POLLERE, INC.
network and architectures analysis

Listening to Networks: Transport Headers Tell All

Google
April 12, 2017

Kathleen Nichols

Some material in this presentation is based upon work supported by the Department of Energy under Award Number DE-SC0009498.

Talk roadmap:

- Packet headers are network diagnostics
- Diagnostic opportunities and challenges
- Pollere's results

Takeaway: Packet header data mining is a frontier

Don’t probe, *listen*

Header information (sequence numbers, timestamps, …) helps transport endpoints figure out what the network has done to their packets.

Network engineers want to know the same thing. Analyzing information from passive packet header capture results in diagnostics that:

- see everything that applications experience
- cover all the infrastructure applications use
- can be deployed anywhere and everywhere without requiring cooperation, coordination, additional agents or extra bandwidth.

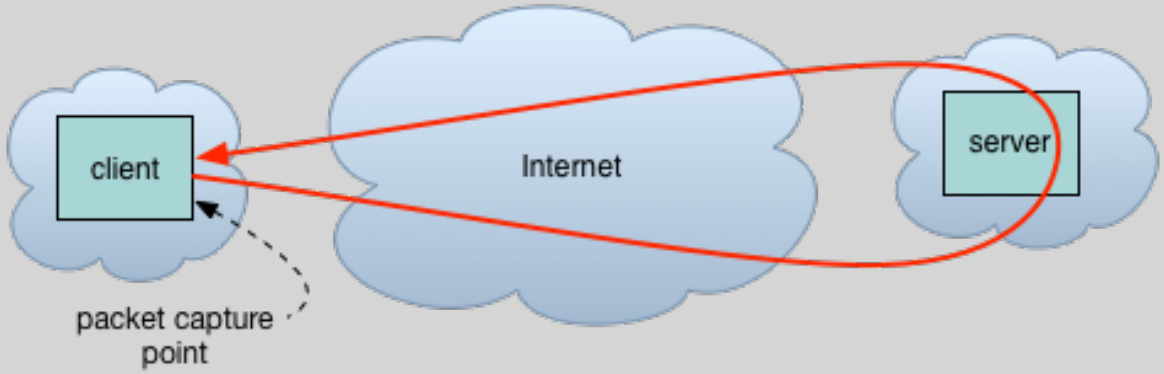
Passive Measurement in Network Performance

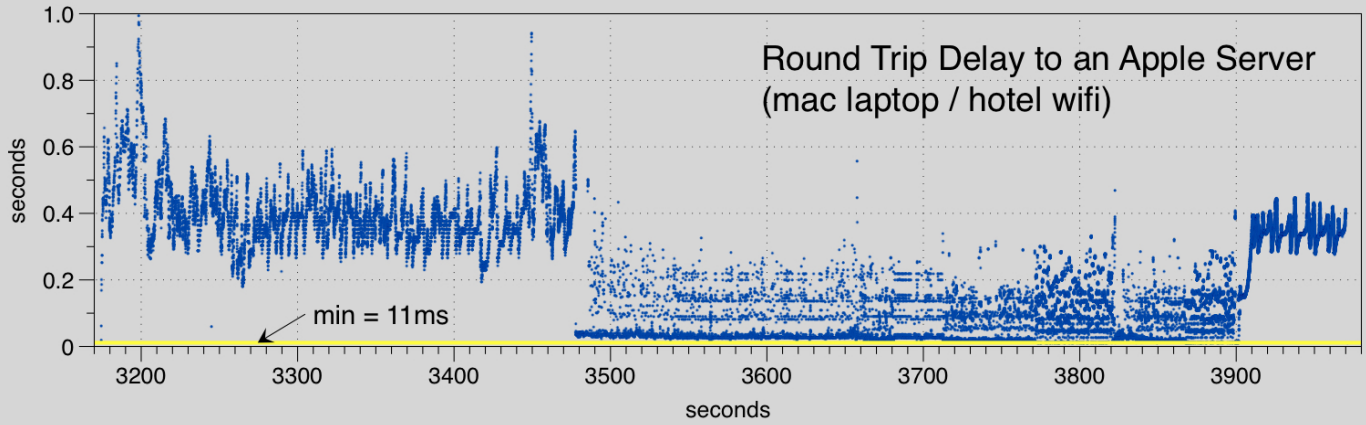
- Historically “network performance data” has meant SNMP counters and netflow samples, augmented with active probing.
- Recently focus has shifted to deriving data from passive packet header monitoring or packet marking active/passive hybrids.
- Tools to do this generally require non-standard switch capabilities, co-location with servers, availability of both flow directions and/or ability to snag all (or all SYN/FIN/RST) packet headers.

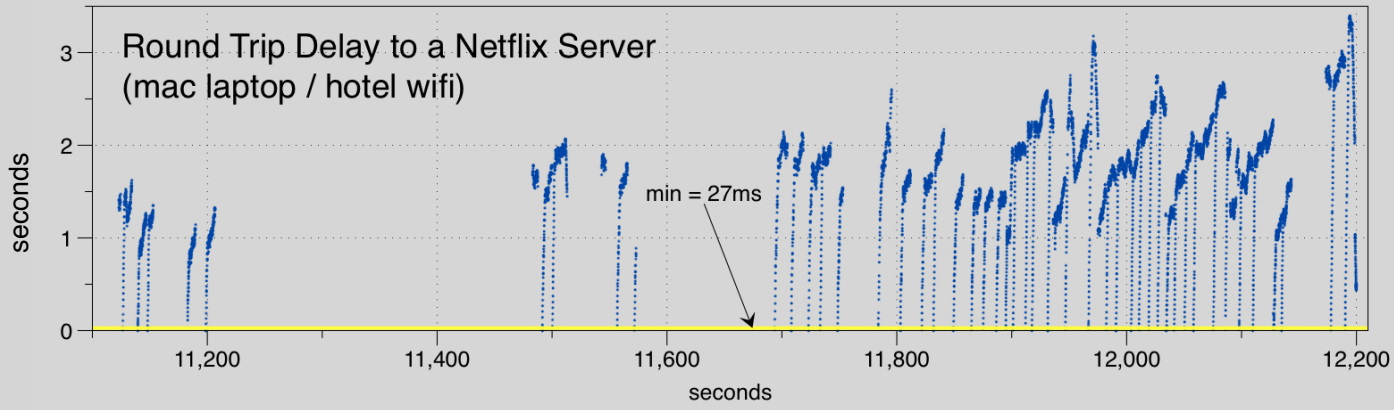
But packet headers can be captured anywhere and are rich in information, much of which is robust to sampling and doesn’t require both flow directions.

Round Trip Time at Endpoints

- RTT is an important transport diagnostic
- **ping** tries to capture the same information via active probing
- but packet capture shows what the transport sees







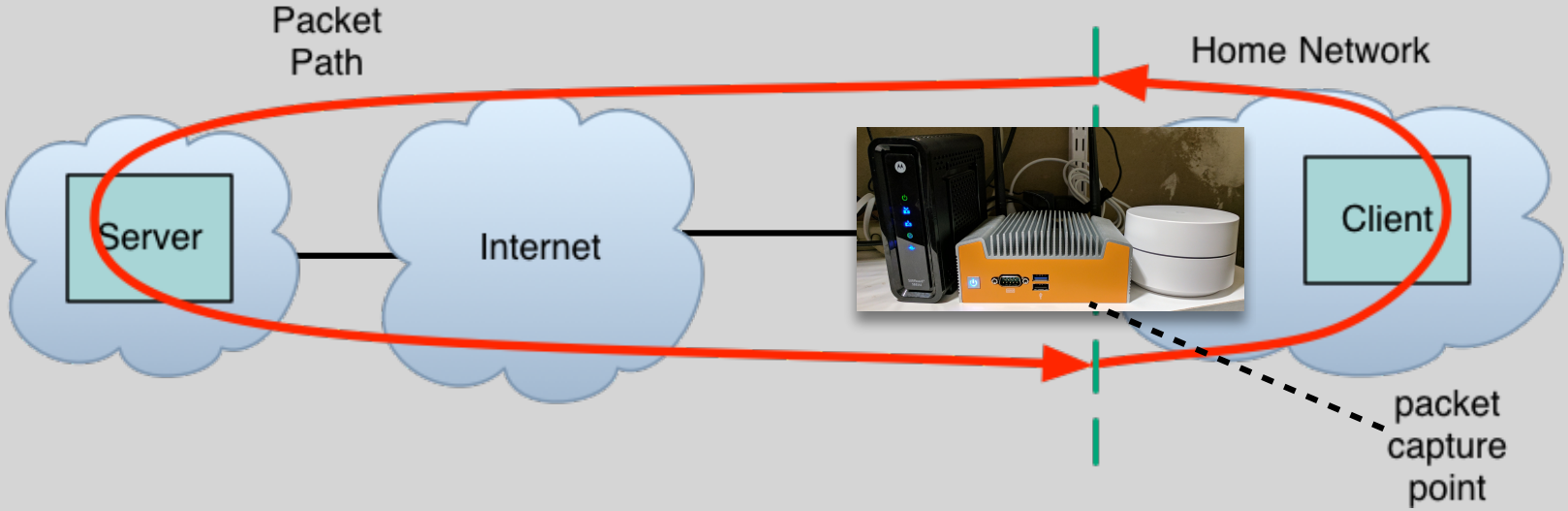
From Packet Traces to On-the-Fly Processing

