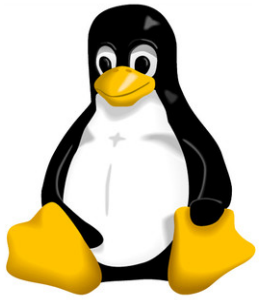


Recent advancements in Linux TCP congestion control

Yuchung Cheng

It’s the latency, stupid!

Many tune-ups to reduce RTTs for small transfers

- Fast Open [draft-tcpm-fastopen]
- Initial congestion window of 10 [6928]
- Faster recovery
  - Initial RTO of 1 sec [6298]
  - Early retransmit [5827]
  - Proportional rate reduction [6937]
  - Tail loss probe [draft-dukkipati-loss-probe]

Transitioning to Laminar

A new framework that separates cleanly various subsystems in TCP

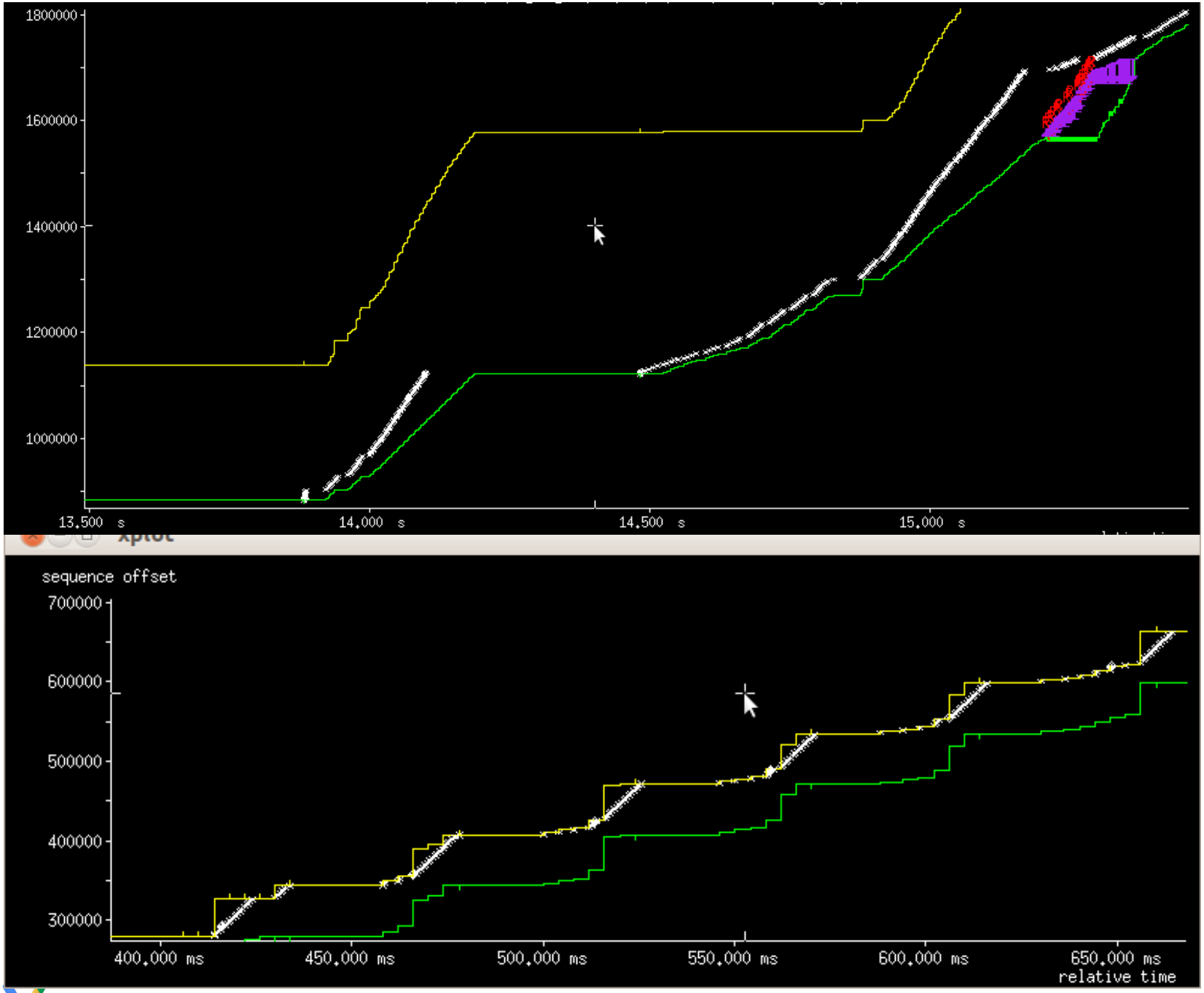
- draft-mathis-tcpm-laminar
- Congestion control: how much is delivered and how much to send per RTT
- Transmission scheduling: when to send within an RTT
- Loss recovery: what to send

