

The NewTCP Project

Optimising TCP for the 21st Century

James Healy, Lawrence Stewart

jhealy@swin.edu.au, lastewart@swin.edu.au

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology



Outline



- TCP Congestion Control
 - History
 - Known Weaknesses
- Proposed Improvements
- HTCP
- Tools
- Future Work



- cwnd: Congestion window
- MSS: Maximum segment size
- ssthresh: slow start threshold
- ACK: TCP Acknowledgment
- RTT: Round trip time
- BDP: Bandwidth-delay product
- RFC: Request For Comment



In The Beginning...



- No congestion control
- Data sent as fast as the receiver's buffers could handle
- No consideration for network buffers
- Meltdown in the mid/late 1980's





- Van Jacobson: 1988 SIGCOMM Paper
 - Maintain a congestion window that caps the number of "in flight" bytes
- RFC1122: Requirements for Internet Hosts (1989)
 - Mandated the implementation of Van Jacobson's congestion control mechanism in all TCPs
- RFC2581: TCP Congestion Control (1999)
 - Colloquially known as "Reno" TCP

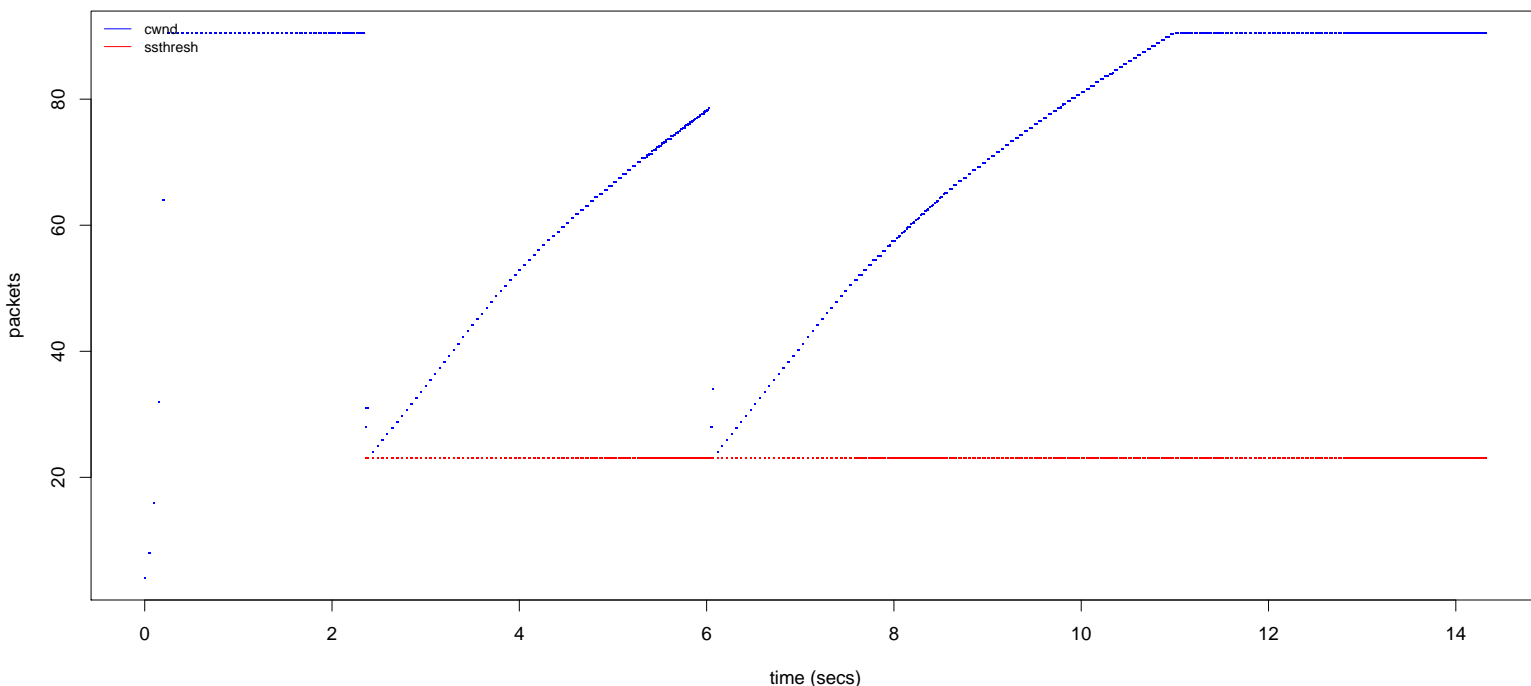


Reno Basics



- Initialisation:
 - $cwnd = MSS$
 - $ssthresh = \infty$
- Increasing $cwnd$:
 - If $cwnd < ssthresh$: $cwnd_{new} = cwnd_{prev} * 2$ for every ACK received
 - If $cwnd \geq ssthresh$: $cwnd_{new} = cwnd_{prev} + MSS$ every RTT
- Decreasing $cwnd$:
 - For every packet lost: $cwnd_{new} = ssthresh = cwnd_{prev} * 0.5$
- Reno is an Additive Increase, Multiplicative Decrease (AIMD) algorithm





If only life were that simple...



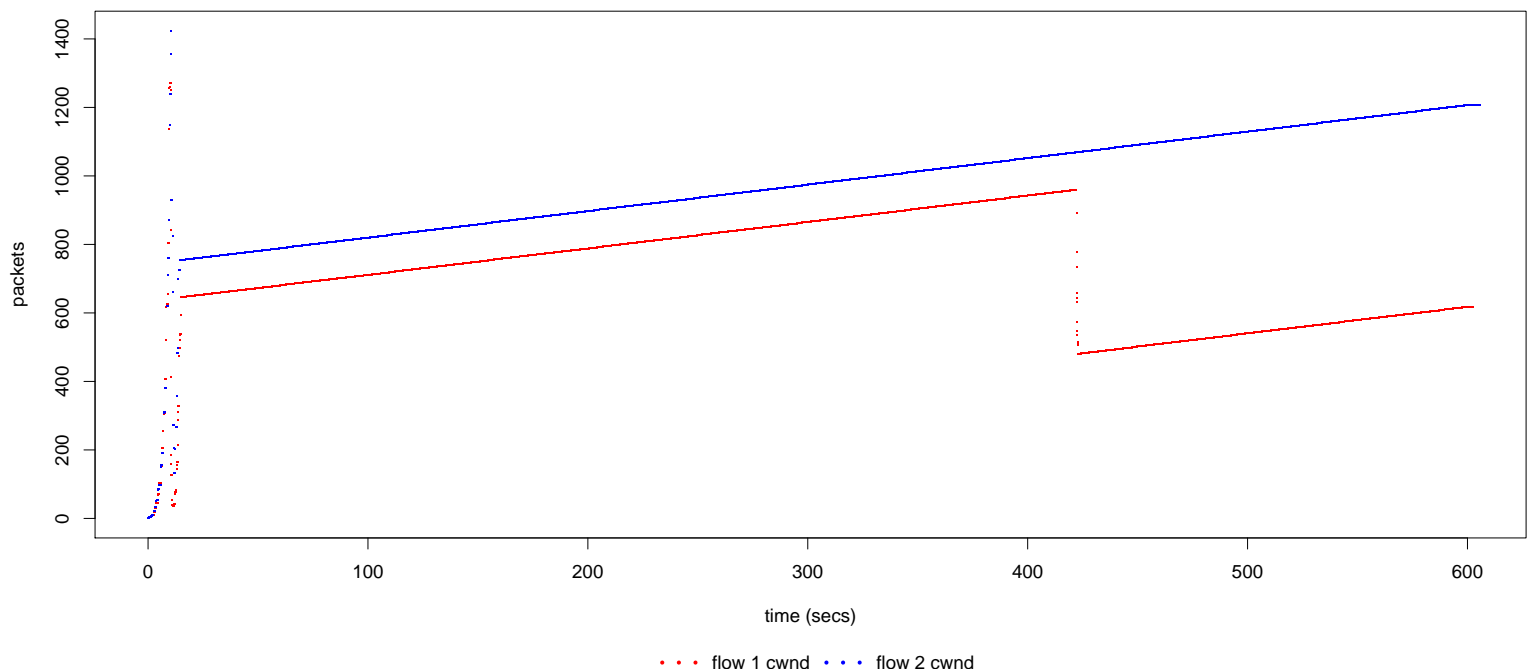
- The Internet's interconnection characteristics are diverse
- Many differences in TCP implementations where RFCs ambiguous or non-specific
- Many edge cases where TCP performance is sub-optimal
- Focus of significant research over the past 2 decades
- Many incremental improvement RFCs flowed from experience and research
 - RFC 2018, 3042, 3390, 3782 (NewReno)...