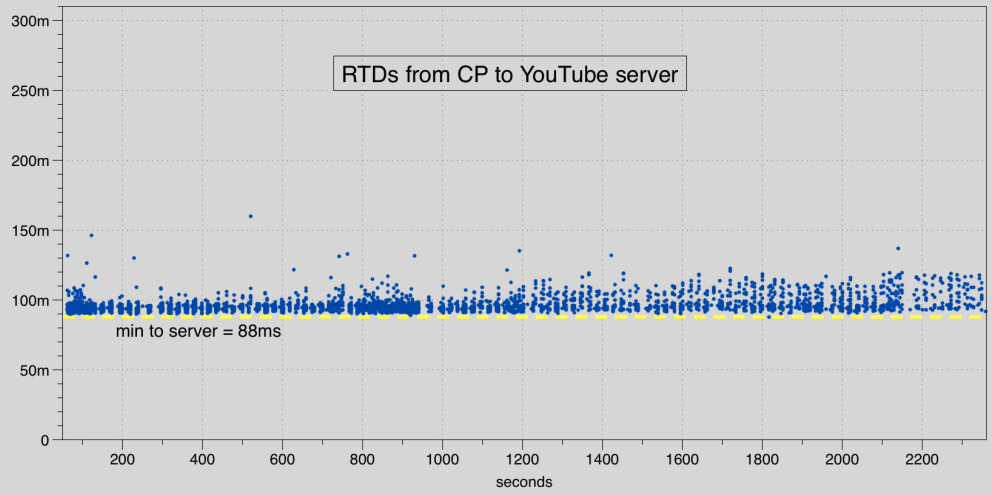
- An independent device allows true “anywhere” operation
- Pollere’s current platform is an Atom-based Linux box deployed as a “bump in the wire” bridge
  - can handle bi-directional line rate GigE or full 802.11ac WiFi
  - expect a Raspberry Pi could handle a residential link; downlink of 200Mbps using a 128 byte snaplen requires 2MBps/direction

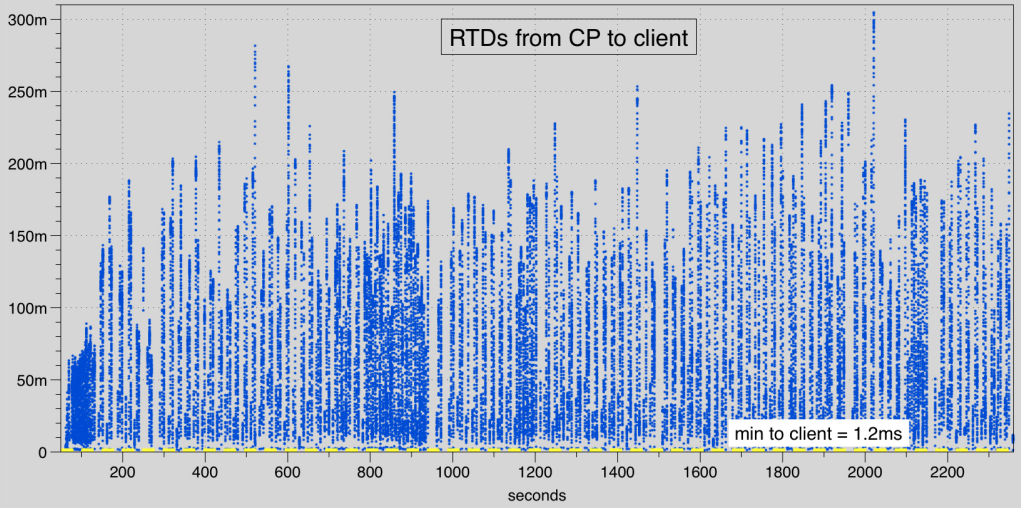
In addition to capture, need enough compute power to process packet headers, store data, and potentially stream metrics through WiFi port

Move Packet Capture into Network

As long as the capture point sees both directions, we can bifurcate delay, creating a “passive ping” between the CP and a host

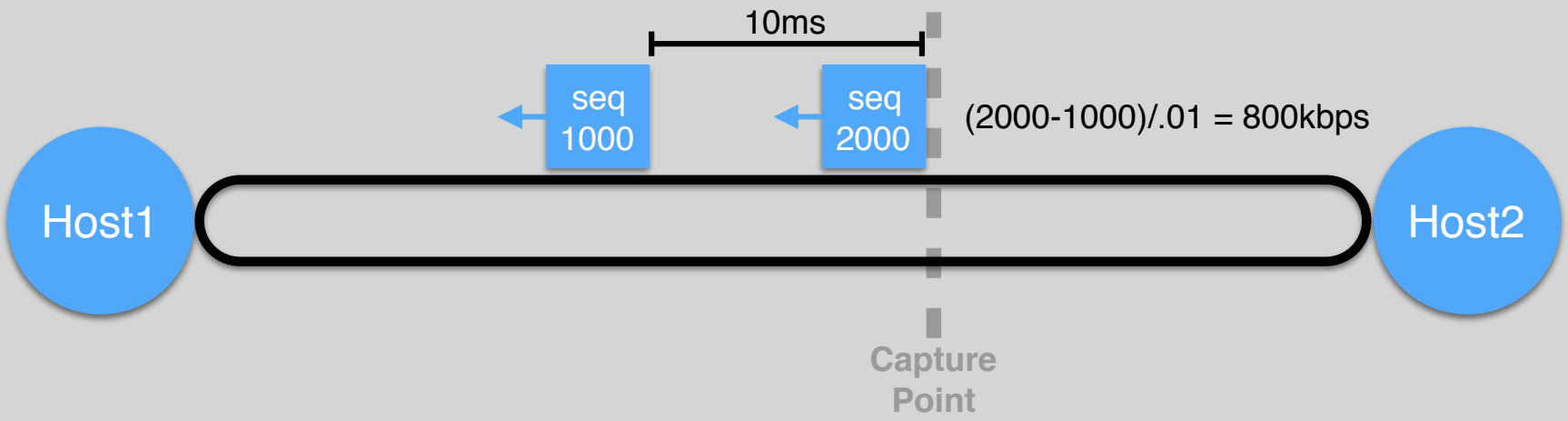




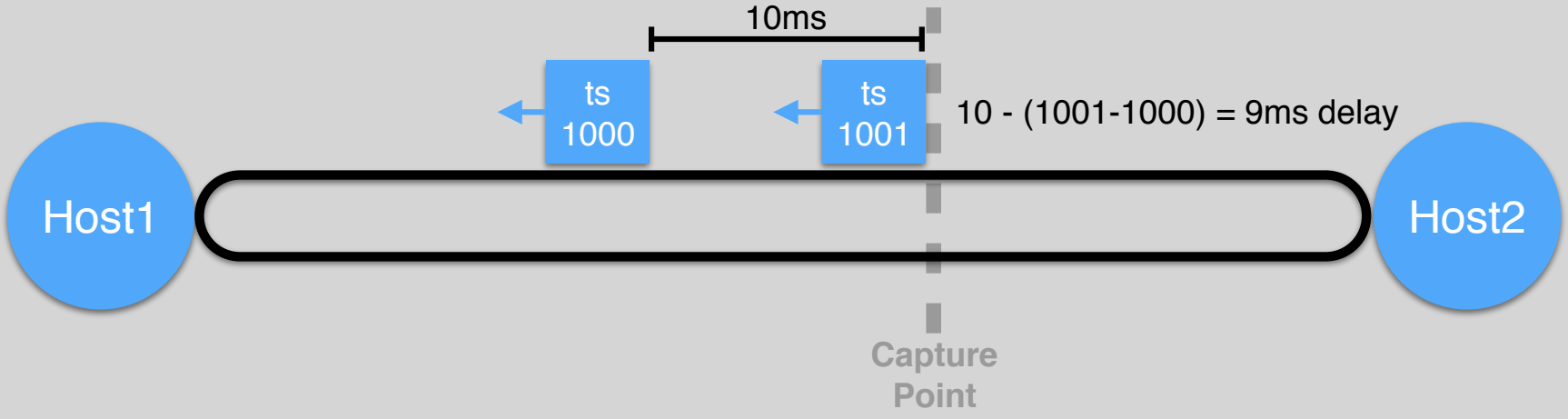


Even in this second tier ISP, home delay (variation) swamps Internet delay!

Packet Headers Have More Stories to Tell



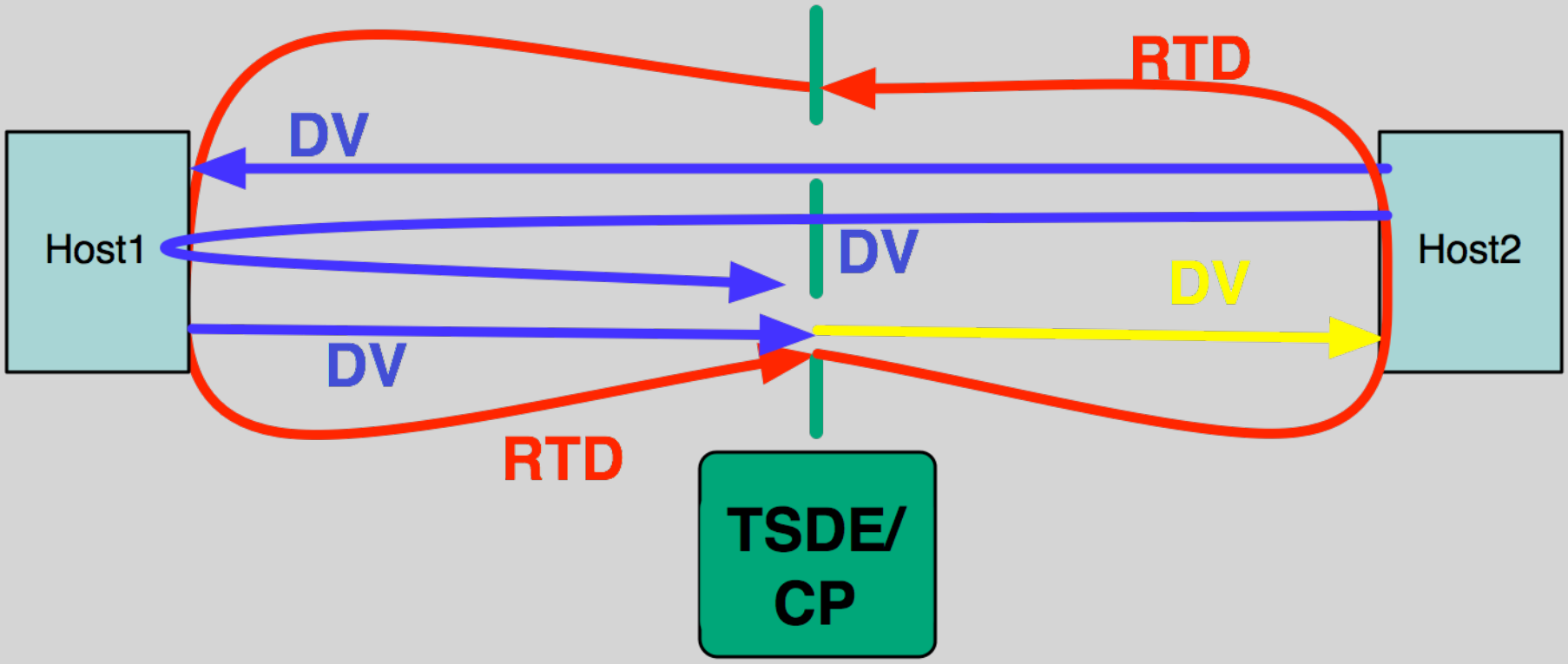
seqno and capture time gives connection's data rate: no need to see every packet or the return path



