Intraflow Performance Diagnostics

Brian Trammell, ETH Zürich

QUIC Interim NYC, 19-20 September 2018
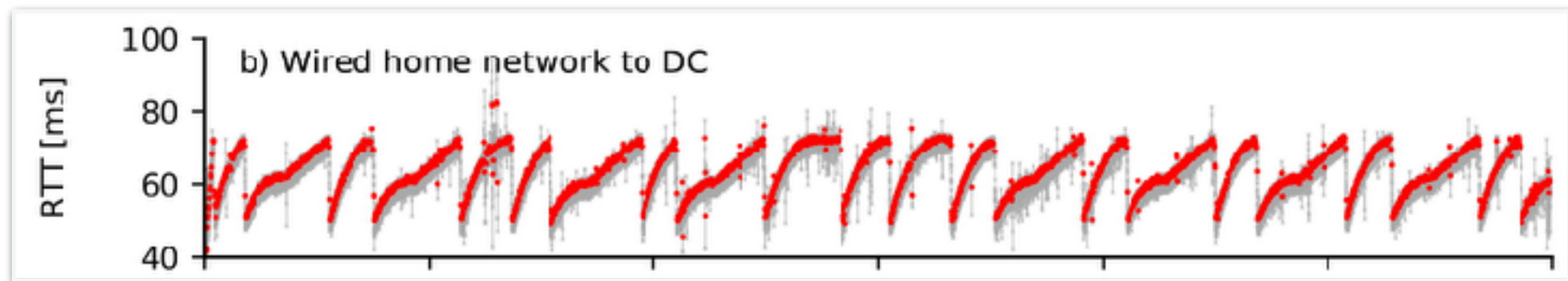
RTT measurement
(the story so far)

- Latency is a key metric for understanding network dynamics, so RTT measurement is useful on its own.

- TCP (accidentally) exposes information that can be used for passive RTT measurement in its wire image.

- Spin bit (or spin+VEC): minimal, intentional information exposure to provide equivalent RTT measurability in a QUIC-dominated, post-TCP world.

but wait, there's more!

- RTT is useful on a per-flow basis: transmission dynamics and buffering are visible.


- Other metrics (available via Wireshark) useful for per-flow diagnostics, too:

  - loss / congestion echo

  - instantaneous inflight and sender dynamics

- Can we support on-path measurement of these in QUIC

  - without adding more bits?

Throughput evoluton

Throughput for 10.226.35.191:56964 → 178.62.216.43:22 (MA)
Wi-Fi: en0
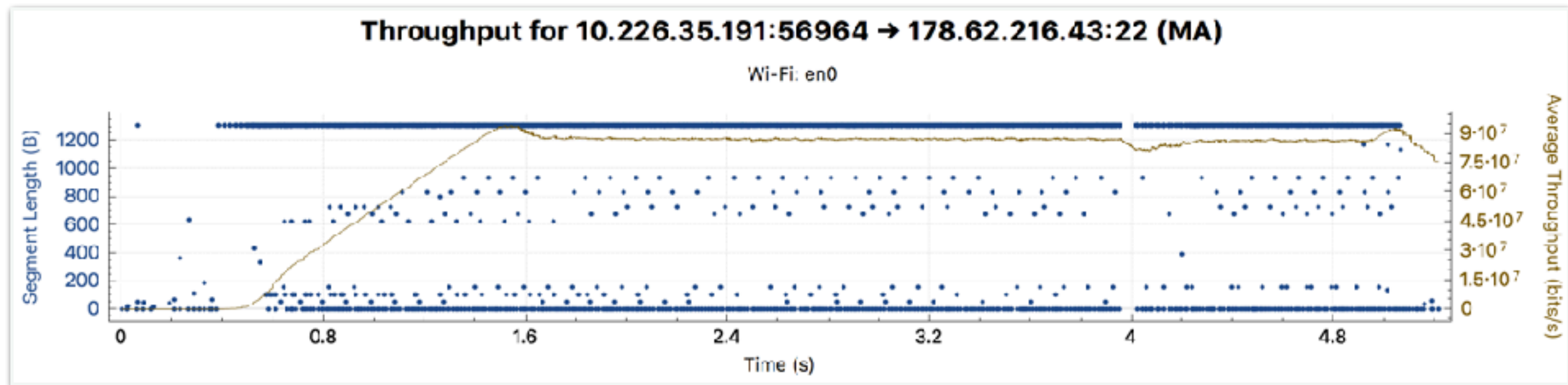
- Throughput evolution: count the size of (non-RTX) packets, divide by time.

- This analysis basically works with QUIC as-is, as long as RTX load is low.

