PRESENTATION 34

Department of Computer Science
University of Petrozavodsk

TCP Sliding Window Size Reconstruction Using IP and TCP Headers.

Vadim A. Ponomarev, Dr. Olga I. Bogoiavlenskaia
(University of Petrozavodsk, Russia).

Abstract

The goal of this project is to build a tool for TCP sliding window reconstruction using IP and TCP headers (actually they are saved in tcpdump file with their timestamps).

TCP sliding window is one of the major parameters used in TCP modeling and analysis, and such a tool could be very useful for researchers working in this area.

Our tool tries to reconstruct TCP sliding window using captured and saved data and implementing TCP Congestion Control algorithm described in RFC 2581.

It can work with any tcpdump capture file without modifications of sender or receiver or interceptor OS, applications or TCP stack implementation. Seawind dumps were used for testing purposes.

The project is under active development, and this presentation shows the current state of the project.

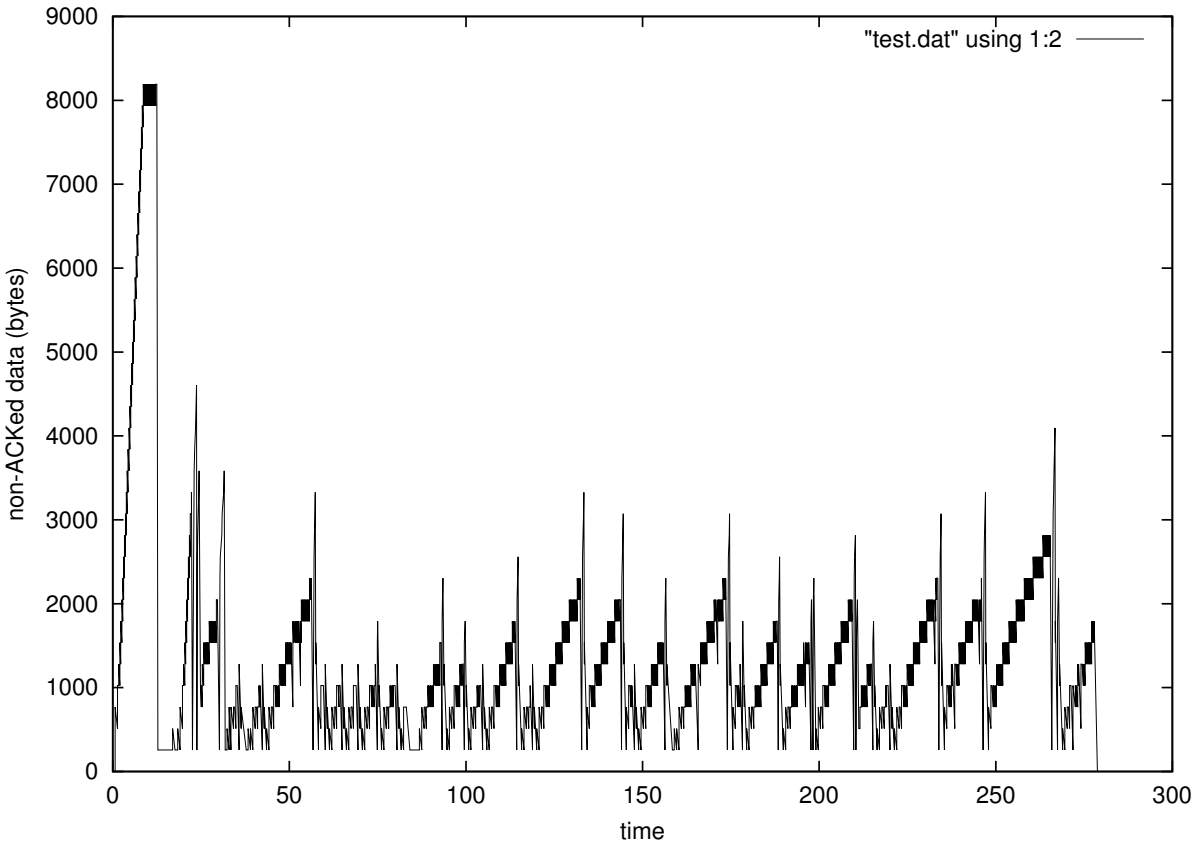

Figure 1: amount of non-acked bytes on the network

There are already some software products, which tries to estimate congestion window size, for example tcptrace (http://irg.cs.ohiou.edu/software/tcptrace/tcptrace.html).

