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

TCP HyStart over Wireless?
11.8 on time (s)
1.3 off time (s)
cwnd capacity

22.9 on time (s)
0.3 off time (s)
cwnd capacity

4G LTE on time (s)
0.1 off time (s)
cwnd capacity

LEO on time (s)
0.1 off time (s)
cwnd capacity

Wi-Fi on time (s)
0.02 off time (s)
cwnd capacity

Wi-Fi on time (s)
0.03 off time (s)
cwnd capacity
Outline

• Motivation (done)
• SEARCH (next)
• Performance Evaluation
• Conclusion
SEARCH – **Slow start** Exit at **Right** CHokepoint

---

**Graph 1:**
- **Y-axis:** Sent bytes
- **X-axis:** Time
- **Points:**
  - $t_1$: 1
  - $t_2$: 2
  - $t_3$: 4
  - $t_4$: 8
- **Legend:**
  - Capacity line

**Graph 2:**
- **Y-axis:** Delivered bytes
- **X-axis:** Time
- **Points:**
  - $t_1$: 1
  - $t_2$: 2
  - $t_3$: 4
  - $t_4$: 8
  - $t_5$: 4
- **Legend:**
  - Capacity line

---

7/23/2024

SEARCH - Better Slow Start for TCP and QUIC
SEARCH – **Slow start Exit at Right CHokepoint**

sent’ = 2 \cdot \text{delv}_{\text{previous}}

diff = sent’ − \text{delv}_{\text{now}}

normalized\_diff = \frac{\text{diff}}{\text{sent’}}

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
Better Slow Start for TCP and QUIC

Parameters:
0) WINDOW_FACTOR = 3.5
1) W = 10
2) EXTRA_BINS = 15
3) NUM_BINS = W + EXTRA_BINS
4) THRESH = 0.35

Initialization():
5) window_size = initial_rtt × WINDOW_FACTOR
6) bin_duration = window_size / W
7) bin[NUM_BINS] = []
8) curr_idx = -1
9) bin_end = row + bin_duration

ACK_arrived(sequence_num, rtt):

// Check if passed bin boundary.
10) if ("row" ≥ bin_end) then
11) update_bins()

// Check if enough data for SEARCH.
12) prev_idx = curr_idx - (rtt / bin_duration)
13) if (prev_idx ≥ W) and
14) (curr_idx = prev_idx) ∧ EXTRA_BINS then

// Run SEARCH check.
15) curr_delv = compute_delv(curr_idx - W, curr_idx)
16) fraction = (rtt mod bin_duration) / bin_duration
17) prev_delv = compute_delv(prev_idx - W, prev_idx, fraction)
18) norm_diff = (2 × prev_delv - curr_delv) / (2 × prev_delv)
19) if (norm_diff ≥ THRESH) then
20) exit_slow_start()
21) end if
22) end if // Enough data for SEARCH.
23) end if // Each ACK.

// Update bin statistics, accounting for cases where more
// than one bin boundary might have been passed.
update_bins()
24) passed_bins = ("row" - bin_end) / bin_duration + 1
25) bin_end = passed_bins × bin_duration
26) for i = (curr_idx + 1) to (curr_idx + passed_bins)
27) if (curr_idx ≥ 0) bin[i mod NUM_BINS] = bin[curr_idx]
28) end for
29) curr_idx = passed_bins
30) 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(ind, 1, 0, fraction = 0):
31) delv = 0
32) delv = bin[1] mod NUM_BINS) × (1 - fraction)
33) delv = (bin[1] mod NUM_BINS) × (1 - fraction)
34) delv = (bin[1] mod NUM_BINS) × fraction
35) return delv

// Exit slow start by setting cwnd and ssthresh.
exit_slow_start():
36) cwnd = curr_idx - 2 × rtt / initial_rtt / bin_duration
37) overshoot = compute_delv(curr_idx, curr_idx)
38) cwnd = overshoot
39) ssthresh = cwnd

7/23/2024
Parameter Selection

• 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 quickly

• 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

---

Improving TCP Slow Start Performance in Wireless Networks with SEARCH
IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)
Perth, Australia, June 2024.
Outline

• Motivation (done)
• SEARCH (done)
• Performance Evaluation (next)
Performance – GEO Satellite

HyStart On

HyStart Off

SEARCH

7/23/2024
Performance – Wi-Fi

~15 MBytes

2 bytes when hystart off exit

HyStart

HyStart off

SEARCH

- Better Slow Start for TCP and QUIC
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) $\rightarrow$ 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)

https://search-ss.wpi.edu/
SEARCH – a New Slow Start Algorithm for TCP and QUIC

Jae Chung
Feng Li
Maryam Ataei Kachooeei
Mark Claypool

IETF CCWG
Vancouver, Canada
July 2024

Thank-you for your attention!
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