RTT evolution
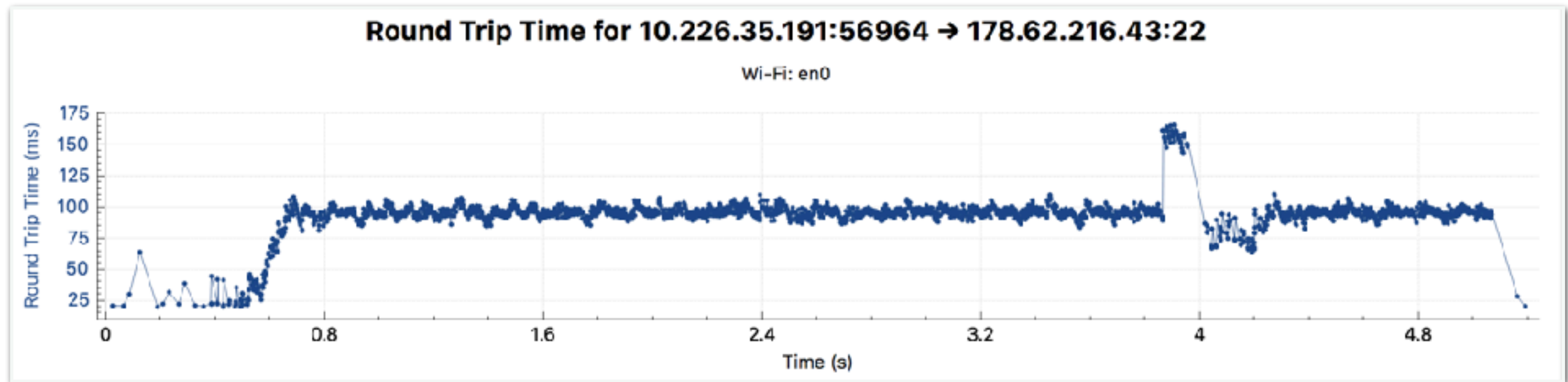
Round Trip Time for 10.226.35.191:56964 → 178.62.216.43:22

Wi-Fi: en0

- RTT evolution for QUIC requires the spin bit (or previous heavier-weight proposals, like PN echo).

- RTT evolution is already much more useful for answering network-focused *why?* questions than throughput.
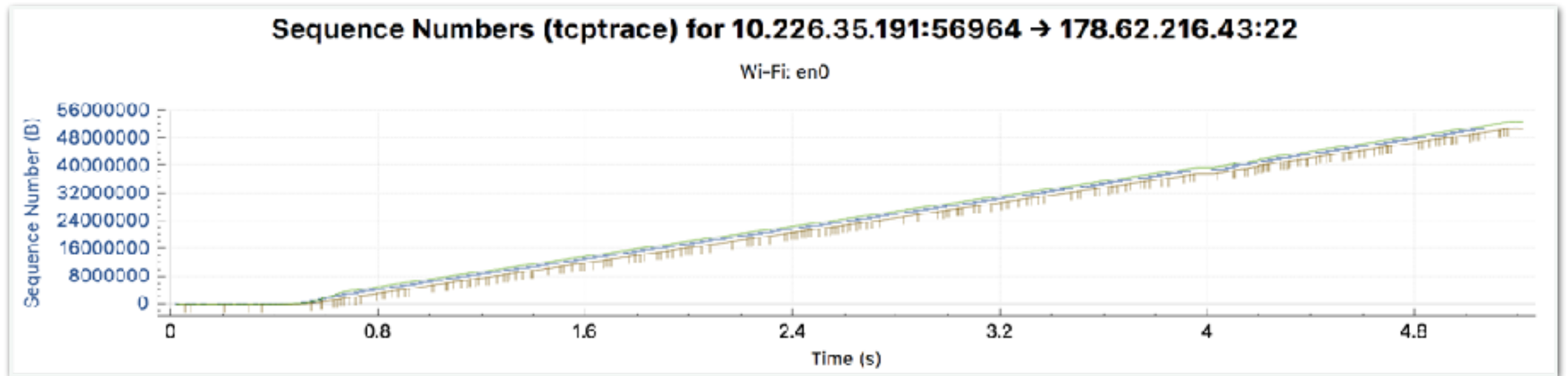
RTT evolution
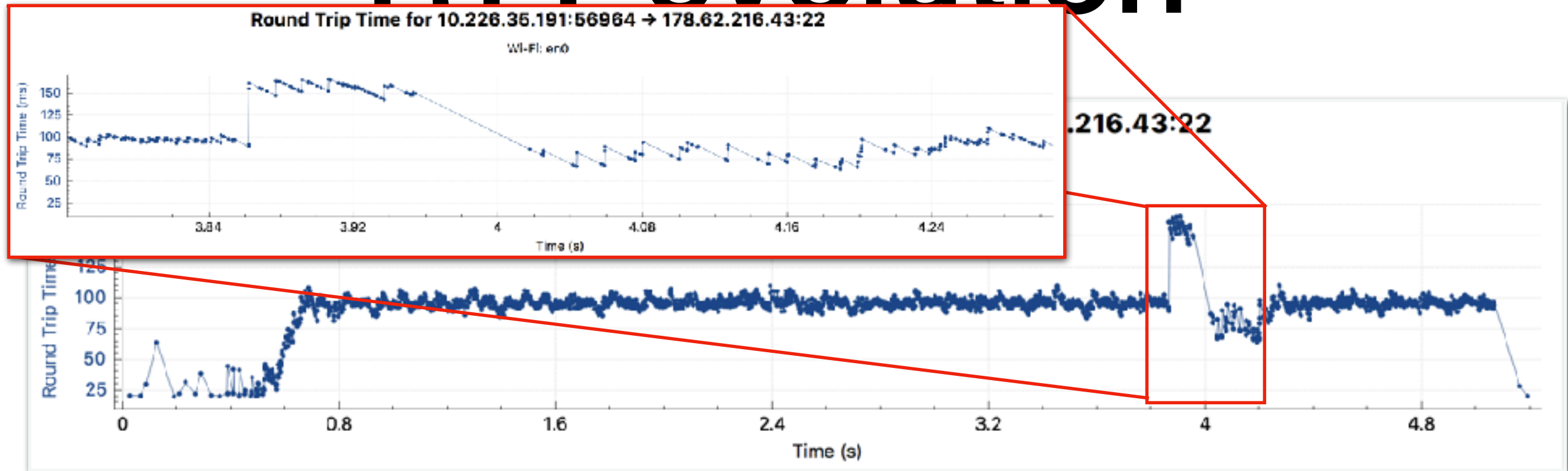
Round Trip Time for 10.226.35.191:56964 → 178.62.216.43:22

