SWINBURNE UNIVERSITY OF TECHNOLOGY

Evolving TCP Using FreeBSD

Code, Tools, Research & Results

Lawrence Stewart

lastewart@swin.edu.au

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

Outline

Detailed outline (section 1 of 4)

1 Recap

- Where are we today
- Open issues

Where are we today

- Many incremental (partially implemented) improvements
- State of the CC union
  - NewReno is defacto standard with warts (LFN, wireless)
  - Many new proposals
  - BSD still uses NewReno
  - Linux uses CUBIC
  - Windows Vista uses Compound
- TCP/IP stack enhancements e.g.
  - CSO/TSO/LRO/TOE
  - Various locking/caching tricks
  - Socket buffer autotuning

Open issues

- High-speed CC algorithms [1]
  - FAST, HS-TCP, H-TCP, CTCP, CUBIC, etc.
- Delay based CC algorithms
- How do we compare and evaluate TCPs?
- Multipath
- CSO/TSO/LRO/TOE obscure behaviours
- Testing/verification of TCP/IP stack behaviour

[1] Nice summary:
http://kb.pert.geant2.net/PERTKB/TcpHighSpeedVariants

Detailed outline (section 2 of 4)

2 FreeBSD As A RnD Platform

- At a Glance
- Modular Congestion Control
- SIFTR
- ALQ
- DPD
- TCP Reassembly Queue

At a Glance

- Modular congestion control
  - In svn project branch, coming to FreeBSD 7 and 8 soon
  - BSD licenced Newreno, HTCP & CUBIC implementations available
  - Sponsord by Cisco Systems
- Statistical Information for TCP Research (SIFTR)
  - FreeBSD kld to gather CSV in-kernel TCP endpoint connection data
  - Similar concept to Web100 with more variables
  - Sponsored by Cisco Systems and the FreeBSD Foundation
- Deterministic Packet Discard (DPD)
  - Adds 'pls' (packet loss set) option for dummynet pipes
  - e.g. ipfw pipe 1 config pls 1,5-10,30 would drop packets 1, 5-10 inclusive and 30
- Dummynet Forensic logging support
  - Log queue state on each packet event

Modular Congestion Control

- NEWS
  - Project moved into public svn repository: projects/tcp_cc_8.x
  - Completed CUBIC implementation (unlikely to be more from me)
  - Significant locking improvements
  - Maintaining both 7.x and 8.x patches
- TODO for 8.x (roughly in order)
  - Commit ABI breaking parts
  - Finish ECN/ABC/VIMAGE integration
  - Complete documentation
  - Commit to 8.x with experimental status i.e. no ABI guarantees
- ISSUES
  - Simple framework may be needed for CC-related algorithm-agnostic tasks
  - Should we consider moving more variables into a CC struct?

Modular Congestion Control

- Defined in `<netinet/cc.h>`

```
struct cc_algo { /* specify one per CC algorithm */
char name[TCP_CA_NAME_MAX];
int (*mod_init) (struct tcpcb *tp);
int (*mod_destroy) (struct tcpcb *tp);
int (*cb_init) (struct tcpcb *tp);
void (*cb_destroy) (struct tcpcb *tp);
void (*conn_init) (struct tcpcb *tp);
void (*ack_received) (struct tcpcb *tp, struct tcphdr *th);
void (*pre_fr) (struct tcpcb *tp, struct tcphdr *th);
void (*post_fr) (struct tcpcb *tp, struct tcphdr *th);
void (*after_idle) (struct tcpcb *tp);
void (*after_timeout) (struct tcpcb *tp);
STAILQ_ENTRY(cc_algo) entries;
};
```

Modular Congestion Control

- Housekeeping

```
/* called during TCP/IP stack initialisation on boot */
void cc_init(void);

/* dynamically registers a new CC algorithm */
int cc_register_algo(struct cc_algo *);

/* dynamically deregisters a CC algorithm */
int cc_deregister_algo(struct cc_algo *);

/* macro that hides housekeeping code from modules */
DECLARE_CC_MODULE(ccname, ccalgo);
```

Modular Congestion Control

- Minor ABI-breaking additions to struct tcpcb

```
struct tcpcb {
    ....

/* CC function pointers to use for this connection */
struct cc_algo *cc_algo;

/* connection specific CC algorithm data */
void    *cc_data;
};
```