Tcptrace can plot the amount of non-acked data on the network (the difference between the highest byte sent and the highest byte ACKed). You can see plot of that difference on Figure 1 above.

It was the first and easiest part of our TCP sliding window reconstruction project. Each TCP packet carries "sequence number", and some TCP packets carries "acknowledgment number", and we can easily calculate the difference between last sent and last acknoledged octets for each packet.

Figure 1 was generated using a capture file from Seawind project.

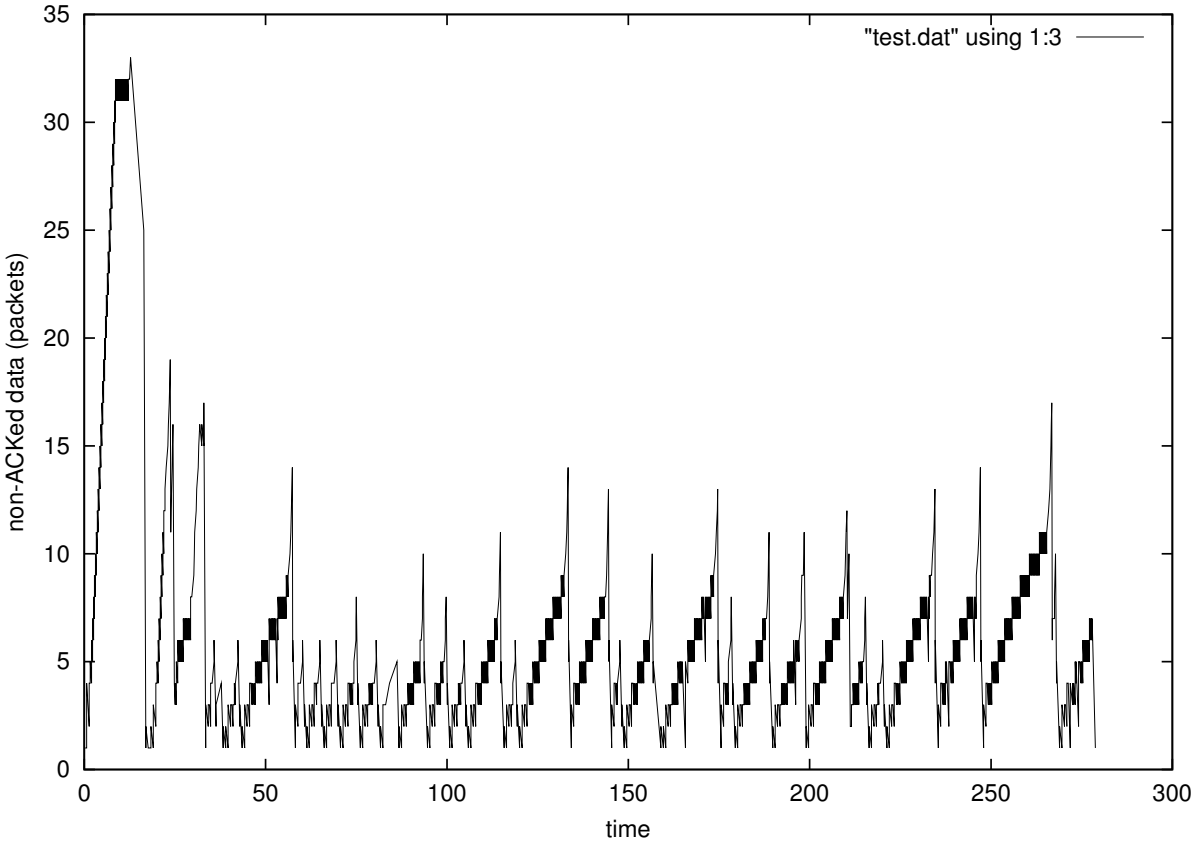

Figure 2: amount of non-acked packets on the network