- RTT evolution for QUIC requires the spin bit (or previous heavier-weight proposals, like PN echo).

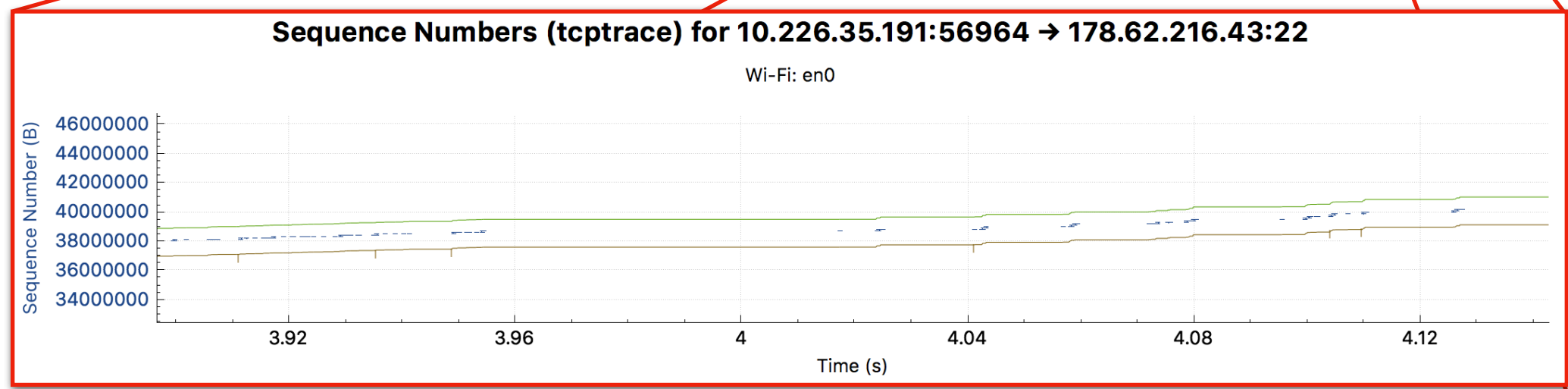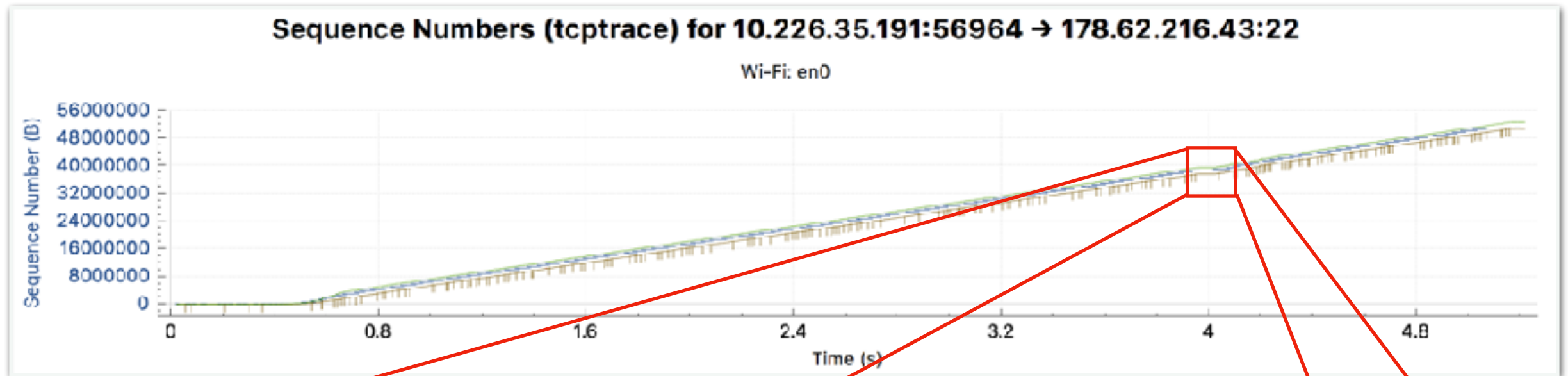- RTT evolution is already much more useful for answering network-focused *why?