User-defined TCP Options Support in Linux

Viet-Hoang Tran, Olivier Bonaventure
(INL, UCLouvain)

Supporting new TCP options

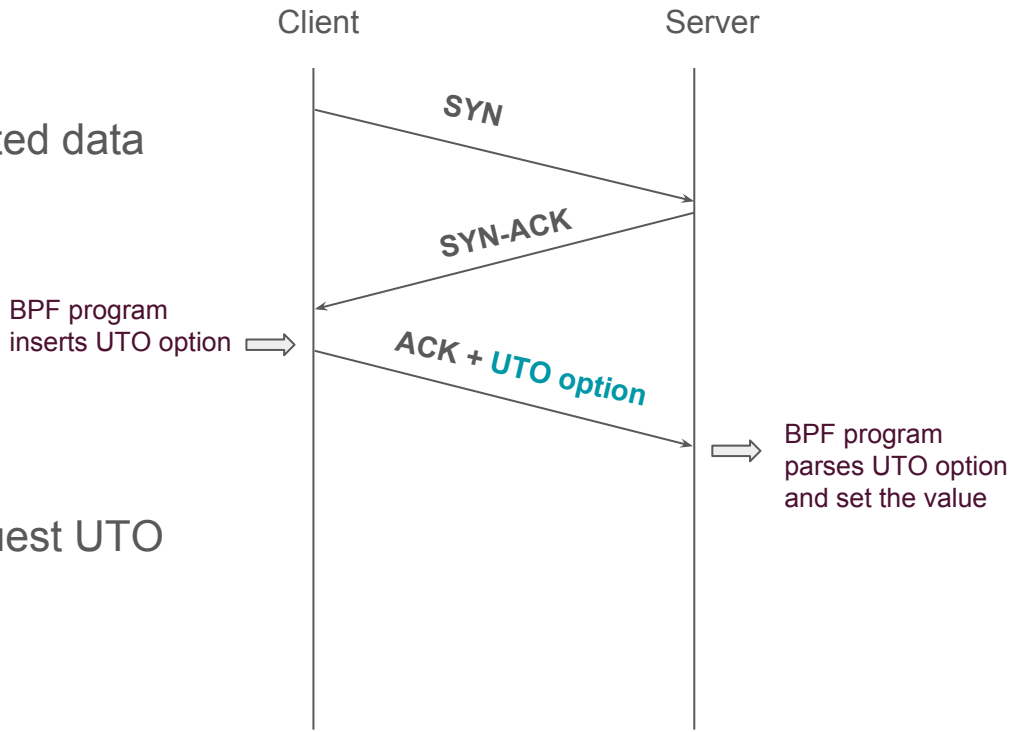
- The standard way to extend TCP
- Implementation: requires kernel changes
- Middlebox interference?
  RFC 6994: “Shared Use of Experimental TCP Options”
- Based on TCP-BPF by Lawrence Brakmo:
  - Hooks at different phases of a TCP connection
    or when connection state changes
  - Read & write to many fields of tcp_sock

User Timeout Option

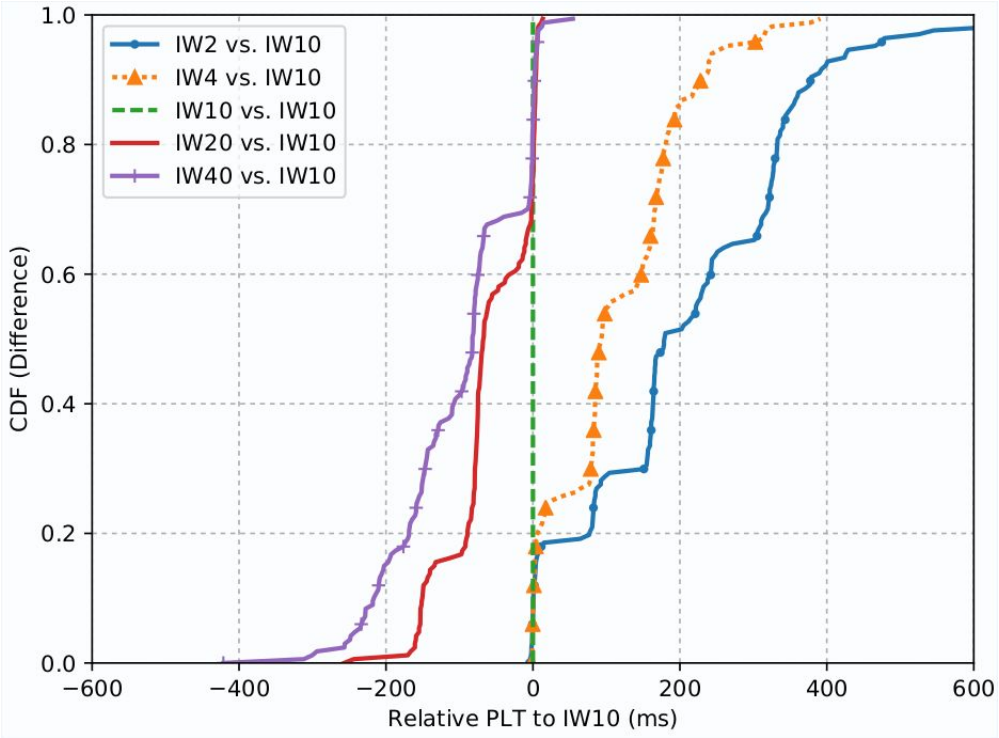
TCP User Timeout (UTO):

