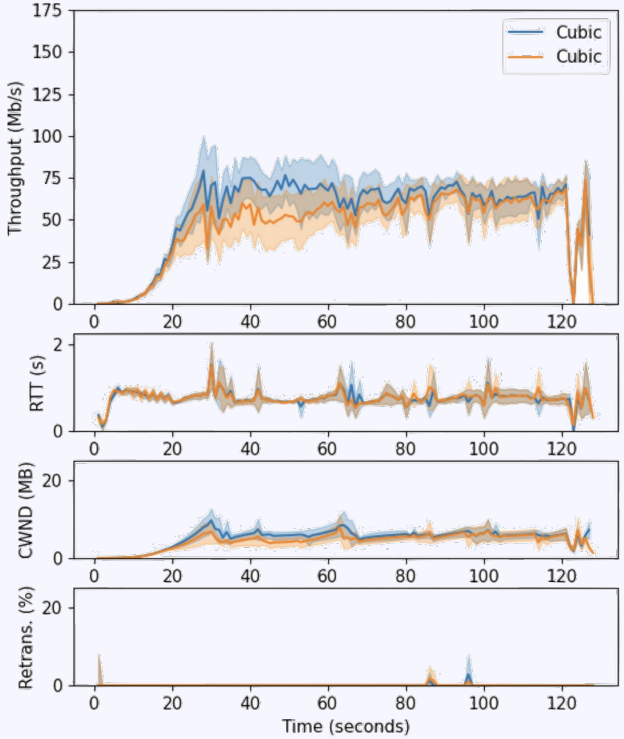
Another server with Cubic, BBR, Hybla, PCC

Servers are connected through a satellite

Servers communicate via simultaneous iperf bulk downloads



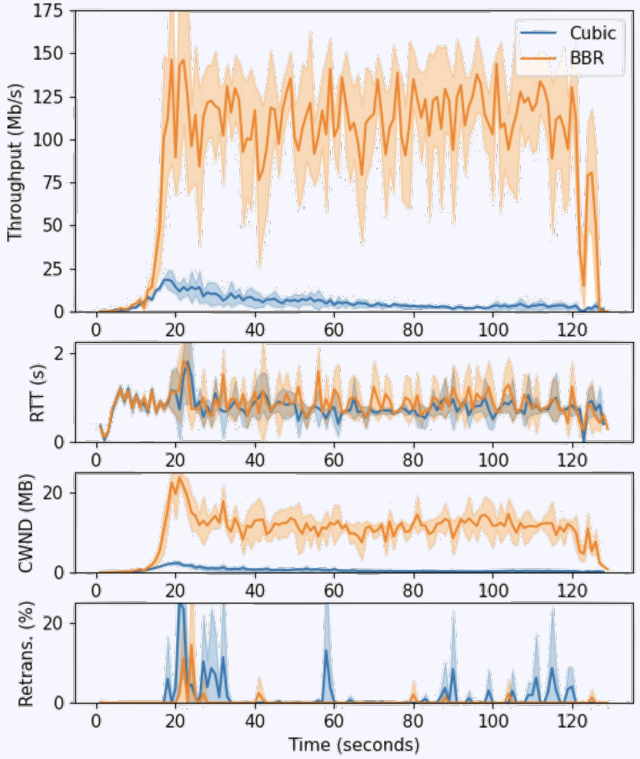
TCP Cubic vs. TCP Cubic



Cubic: less aggressive in steady-state, adjusts quickly when needed

For both servers, throughput and RTTs are very similar

TCP Cubic vs. TCP BBR

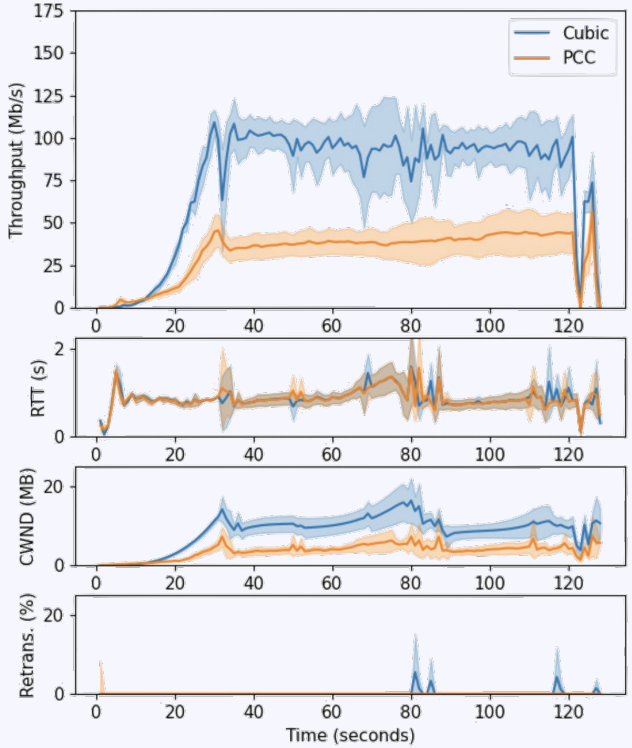


BBR (Bottleneck Bandwidth and RTT): uses max bandwidth and min RTT to calculate CWND size

