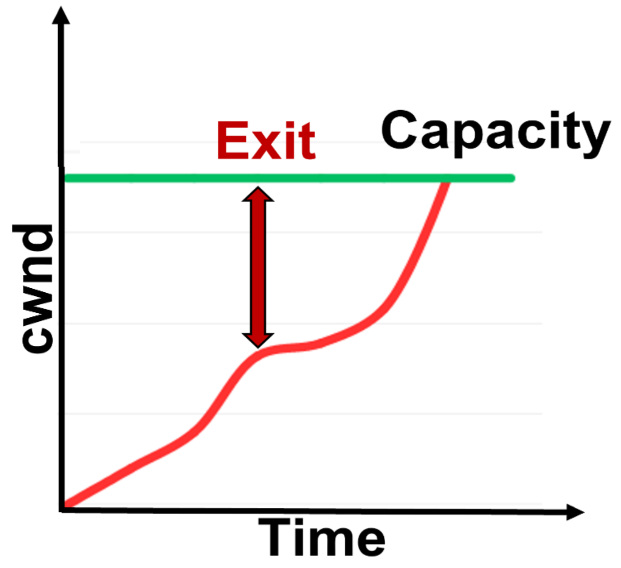
SEARCH – a New Slow Start Algorithm for TCP and QUIC

