# All Quiet on the Western Front – News from TCP

*History, Status Quo and Perspective*

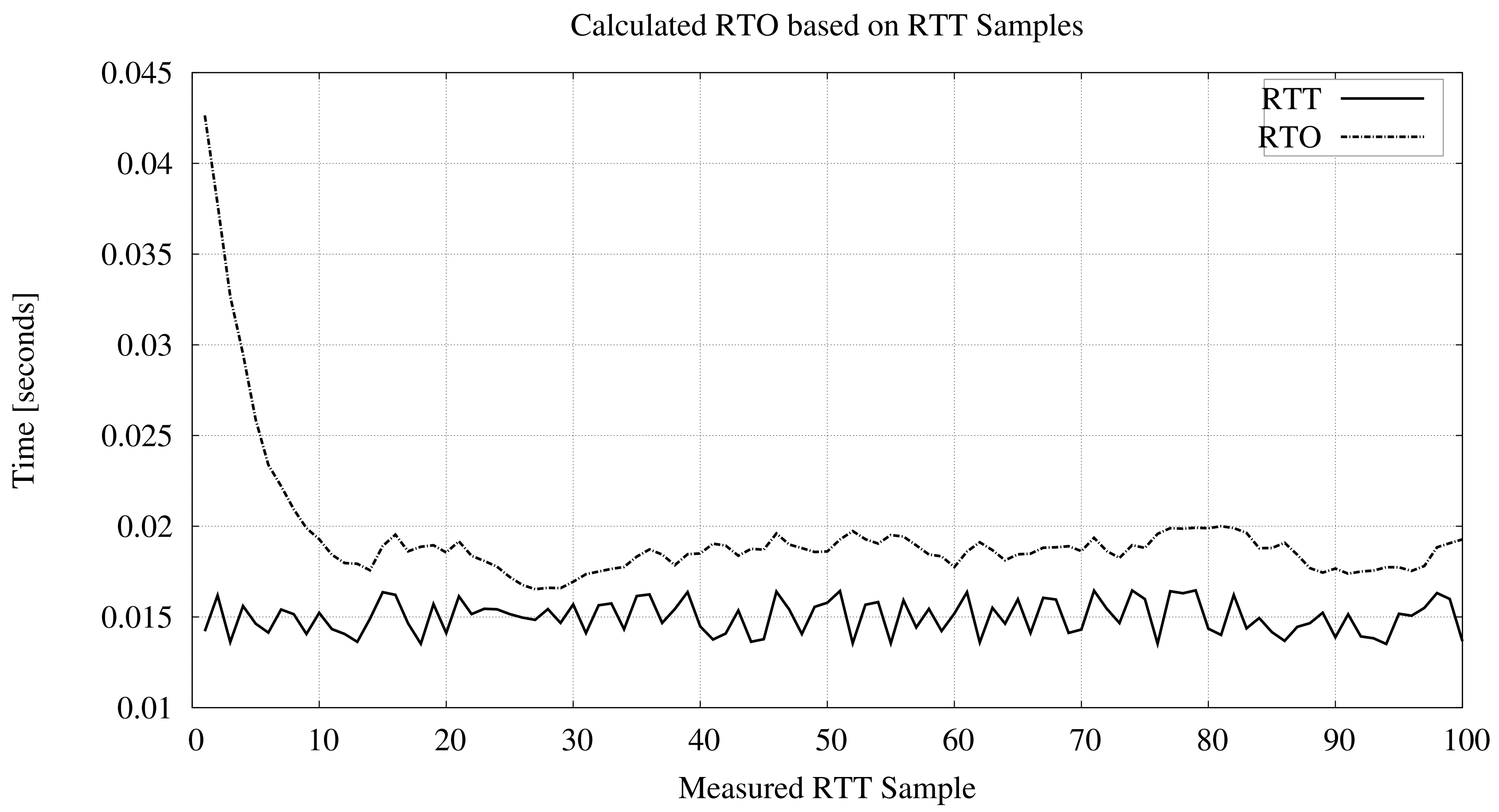
Hagen Paul Pfeifer
hagen.pfeifer@protocollabs.com