TSval gives a low resolution time of packet departure from sender; this scenario indicates delay upstream of the CP

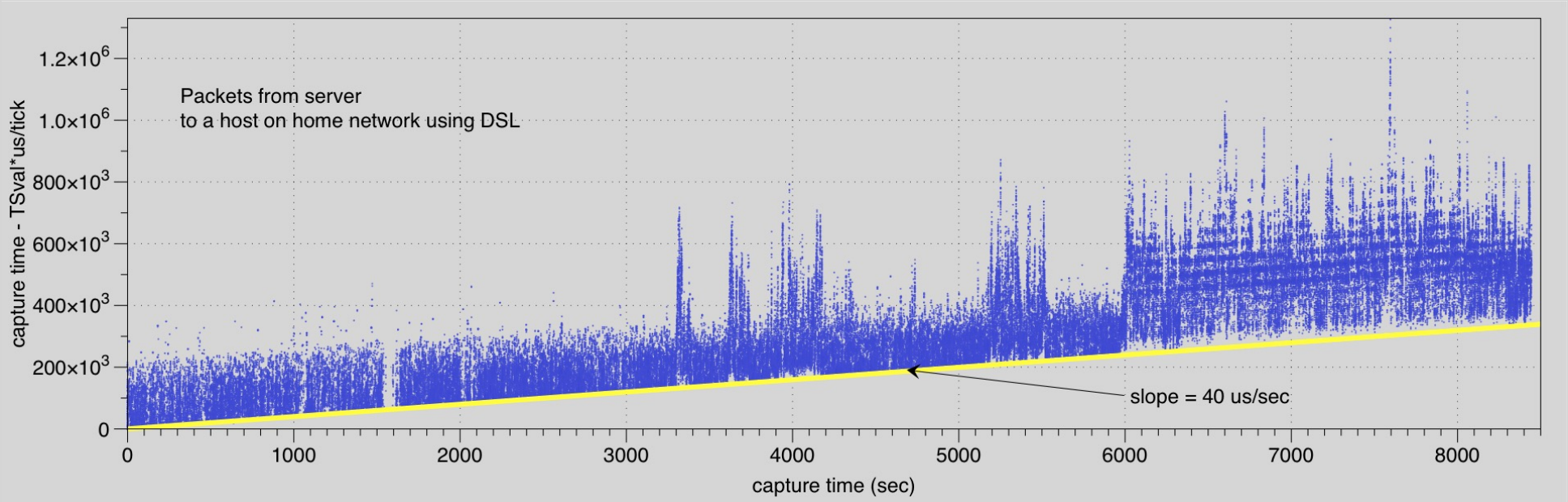
Pollere's Focus: Transport Segment Delay Estimation

- Every packet provides delay estimates for several path segments
- Packet header data can be used to localize delay (relative to CP)
- Blue lines are metrics from unidirectional packet flow



Challenge: Clock Issues

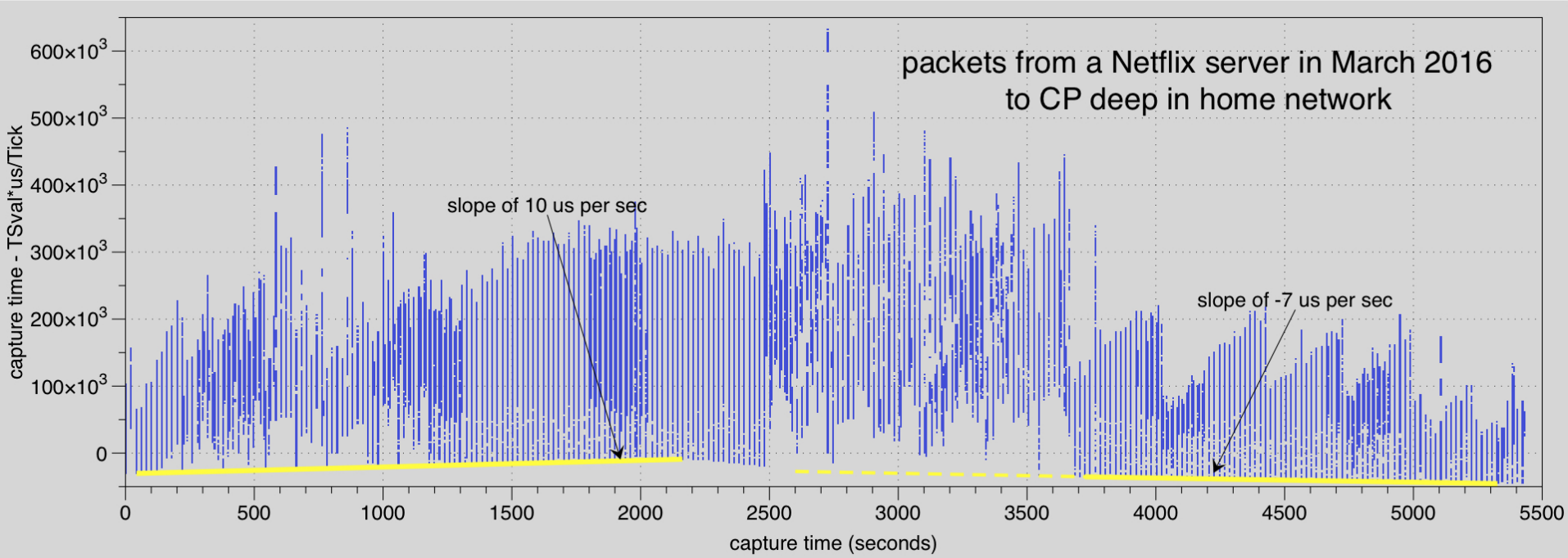
- In a world of synchronized clocks, the delay variation of a packet is the capture time minus the sender time as recorded (coarsely quantized) in its TSval.
- In this world, we can extract the ms per TS increment and (often) the relative skew from the packet header stream to convert the TS ticks to time and plot the capture time vs this delay variation



- This server used 1ms per TS increment or “tick”.
- Delay from the server to the capture point was frequently in the 100s of milliseconds.

Challenge: More Clock Issues

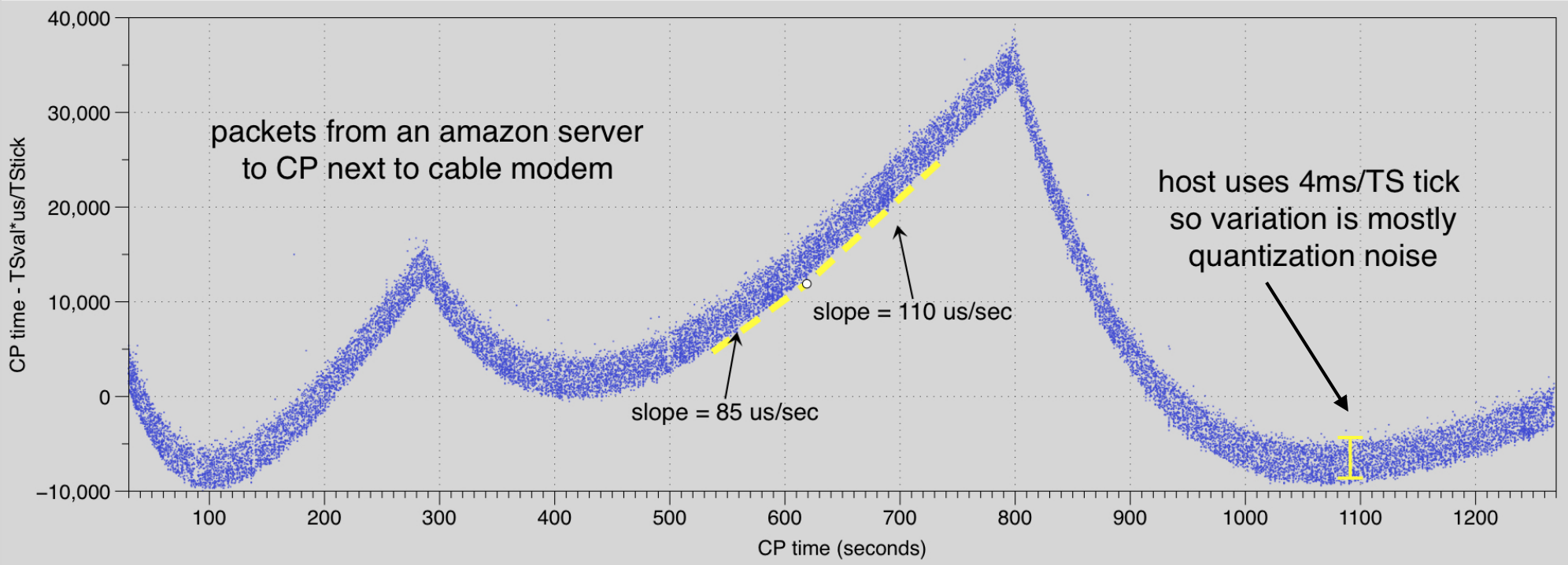
- Even on the same flow, there can be events that affect the relative skew
- Often, as below, the skew magnitude and sign both change