- There are still problems with TCP
- Long, fast links (large BDP)
 - Increasingly prevalent for international connections
 - 100's of Mbps
 - RTTs in excess of 100ms
- Wasted link capacity
 - Takes minutes/hours for cwnd to expand
 - A single packet loss can devastate throughput
- Solution is not clear cut
 - Increase cwnd more/faster when all is well?
 - Reduce it less/slower when packets are lost?
 - Numerous tradeoffs must be balanced with any change



fairness_reno_320ms_100mbit_1.0per-bdp-1



Let the battle begin!



- Competing congestion control proposals
 - Fast TCP
 - H TCP
 - Vegas TCP
 - Scalable TCP
 - BIC/CUBIC TCP
 - Compound TCP
 - ... the list goes on
- All identical "on the wire"
- Vary the cwnd increase/decrease functions



Why Has the IETF Not Rescued Us?



- Little agreement on how to compare and evaluate proposals
- Metric definitions are blurry or undefined, e.g.
 - Fairness between competing flows
 - Convergence time
- Are some metrics more important/relevant than others?
- Difficult to model dynamic systems in large scale



The Wild Wild West



- Several operating systems have taken the law into their own hands
 - Linux uses CUBIC TCP
 - Windows Vista uses Compound TCP
- Reality check: several TCP variants are already interacting with each other in the wild



Caia Seminar

<http://www.caia.swin.edu.au> jhealy@swin.edu.au, lastewart@swin.edu.au

October 16, 2007 13

CAIA's NewTCP Project



- Implement one of the proposals (HTCP) in FreeBSD
- Provide analysis of HTCP's performance
- Contribute to the community's greater knowledge along the way