ProtocolLabs
http://www.protocollabs.com

## Retransmission Timer

- TCP uses a timer to ensure data delivery in the absence of any ACK packet – called *Retransmission Timer*
- Duration: Retransmission Timeout (RTO)
- RFC 1122 specifies that the timer should be calculated aligned with the famous Jacobson Paper (Jac88)
- Initial RTO (no measurements): $\geq 3$ seconds
- RFC 2988 specifies that the minimum TCP Retransmission Timeout (RTO) SHOULD be 1 second
  - Keep TCP conservative
  - Avoid spurious retransmissions
- But:
  - Analysis had demonstrated that a RTO of 1 second badly breaks throughput in environments faster than 33kB with minor packet loss rate (e.g. 1%)

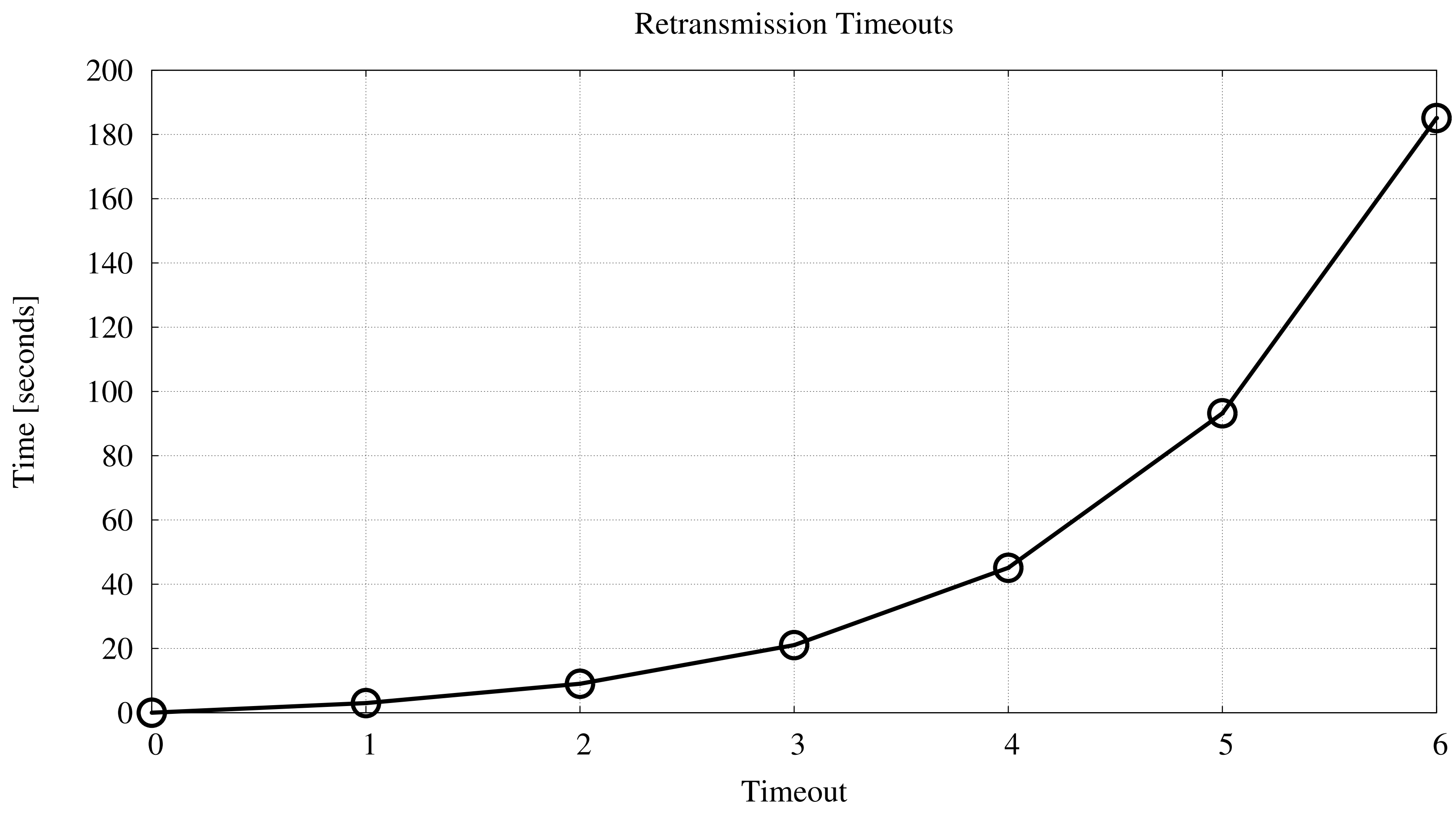
# Retransmission Timer



## Retransmission Timer

- RFC 6298:
  - Initial RTO to 1 second
- Minimum RTO in ESTABLISHED state is 200 ms for Linux (30 ms for FreeBSD)
- Why
  - Clock granularity much higher
- Links:
  - `http://utopia.duth.gr/~ipsaras/minrto-networking07-psaras.pdf`