max time waiting for the ACK of transmitted data
before aborting the connection

RFC 5482: TCP option to announce/request UTO
Not yet implemented in Linux



Initial CWND option



When the receivers know more about the network bottleneck.

e.g. clients know about WiFi/Cellular capacity

Delayed ACK Option

- Motivation: Too many ACKs or too few ACKs is not good.
  - ACK processing overhead, wireless channel contention,...
  - or large bursts, too high minRTO, …

  → The need to request the remote's ACK delay strategy

- Google proposed *Low Latency Option* at IETF 97-99:

  Absolute Delack timeout + TS resolution

- Instead, we use a purely delayed-ACK option, which carries two values:

  Segs count: Number of received segs before sending an ACK

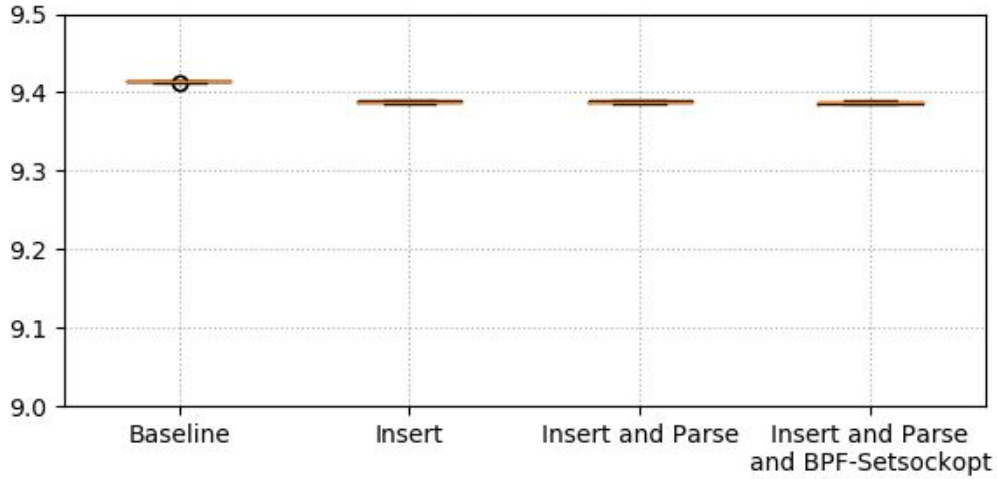
  Delack timeout: relatively as a fraction of RTT

  Value: `log2(RTT/Timeout)`

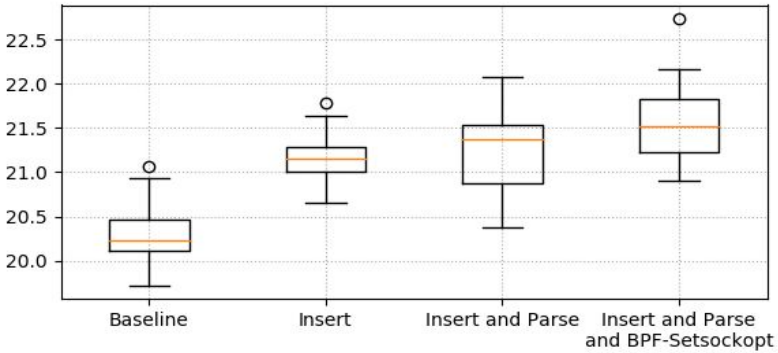
  based on sRTT or minRTT?