Congestion window can be measured not only in bytes, but in packets too. Figure 2 shows the number of packets sent, but unacknoledged yet. There is an "unacknoledged packet counter" for each TCP connection (actually, for each direction of TCP connection), and each sent packet increases that counter, and each acknoledged packet decreases counter.

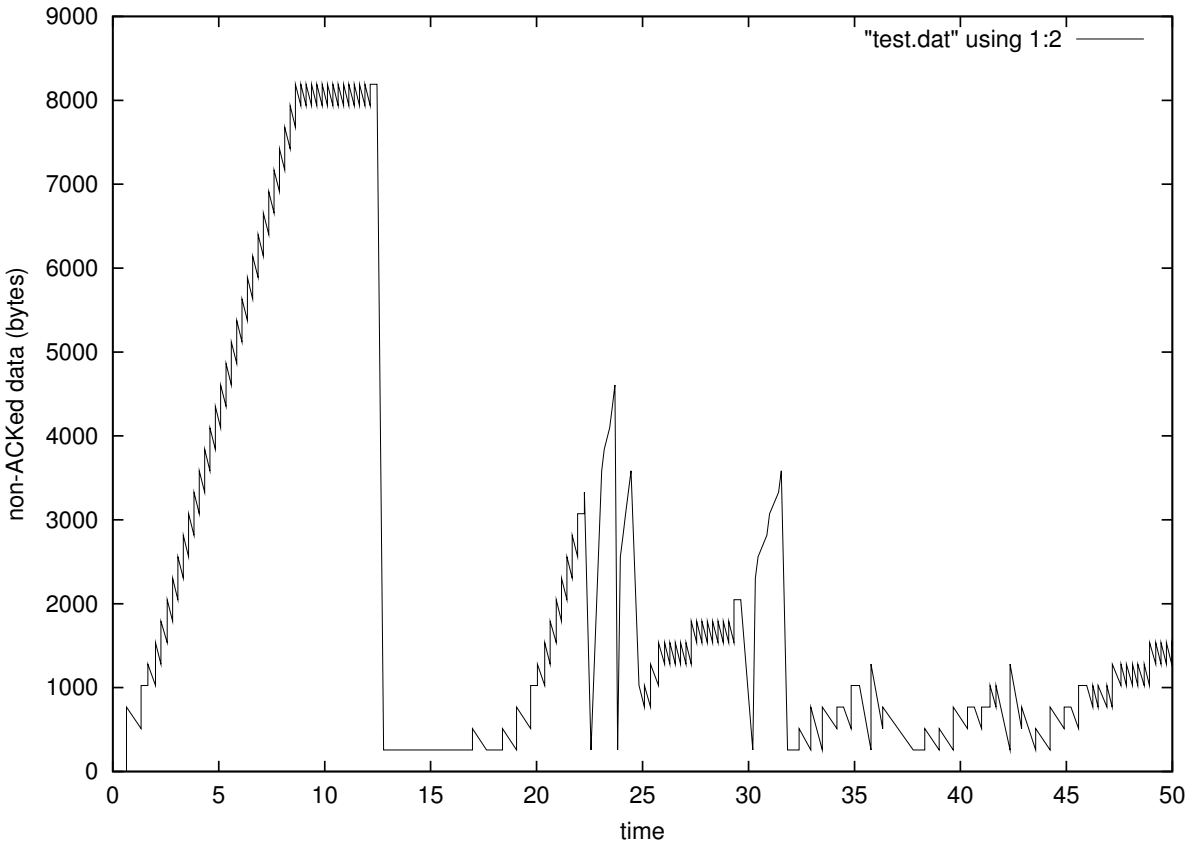

Figure 3: zoomed amount of non-acked bytes on the network

But there is a problem with shown "amount of unacknoledged data" plots. The amount of unacknoledged data is only some kind of heuristic, and the real congestion window is not necessary equals amount of unacknoledged data.