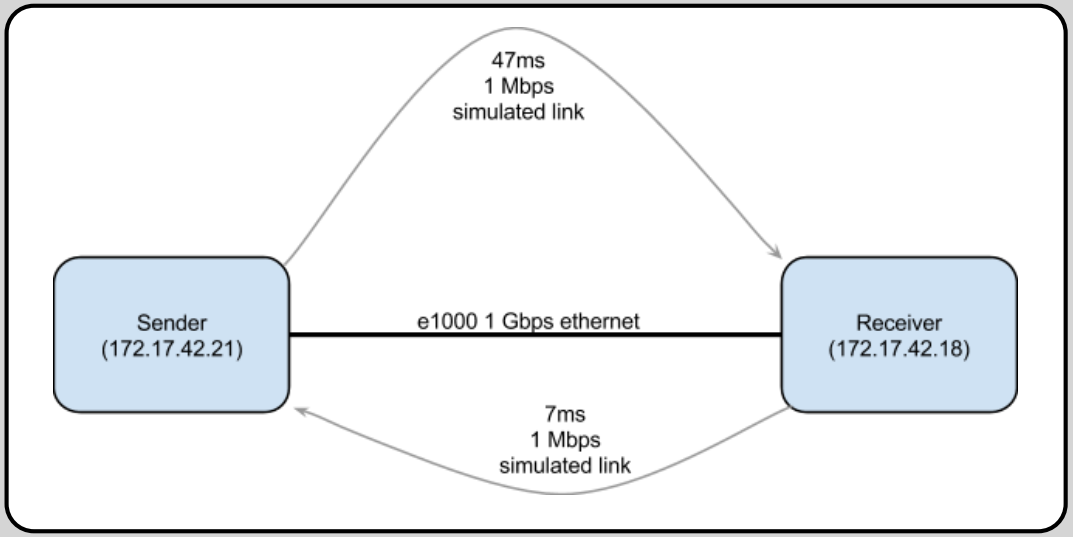
- This server used 1ms per TS tick
- Delay from the server to the capture point was frequently in the 100s of milliseconds so quantization noise is not significant

Challenge: Serious Clock Issues

- Sometimes, it's even worse!
- A human can look at this data and see that the variation is likely due to thermal effects but it's pretty challenging for an automated approach
- Opportunity: algorithms and filters to detect and follow skew



Challenge: Validation



After simulation studies, moved to Pollere's lab.

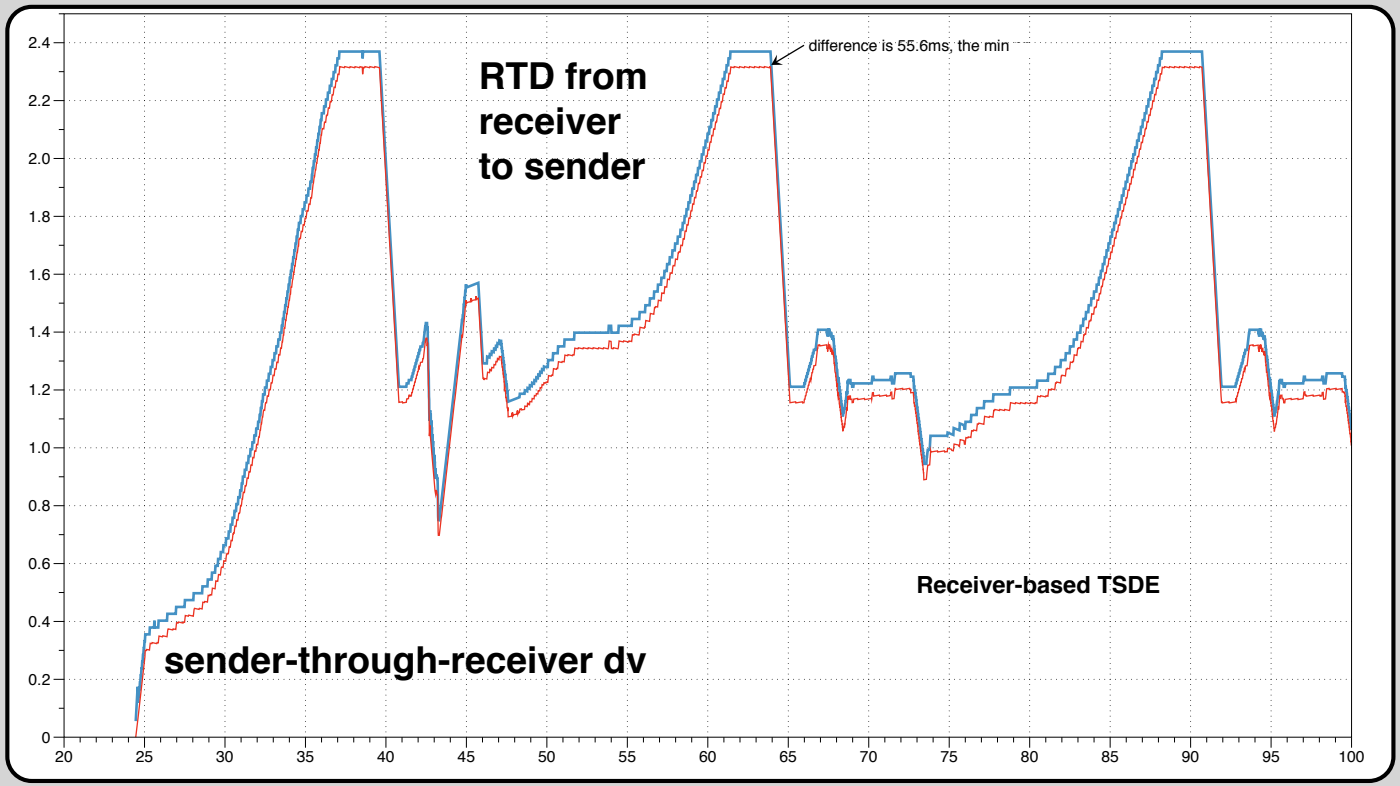
Linux *netem* emulates 1Mbps link with 47ms of delay in the sender-to-receiver direction and 7ms of delay in the receiver-to-sender direction

A single TCP connection bloats the buffer.

RTD determined by "passive ping"

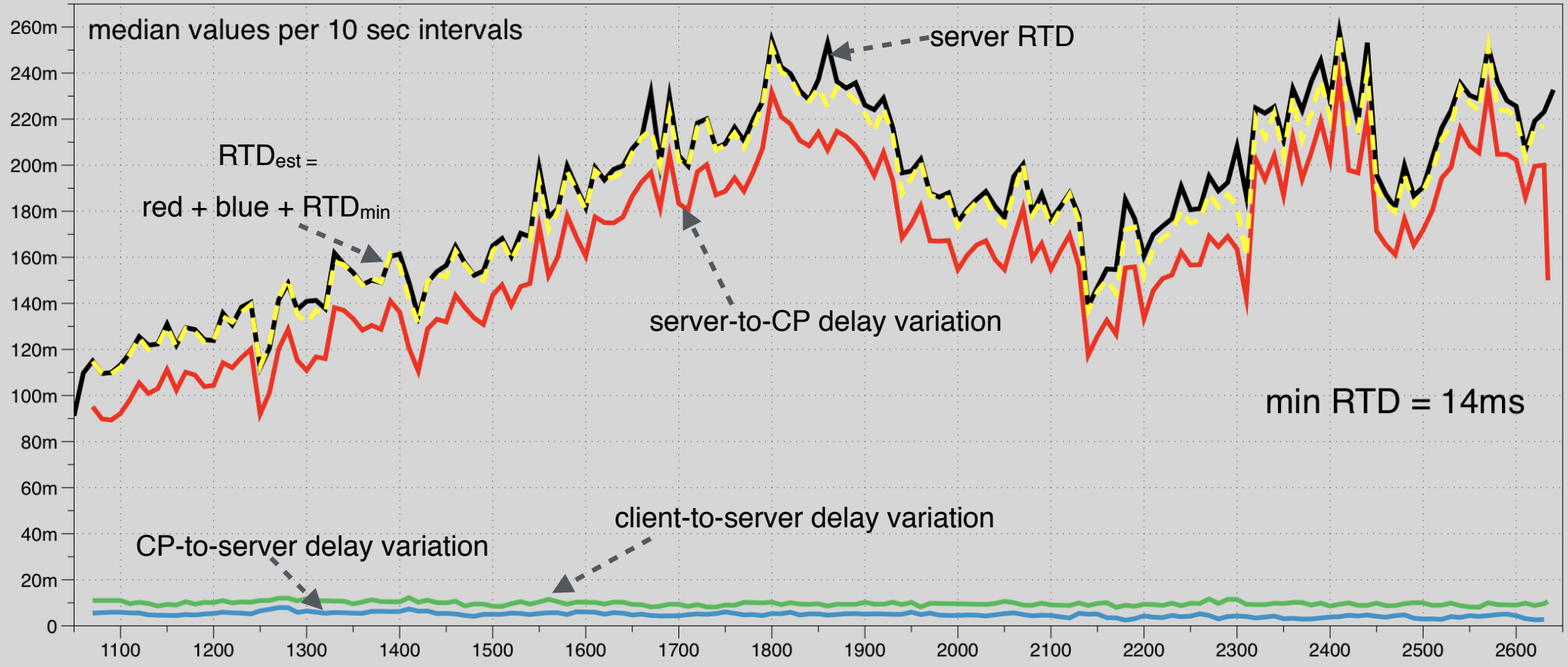
DV metric uses only one direction's packets.

Differs from the RTD by 55.6ms, the min RTD.