Caia Seminar

<http://www.caia.swin.edu.au> jhealy@swin.edu.au, lastewart@swin.edu.au

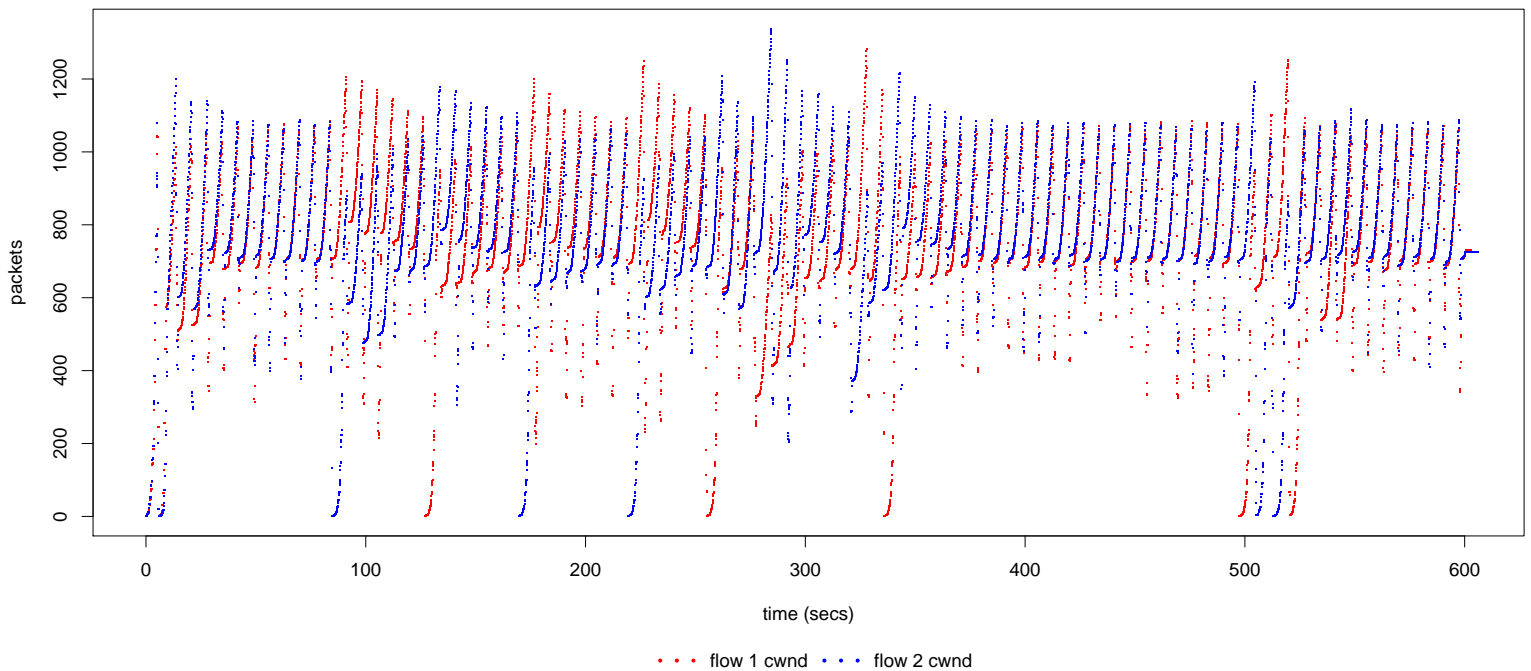
October 16, 2007 14



- Proposed by the Hamilton Institute, Ireland
- Increase factor a function of time since last congestion event
 - The further we get from congestion, the larger the increase
- Multiplicative decrease factor based on RTT
 - Can vary between [0.5, 0.8]
 - The larger the RTT, the smaller the decrease
 - Remains multiplicative
- Available in Linux for the last 2 years