Overhead

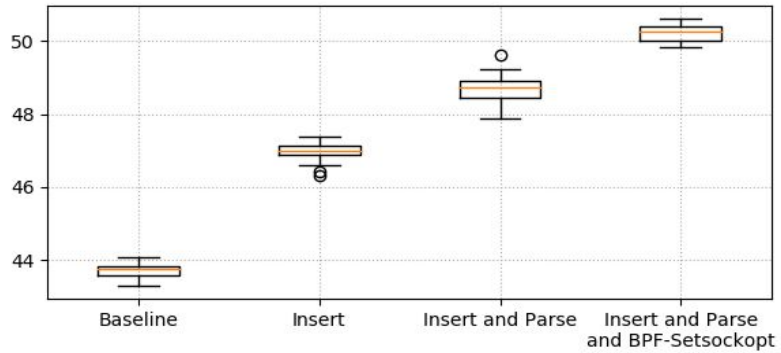
- iperf3 transfer over 10 Gbps link
- trigger on every packet



Average Throughput (Gbps)



Sender's CPU usage (%)



Receiver's CPU usage (%)

Code Status

	Kernel changes	BPF program
TCP Option framework	75	-
Use case: TCP User Timeout	16	76
Use case: Congestion Control	0	92
Use case: Initial Window	0	76
Use case: Delayed ACK	94	77

Caveats

- Option size <= 4 Bytes, extensible to 16 Bytes
- Decouple from cgroup-v2?

Conclusion

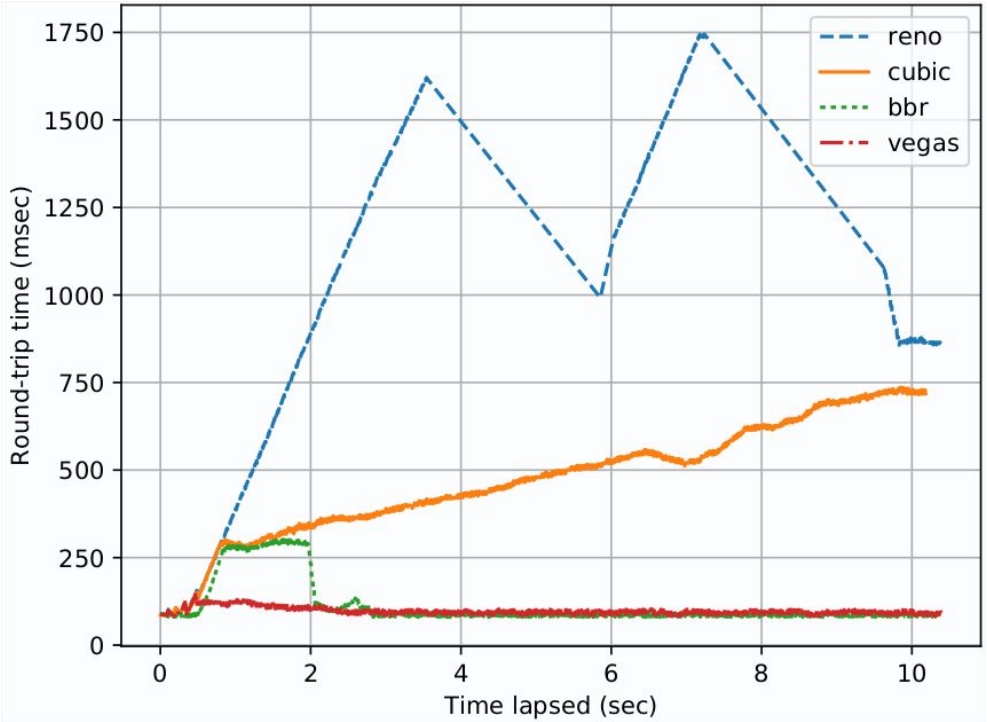
- eBPF makes it easier to extend TCP
  or adapt TCP to various environments.

- More in paper: https://arxiv.org/abs/1901.01863
- Git repository: https://github.com/hoang-tranviet/tcp-options-bpf

  Feel free to try and extend it!

