Improving QUIC Slow Start Performance with SEARCH

Amber Cronin*, Maryam Ataei Kachooei*,
Jae Chung†, Feng Li†,
Benjamin Peters†, Mark Claypool*

* Worcester Polytechnic Institute, † Viasat

Why study QUIC?

2012: Google begins development of a modern replacement for TCP

2021: QUIC codified by IETF with RFC9000

2023: QUIC carries ~30% of web traffic [Cloudflare]

Benefits over TCP:

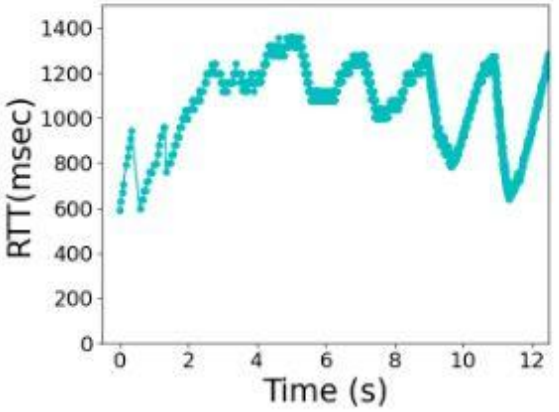
- Faster protocol encryption setups (by RTTs)
- Multiple streams per connection
- Avoids middlebox ossification by use of UDP

Satellite link fixes (PEPs) no longer function

Satellite Links

GEO links: High bandwidth, high latency

- 600 ms RTT @ 144 Mbps
- Bandwidth Delay Product (BDP) = BW x RTT = 10.8 MB



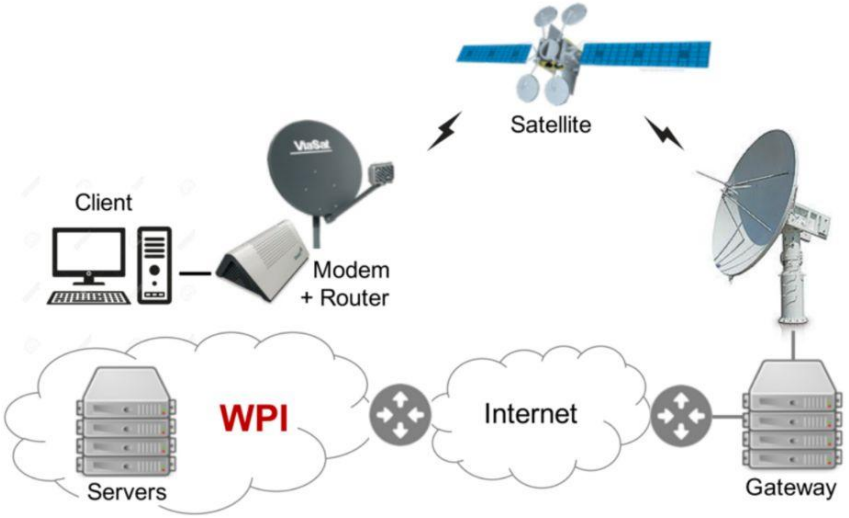
(b) Round-trip time.