New transmission scheduling: fq/pacing

Goal is to reduce TCP bursts

1. CC (cwnd) controls amount to send per round trip
2. Transmission is ACK clocked if data is available
3. Otherwise pace large burst at cwnd / RTT
   a. Send microburst as a TSO frame into the NIC
4. Use fair queuing among flows to improve mixing and fairness
5. Try it out on 3.11-rc6! `$tc qdisc add dev eth0 root fq`

More info at http://lwn.net/Articles/564978/


1800000
1600000
1400000
1200000
1000000
13.500 s
14.000 s
14.500 s
15.000 s
sequence offset
700000
600000
500000
400000
300000
400.000 ms
450.000 ms
500.000 ms
550.000 ms
600.000 ms
650.000 ms
relative time

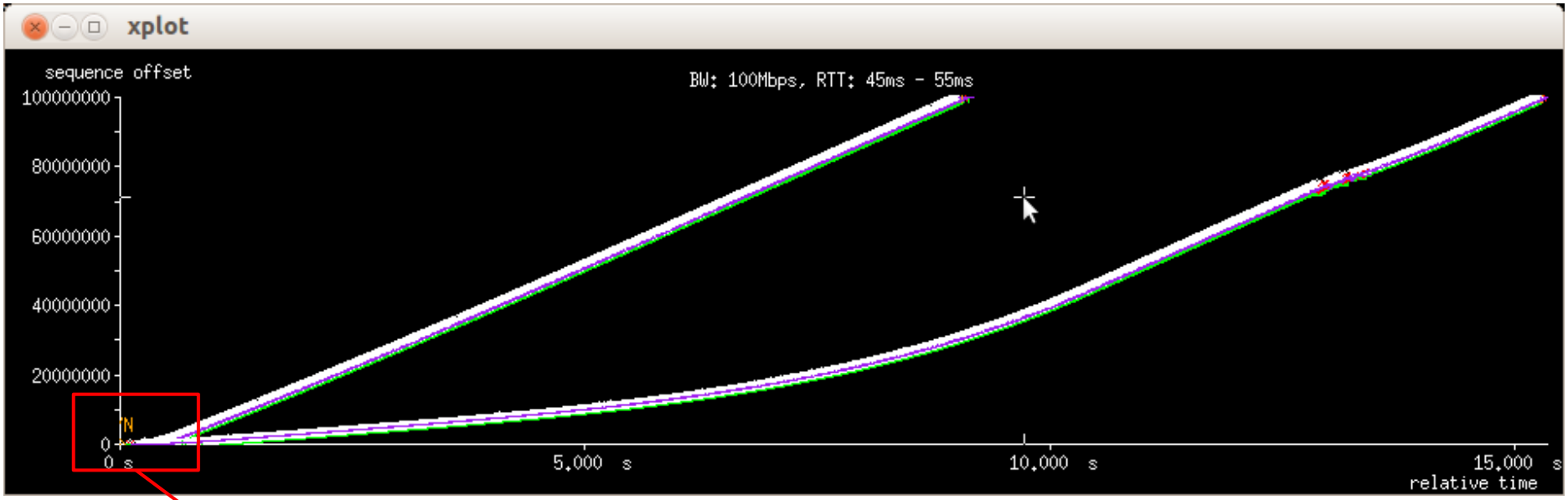
Reordering resilience

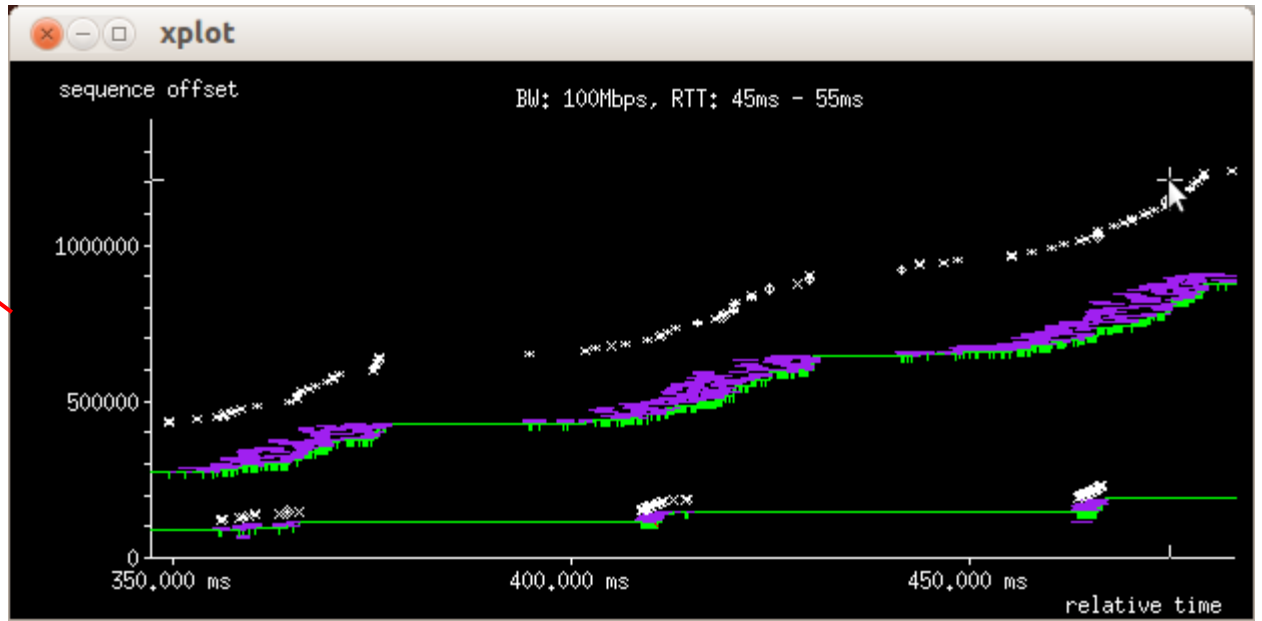
Goal: utilize network at 100% under frequent packet reorderings

1. Use packets newly acked or sacked as an input to CC
   a. Instead of per ACK
   b. Account for stretch ACKs, ACK losses, and SACKs
   c. CC is agnostic to reordering