KPI/API/Configuration

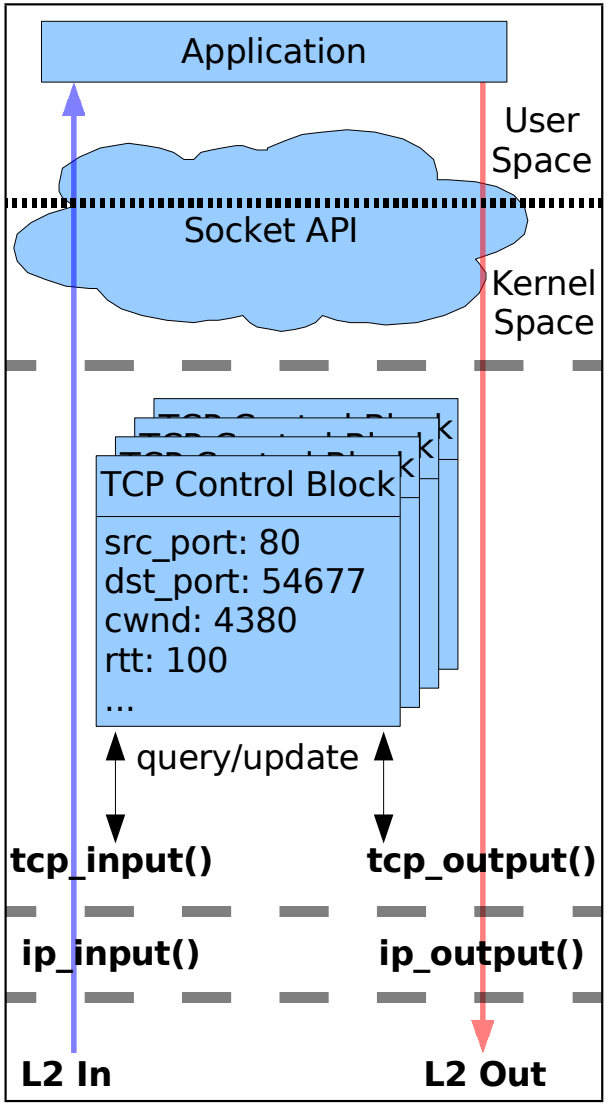
- New `net.inet.tcp.cc` sysctl tree with variables:
  - `available`: comma-separated list of available CC algorithms
  - `algorithm`: current system default CC algorithm
- Removed `net.inet.tcp.newreno` sysctl variable
- New socket option `TCP_CONGESTION` defined in `tcp.h`
  - Override system default CC algorithm using setsockopt(2)
  - Same as Linux define e.g. Iperf -Z option works

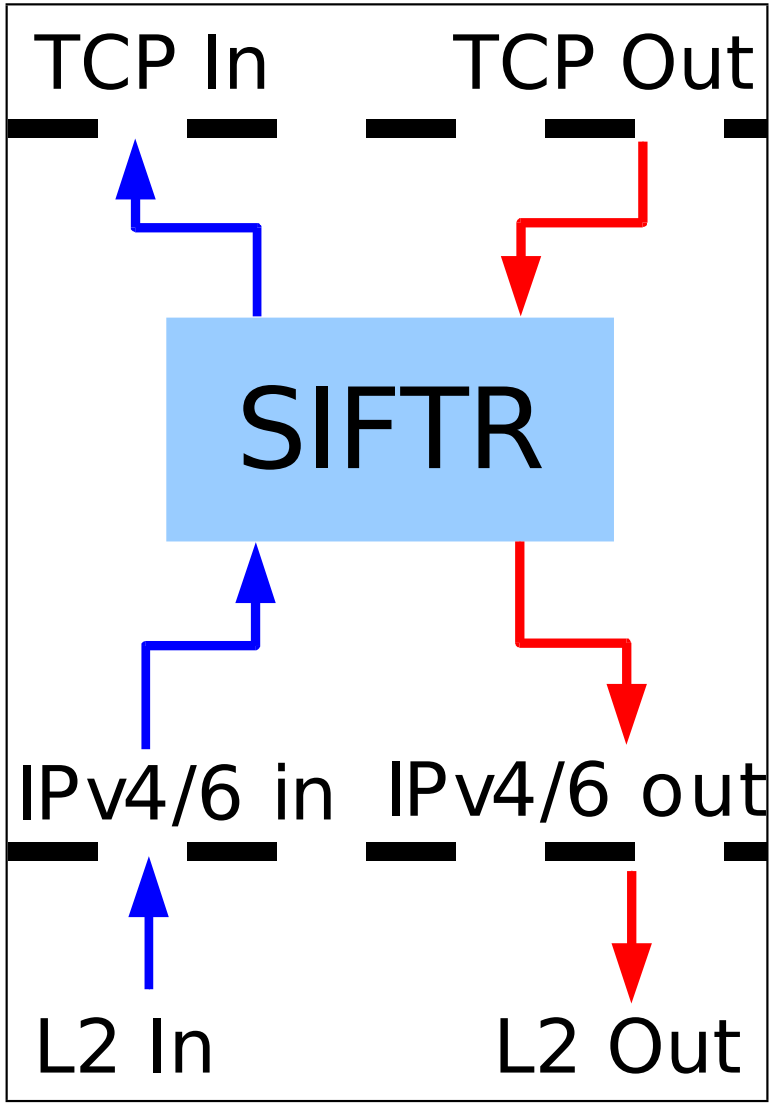
SIFTR

- Statistical Information For TCP Research
- FreeBSD [6,7,8] kernel module
- BSD licenced source [2]
- Similar base concept to Web100
- Event triggered (not poll based)
- Currently logs 25 different variables to file as CSV data [3]
- Plan to integrate into base system for 8.x
- Work on v1.2.x sponsored by the FreeBSD Foundation