BBR gets 10x higher throughput than Cubic

Cubic likely has low throughput due to intermittent loss

TCP Cubic vs. TCP PCC

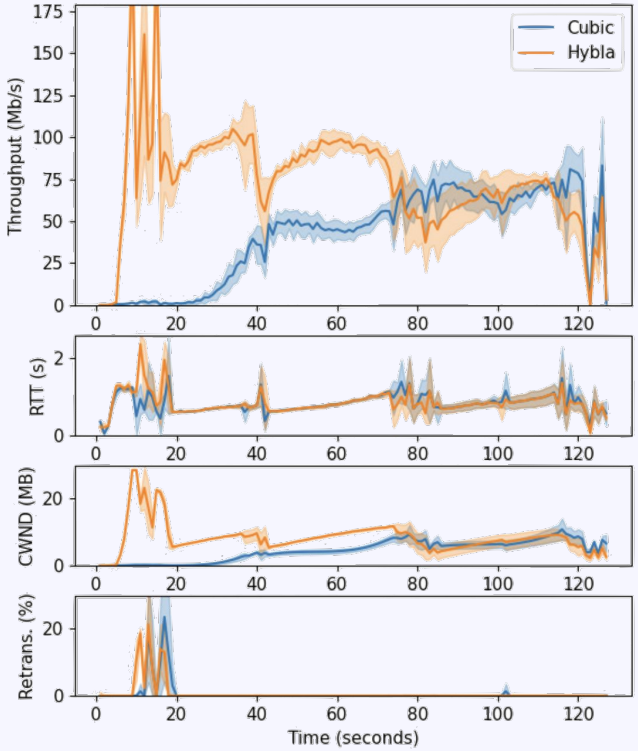


PCC: continuously observes performance

Cubic gets 2x higher throughput

PCC prioritizes low delay, Cubic doesn't consider delay

TCP Cubic vs. TCP Hybla

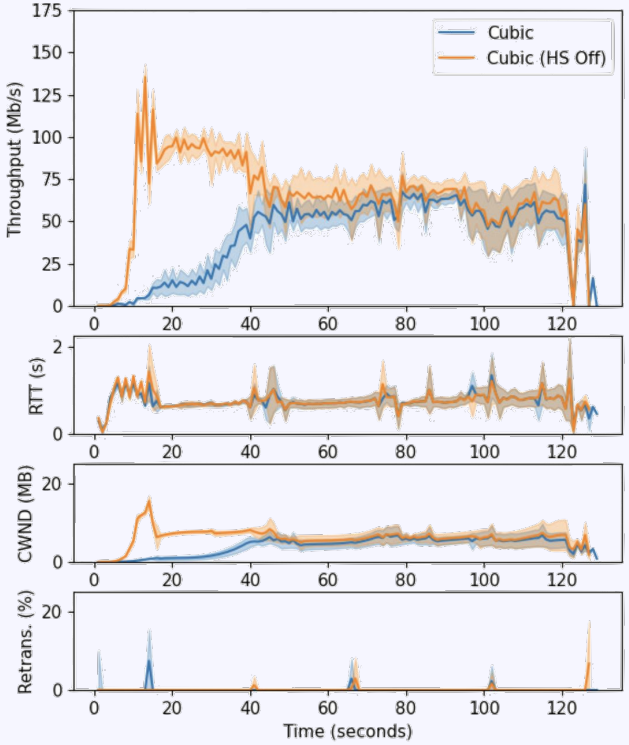


Hybla: good for high-latency links, adjusts CWND based on RTT

Hybla gets much more throughput during start-up

Initial start-up for Hybla causes retransmissions and higher RTTs

TCP Cubic vs. TCP Cubic (Hystart off)



Hystart: Cubic has Hystart on by default

Originally thought that Cubic with Hystart causes connections over satellite to exit slow-start before CWND reaches max

Cubic without Hystart gets more throughput during start-up, but then has same throughput as Cubic with Hystart

Concluding Thoughts

Using TCP Cubic, the default congestion control algorithm, with other algorithms over a satellite connection can affect network performance.