Backup slides

Congestion Control Request Option

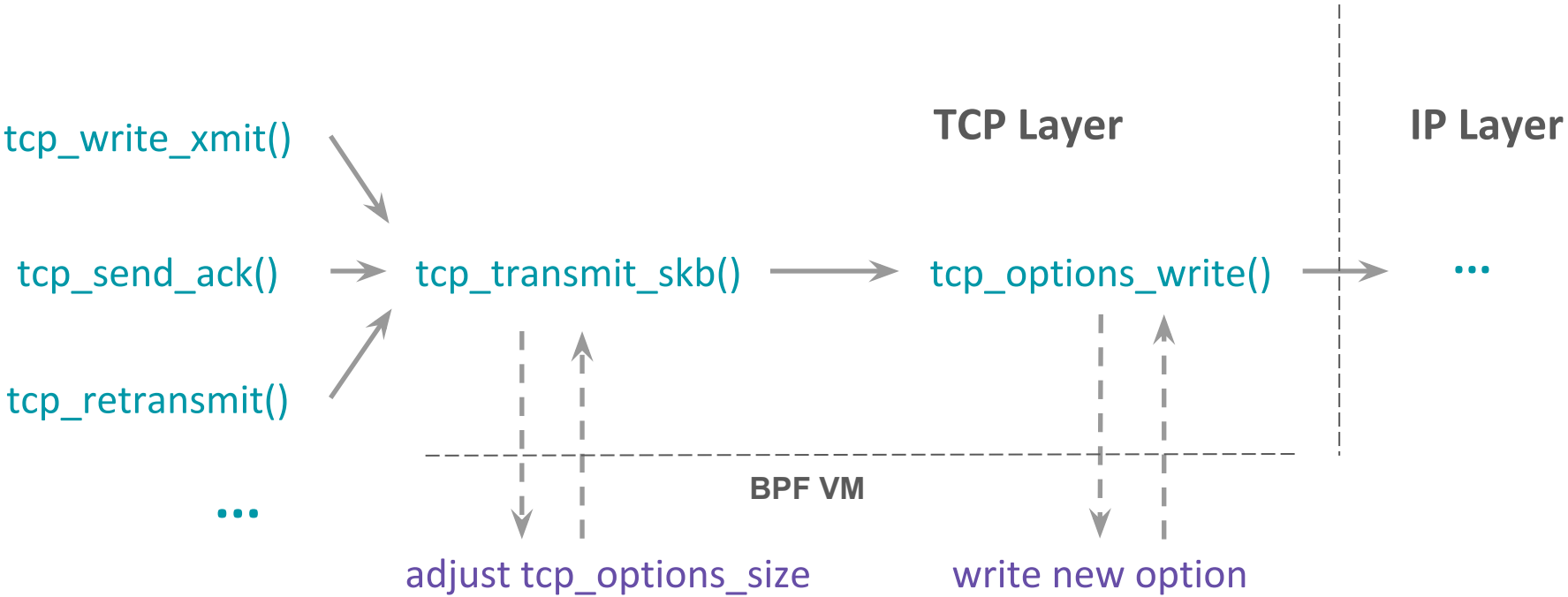


Receiver requests the sender to use a desired CC algorithm for the connection

E.g. Clients prefer low latency over throughput

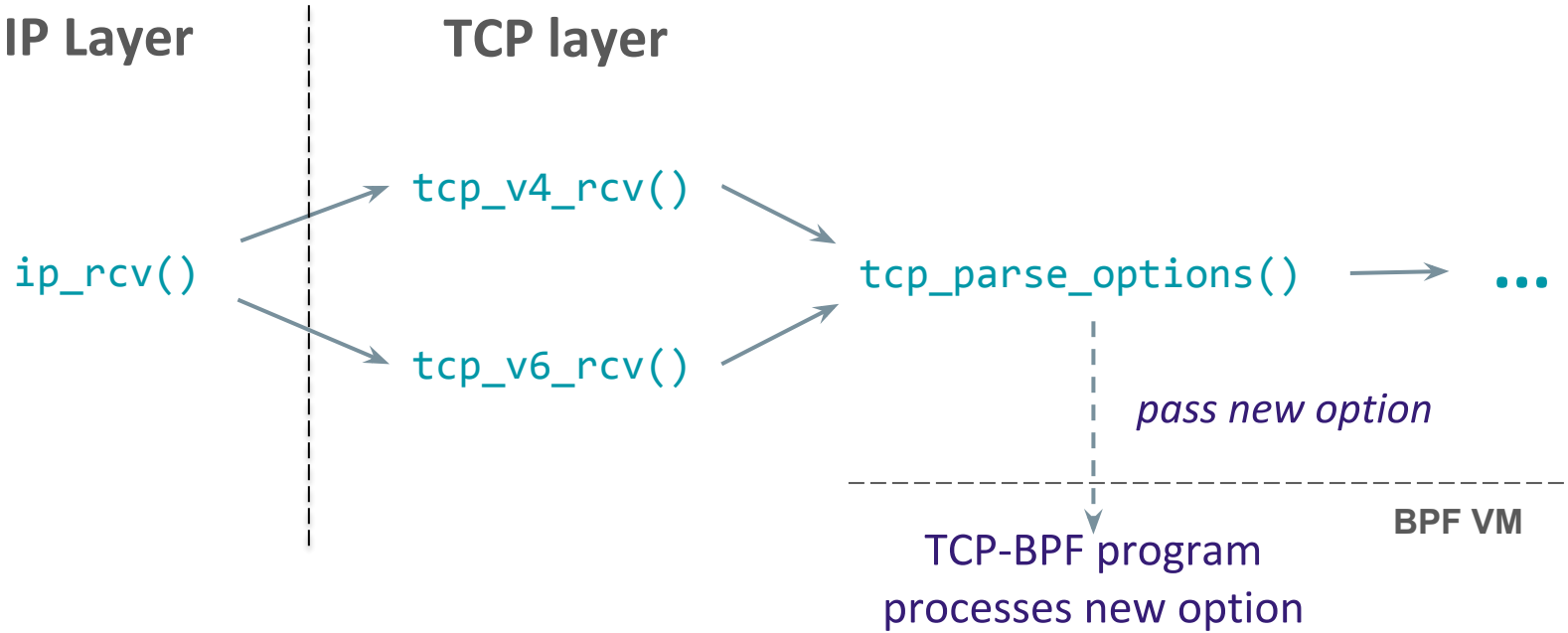
Two sides shared the list of CC beforehand