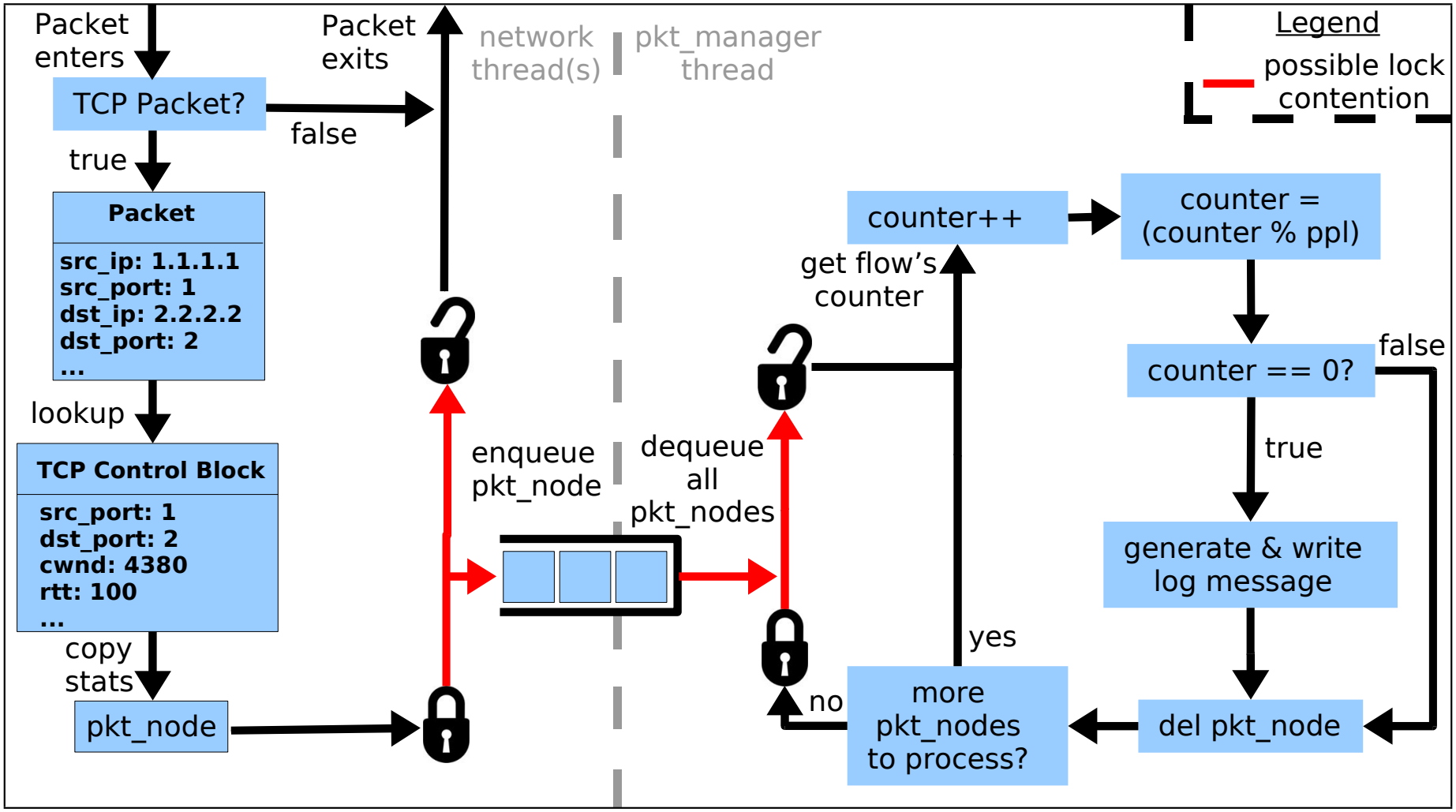
[2] Available from: http://caia.swin.edu.au/urp/newtcp/tools.html
[3] See README in SIFTR distribution for specific details

SIFTR


Application
User Space
Socket API
Kernel Space
TCP Control Block
src_port: 80
dst_port: 54677
cwnd: 4380
rtt: 100
...
query/update
tcp_input()
tcp_output()
ip_input()
ip_output()
L2 In
L2 Out


TCP In
TCP Out
SIFTR
IPv4/6 in
IPv4/6 out
L2 In
L2 Out

SIFTR


Packet enters
TCP Packet?
false
Packet exits
true
Packet
src_ip: 1.1.1.1
src_port: 1
dst_ip: 2.2.2.2
dst_port: 2
...
lookup
TCP Control Block
src_port: 1
dst_port: 2
cwnd: 4380
rtt: 100
...
copy stats
pkt_node
network thread(s)
pkt_manager thread
enqueue pkt_node
dequeue all pkt_nodes
Legend
possible lock contention
counter++
counter = (counter % ppl)
get flow's counter
counter == 0?
false
true
generate & write log message
yes
no
more pkt_nodes to process?
del pkt_node

Asynchronous Logging Queues (ALQ)

- Jeff Roberson's KPI for in-kernel file logging
- Made it build as a LKM
- Extended KPI to allow variable length message support
- Under-the-hood reworked to use a circular buffer
- Useful fallout from SIFTR work
- Would like to add high water mark triggered flushing
- Will commit to 8.x, also backportable

Asynchronous Logging Queues (ALQ)

```
/* unchanged. count=0 now means size arg specifies buffer size */
int alq_open(struct alq **, const char *file, struct ucred *cred,
             int cmode, int size, int count);

/* legacy fixed length write, wraps alq_writen() */
int alq_write(struct alq *alq, void *data, int flags);

/* new variable length write */
int alq_writen(struct alq *alq, void *data, int len, int flags);

/* legacy fixed length ale, wraps alq_getn()*/
struct ale *alq_get(struct alq *alq, int flags);

/* new variable length ale */
struct ale *alq_getn(struct alq *alq, int len, int flags);
```

Deterministic Packet Discard (DPD)

- Patch against FreeBSD 8.x IPFW/Dummynet
- BSD licenced source [4]
- Useful for protocol (not just TCP!) verification and testing
- Adds 'pls' (packet loss set) option for dummynet pipes
- e.g. ipfw pipe 1 config pls 1,5-10,30 would drop packets 1, 5-10 inclusive and 30
- Need to catch up with Luigi's work
- Low priority, but hope to commit to 7.x and 8.x soon