Jae Chung
Feng Li

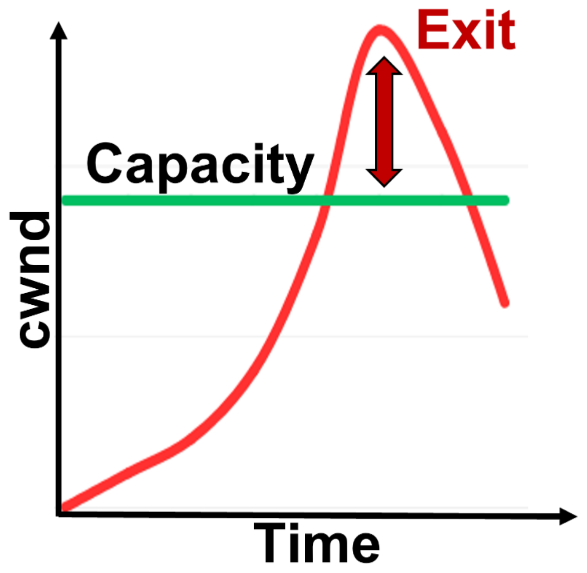
Maryam Ataei Kachooei
Mark Claypool

IETF CCWG
Vancouver, Canada
July 2024

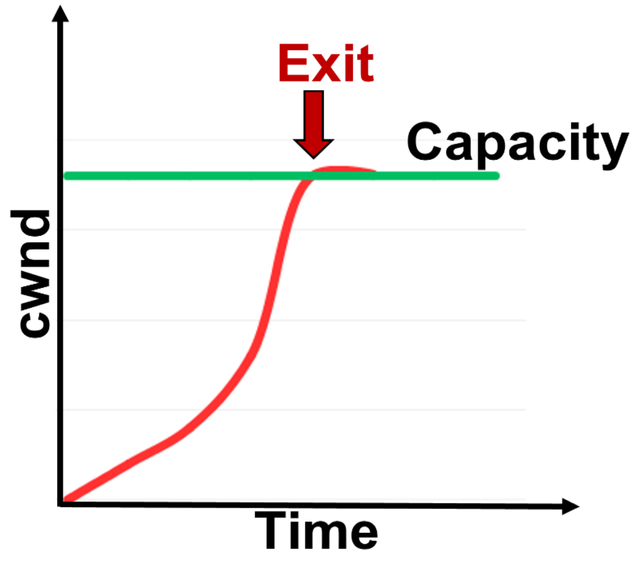
Motivation



Exit Too Early

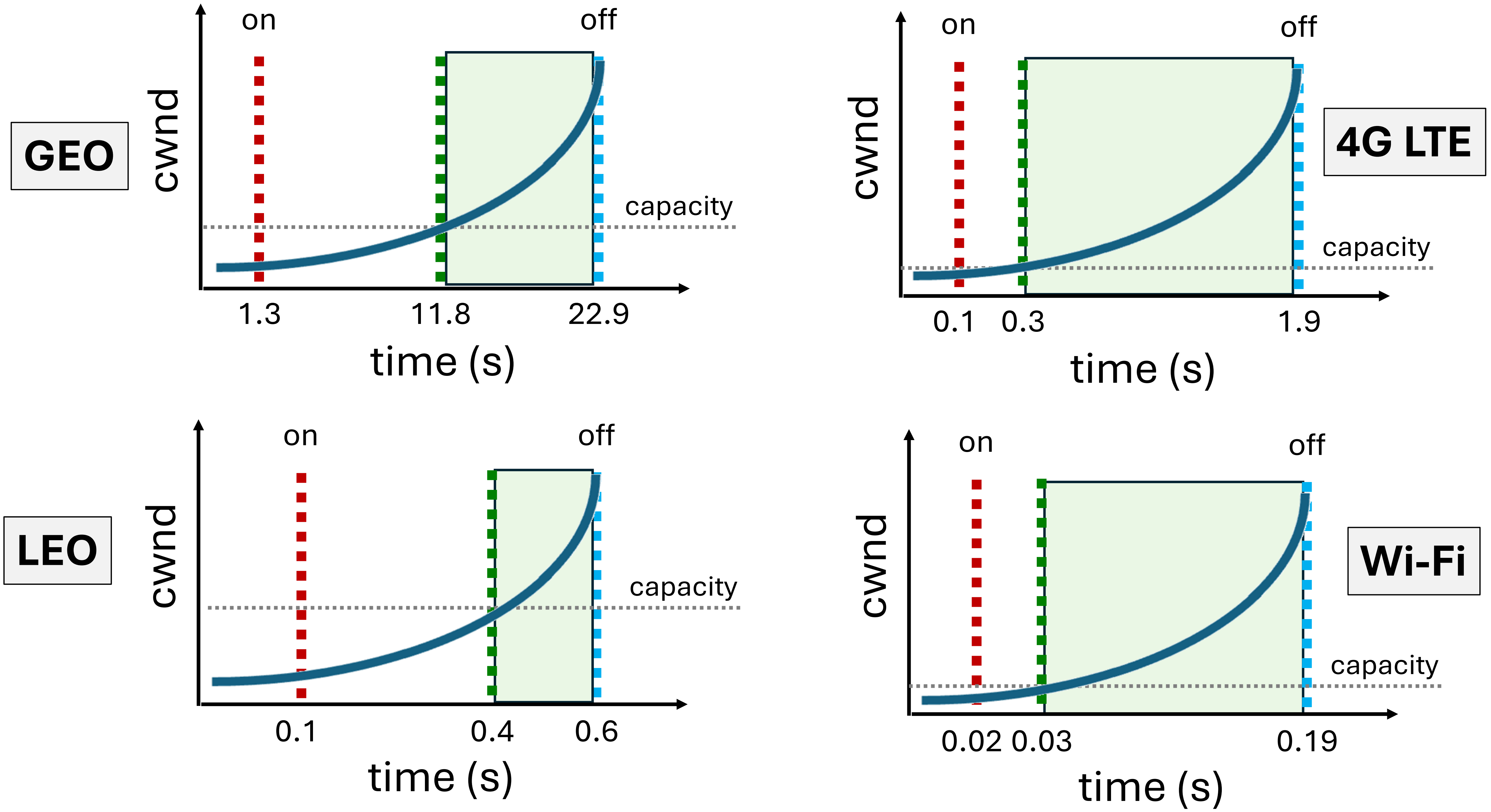


Exit Too Late



Exit at Choke Point

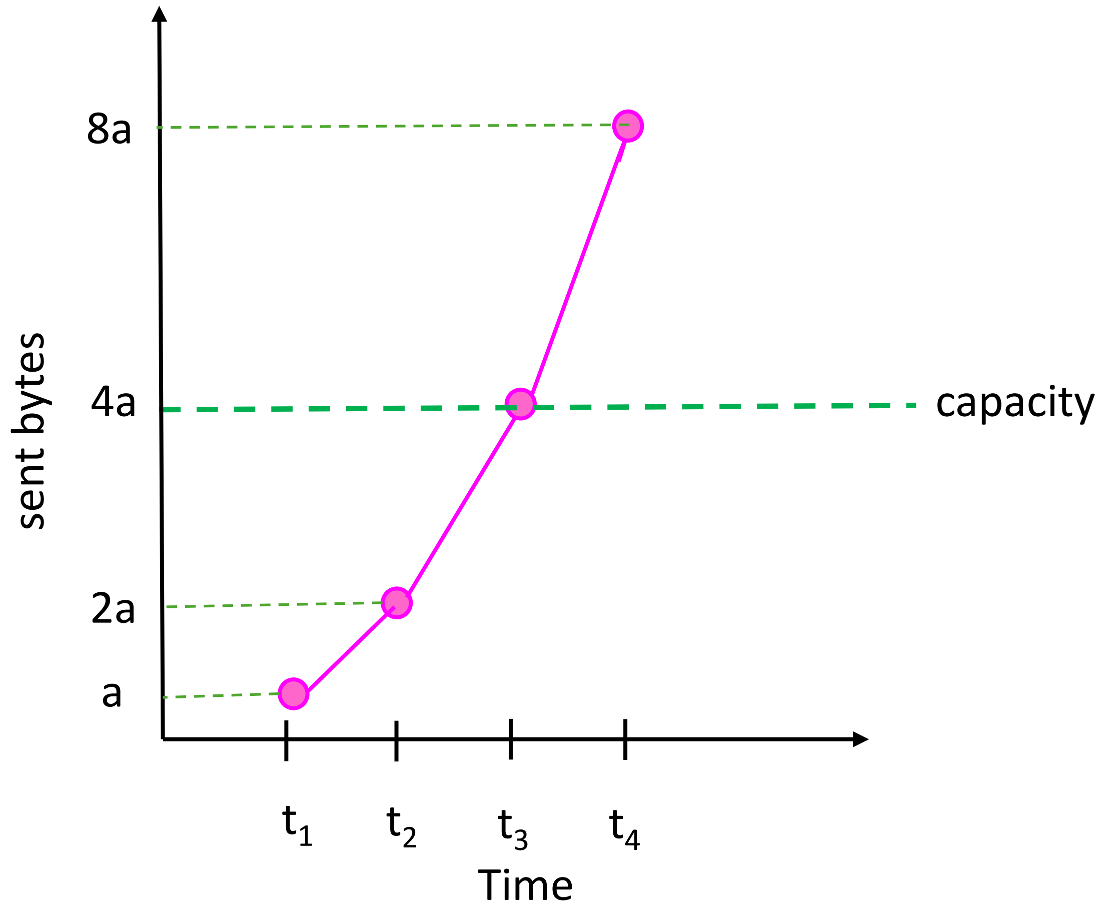
TCP HyStart over Wireless?