Figure 3 shows plot 1 zoomed at [0-50]. You can see some kind of "saw", and general instability of calculated amount of unacknoledged data when packet losses or timeouts occurs.

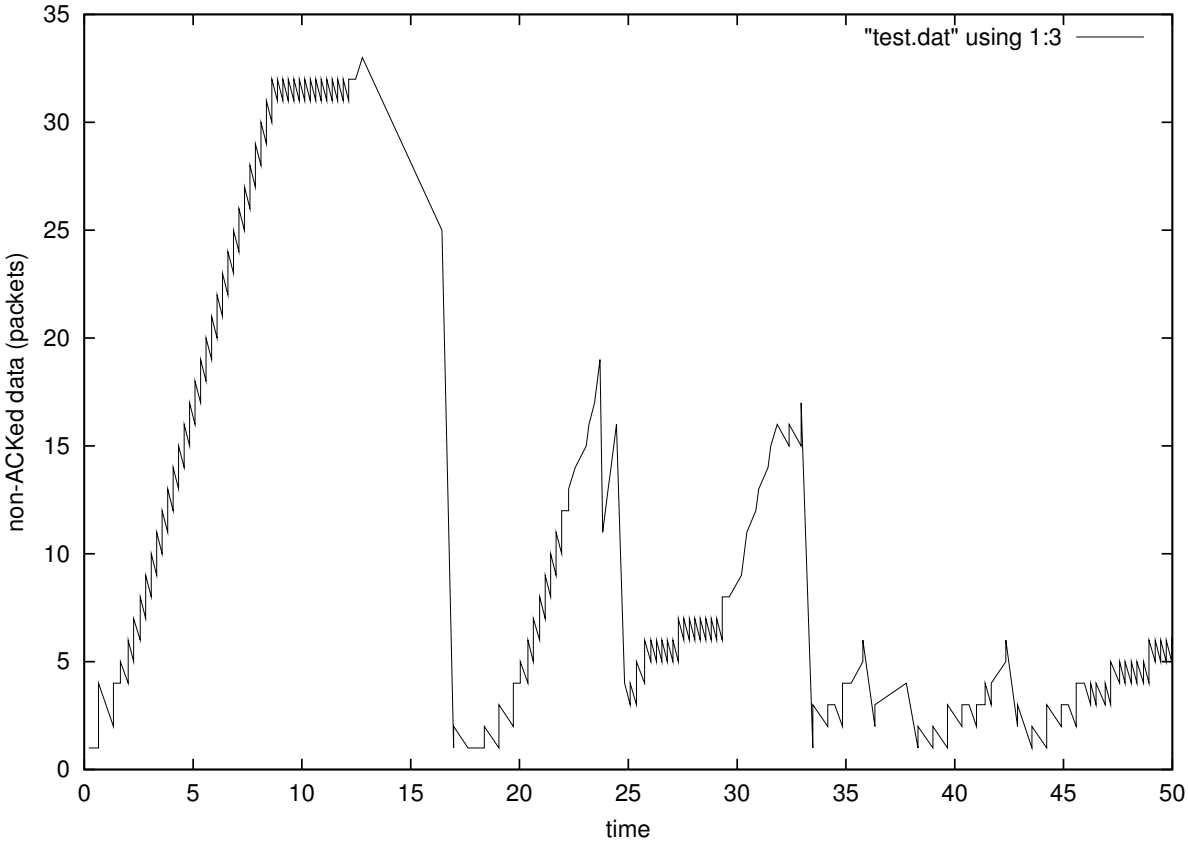

Figure 4: zoomed amount of non-acked packets on the network

Just for completeness - Figure 4 shows plot 2 zoomed at [0-50]. We can see "saw" and instability again.