Queues, Queues, Queues

In practice, RTTs reach 1000+ ms

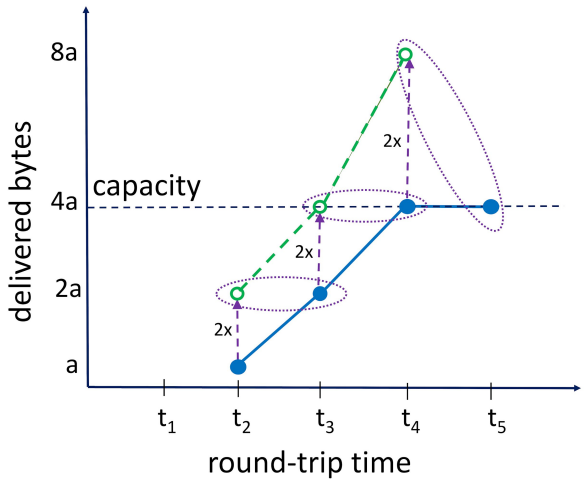
- BDP = 18+ MB

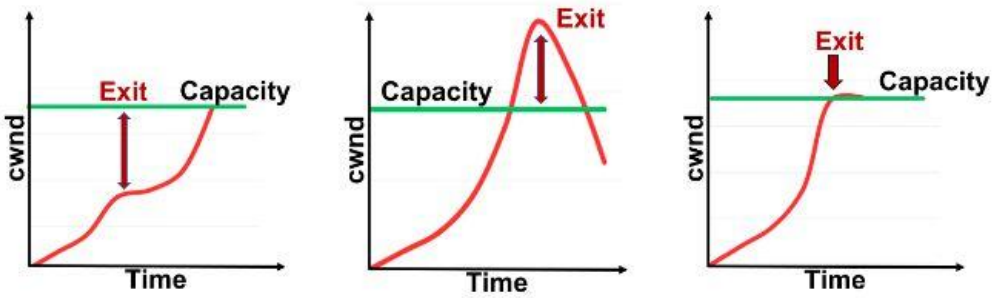
HyStart unstable

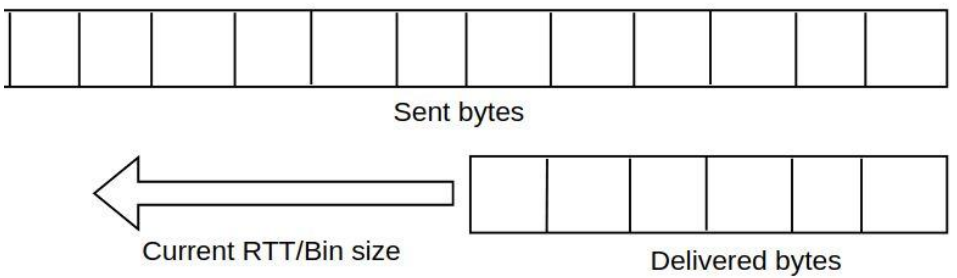


The SEARCH Algorithm

Track the delivery rate rolling average to reduce the impact of noise on link capacity detection

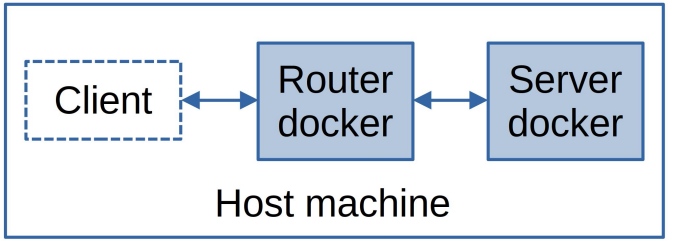






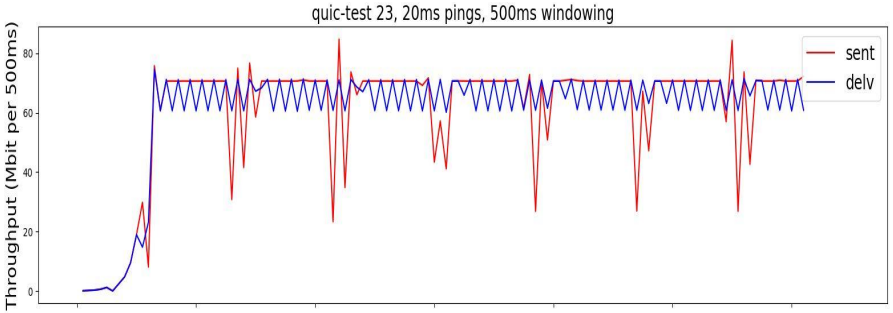
Initial Tests

QUIC implementation: Quicly
Testbench: Qperf (with modifications)

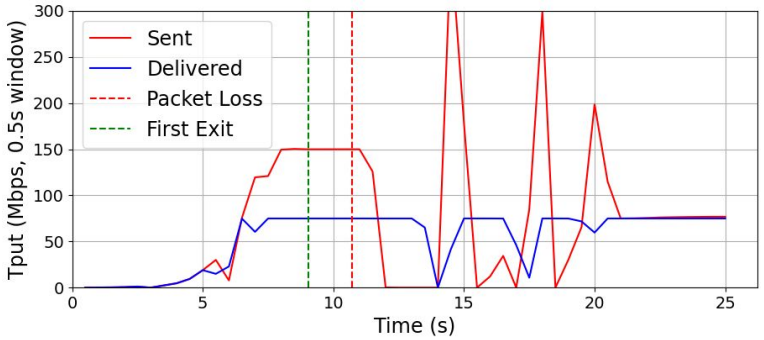


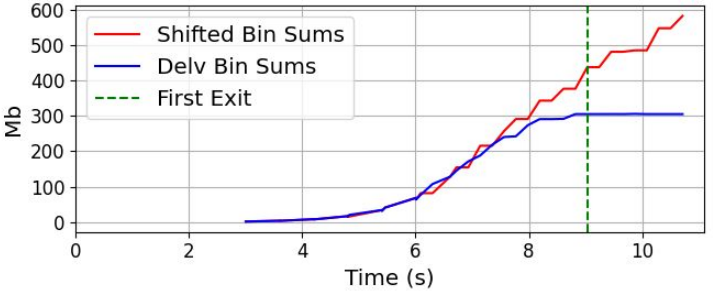
Before:

Congestion controller limited by MAX_DATA transmissions...