Validation on Home Network Data

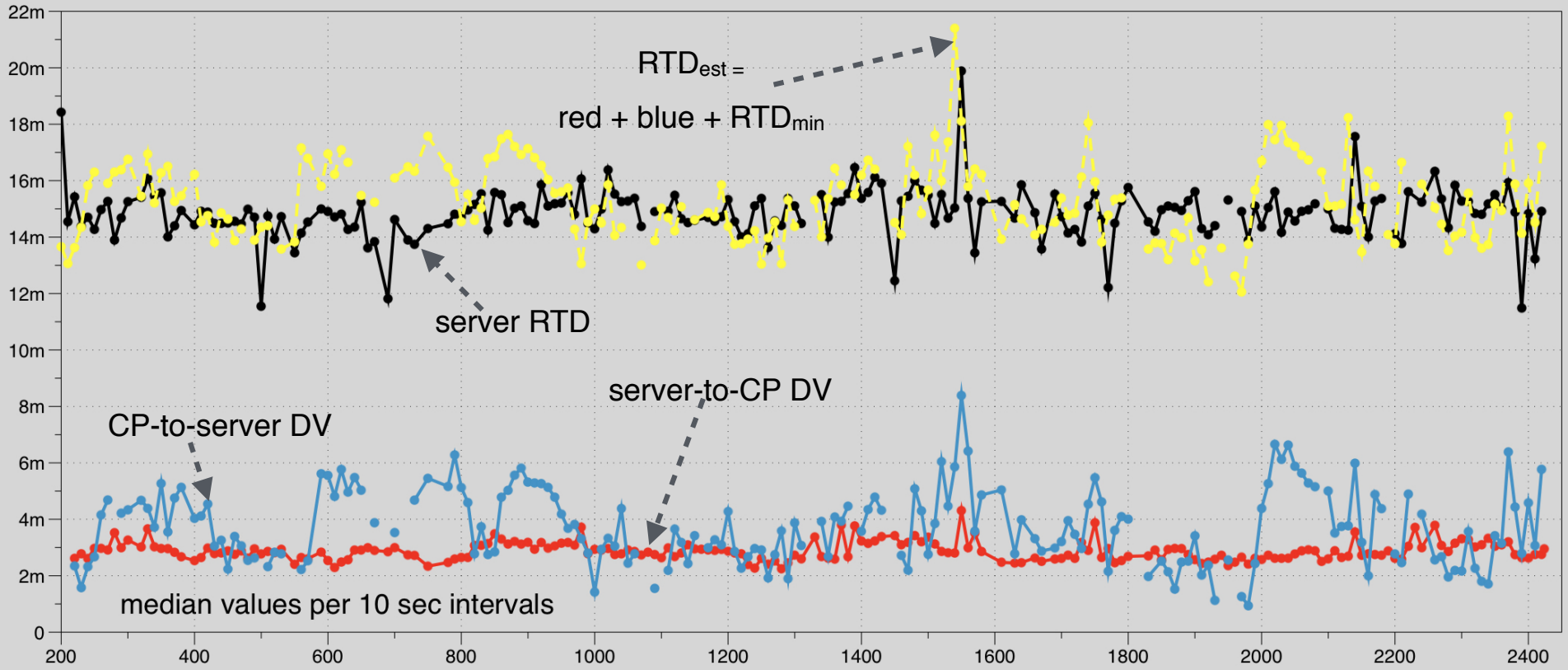
Check consistency of uni-directional and bi-directional metrics



- Home wireless and wired network lies between CP and server
- The large delay variation appears to be in server to CP direction (but likely in the home network)

Validation on Home Network Data

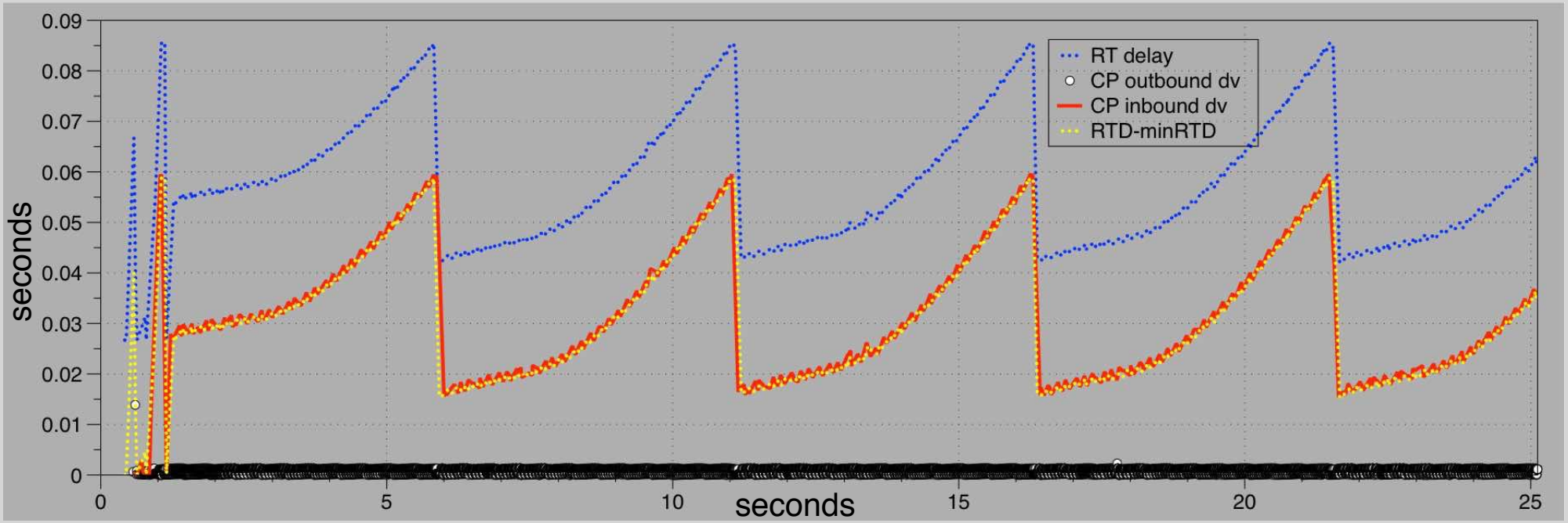
Check consistency of uni-directional and bi-directional metrics



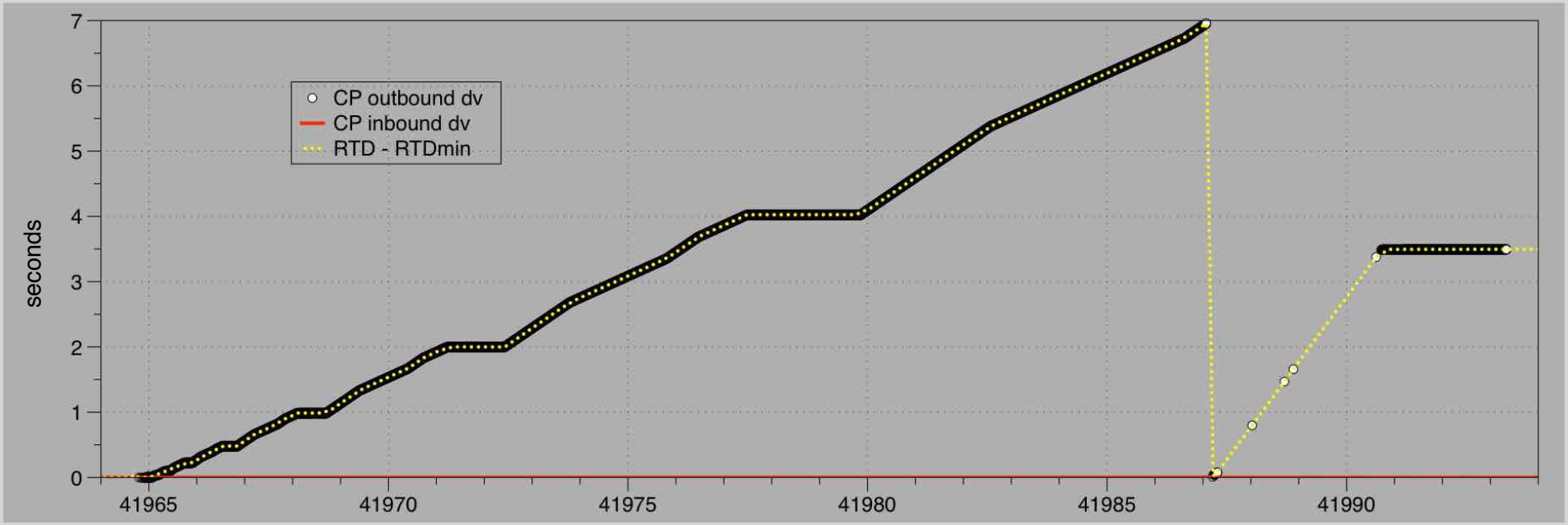
- CP is next to cable modem; Google wifi between client and CP
- Client delay is mostly delay variation (min RTD is 0.9ms) and appears to be outbound from the wifi router but 10ms TSval “ticks” obscure this
- Useful TSvals please!

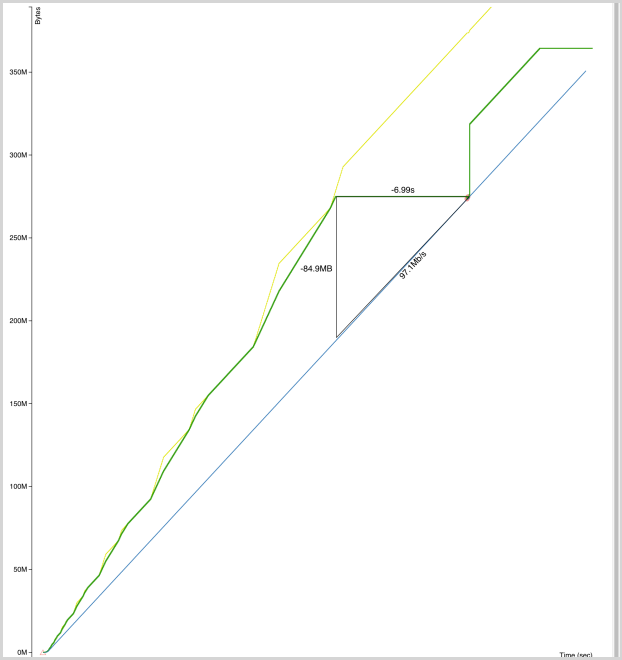
Validating with Research Network Data

- Portion of a TCP connection
- Outbound, CP-to-host is the ACK path with small delay variation
- Inbound delay variation has ~16ms of standing queue over the connection lifetime. There is a minimum RTD of 26.7ms over the connection lifetime (at start)
- Losses occur whenever the queue delay reaches 60ms, indicating a 750 kbyte buffer in the path (.06 sec x 100 Mbps / 8 bits/byte)