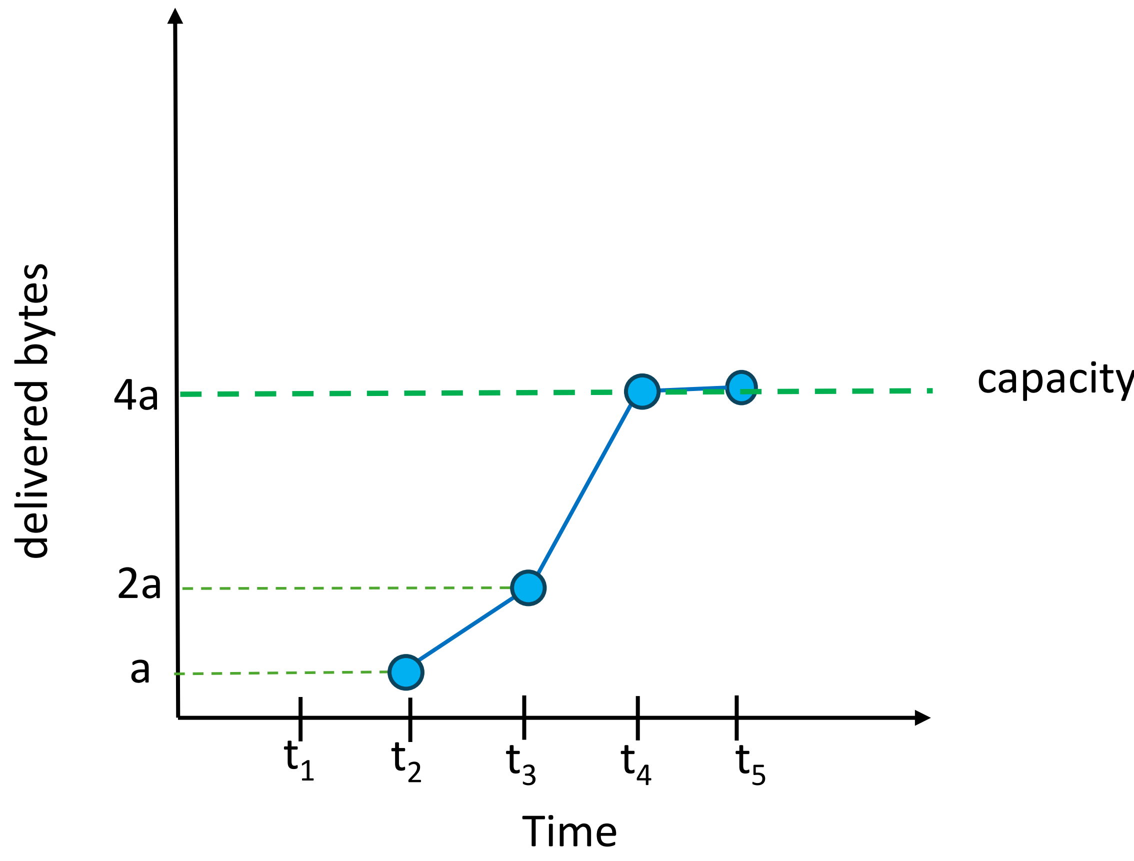

GEO
cwnd
on
off
capacity
1.3
11.8
22.9
time (s)
4G LTE
cwnd
on
off
capacity
0.1
0.3
1.9
time (s)
LEO
cwnd
on
off
capacity
0.1
0.4
0.6
time (s)
Wi-Fi
cwnd
on
off
capacity
0.02
0.03
0.19
time (s)

Outline

- Motivation (done)
- SEARCH (next)
- Performance Evaluation
- Conclusion

SEARCH – Slow start Exit at Right CHokepoint





SEARCH – Slow start Exit at Right CHokepoint

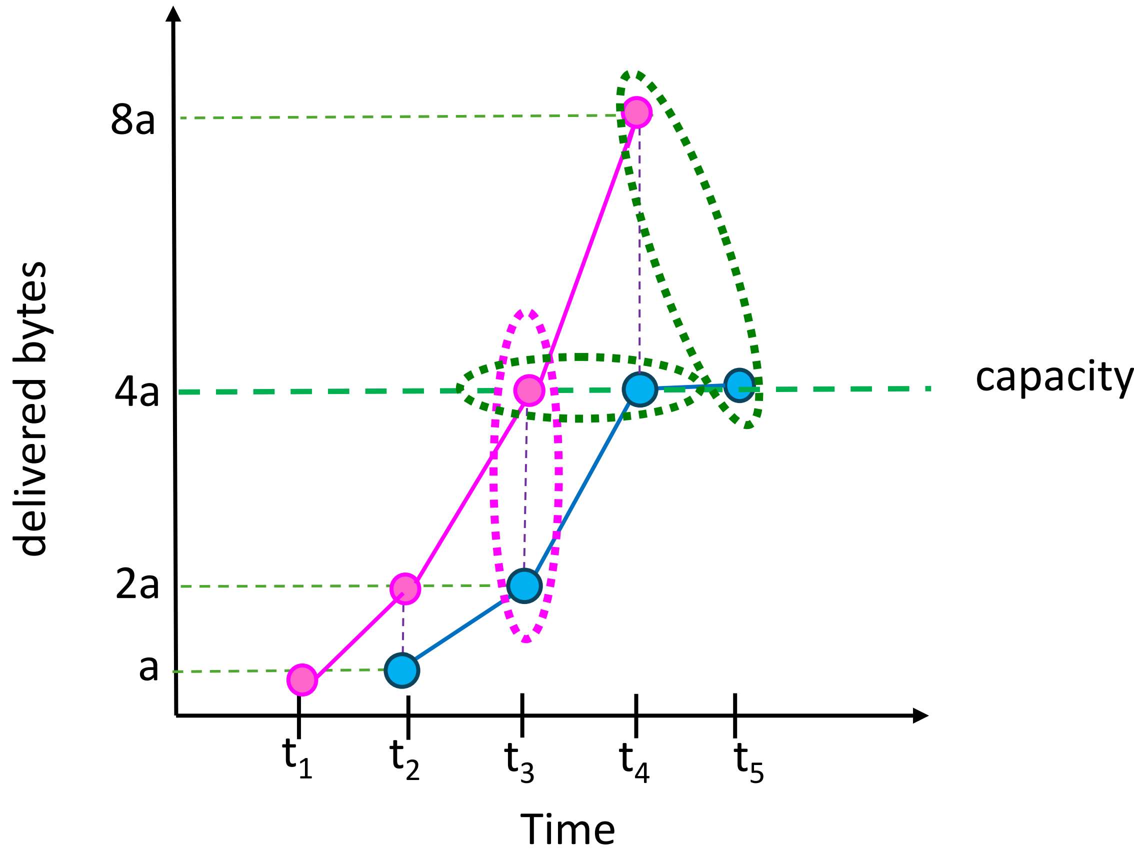
$\text{sent}' = 2 \cdot \text{delv}_{\text{previous}}$