## Retransmission Timeout



## TCP User Timeout Option

- Background: TCP Timeout (don't confuse with TCP's Retransmission Timer)
- Definition:
  - Controls how long transmitted data may remain unacknowledged before a <u>connection is closed</u>
- *RFC 793* default timeout of 5 Minutes
- *RFC 1122* – Host Requirement:
  - Two thresholds $R1$ and $R2$ (time units or count of retransmissions)
  - $R1$ SHOULD correspond to at least 3 retransmissions, at the current RTO
  - The value of $R2$ SHOULD correspond to at least 100 seconds
  - $R2$ for a SYN segment MUST be set large enough to provide retransmission of the segment for at least 3 minutes

## TCP User Timeout Option

- TCP has no in-protocol mechanism to signal changes to the local user timeout to the other end of a connection
- Local changes to increase/decrease the Timeout are ineffective because the peer will close the connection based on the local setup
- Sidenote: Solaris provides three timeouts depending on state: `SYN-SENT`, `SYN-RECEIVED`, or `ESTABLISHED`
- Often this timeout is too long or too short (under Linux for a WAN connection it can be 20 Minutes till `ETIMEDOUT` is returned)
- Socket Option:
  - `TCP_USER_TIMEOUT` socket level option (since 08.2011)
  - Legal in any state, effective only in established states (`ESTABLISHED`, `FIN-WAIT-1`, ...)
- The TCP Timeout belongs to the Retransmission Timer – but this is an implementation decision!

## The Raise of New Congestion Control Algorithms

- Question: Given a 1500 byte packet size, an RTT of 100ms and a 10 Gbps link what is the congestion window to fill the pipe?

## The Raise of New Congestion Control Algorithms

- Question: Given a 1500 byte packet size, an RTT of 95[1] and a 10 Gbps link what is the congestion window to fill the pipe?
  - BDP: 125000000 byte $\rightarrow$ Congestion Window: $\underline{\text{833333 packets}}$
  - Slow start: after 19 RTT's the sending rate reaches 10Gbps
- "HighSpeed TCP for Large Congestion Windows" RFC 3649, 2003[11]
- Two areas:
  - Slow start (connection start, timeout and longer time without RX/TX)
  - Congestion avoidance
- Congestion Avoidance:
  - At least 1.6 hours between packet drops
- Slow start:
  - Tens of thousands of packets dropped from one window of data

[1] $(1/(299792458 * 0.7)) * (10000000 * 1000)$

## Network Buffering Problem

- Historic Considerations:
  - Active Queue Management (AQM, RED, ...) was introduced to signal TCP to reduce sending rate (NOTE: congestion control in TCP was the underlying mechanism)
  - Sidenote: not yet written but this signaling was expected to be timely
  - Other transport protocol are judged on TCP – there is no other protocol which is unfair (SCTP, DCCP)
- Problem nowadays:
  - Large buffers are everywhere in the path (memory price)
  - Standard SoHO router:
    - Queues
    - Ring buffer in Network Adapter
  - But there are other problems as well: RED is often not enabled (to complicated for operators)
  - Result: in the case of congestion - no packets are dropped (ECN'ed) at all - they are queued somewhere in the path

- But: timely <u>packet drop is essential for TCP!</u>
- In the end: if you buffer everything the control loop does not work well