* questions than throughput.
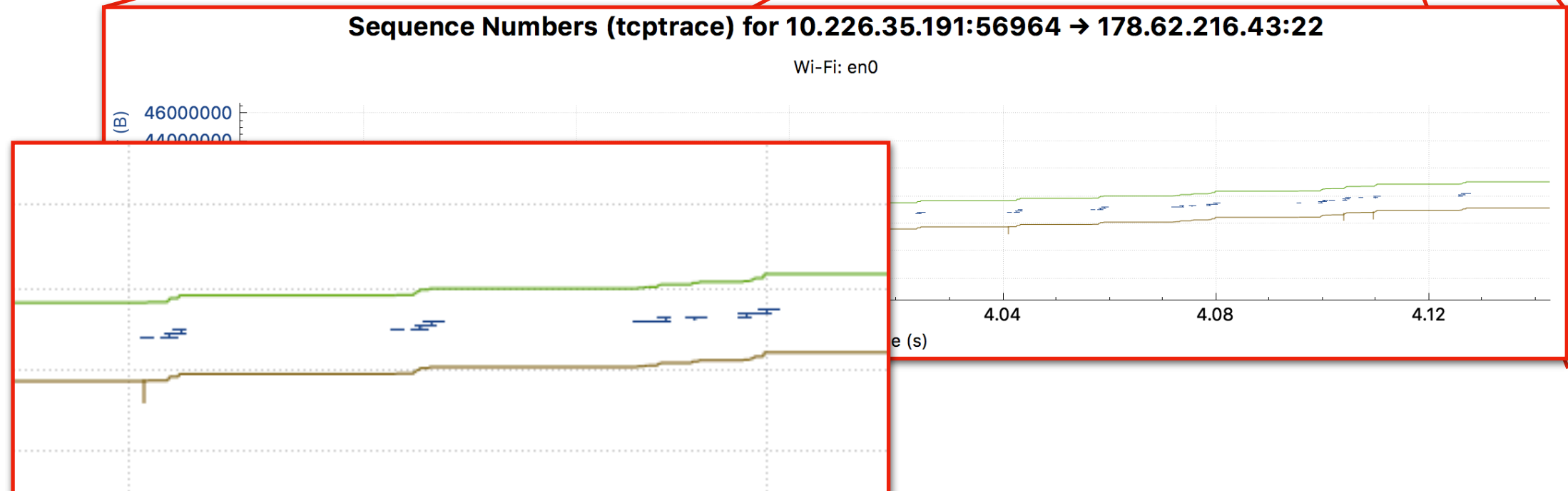
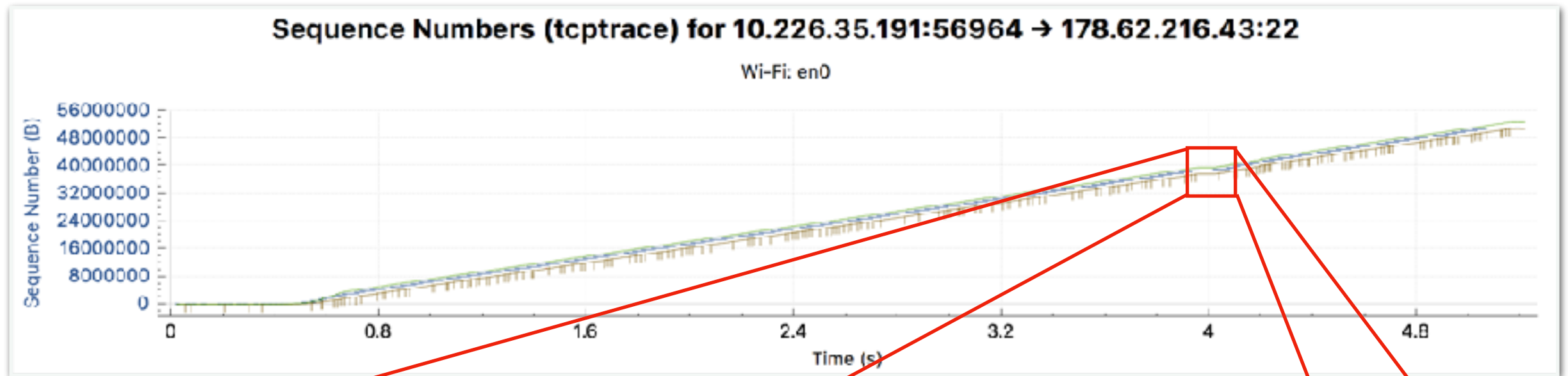# tcptrace
(or, how the TCP-savvy debug networks)

tcptrace
(or, how the TCP-savvy debug networks)