$\text{diff} = \text{sent}' - \text{delv}_{\text{now}}$

$\text{normalized\_diff} = \text{diff} / \text{sent}'$

$\text{normalized\_diff} \geq \text{threshold?}$

→ exit slow start



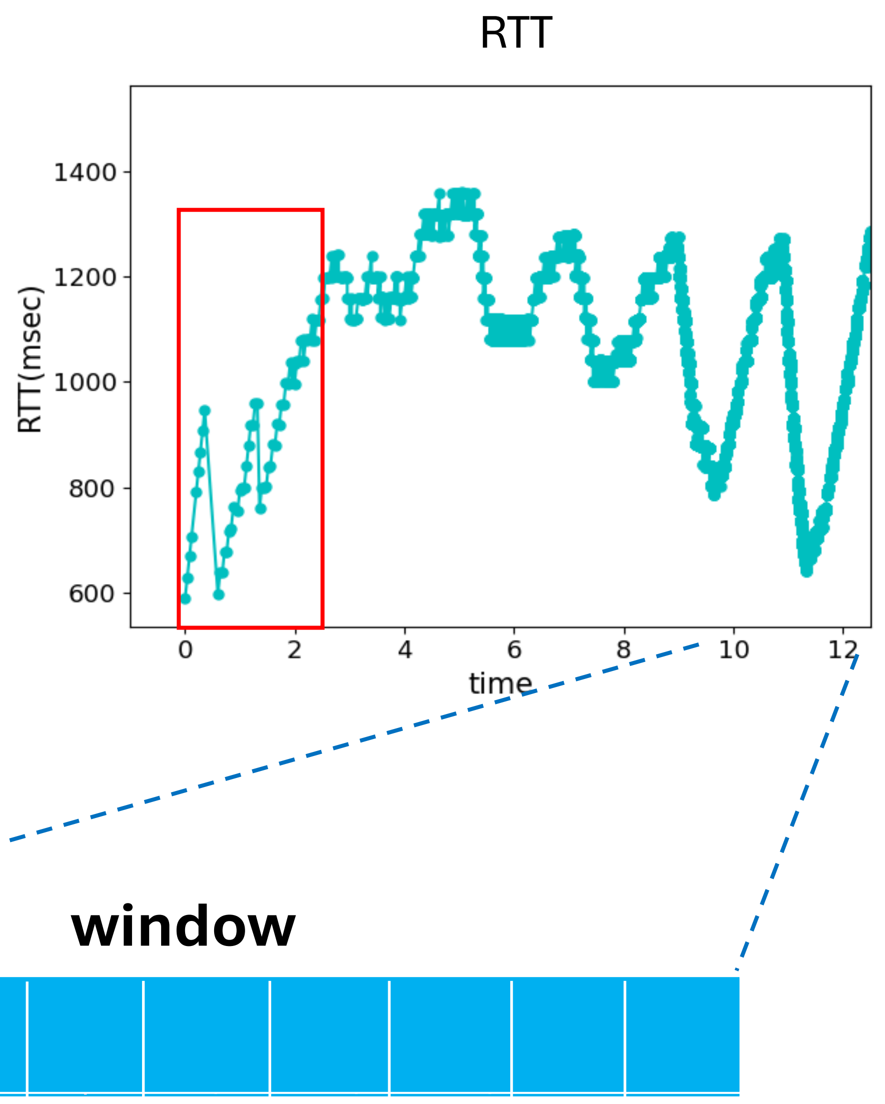
Challenges

Variable RTTs

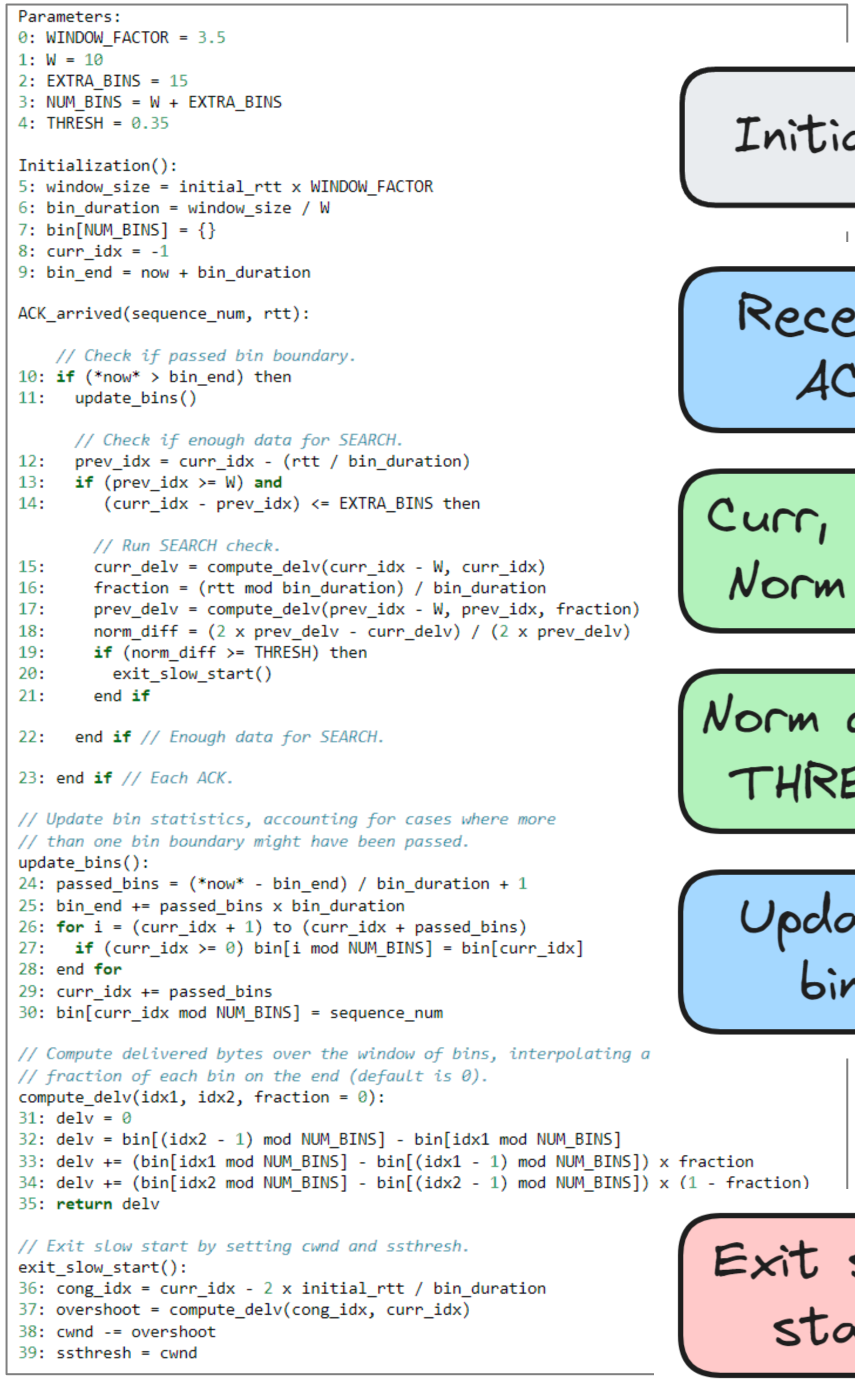
- Caused by uplink ACK schedule timing
- *Not* caused by congestion on forward link