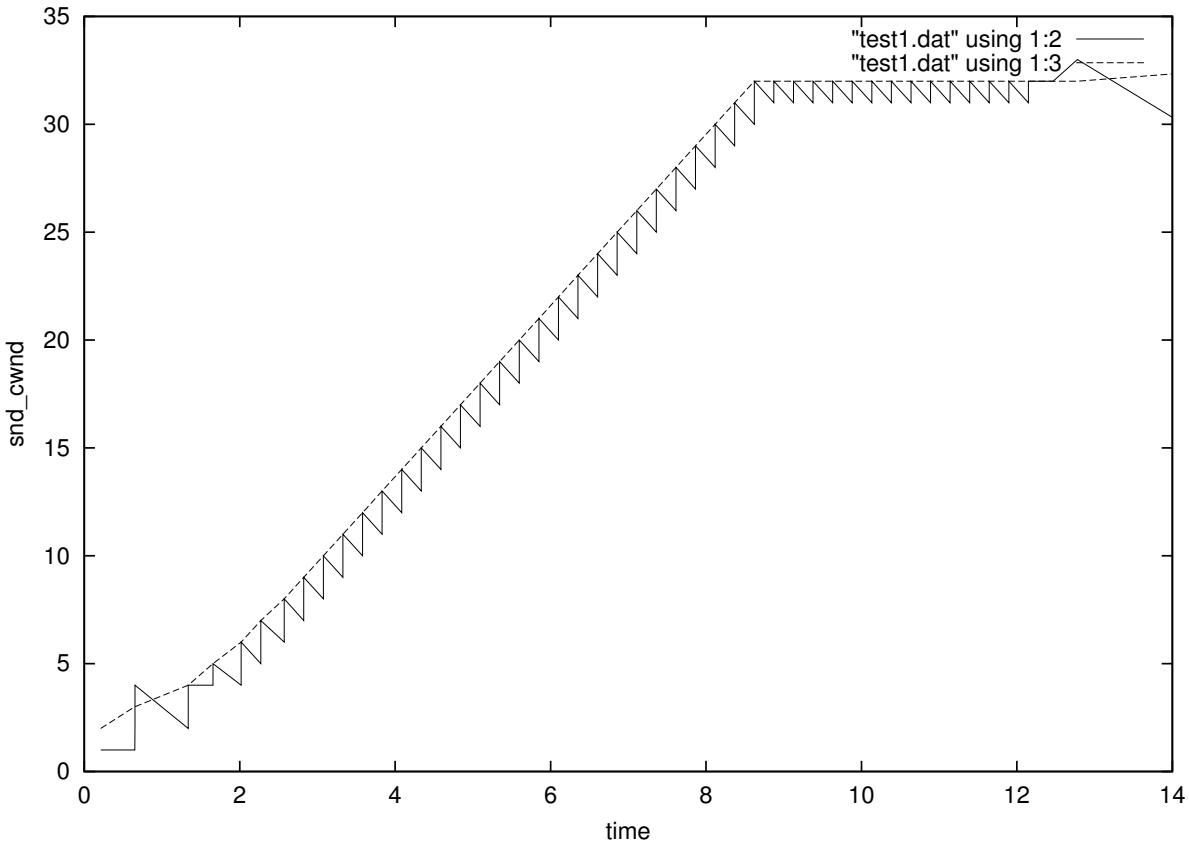

Figure 5: slow start and congestion avoidance

Our projest tries to implement algorithm described in RFC2581. Each packet has source socket and destination socket. While processing packet, we must process it at the sending side and the receiving side. At the sending side, we can only put packet in the queue of unacknoledged packets and modify snd_nxt. At the receiving side, algorithm described in RFC2581 must be implemented (slow start, congestion avoidance, fast retransmit and fast recovery).

Note that each side can be sending in one moment or time, and receiving in another moment of time.

Figure 5 shows Figure 2 zoomed at [0-12], where solid line is heuristic described above (number of unacknoledged data) and dotted line is our reconstructed congestion window (snd_cwnd). This part of plot shows slow start, a bit of congestion avoidance and packet loss.