Later, in the opposite transfer direction



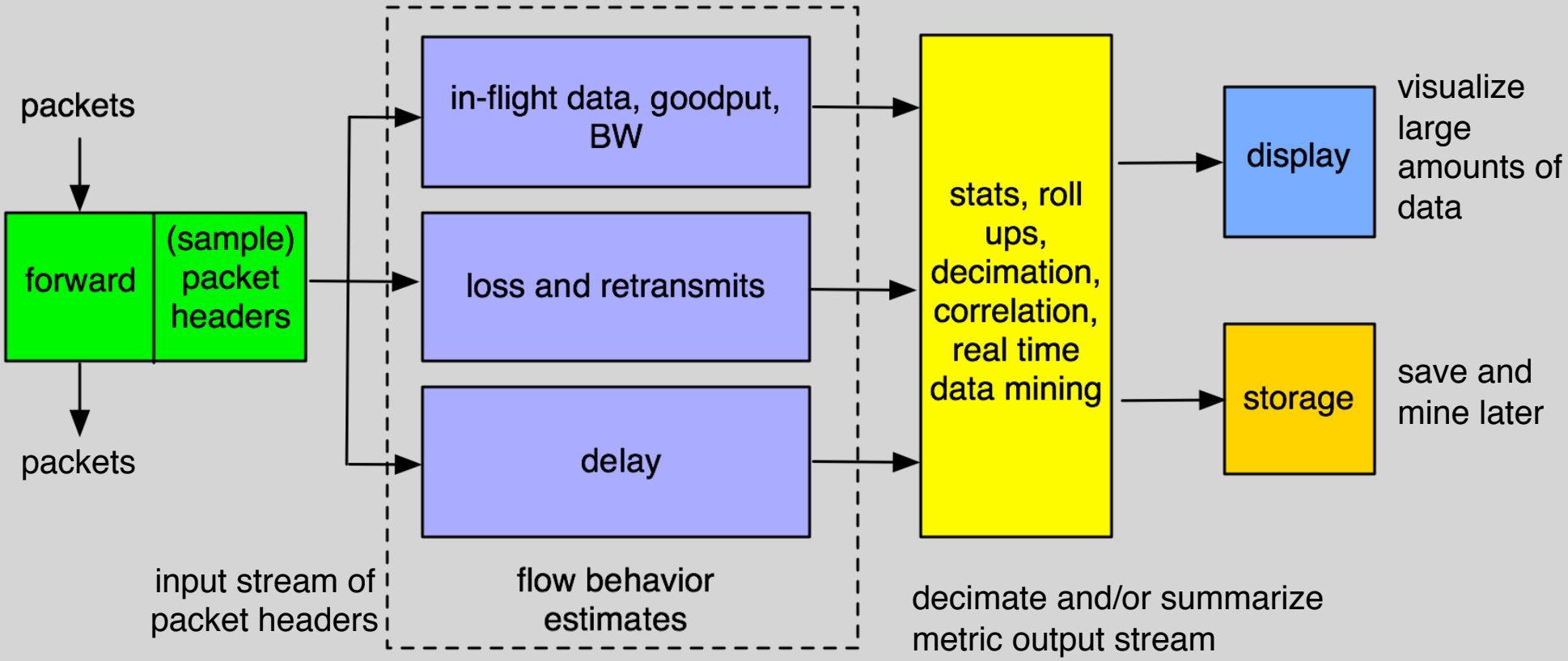


Asked Van Jacobson to take a look to see if this delay was “for real”

His TCP analysis tool showed it’s really there

Challenge: A Lot of Data

- Per-packet metrics can mean a **lot** of data
- Opportunity: address this with a good “bag of tricks”

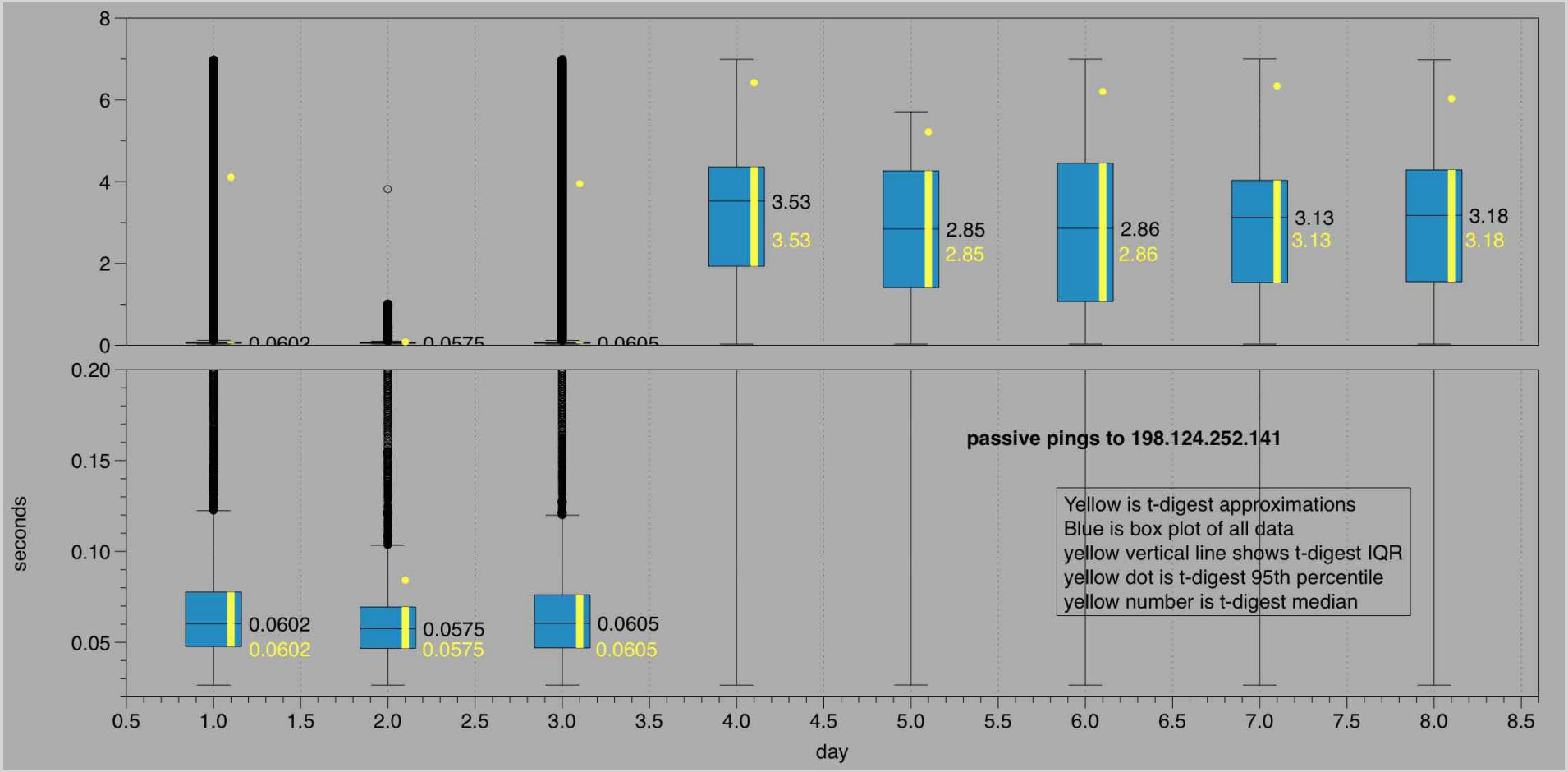


In the Bag of Tricks: Statistics on the Fly

Use *tdigest* algorithm to estimate CDF on the fly.

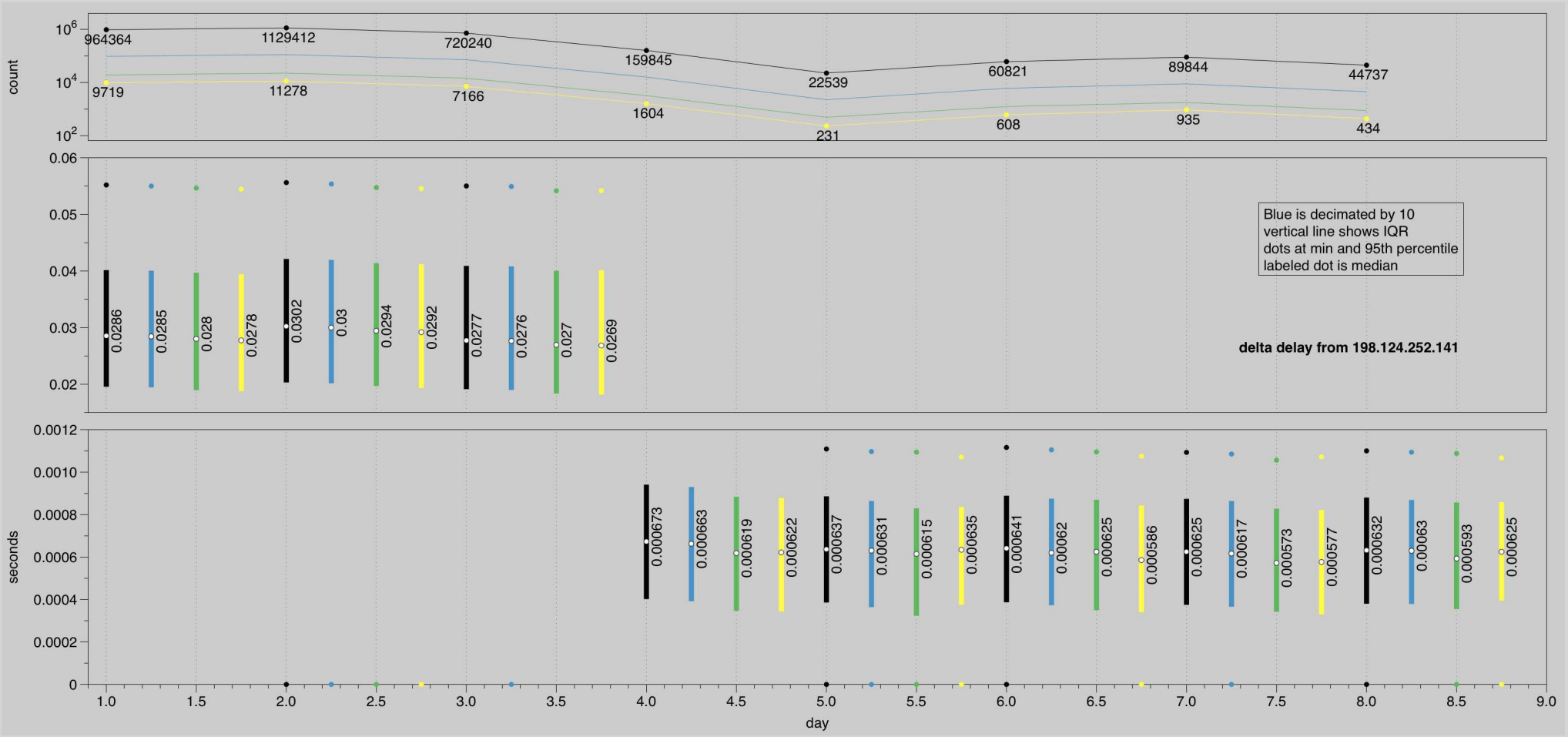
Applied to the inbound (to CP) delay variation of the long-lived file transfer with the bloat

Box plots are exact statistics. *tdigest* estimates are shown as yellow stripes and dots. Agreement with exact measures is excellent.