tcptrace
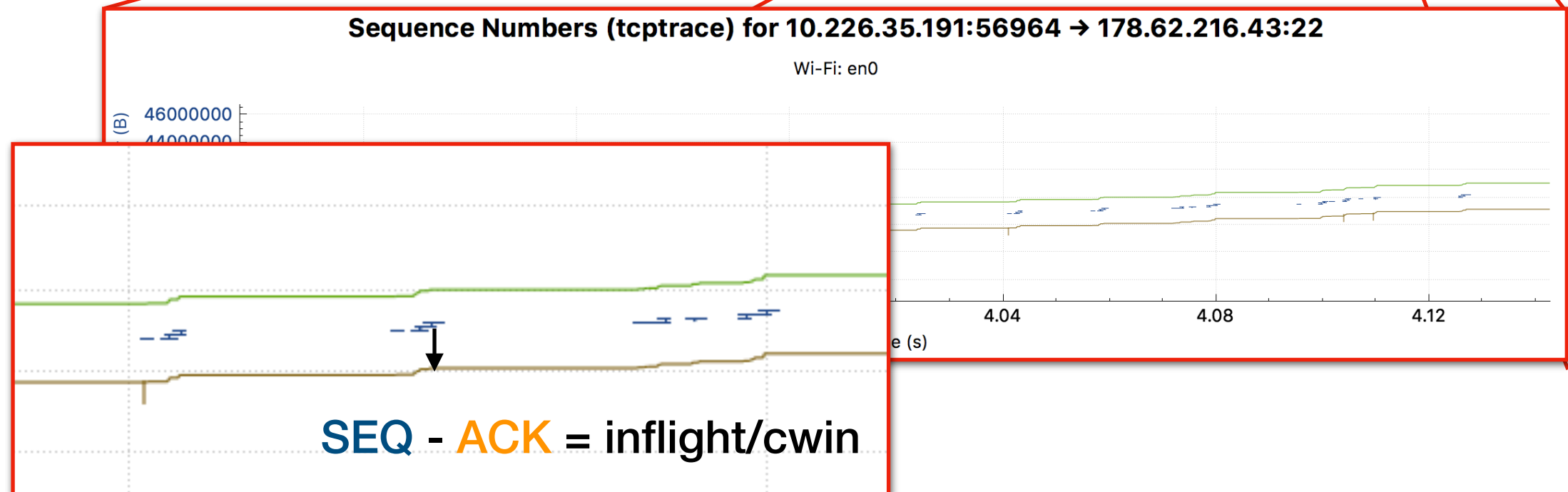## (or, how the TCP-savvy debug networks)

tcptrace
(or, how the TCP-savvy debug networks)

Sequence Numbers (tcptrace) for 10.226.35.191:56964 → 178.62.216.43:22

Wi-Fi: en0

SEQ - ACK = inflight/cwin

tcptrace
(or, how the TCP-savvy debug networks)

Sequence Numbers (tcptrace) for 10.226.35.191:56964 → 178.62.216.43:22

Wi-Fi: en0

WIN - ACK = rwin

SEQ - ACK = inflight/cwin

tcptrace
(or, how the TCP-savvy debug networks)