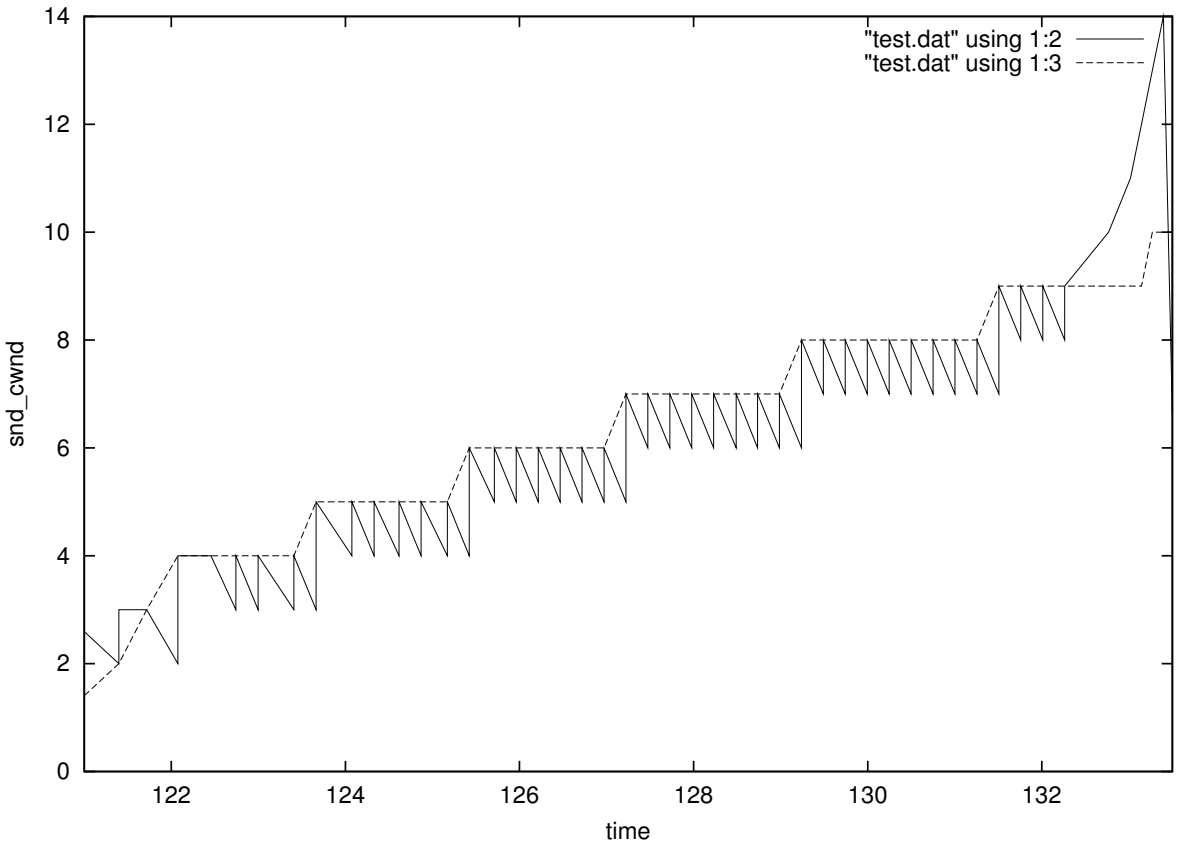

Figure 6: congestion avoidance

Figure 6 shows Figure 2 zoomed at [121-133] with solid line is heuristic described above (number of unacknoledged data) and dotted line is our reconstructed congestion window (snd_cwnd). This part of plot shows several steps of congestion avoidance algorithm with following packet loss.

Now the conclusion. Seems that basic idea works ok, and our reconstructed congestion window (at least) does not contradict with real amount of unacknoledged data in a network. Slow start and congestion avoidance parts are implemented somehow. Slow start threshold and fast retransmit/recovery parts are under development now.