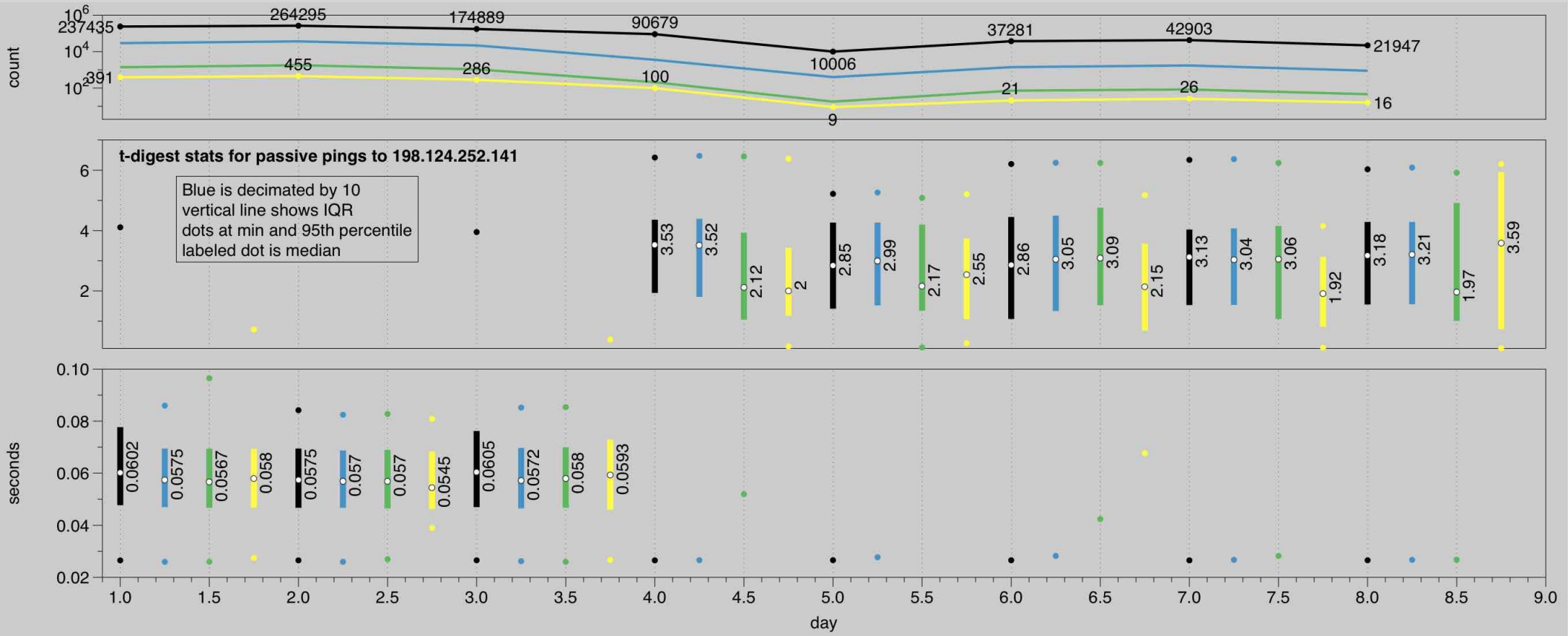
In the Bag of Tricks: Sampling

- Randomly sampled the packet trace at means of 1 in 10, 1 in 50, and 1 in 100. Black lines are “every packet” data
- Statistics remain fairly stable even with decimation by 100
- With **flow-based** sampling in our bag of tricks, even better results are likely

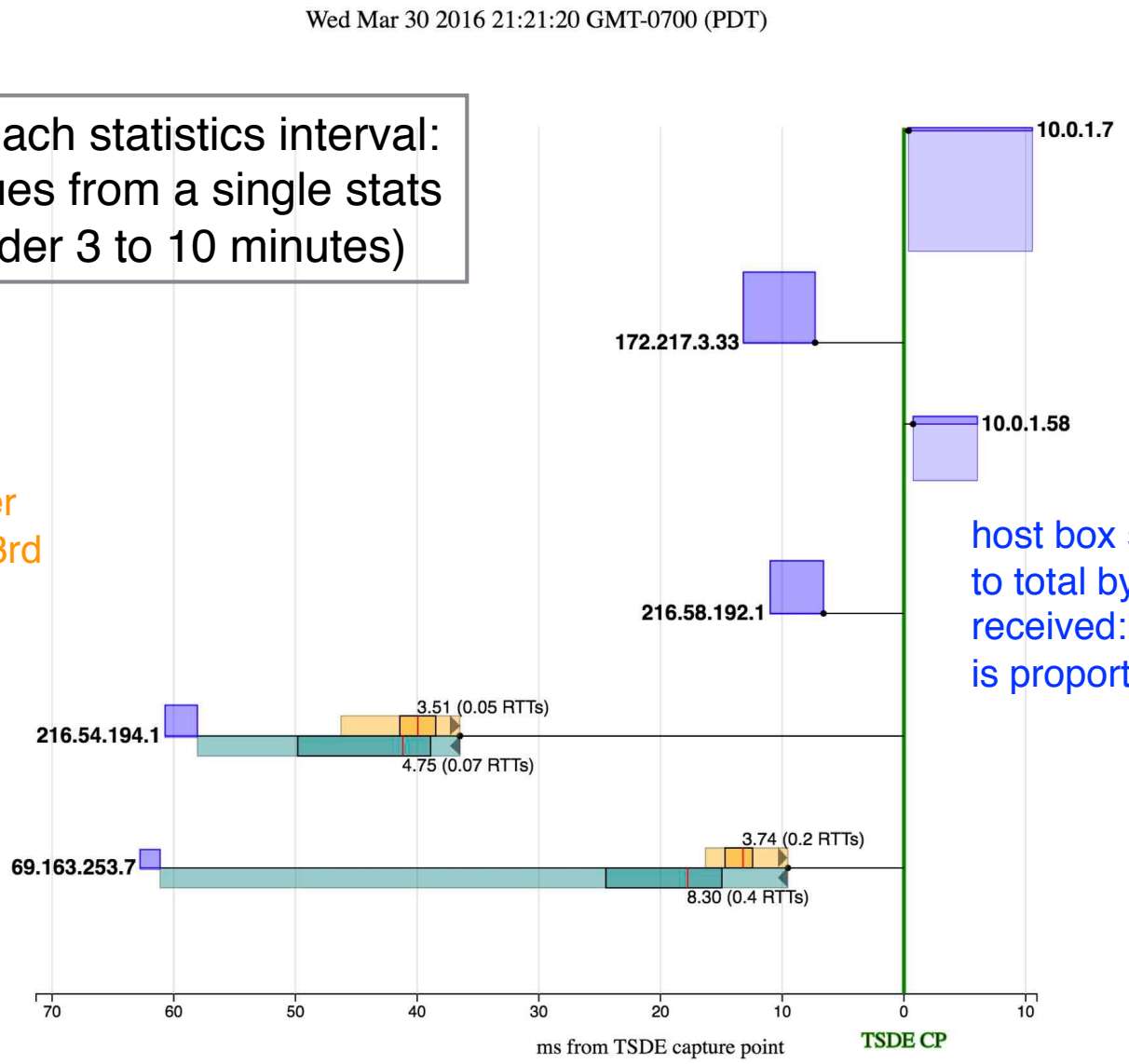


Paired RT Delay Metric Statistics

- Where there are very few samples to work with it's hard to get the underlying median from a ramping delay
- Non-bloated portion does better; also samples still sketch out the curve. Again, flow-based sampling in the bag of tricks would improve performance



Delay Topology Map Diagnostic uses t-digest Quantiles



Changes each statistics interval: shows values from a single stats interval (order 3 to 10 minutes)

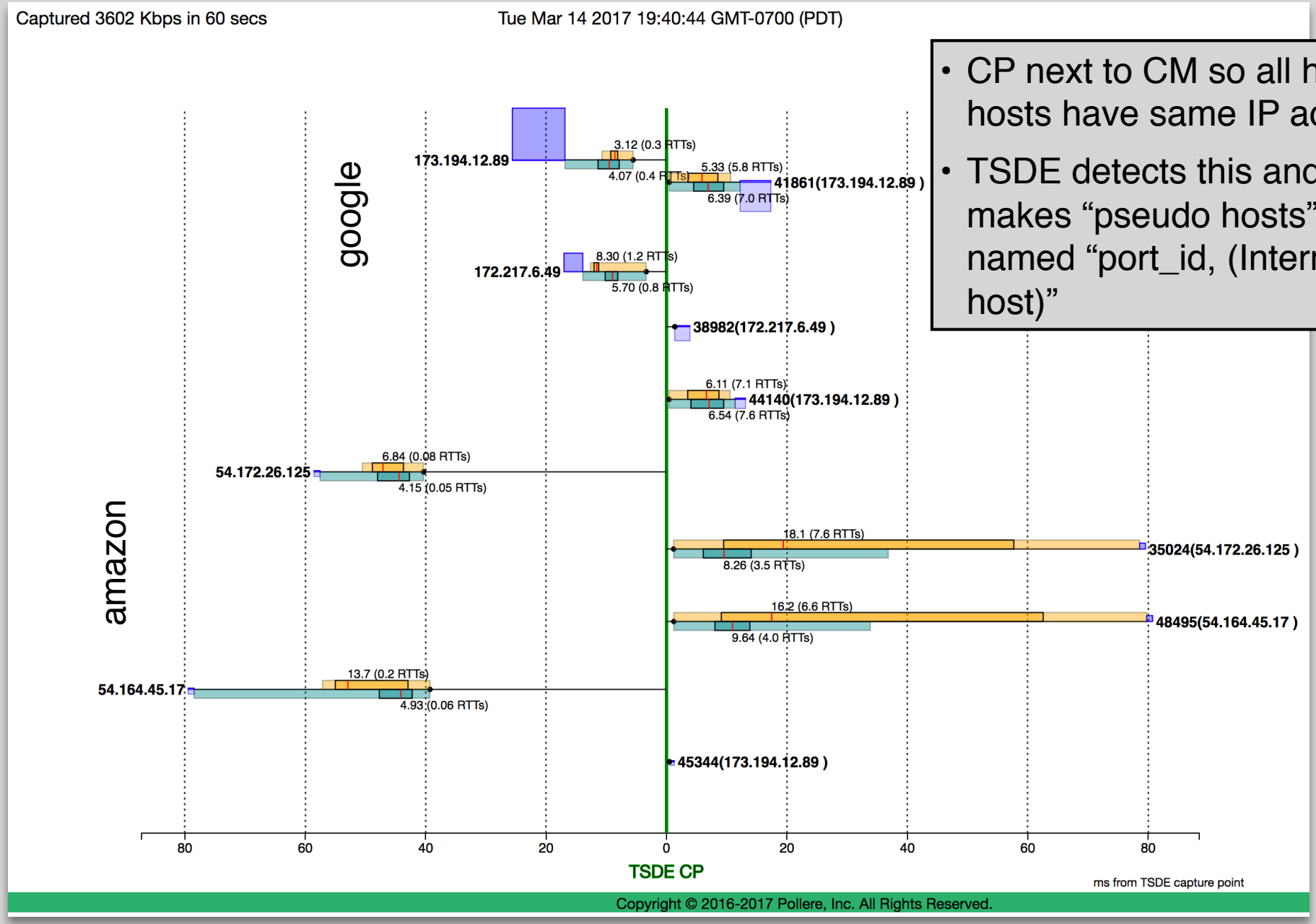
inbound dv: inner box for 1st and 3rd quartiles