Limit memory on server

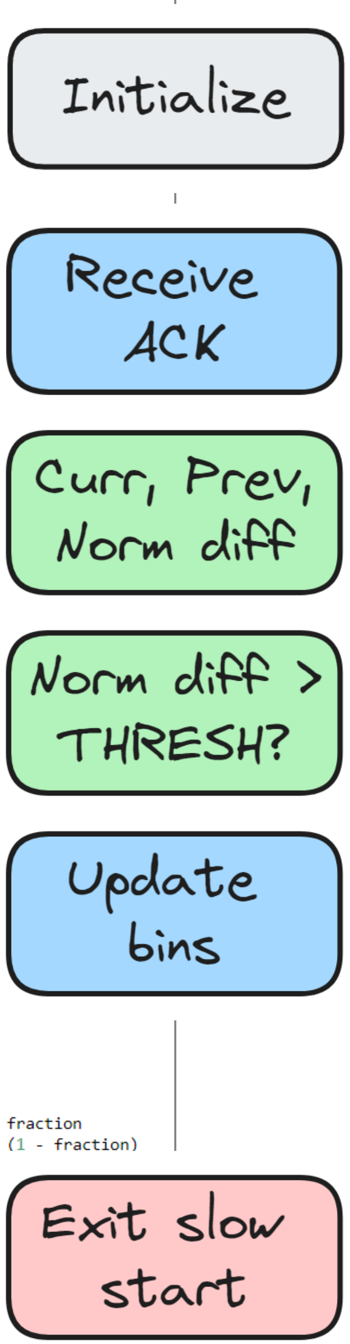
- Memory allocated per flow
- Can't store history for each ACK



SEARCH



```
Parameters:
WINDOW_FACTOR = 3.5
W = 10
EXTRA_BINS = 15
NUM_BINS = W + EXTRA_BINS
THRESH = 0.35

Initialization():
window_size = initial_rtt x WINDOW_FACTOR
bin_duration = window_size / W
bin[NUM_BINS] = {}
curr_idx = -1
bin_end = now + bin_duration

ACK_arrived(sequence_num, rtt):

  // Check if passed bin boundary.
  if (*now* > bin_end) then
    update_bins()

    // Check if enough data for SEARCH.
    prev_idx = curr_idx - (rtt / bin_duration)
    if (prev_idx >= W) and
       (curr_idx - prev_idx) <= EXTRA_BINS then

      // Run SEARCH check.
      curr_delv = compute_delv(curr_idx - W, curr_idx)
      fraction = (rtt mod bin_duration) / bin_duration
      prev_delv = compute_delv(prev_idx - W, prev_idx, fraction)
      norm_diff = (2 x prev_delv - curr_delv) / (2 x prev_delv)
      if (norm_diff >= THRESH) then
        exit_slow_start()
      end if

    end if // Enough data for SEARCH.

  end if // Each ACK.

// Update bin statistics, accounting for cases where more
// than one bin boundary might have been passed.
update_bins():
  passed_bins = (*now* - bin_end) / bin_duration + 1
  bin_end += passed_bins x bin_duration
  for i = (curr_idx + 1) to (curr_idx + passed_bins)
    if (curr_idx >= 0) bin[i mod NUM_BINS] = bin[curr_idx]
  end for
  curr_idx += passed_bins
  bin[curr_idx mod NUM_BINS] = sequence_num

// Compute delivered bytes over the window of bins, interpolating a
// fraction of each bin on the end (default is 0).
compute_delv(idx1, idx2, fraction = 0):
  delv = 0
  delv = bin[(idx2 - 1) mod NUM_BINS] - bin[idx1 mod NUM_BINS]
  delv += (bin[idx1 mod NUM_BINS] - bin[(idx1 - 1) mod NUM_BINS]) x fraction
  delv += (bin[idx2 mod NUM_BINS] - bin[(idx2 - 1) mod NUM_BINS]) x (1 - fraction)
  return delv

// Exit slow start by setting cwnd and ssthresh.
exit_slow_start():
  cong_idx = curr_idx - 2 x initial_rtt / bin_duration
  overshoot = compute_delv(cong_idx, curr_idx)
  cwnd -= overshoot
  ssthresh = cwnd
```



Parameter Selection

Improving TCP Slow Start Performance in Wireless Networks with SEARCH

IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)