After:





Data Gathering

	First-RTT Loss	Wireless Loss	Clean	**Total**
Baseline	103	12	544	659
SEARCH	116	8	535	659
Total	219	20	1079	1318

	Total	50 MB	100 MB	150 MB	200 MB
Baseline	544	449	421	386	358
SEARCH	535	445	419	380	323
Total	1079	894	840	766	681

Dataset makeup

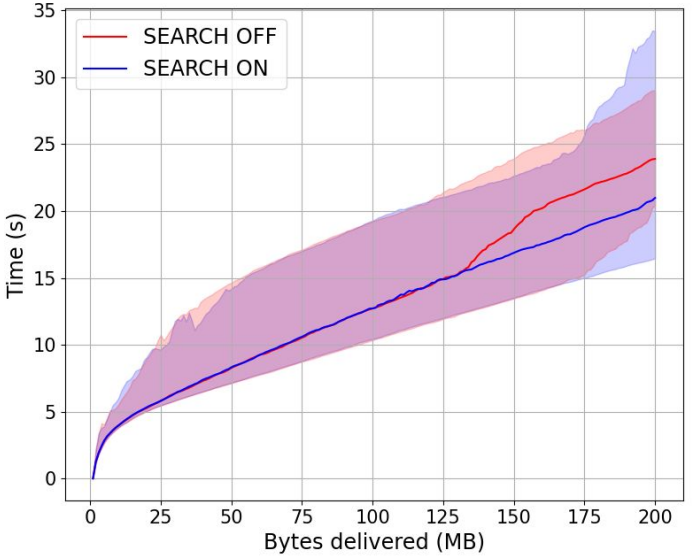
Two runs over the satellite platform

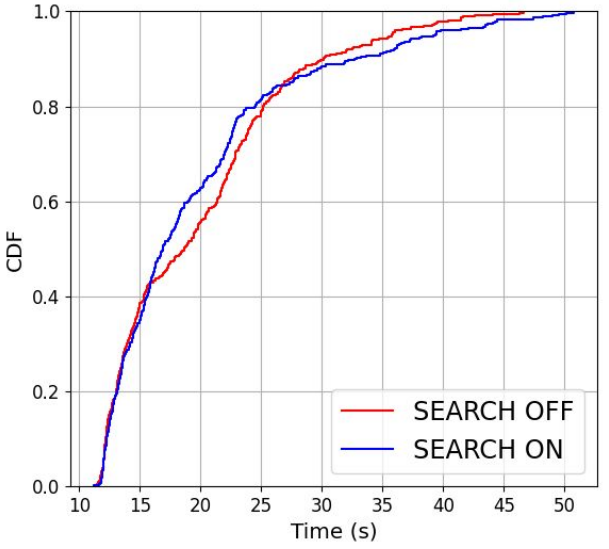
- 20 hours
- 24 hours

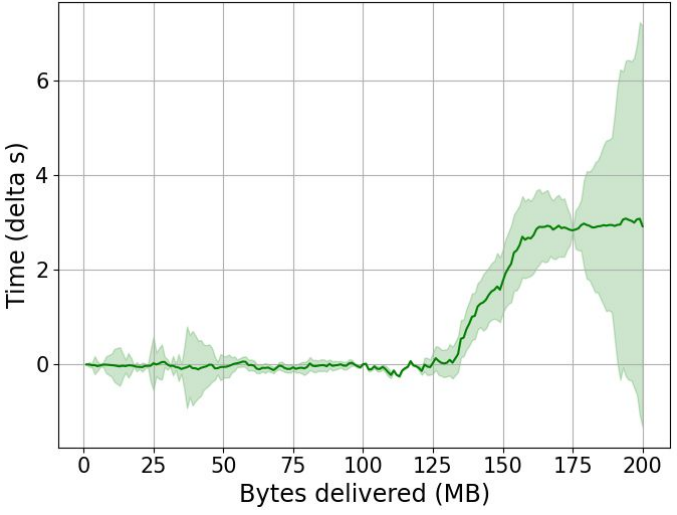
Server set to 150Mbps limit

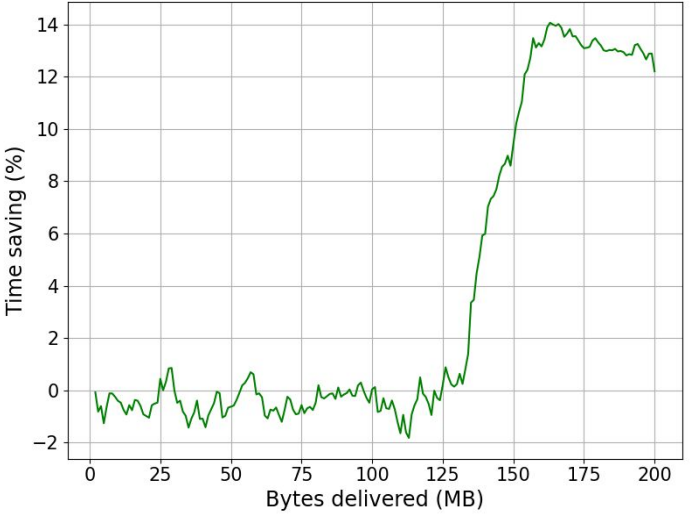
Results

- SEARCH improves median goodput
- 3 second, or 14% improvement over base case





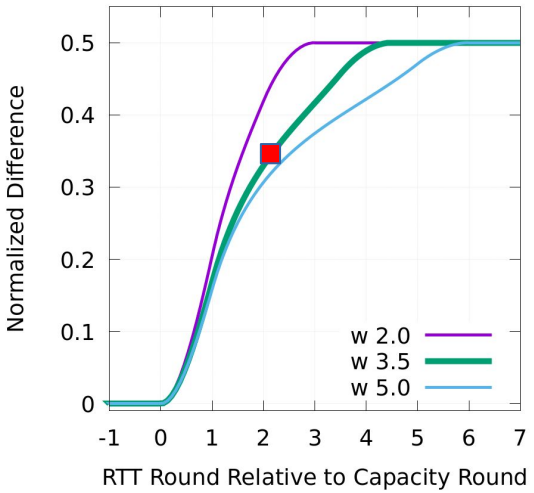




Congestion Window Modification

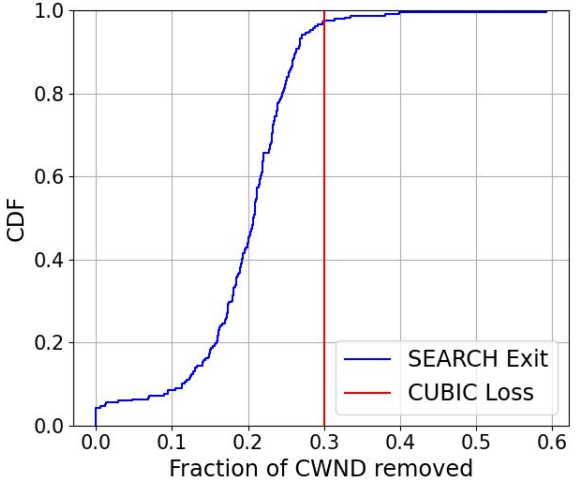
Tracking data on link capacity
Theoretical proofs of SEARCH detection (1.96 RTTs)

= Direct knowledge of link capacity



	First-RTT Loss	Wireless Loss	Clean	**Total**
Baseline	106	22	232	360
SEARCH	83	23	254	360
Total	189	45	486	720

Dataset makeup



Summary

- Extended existing QUIC implementation with SEARCH
- Showed SEARCH improves QUIC’s goodput by 3 seconds (14%)
- Showed SEARCH detects chokepoint and exits before loss
