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: \( \log_2(\text{RTT}/\text{Timeout}) \)
    - based on sRTT or minRTT?
Overhead

- iperf3 transfer over 10 Gbps link
- trigger on every packet
## Code Status

<table>
<thead>
<tr>
<th>Use case</th>
<th>Kernel changes</th>
<th>BPF program</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Option framework</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Use case: TCP User Timeout</td>
<td>16</td>
<td>76</td>
</tr>
<tr>
<td>Use case: Congestion Control</td>
<td>0</td>
<td>92</td>
</tr>
<tr>
<td>Use case: Initial Window</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>Use case: Delayed ACK</td>
<td>94</td>
<td>77</td>
</tr>
</tbody>
</table>

## 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.

- 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

tcp_write_xmit()

TCP Layer

tcp_send_ack() → tcp_transmit_skb() → tcp_options_write()

IP Layer

...tcp_retransmit()...

adjust tcp_options_size

write new option

One more thing: update current MSS
Parse new option

- IP Layer
  - `ip_rcv()`

- TCP Layer
  - `tcp_v4_rcv()`
  - `tcp_v6_rcv()`
  - `tcp_parse_options()`

TCP-BPF program processes new option

BPF VM

pass new option
Extreme (and unrealistic) benchmark

over loopback interface
trigger on every packet