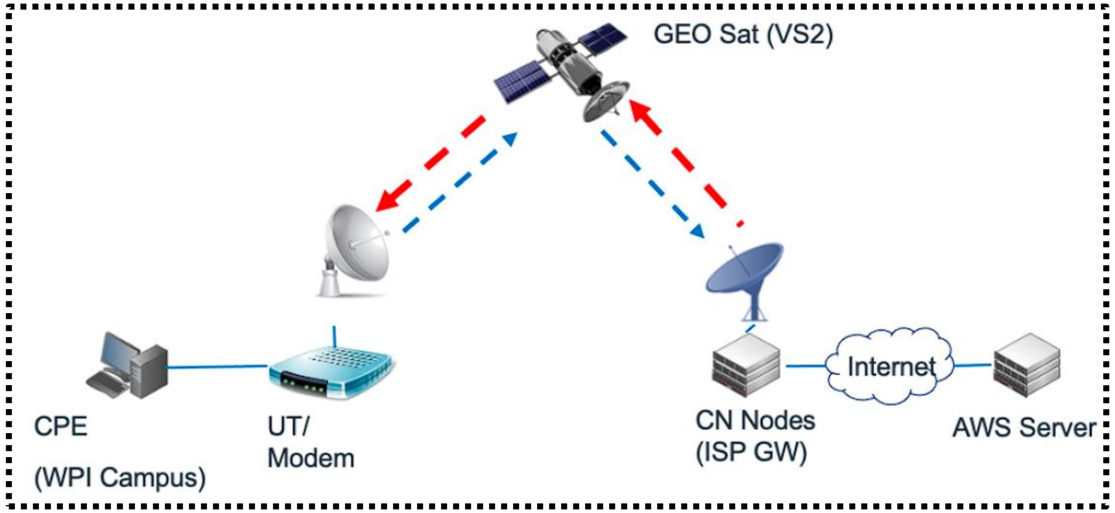
Perth, Australia, June 2024.

- Window size:
  - Large enough for link RTT variation
  - Small enough to respond quickly
- Number of bins:
  - Large enough to reduce load
  - Small enough to maintain fidelity and respond quicky
- Threshold:
  - Large enough so above noise
  - Small enough to respond quickly

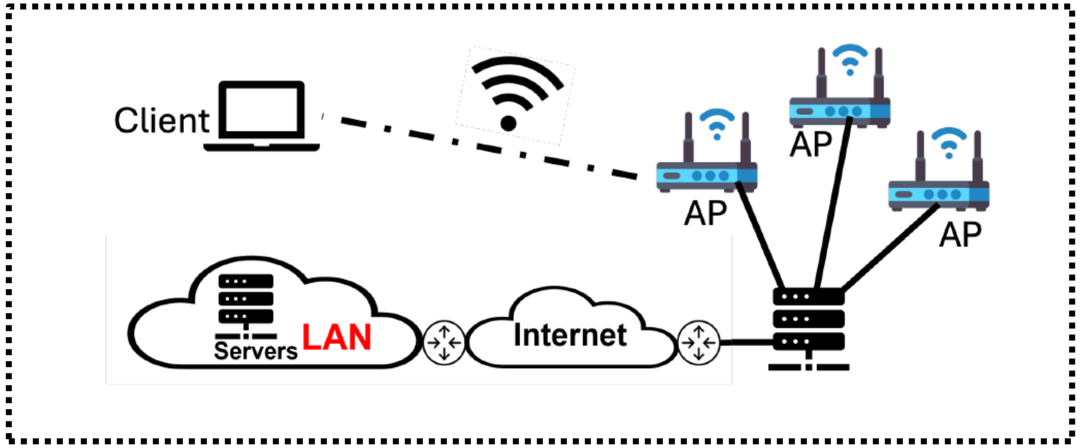
```
0: WINDOW_FACTOR = 3.5


1: W = 10
2: EXTRA_BINS = 15
3: NUM_BINS = W + EXTRA_BINS


4: THRESHOLD = 0.35
```

Outline

- Motivation (done)
- SEARCH (done)
- Performance Evaluation (next)