tcptrace
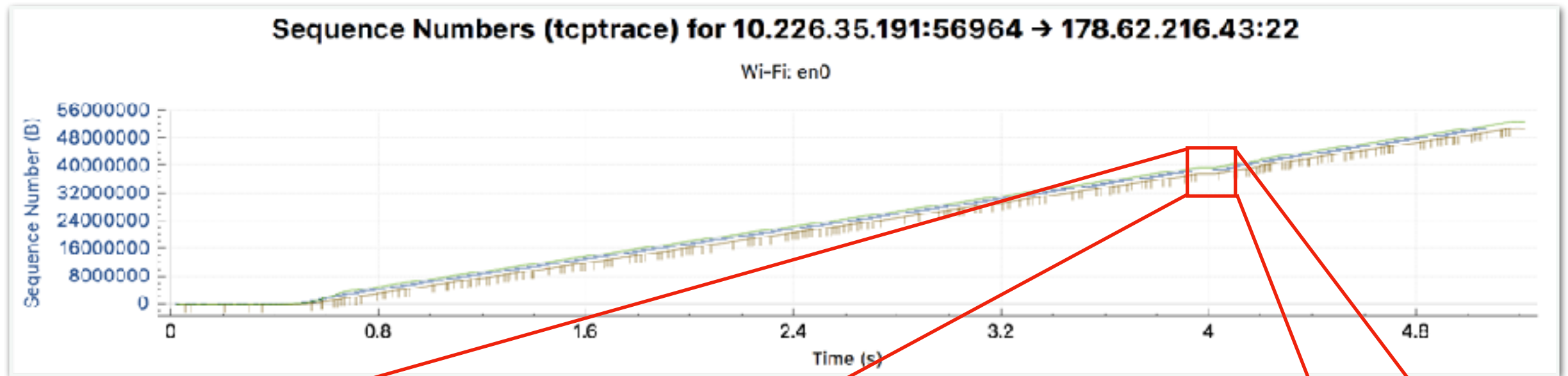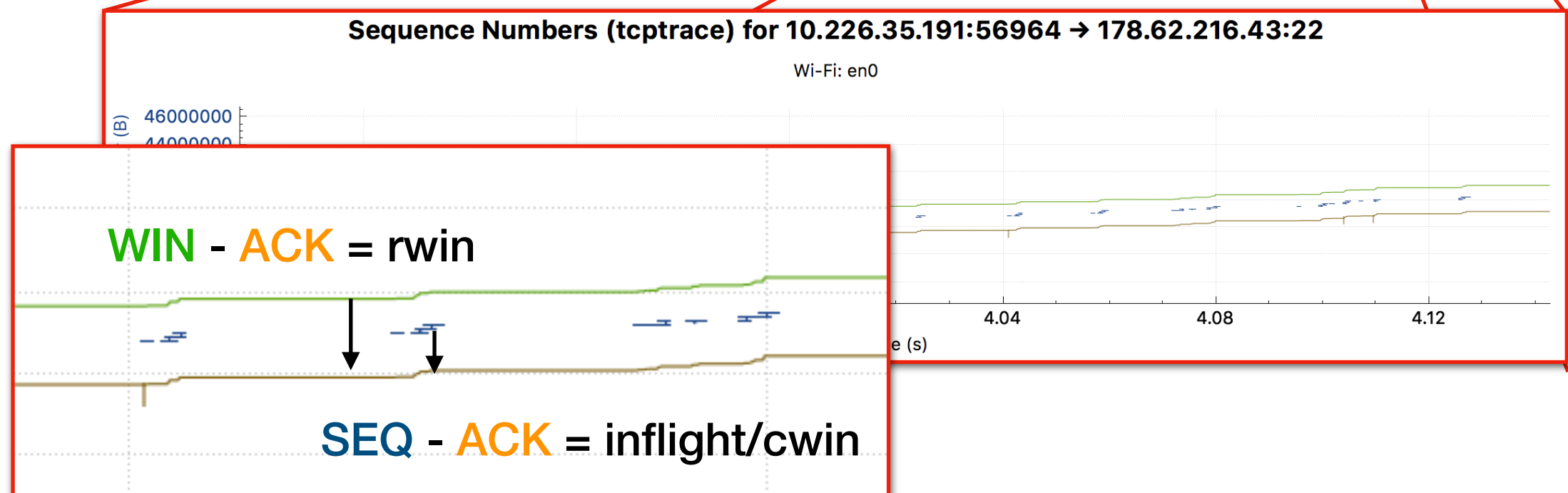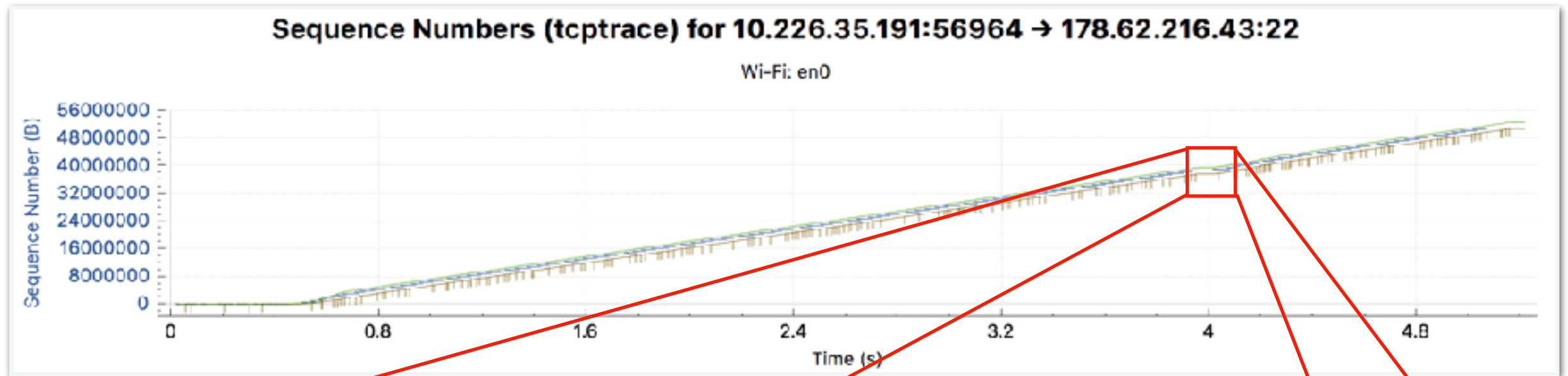