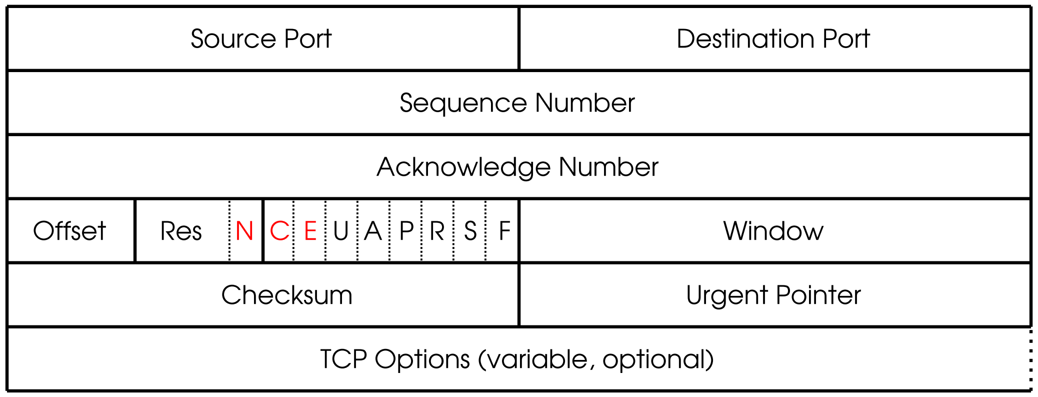
- If latency is an issue analyze your buffers
- If the network is congested: signal it timely – drop packets!
- In the beginning ECN was not used (middle box bugs)
- QoS cannot help you: User IP voice/interactive traffic is classified as data in their networks
- Jim Getty (X Window, HTTP 1.1 Spec, ...)

## ECN

- Every ECN capable sender sets an ECN capable transport (ECT) codepoint in the IP header of every packet
- ”Network devices can then explicitly signal congestion to the receiver by changing the codepoint in the IP header from ECT to ECN (1 bit change) of such packets“
- ”The transport receiver communicates these ECN signals back to the sender, which then performs the appropriate congestion control rate reduction.”
- http://www.imperialviolet.org/binary/ecntest.pdf
  - Responds to SYNs with ECN bits 1,349,711 (99.44%)
  - ECN successfully negotiated 14,407 (1.07%)
  - ECN not supported 14,407 (98.93%)
  - No response to SYNs with ECN bits 7,627 (0.56%)
- References: [29, 6, 21, 32]

## ECN II

- Question:
  - What happens if the receiver does not set ECE if the IP header is EC marked?
  - What happens if some middle-box removes ECE?
- Performance advantage at the expense of competing connections that behave properly
- ECN-nonce is a simple, efficient mechanism to eliminate the potential abuse of ECN



## TCP Option Space

- Background:
  - TCP Options are a mechanism to evolve the TCP (e.g. SACK, TS)
  - With new protocols like MPTCP the TCP Option space is more and more a real issue
  - Options are not exchanged reliable (only in the TWHS phase)
  - Options are not reliable acked by a data ACK (Options are not sticked with a seq number, but even if you stick it to data: what happens if one end does not transfer any data? :-)
  - Real Problem: it is difficult to increase the space of the first SYN - that is the challenge
- Simultaneous open: problem; middle boxes: problem
- NAT: will probably misinterpret options as data (because they are not required to read the options data to know about the option extension)
- TCP Options Overview:
  - SACK (max 18 byte, plus padding of 2 byte)
  - Timestamp (12 byte, plus padding)

- TCP Option Space: max 40 Byte general: the first packet (SYN, SYN/ACK) is perfectly acceptable to send data
- TWHS TCP Option Space: In SYN frames nothing about the destination can be assumed
- Other Issues with TCP Options
  - Some middle-boxes are sensitive about option ordering
  - Some middle-boxes are sensitive about unknown TCP options
- Eventually a 4WHS may be an answer, but it never increases the SYN option space: show stopper! (4WHS Complex, NAT/Firewall problems)

## Rate Halfing

- Fast Retransmit
  - Third Dup-ACK
  - Linux implements a more sophisticated mechanism: dynamically observe packet reordering, result that more than three Dup-ACKs are required to trigger a fast retransmission
- Fast Recovery
  - $RFC2582$ suggests to reduce the window at once
  - Rate halving: Instead, the sender decreases the congestion window size gradually, by one segment per two incoming acknowledgements, until the congestion window reaches half of its original value.
- Rate Halving:
  - Rate-halving avoids pauses in transmission, but is slightly too aggressive after the congestion notification, until the congestion window has reached a proper size.