[4] Available from `http://caia.swin.edu.au/urp/newtcp/tools.html`

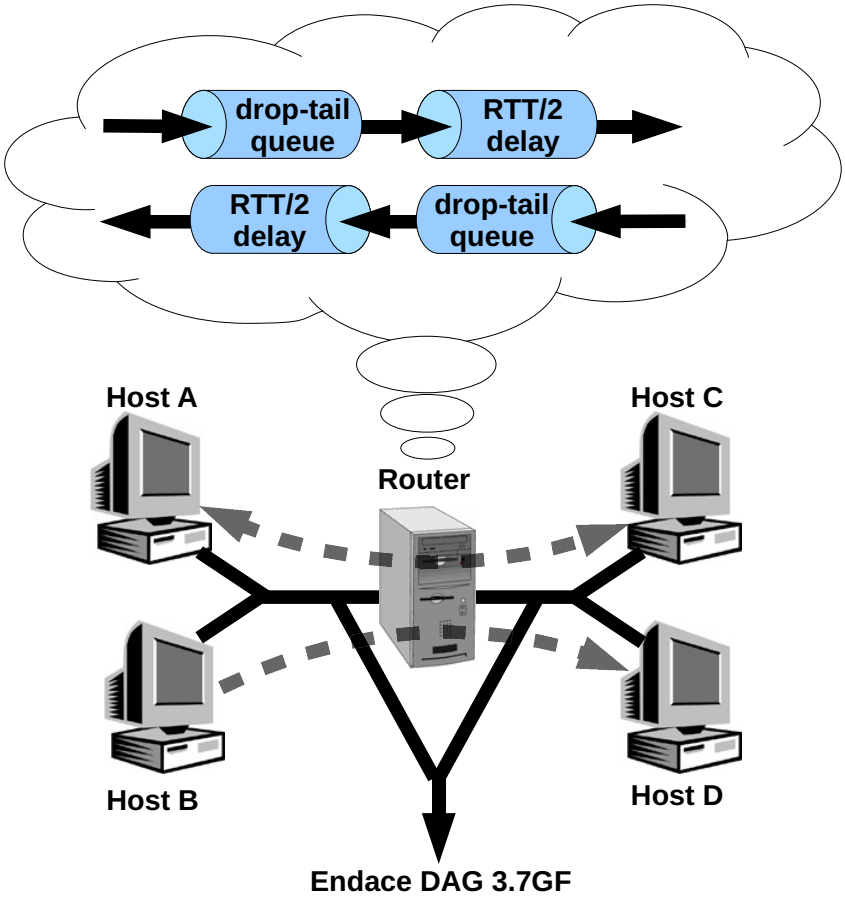
TCP Reassembly Queue

- TCP reassembly queue tuning is inherently connection specific
- Current method is wasteful and can severely damage TCP performance
- Aim to do away with net.inet.tcp.reass.maxqlen
- Adapt reassembly queue based on connection dynamics
- Somewhat akin to socket buffer auto tuning
- Currently WIP (building on Andre's work)
- Sponsored by the FreeBSD Foundation

Detailed outline (section 3 of 4)

