Proposal: TCP Daytona

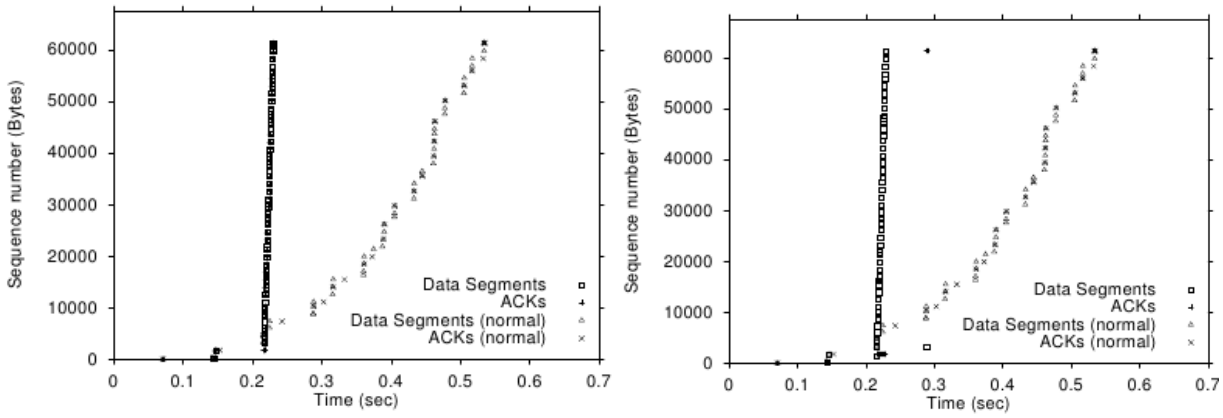
Max Meyers
Roan Kattouw

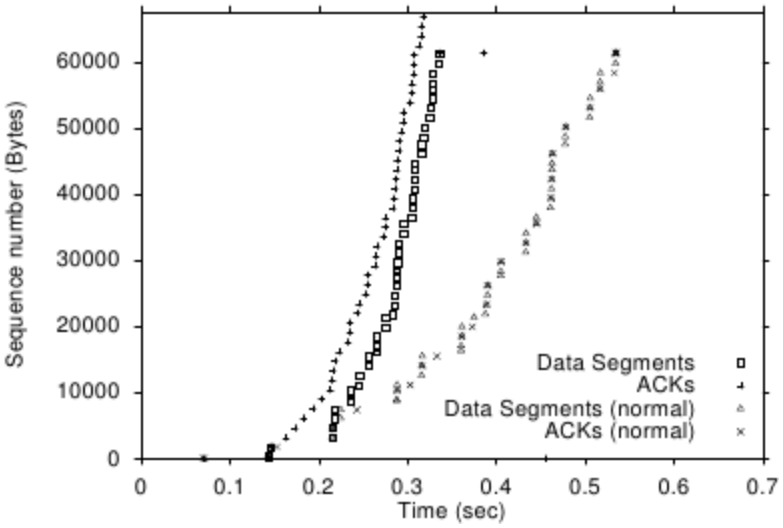
The paper “TCP Congestion Control with a Misbehaving Receiver” (Savage et al., 1999) describes three attacks on TCP congestion control that allow a non-conformant receiver to make the sender send data arbitrarily fast, taking more than its fair share of the available bandwidth and defeating the purpose of congestion control. The authors jokingly refer to the resulting TCP implementation as “TCP Daytona”.

The paper describes three attacks:

1. When receiving a data segment, send not one but multiple ACKs, each for disjoint parts of the received segment. This causes the sender to increase its cwnd multiple times.
2. When receiving a data segment, send lots of ACKs for the previous segment. The enter fast retransmit, first retransmitting the original data, then inflating the cwnd and sending many more data segments.
3. When receiving a data segment, ACK it, but also send a number of ACKs for sequence numbers in the future. This tricks the sender into thinking the RTT is lower, causing it to send more data faster.

We propose to reproduce the results in figures 4, 5 and 6 in the paper, showing the speedup achieved by each of these attacks:





To reproduce these, we would have to modify the TCP congestion control algorithm, either in the kernel or, if supported, in Mininet (the documentation doesn't seem to have any information about this; maybe the TAs know more?). Linux has multiple congestion control algorithms and offers programs a choice of which one to use (iperf's -Z flag exposes this), so we could clone an existing congestion control algorithm and modify it slightly to make it misbehave.

We would also have to create a topology in Mininet, run a web server and an iperf server, and hit them from a client running our modified TCP congestion (non-)control code. Running this on the internet targeting actual web sites would be very interesting, but is probably not a very good idea.