## Proportional Rate Reduction

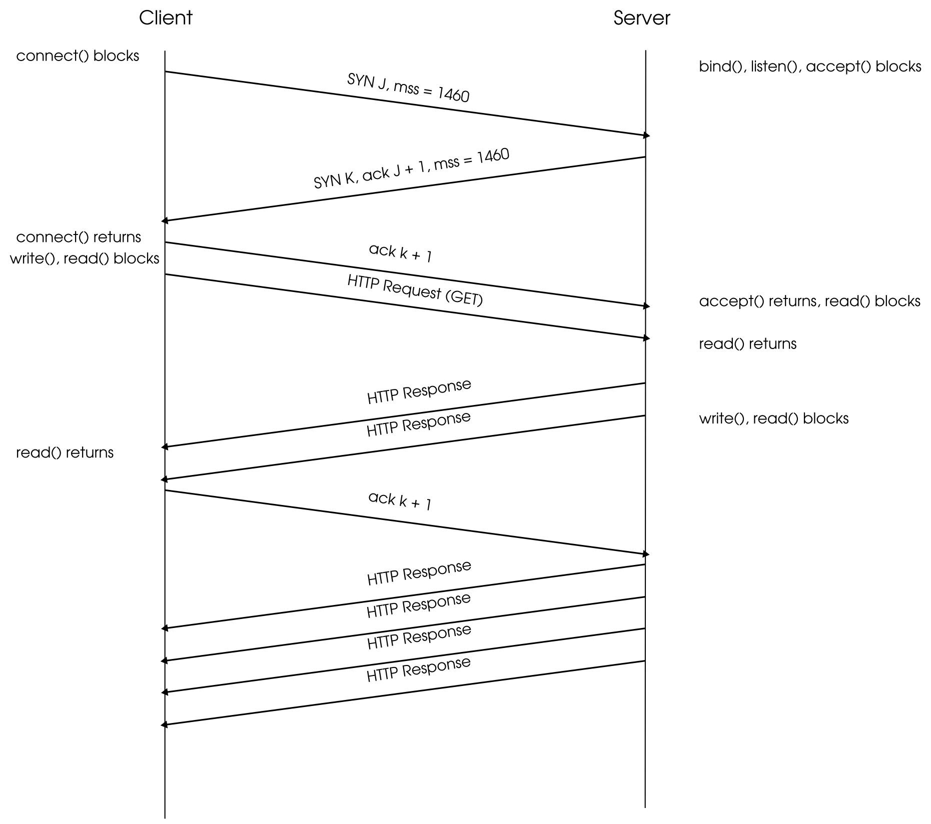
- Rate Reduction Scheme ("successor" to Rate Reduction)
- During loss recovery the amount of data tends to be inaccurate
- Background:
  - If loss is detected at the sender, the sender is required to reduce the CW
  - This reduction occurs in the same RTT as the repair to the loss occurs (retransmission).
  - Normally this is done by <u>not</u> transmitting any data if new ACK arrives
  - Window reduction is implemented by two algorithms:
    1. Fast Recovery
    2. Rate Reduction
  - <u>But:</u> both algorithms are burst-loss-sensitive! Tend to be ineffective if the number of returning ACKs is small ($\leq oldCW/2$) that the effective CW fall below the new calculated CW

## Proportional Rate Reduction

- Proportional Rate Reduction - PRR
  - Goal: at the end of recovery the CW will be close to the calculated CW
- Open Question: if the used CC is wrong, is PRR also wrong? I.e. to aggressive?

# Initial Congestion Window 10

- A larger share of TCP flows is short-lifed (e.g. HTTP) - start up behavior is important
- Actual Web Browsers often open more than one connection
- Problem: round trip time



## TCP Fast Open (TFO)

- Some traffic is short-lived:
  - HTTP (P-HTTP has too many limitations limiting the level of persistency, often middle boxes terminate idle TCP connections, With mobile networks, aggressive power saving algorithms are putting more constraint on the length of TCP connections in order to avoid any unnecessary traffic)
  - DNS (via TCP, except AXFR/IXFR records)
- Idea:
  - Allows data to be carried in the SYN or SYN-ACK
  - Cookie based design
    - Network stack state overhead small
    - Cookies are well known and good understand
- Literature: [19]
- `http://tools.ietf.org/html/draft-cheng-tcpm-fastopen-00`

## Congestion Exposure (ConEx)

- Today:
  - Network may signal congestion to the receiver by ECN markings
  - Receiver passes this information back to the sender in transport-layer feedback
- ConEx: senders inform the network about the congestion encountered by packets in the same flow!
- Senders <u>also relay congestion information</u> back into the network
- All devices across the path can use this information

## Thank You!

### Contact

- Hagen Paul Pfeifer
- hagen.pfeifer@protocollabs.com
- Key-Id: 0x98350C22
- IETF Statistics generated with getauthors script (Jari Arkko)

## Literatur

[1] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (Draft Standard), September 2009.