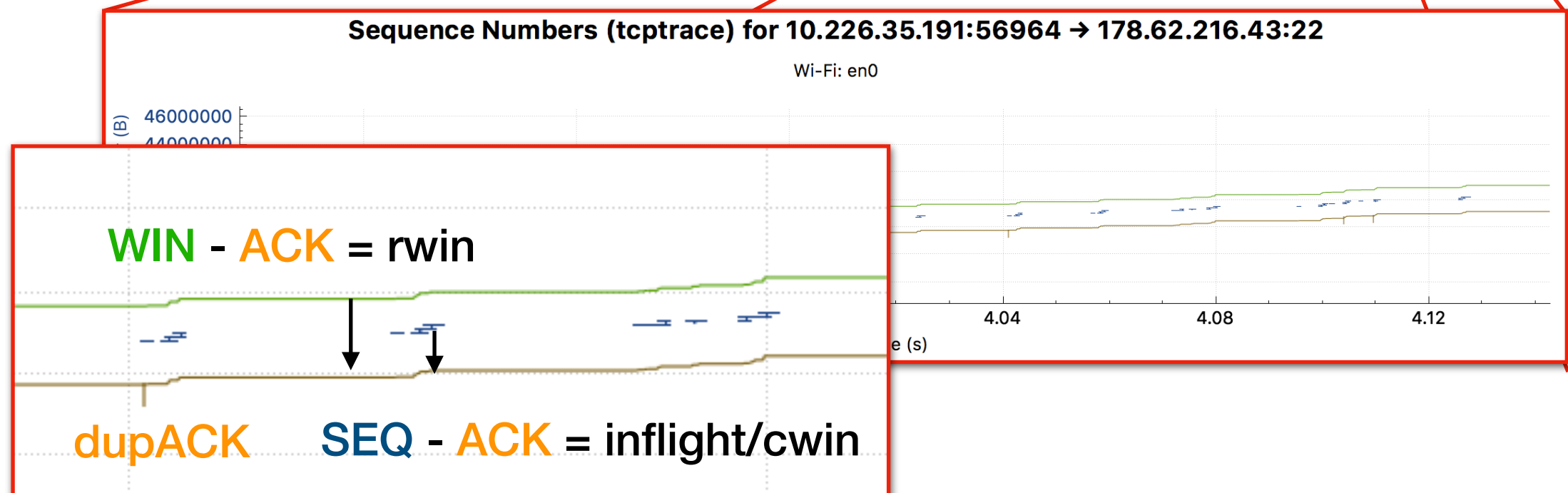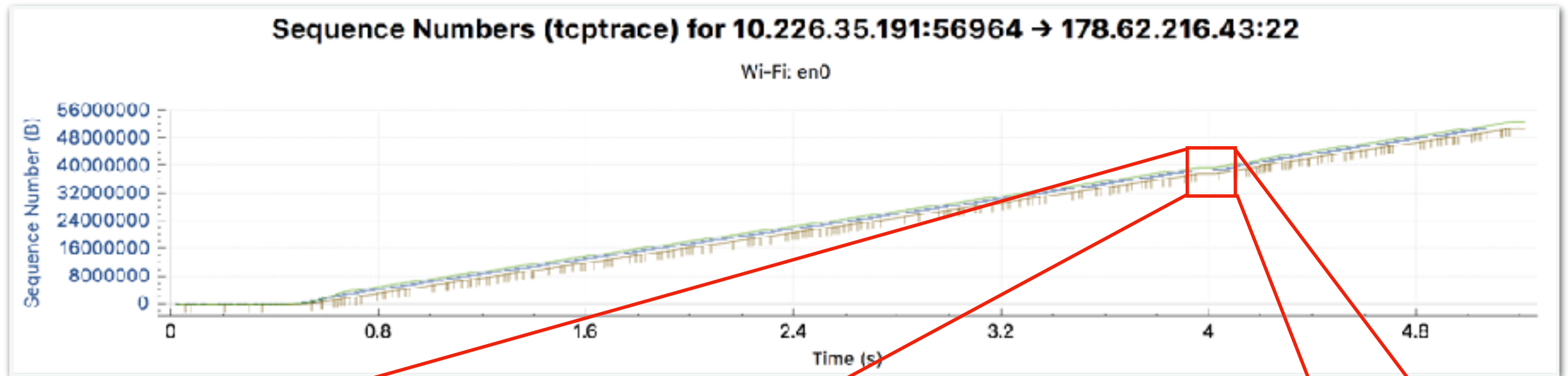## (or, how the TCP-savvy debug networks)