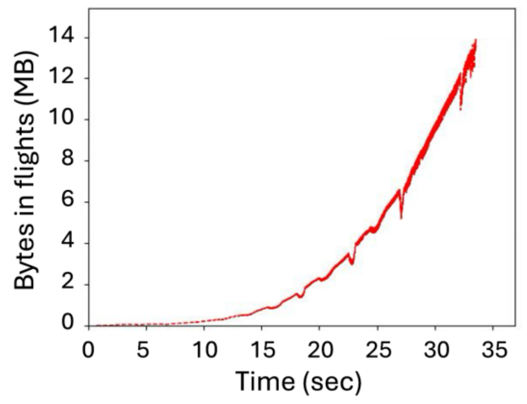
GEO

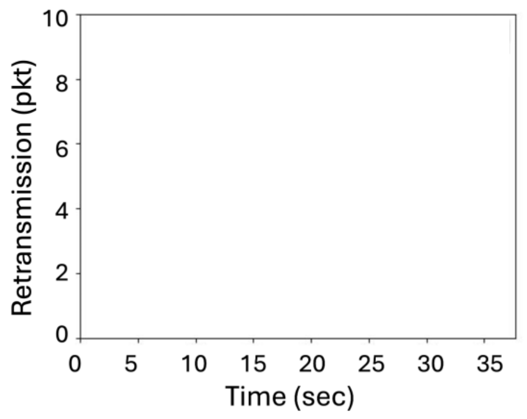


Wi-Fi

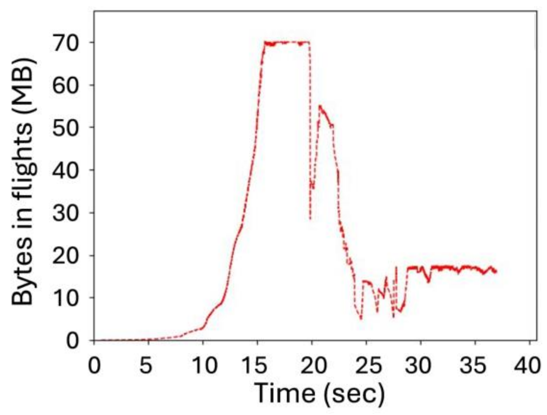
Performance – GEO Satellite

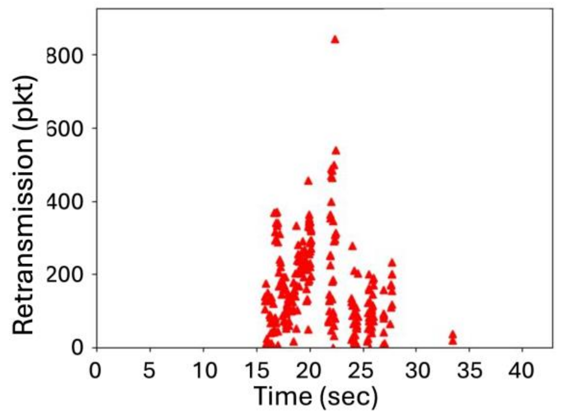
HyStart On



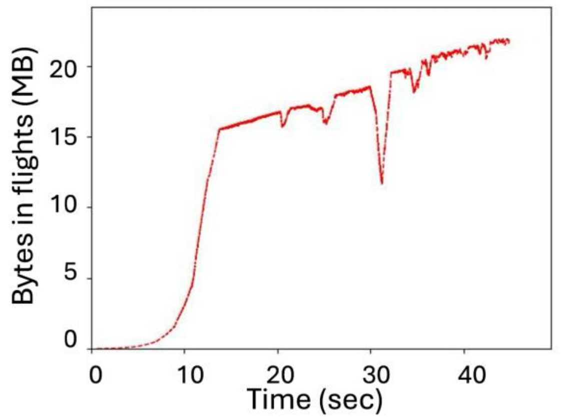


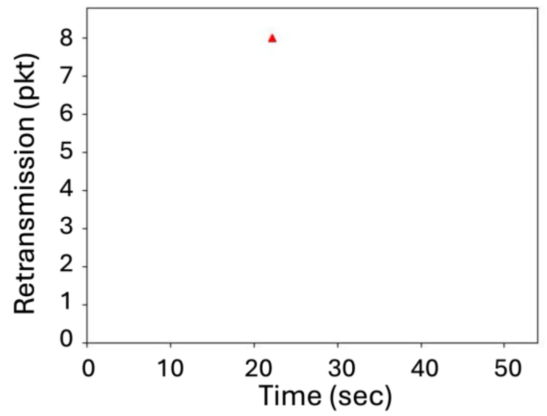
HyStart Off

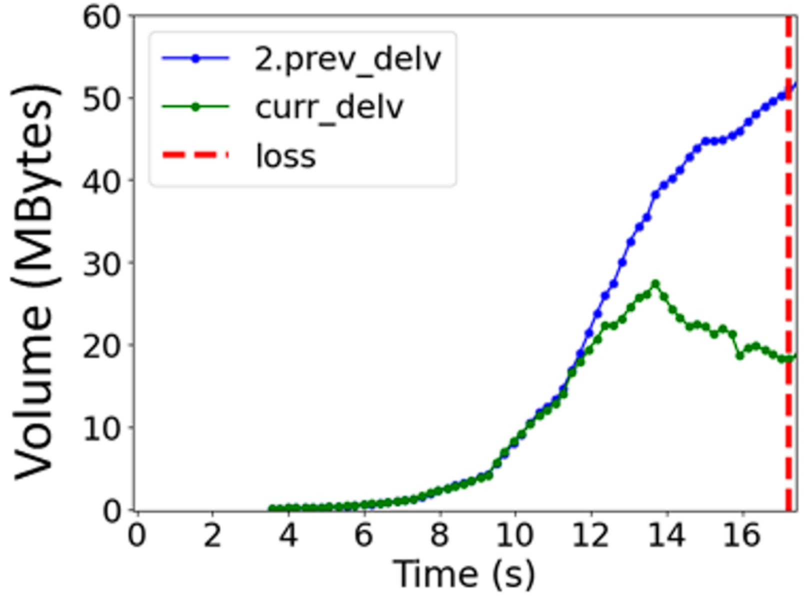


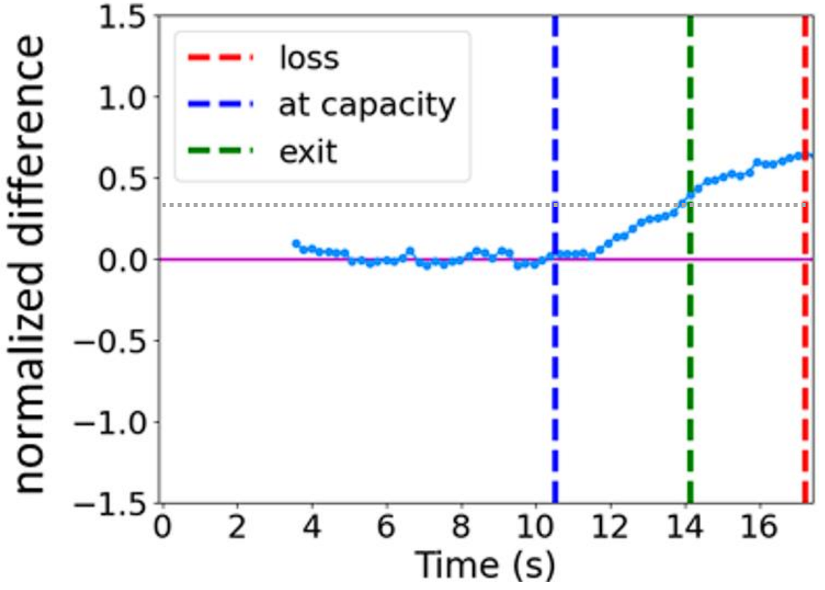


SEARCH

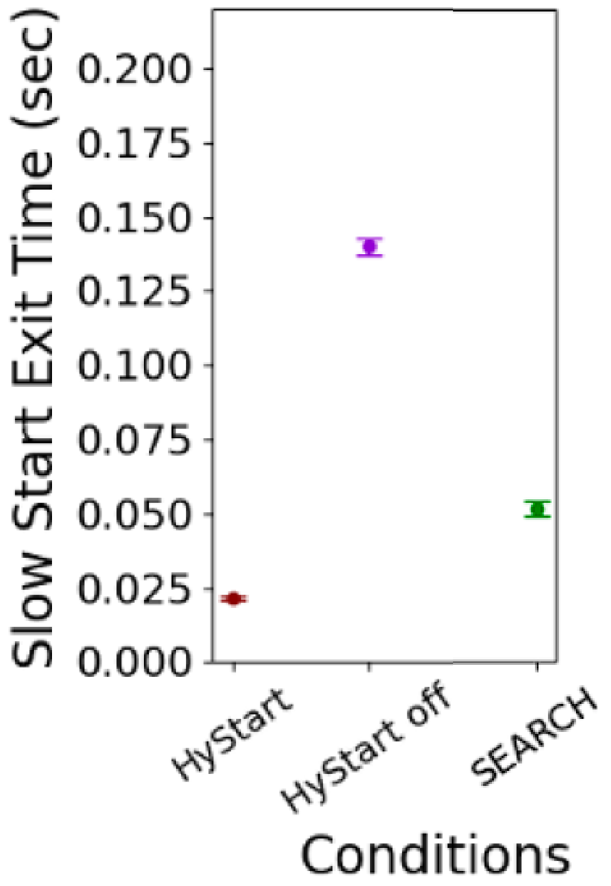




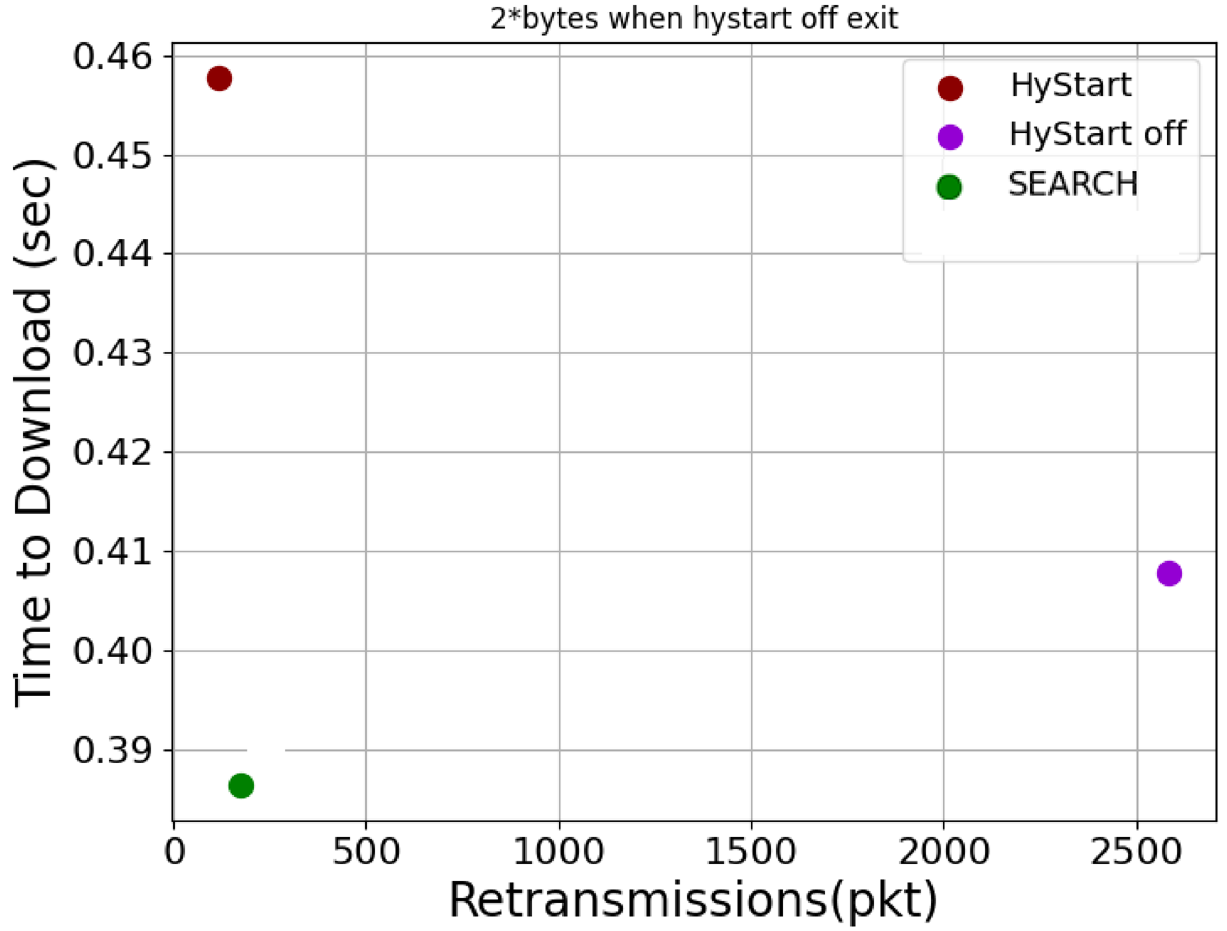




Performance – Wi-Fi



~15 MBytes



Implementation Status

- Linux Kernel Modules
  - 5.13.x series kernel (main branch)
    https://github.com/Project-Faster/tcp_ss_search.git
  - 6.10rc2 based (net-next-6.10rc2 branch)
    https://github.com/Project-Faster/tcp_ss_search/tree/net-next-6.10rc2
- QUIC H2O/Quicly
  https://github.com/Project-Faster/quicly/tree/generic-slowstart
- Upstream into Linux mainstream and open source QUIC

Conclusion

- HyStart does not work in wireless environments (GEO, LEO, 4G LTE, Wi-Fi) → premature slow start exits
- SEARCH
  - Determines “choke point” from expected delivered bytes
  - Exits slow start after congestion point, before loss
  - Improves utilization, reduces packet loss (versus off)

SEARCH https://search-ss.wpi.edu/

Thank-you for your attention!

SEARCH – a New Slow Start Algorithm for TCP and QUIC

Jae Chung
Feng Li

Maryam Ataei Kachooei
Mark Claypool

IETF CCWG
Vancouver, Canada
July 2024

References

- Improving TCP Slow Start Performance in Wireless Networks with SEARCH
  - *IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)*
  - Perth, Australia, June 2024.
- Improving QUIC Slow Start Behavior in Wireless Networks with SEARCH
  - *IEEE Local and Metropolitan Area Networks (LANMAN)*
  - Boston, Massachusetts, USA, July 2024
- Implementation of the SEARCH Slow Start Algorithm in the Linux Kernel
  - *0x18 NetDev Conference*
  - Santa Clara, California, USA, July 2024