median delay variation line

outbound dv: inner box for 1st and 3rd quartiles

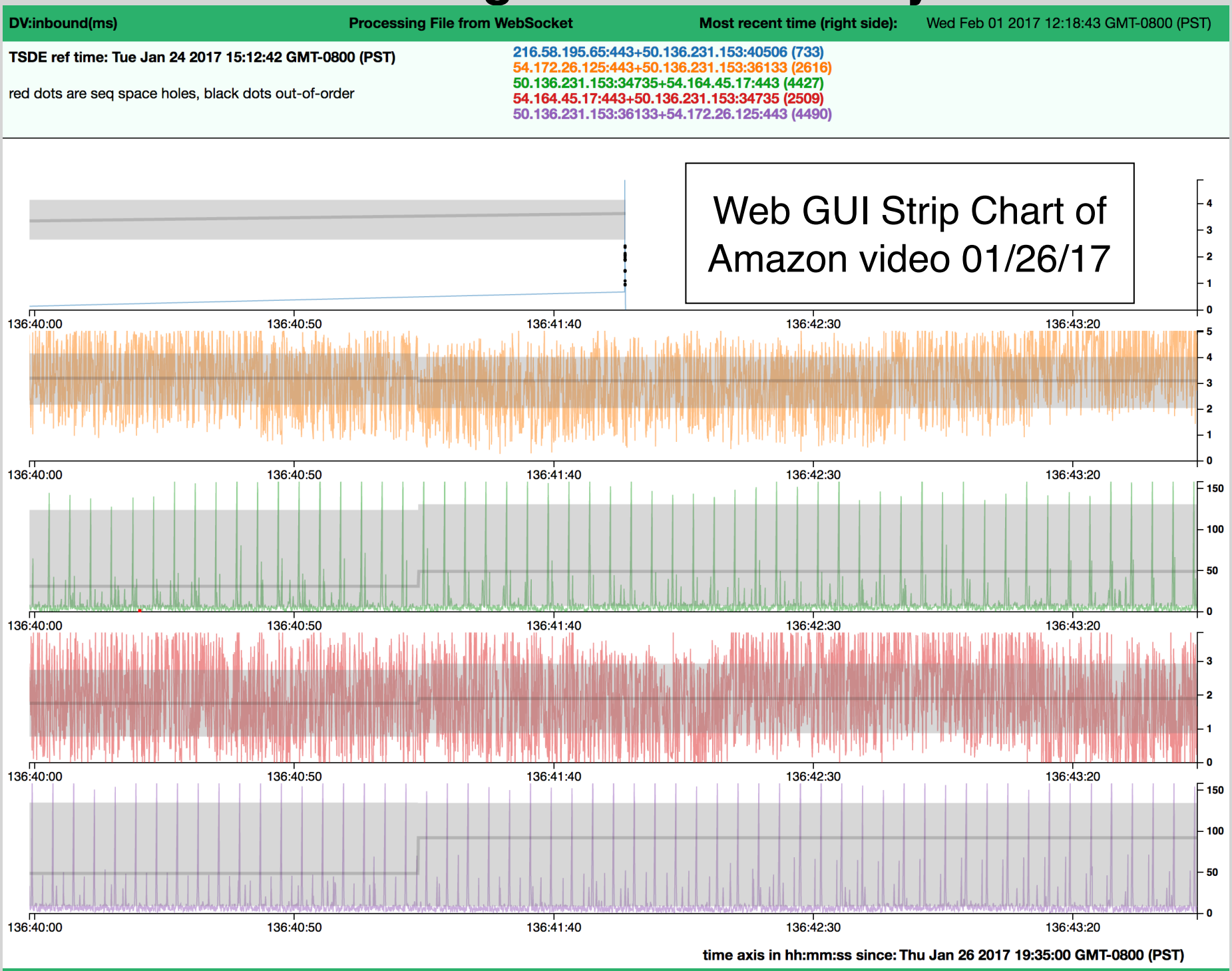
host box size is relative to total bytes sent and received: dark shading is proportion *sent*

Screen Shot (High Definition YouTube)

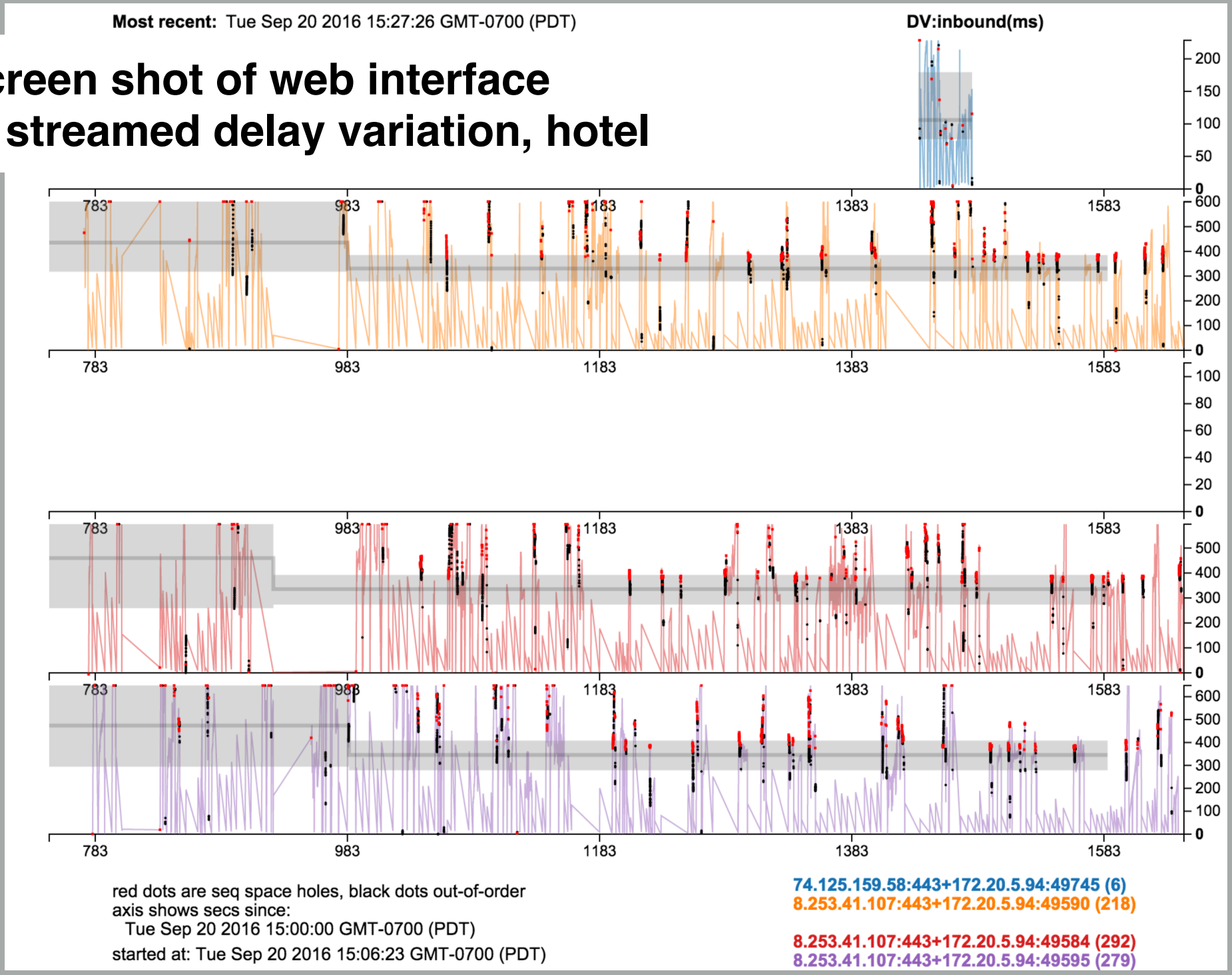


- CP next to CM so all home hosts have same IP address
- TSDE detects this and makes “pseudo hosts” named “port_id, (Internet host)”

Visual Diagnostics on the Fly



Screen shot of web interface of streamed delay variation, hotel



Summary: Entering the Frontier

- Diagnostic use of header captures requires:
  - understanding the underlying protocols
  - useful transport protocol headers, useful TSvals
  - an analysis bag of tricks to help hear everything the packets have to say
- Extracting data from headers and displaying it on-the-fly is harder than post-processing but exposes problems as they're happening.
- Mining all the header information allows a single capture point to find different kinds of problems but coherent display of the result is challenging.

Some Final Points

- Probing a crappy (e.g. third world) link seems like a bad idea. Use the data itself to diagnose performance issues
- Packet headers provide rich information (payload encryption doesn't matter) that active probes can't get
- Data can be mined by inexpensive devices deployed *anywhere*

The availability of this rich information is in danger from application layer protocols that encrypt transport headers