Sequence Numbers (tcptrace) for 10.226.35.191:56964 → 178.62.216.43:22

We can't do this in QUIC without exposing too much about transport internals.
Goal: a different view with equivalent utility.

Sequence Numbers (tcptrace) for 10.226.35.191:56964 → 178.62.216.43:22

Wi-Fi: en0

WIN - ACK = rwin

dupACK   SEQ - ACK = inflight/cwin
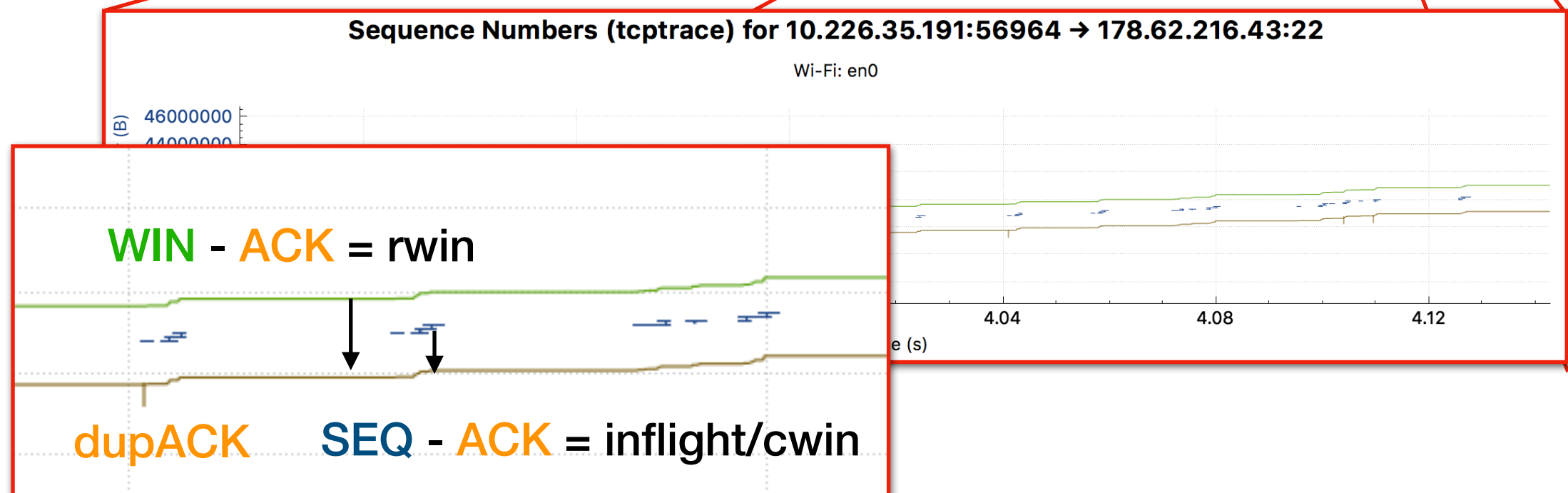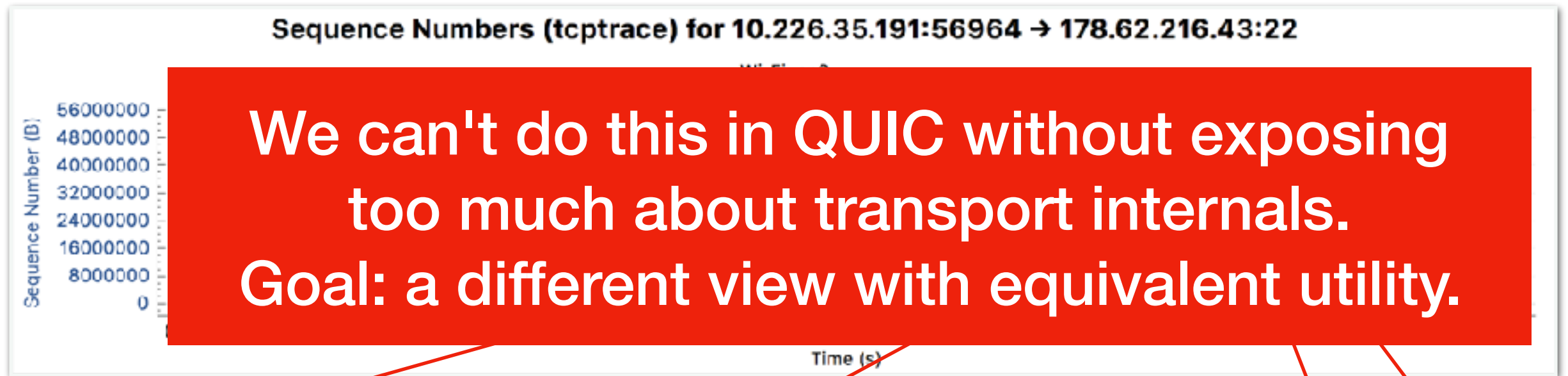
Questions to Answer

- Can RTT evolution + throughput evolution (i.e., inflight/cwin/pacing evolution) allow useful tcptrace-like analysis for QUIC flows?

- Can the VEC provide additional input to this analysis?

  - VEC sample rate is related to loss and reordering by

$$r_s = \frac{(1 - p_{L \cup R})^3}{RTT} \longrightarrow p_{L \cup R} = 1 - (r_s \times RTT)^{-3}$$

- These (and related questions) are the subject of an MSc thesis at ETH to run October 2018 - April 2019

A Way Forward

- The *fitness for purpose* of the spin bit seems certain.

- VEC / other approaches not as certain before Bangkok.

- So let's accommodate continued experimentation as well as early deployment of one-bit spin:

  - Server SHOULD set bit S to the value of bit S received on highest-numbered packet from client.

  - Client behavior undefined in -transport, MUST grease if no other purpose.

  - Reference to draft-*-tsvwg-spin for client-side RTT behavior.

  - Remaining two measurement/experiment bits held for experimentation, MUST grease if not experimenting.