- Tested on a commercial satellite platform

SEARCH

Algorithm 1
SEARCH: Slow start Exit At Right CHokepoint.

Parameters:
WINDOW_SIZE = Initial_RTT $\times$ 3.5
W = 10
EXTRA_BINS = 15
NUM_BINS = W + EXTRA_BINS
BIN_DURATION = WINDOW_SIZE / W
THRESH = 0.35

Initialization:
bin[NUM_BINS]
curr = 0
bin_end = **now** + BIN_DURATION

Each acknowledgement:
if (**now** $>$ bin_end) **then**
  bin_end += BIN_DURATION
  curr += 1
  bin[curr mod NUM_BINS] = 0
  prev = curr - (RTT / BIN_DURATION)
  if (prev $\geq$ W) and (curr - prev) $\leq$ EXTRA_BINS **then**
    // Check if SEARCH should exit
    curr_delv = $\sum_{\text{curr-W}}^{\text{curr}}$ bin[i mod NUM_BINS]
    prev_delv = $\sum_{\text{prev-W}}^{\text{prev}}$ bin[i mod NUM_BINS]
    norm_diff = $\dfrac{2 \cdot \text{prev\_delv} - \text{curr\_delv}}{2 \cdot \text{prev\_delv}}$
    if (norm_diff $\geq$ THRESH) **then**
      // Exit slow start
      back = $\dfrac{\text{Initial\_RTT} \cdot 2}{\text{BIN\_DURATION}}$
      over = $\sum_{\text{curr - back}}^{\text{curr}}$ bin[i mod NUM_BINS]
      set `ssthresh` and `cwnd` to `(cwnd - over)`
    end if
  end if
end if
bin[curr mod NUM_BINS] += bytes_delivered