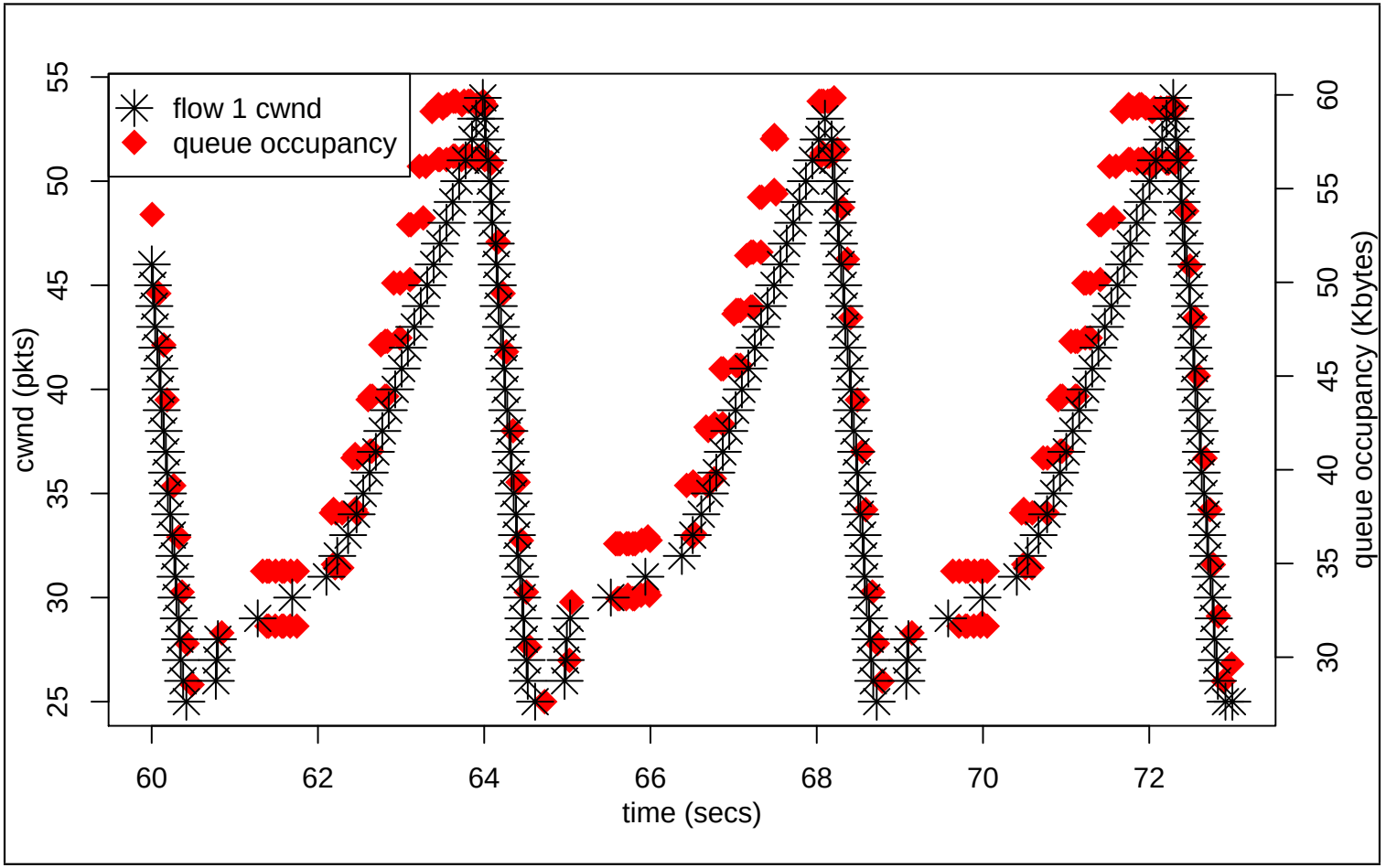
Testbed

- Linux/FreeBSD hosts
- Modular congestion control
- Web100/SIFTR for Linux/FreeBSD testing
- Iperf/Tcpreplay for traffic generation
- FreeBSD dummynet router
- Endace DAG 3.7GF capture card