Add new option: 2 steps



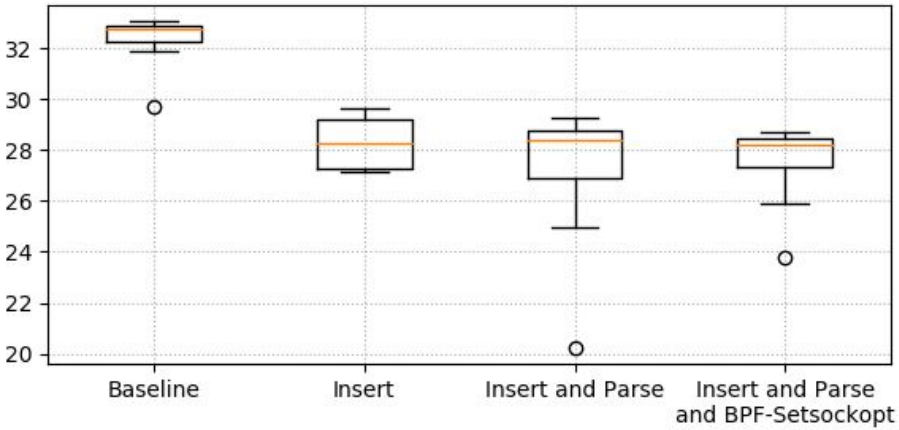
One more thing: update current MSS

Parse new option

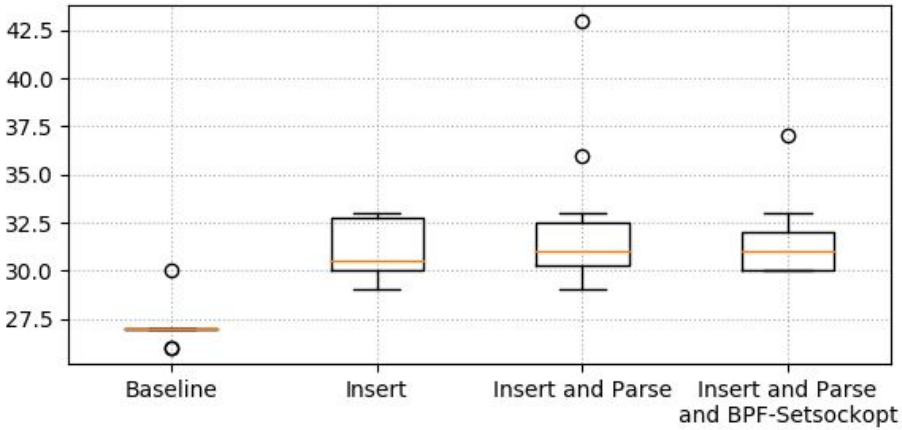


Extreme (and unrealistic) benchmark

over loopback interface
trigger on every packet



Average Throughput (Gbps)



RTT (usecs)