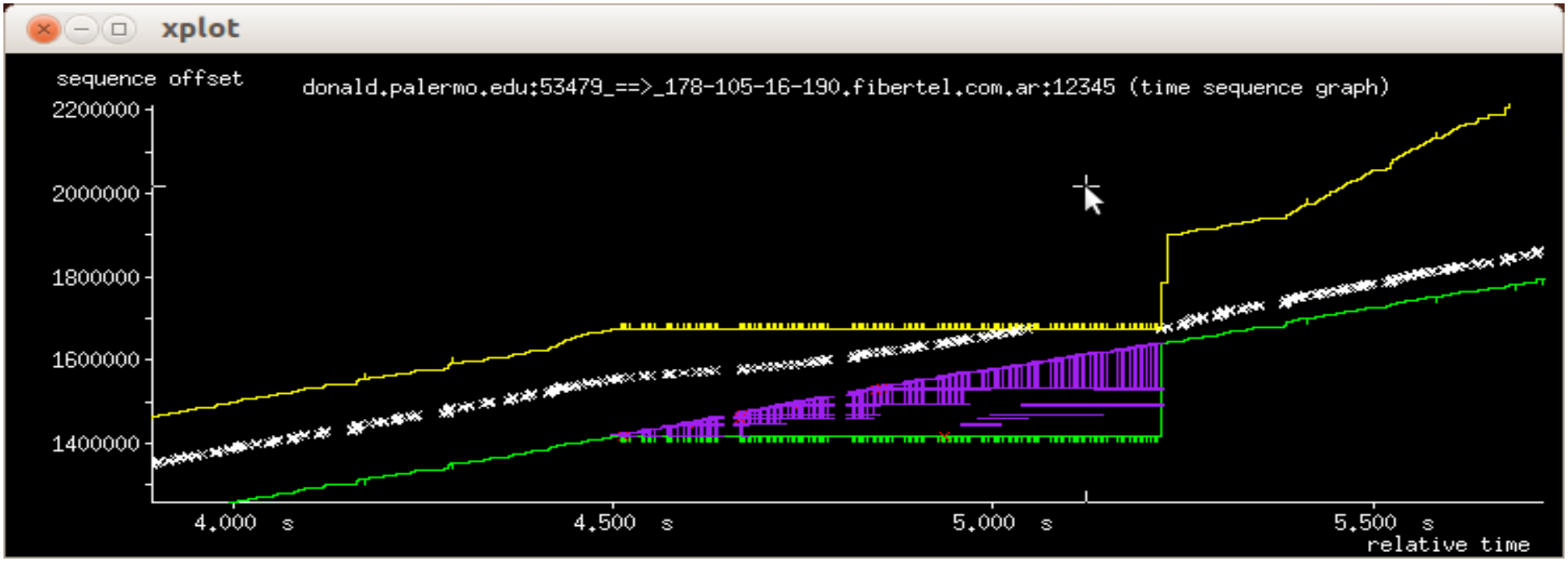
2. Measure RTT from newly SACKed data
   a. Improve RTT estimation under bufferbloat
   b. More useful than RTT from TCP Timestamps

Reordering resilience (cont'd)





Receiver buffer auto-tuning enhancement



Receive window is stopping limited-transmit (new data) in fast recovery!

- Accounts for out-of-order packet
- Announce rwin of 2x - 4x bytes expected to receive in next RTT
- Initial receive window is 20 pkts

New socket option TCP_NOTSENT_LOWAT

Some protocols want to delay writing to the socket as late as possible

- E.g., prioritization in SPDY
- Bytes written to (kernel) TCP is committed
- Keep just enough buffering in TCP to keep utilization high

TCP_NOTSENT_LOWAT

- Specifies the amount of “unsent” data in the send buffer
- POLLOUT is raised when the threshold is met

Other notable changes

1. TSO deferral fix that reduces burst size by 4 - 10x
2. Fixed a hystart bug that disables itself every other 25 days
3. TCP small queue: no more than two TSO frame per flow in qdisc
   https://lwn.net/Articles/506237/
4. Defense of Blind In-Window Attacks [5961]
5. Removed unnecessary features
   a. cwnd moderation [5681*]
   b. Limited slow start [3742]
   c. Cookie transaction [6013]
   d. Appropriate byte counting [3465]
6. F-RTO [5682] rewrite from scratch

Calling for attentions

1. Hystart is critical to reduce SS overshoot but has received little attention
2. Better congestion control to defend bufferbloat and/or high loss network
   a. Use delay and/or ECN, in addition to loss
   b. Reacting to BW changes within an RTT (e.g., mobile)
3. Dynamic prioritization of flows from the same receiver application
4. Unordered receive socket API to reduce HoL on multiplexed TCP (minion)
5. Simpler and effective loss recovery
   a. dupthresh is a flawed metric under high reordering
   b. Use the notion of time instead of packet count?