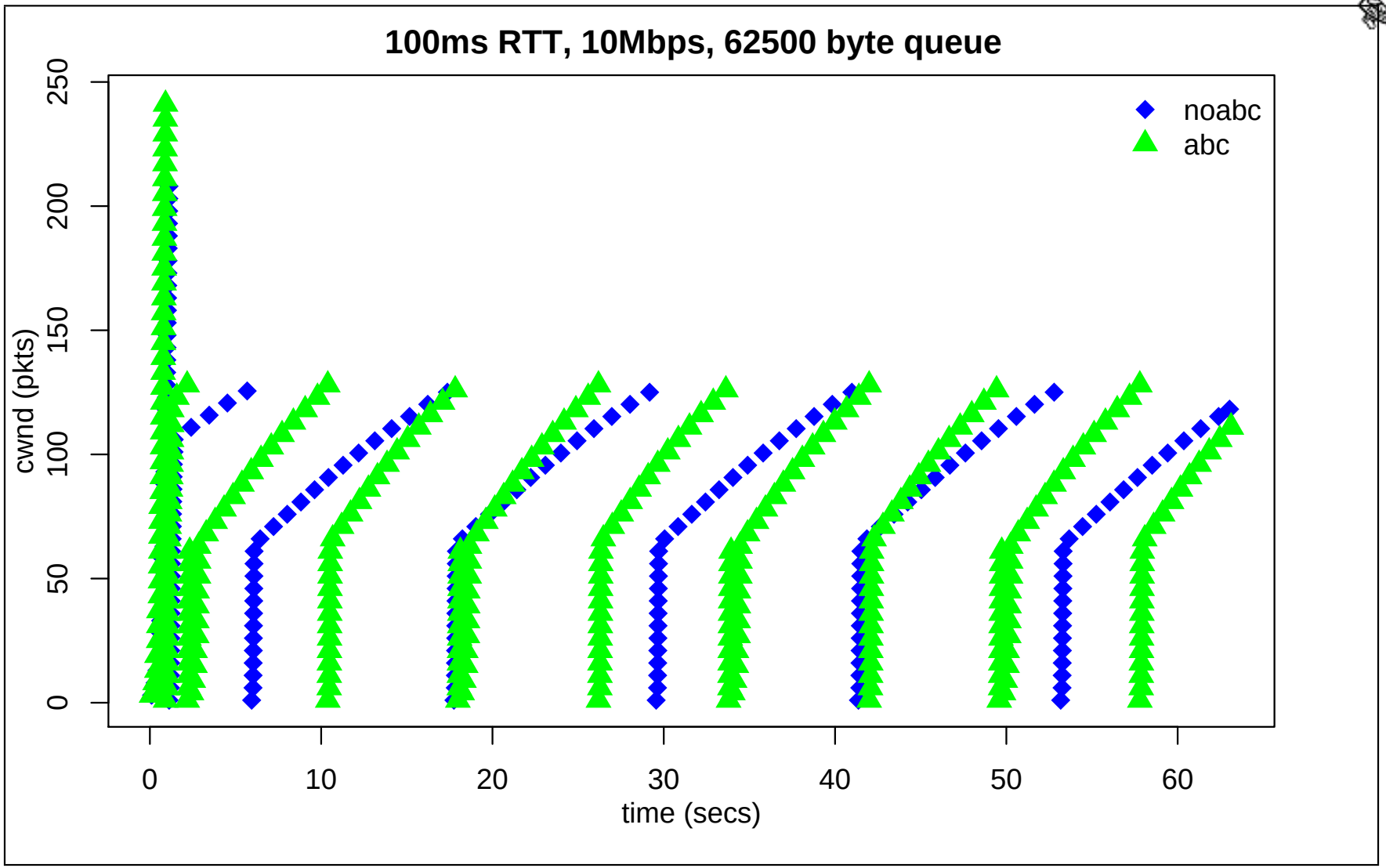


Connection Dynamics

- 1 TCP flow, H-TCP, 100ms RTT, 1Mbps, 60000 byte queue

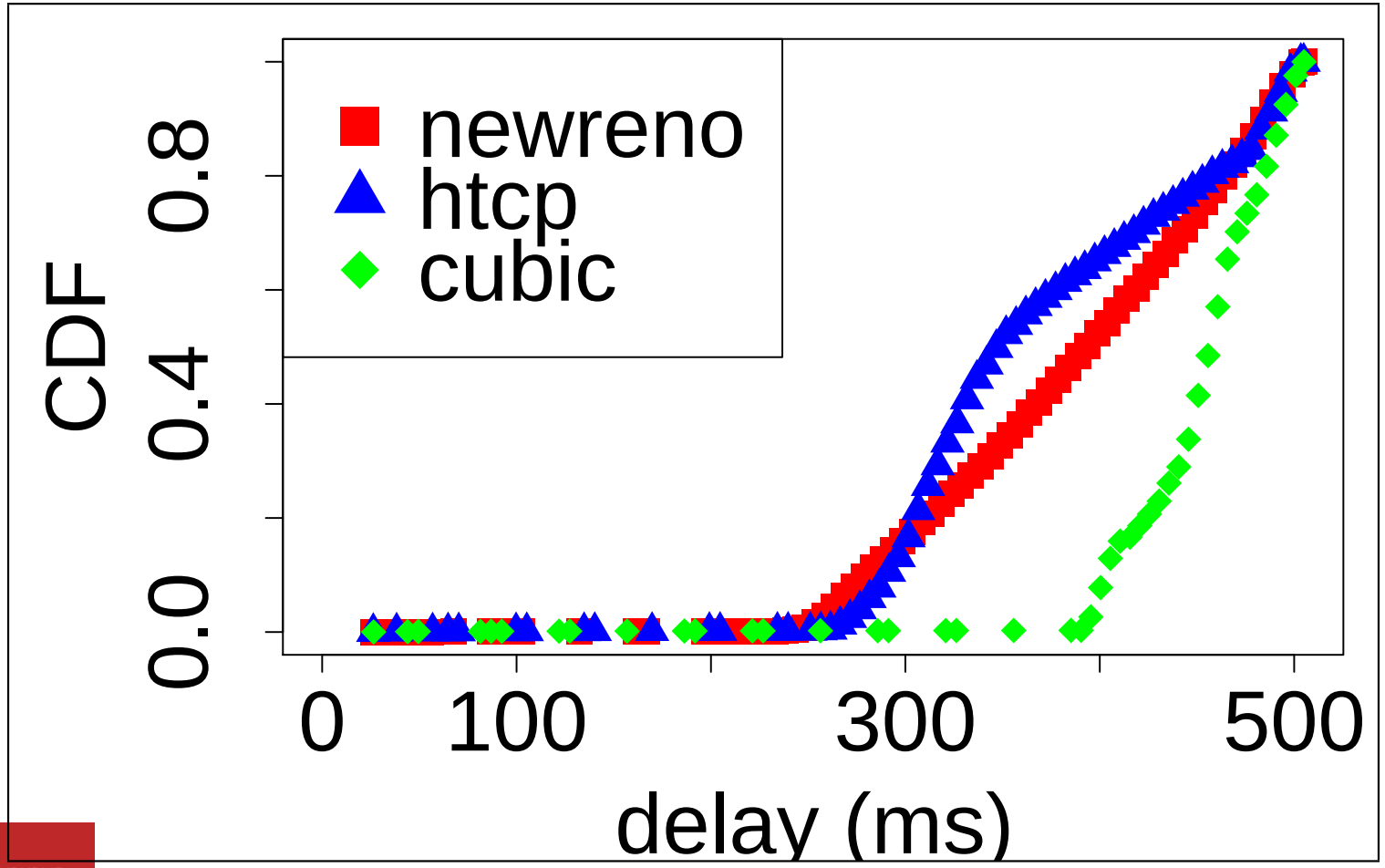


Appropriate Byte Counting (ABC)