[2] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC 2581 (Proposed Standard), April 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[3] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309 (Informational), April 1998.

[4] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298.

[5] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *Computer*, 33(5):59–67, 2000.

[6] B. Briscoe. Tunnelling of Explicit Congestion Notification. RFC 6040 (Proposed Standard), November 2010.

[7] B. Constantine, G. Forget, R. Geib, and R. Schrage. Framework for TCP Throughput Testing. RFC 6349 (Informational), August 2011.

[8] M. Duke, R. Braden, W. Eddy, and E. Blanton. A Roadmap for Transmission Control Protocol (TCP) Specification Documents. RFC 4614 (Informational), September 2006. Updated by RFC 6247.

[9] L. Eggert. Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status. RFC 6247 (Informational), May 2011.

[10] L. Eggert and F. Gont. TCP User Timeout Option. RFC 5482 (Proposed Standard), March 2009.

[11] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[12] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Quick-Start for TCP and IP. RFC 4782 (Experimental), January 2007.

[13] S. Floyd, A. Arcia, D. Ros, and J. Iyengar. Adding Acknowledgement Congestion Control to TCP. RFC 5690 (Informational), February 2010.

[14] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582 (Experimental), April 1999. Obsoleted by RFC 3782.

[15] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9:392–403, 2001.

[16] F. Gont. TCP's Reaction to Soft Errors. RFC 5461 (Informational), February 2009.

[17] F. Gont. Reducing the TIME-WAIT State Using TCP Timestamps. RFC 6191 (Best Current Practice), April 2011.

[18] F. Gont and A. Yourtchenko. On the Implementation of the TCP Urgent Mechanism. RFC 6093 (Proposed Standard), January 2011.

[19] Seppo Hätönen, Aki Nyrhinen, Lars Eggert, Stephen Strowes, Pasi Sarolahti, and Markku Kojo. An experimental study of home gateway characteristics. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 260–266, New York, NY, USA, 2010. ACM.

[20] R. Hay and W. Turkal. TCP Option to Denote Packet Mood. RFC 5841 (Informational), April 2010.

[21] A. Kuzmanovic, A. Mondal, S. Floyd, and K. Ramakrishnan. Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets. RFC 5562 (Experimental), June 2009.

[22] G. Lebovitz and E. Rescorla. Cryptographic Algorithms for the TCP Authentication Option (TCP-AO). RFC 5926 (Proposed Standard), June 2010.

[23] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm. RFC 6206 (Proposed Standard), March 2011.

[24] V. Paxson and M. Allman. Computing TCP's Retransmission Timer. RFC 2988 (Proposed Standard), November 2000. Obsoleted by RFC 6298.

[25] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's Retransmission Timer. RFC 6298 (Proposed Standard), June 2011.

[26] K. Pister, P. Thubert, S. Dwars, and T. Phinney. Industrial Routing Requirements in Low-Power and Lossy Networks. RFC 5673 (Informational), October 2009.

[27] J. Postel. DoD standard Transmission Control Protocol. RFC 761, January 1980. Obsoleted by RFC 793.

[28] A. Ramaiah, R. Stewart, and M. Dalal. Improving TCP's Robustness to Blind In-Window Attacks. RFC 5961 (Proposed Standard), August 2010.

[29] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. Updated by RFCs 4301, 6040.

[30] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata. Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP. RFC 5682 (Proposed Standard), September 2009.

[31] W. Simpson. TCP Cookie Transactions (TCPCT). RFC 6013 (Experimental), January 2011.

[32] N. Spring, D. Wetherall, and D. Ely. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. RFC 3540 (Experimental), June 2003.

[33] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001 (Proposed Standard), January 1997. Obsoleted by RFC 2581.

[34] X. Xiao, A. Hannan, V. Paxson, and E. Crabbe. TCP Processing of the IPv4 Precedence Field. RFC 2873 (Proposed Standard), June 2000.

[35] A. Zimmermann and A. Hannemann. Making TCP More Robust to Long Connectivity Disruptions (TCP-LCD). RFC 6069 (Experimental), December 2010.