fairness_htcp_160ms_100mbit_1.0per-bdp-1

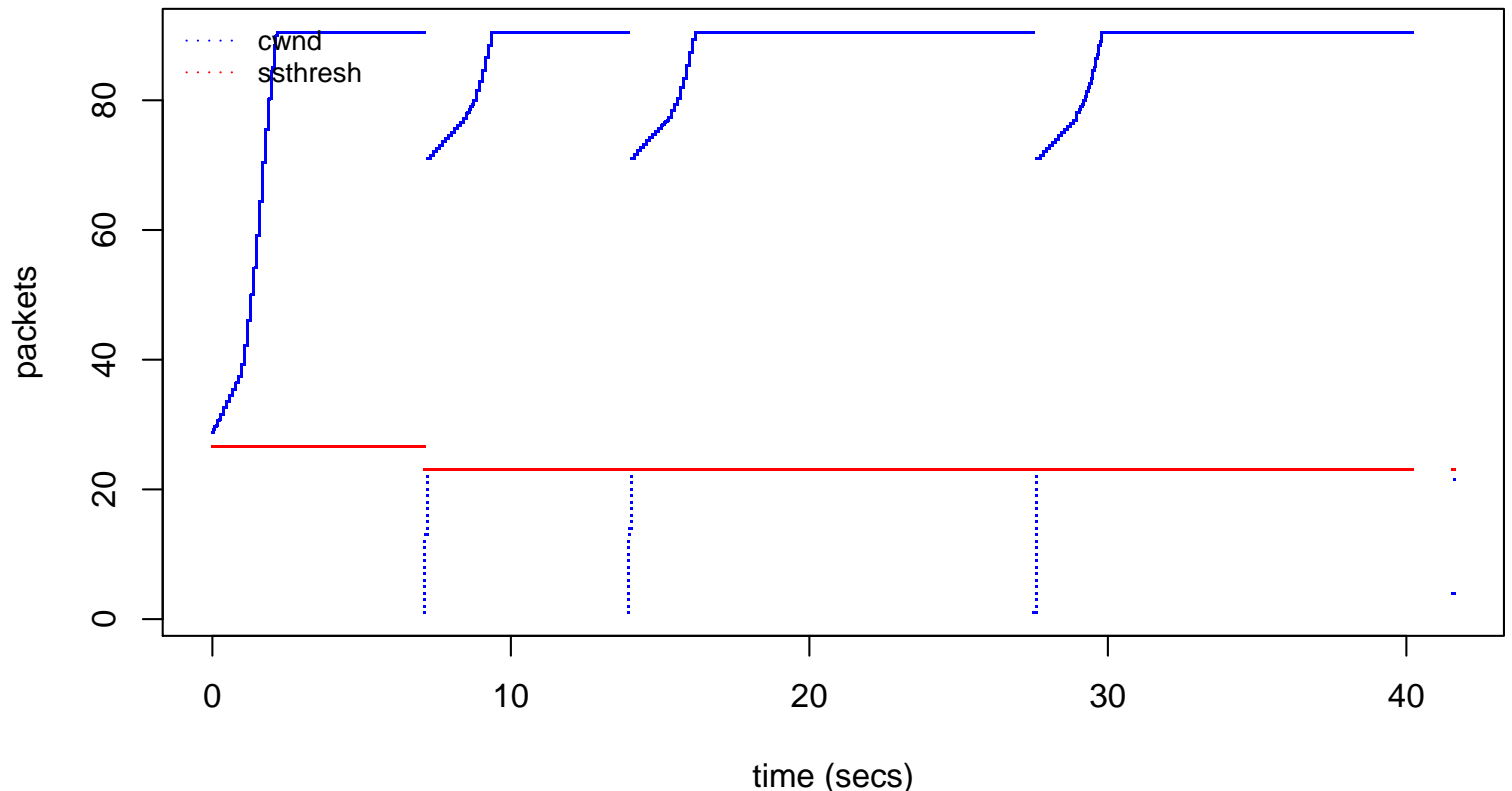




- "Clean room" implementation for FreeBSD 7
- Ignored the existing Linux code
 - Are the specs sufficient?
 - Avoid licencing issues
- Architecture
 - Pluggable congestion control
 - Replaced hard coded NewReno with hooks
 - Loadable module implementations for each algorithm
 - Only NewReno and HTCP at this stage



**FreeBSD 7.0 htcp
(100ms RTT, reliable loss, inflight disabled)**



Implementation Goodies



- Pluggable congestion control is a novel addition to FreeBSD
- New algorithms can be implemented and evaluated quickly
 - HTCP: 230 lines of code

```
struct tcp_cc_functions {
    char name[TCP_CC_MAX_ALGORITHM_NAME_LEN];
    int (*init) (struct tcpcb *tp);
    void (*deinit) (struct tcpcb *tp);
    void (*tcp_ssthresh_update) (struct tcpcb *tp);
    void (*tcp_cwnd_init) (struct tcpcb *tp);
    void (*tcp_cwnd_increase) (struct tcpcb *tp);
    void (*tcp_cwnd_decrease) (struct tcpcb *tp, struct tcphdr *th);
    void (*tcp_cwnd_update_after_idle) (struct tcpcb *tp);
    void (*tcp_cwnd_update_after_timeout) (struct tcpcb *tp);
    void (*tcp_pre_fr) (struct tcpcb *tp);
};
```



Useful Tools



- Statistical Information for TCP Research (SIFTR)
- Deterministic Packet Dropper (DPD)
- Iperf





- Loadable FreeBSD 5/6/7 kernel module
- Logs state on all active TCP connections to a CSV file
- Currently logs 18 variables
 - Flow source/destination
 - Cwnd
 - Ssthresh
 - Estimated RTT
 - etc
- BSD licenced, available from <http://caia.swin.edu.au/newtcp/tools.html>



DPD



- Loadable FreeBSD 5/6/7 kernel module
- Drop arbitrary packets from TCP flows
 - Configuration example: 10,20,100-105,200
- Useful for comparing behaviour related to loss events
- Not for simulating real packet loss scenarios
- BSD licenced, available from <http://caia.swin.edu.au/newtcp/tools.html>





- Network benchmarking tool
- Developed outside of CAIA
- NewTCP has submitted patches upstream



Future Work



- Complete FreeBSD HTCP testing
- Release TCP test and analysis suite
- Write up research results for publication



Conclusion



- TCP has served us very well to date
- TCP can be incrementally improved to increase performance
 - Current research focus is on high BDP paths
- HTCP is one of many variants vying for the spotlight
- We have:
 - Implemented HTCP for FreeBSD purely from specifications
 - Begun testing our implementation
 - Developed useful TCP testing and analysis tools



Questions?