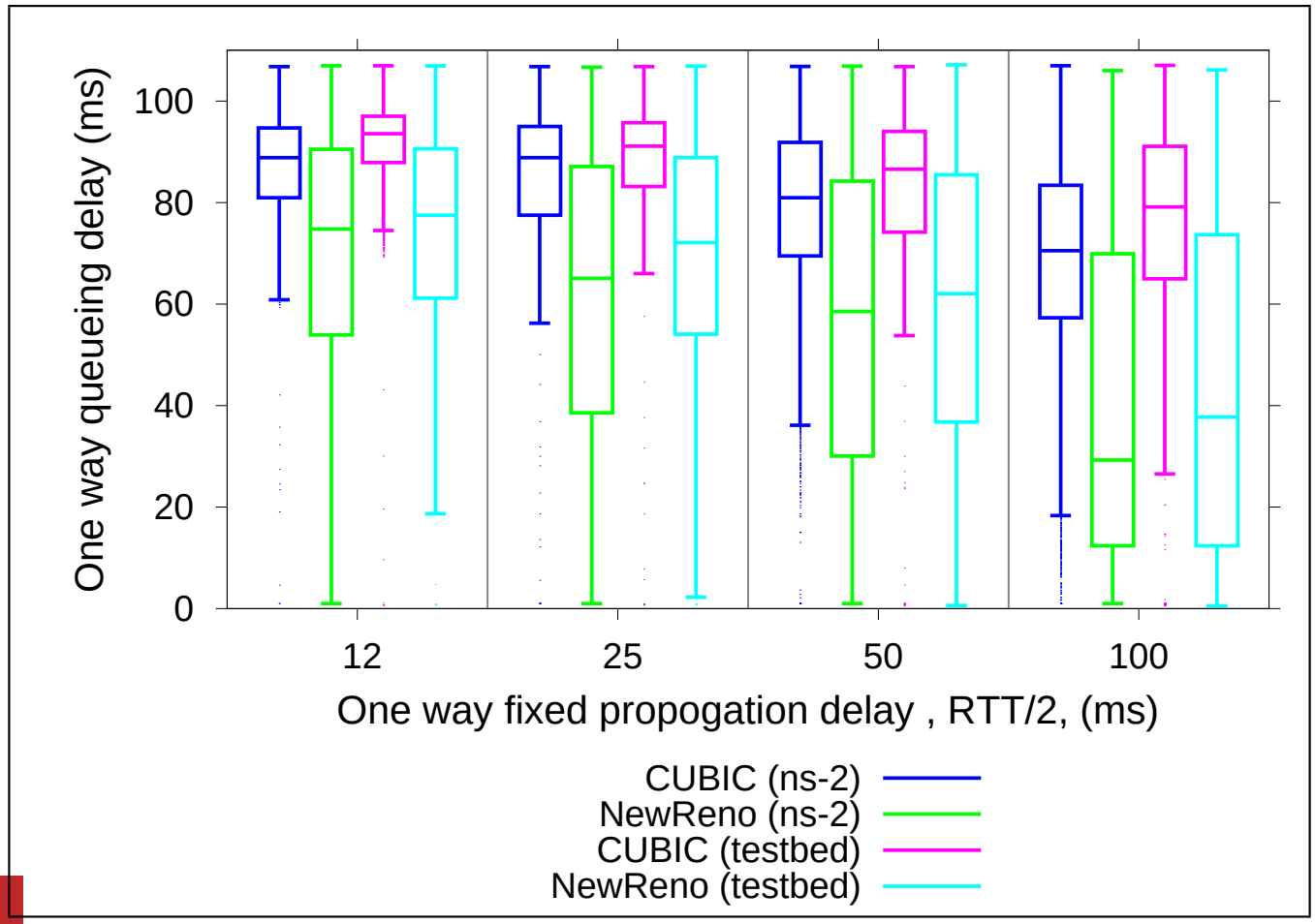
Collateral Damage

- Induced delay: 1 TCP vs 1 CBR UDP flow, 50ms RTT, 1Mbps, 60000 byte queue



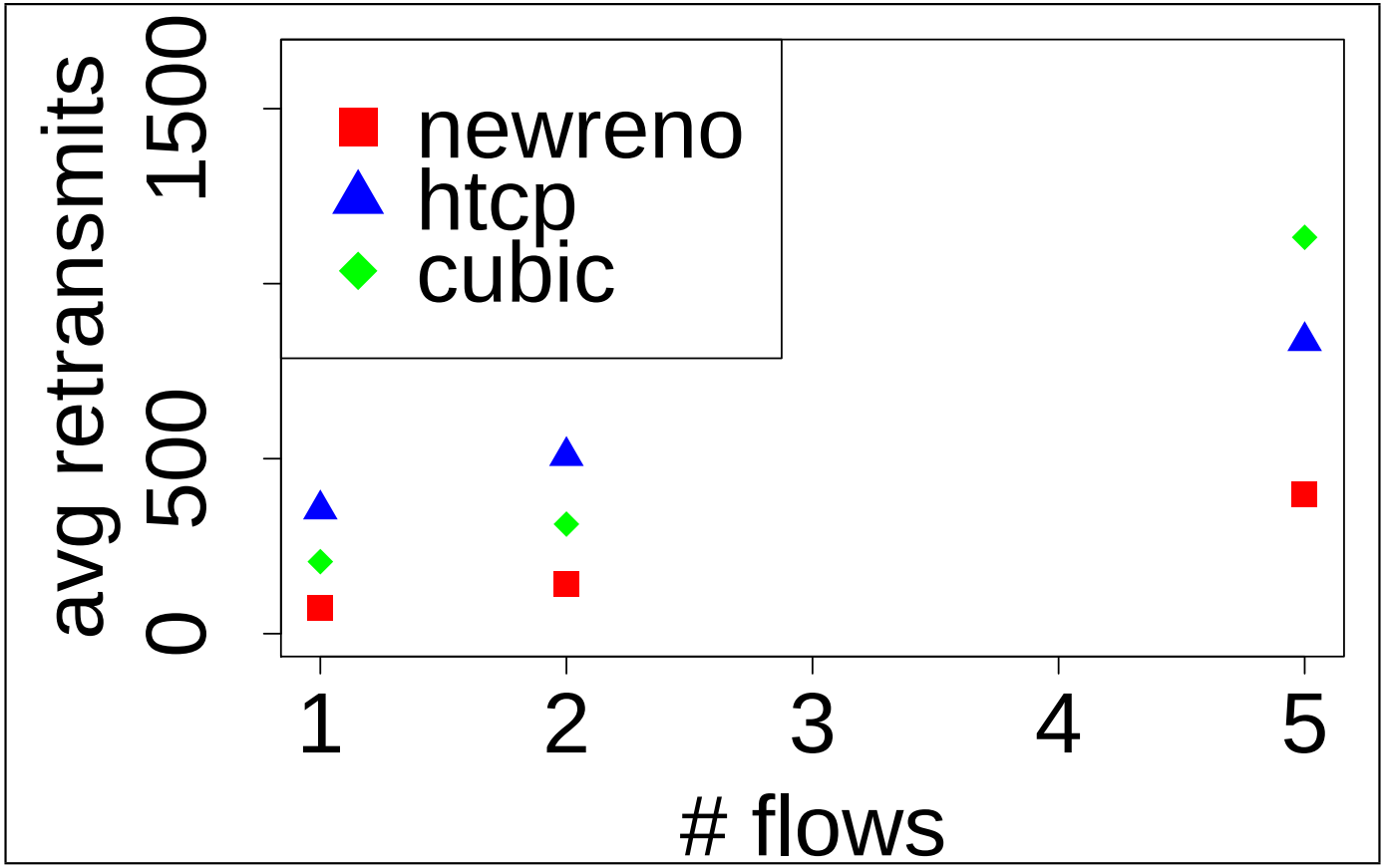
Collateral Damage

- Induced delay: 1 TCP vs 1 CBR UDP flow, 50ms RTT, 1.5Mbps/256Kbps, 20000 byte queue



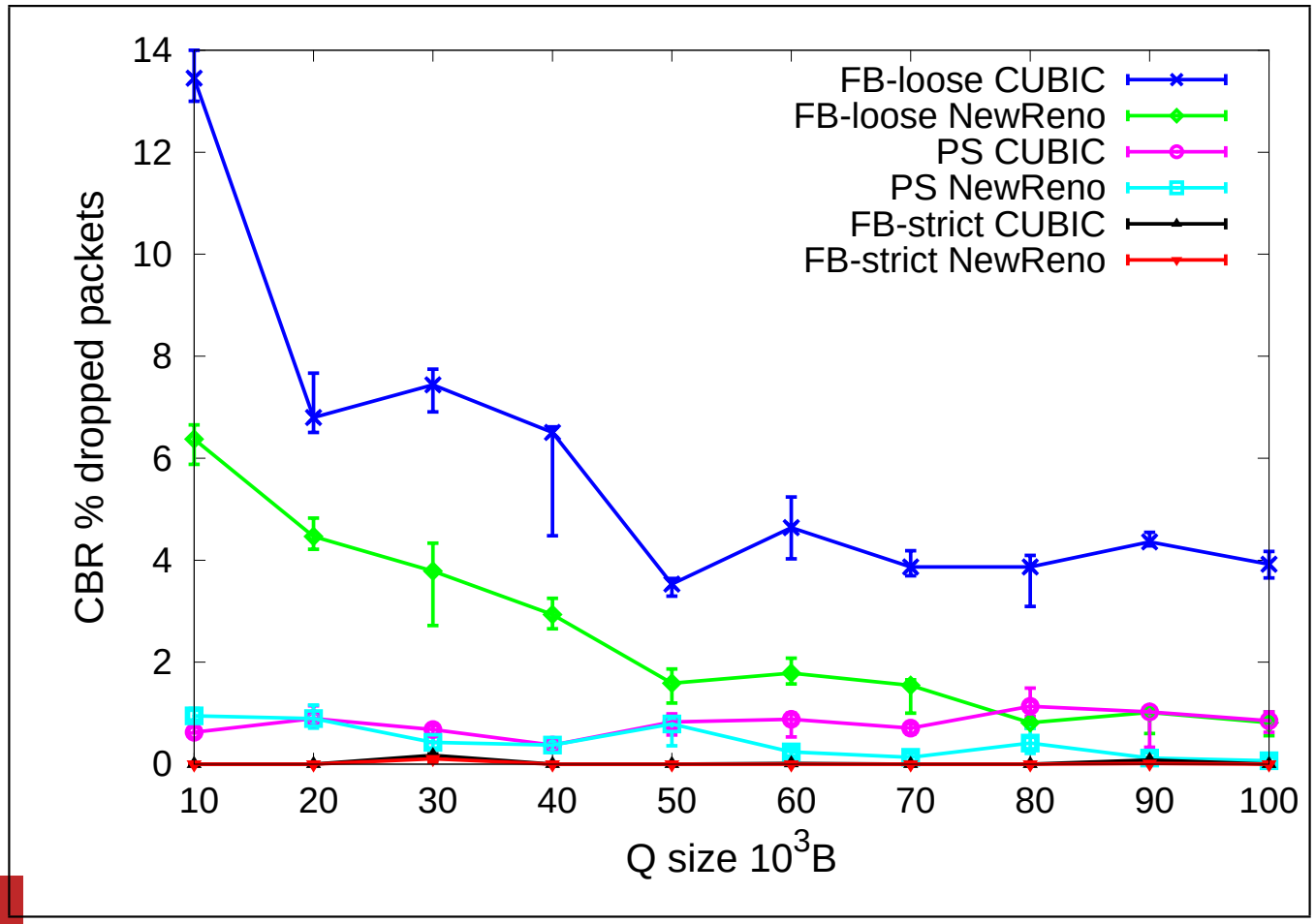
Collateral Damage

- Retransmissions: n TCP vs 1 CBR UDP flow, 50ms RTT, 1Mbps, 60000 byte queue



Subtle Queuing Implications

- Induced CBR loss: 1 TCP vs 1 CBR UDP flow, 100ms RTT, 1.5Mbps/256Kbps, NS



Detailed outline (section 4 of 4)

Ideas for Future Work

- TCP specific:
  - Improve RTT estimator
  - Share CC between TCP/SCTP
  - Rework the host cache
  - Comprehensive RFC compliance check
  - Fix slow-start, FR/FR
- TCP/IP stack in general:
  - Framework for dealing with CSO/TSO/LRO/TOE
  - DTRACEesque instrumentation
  - Testing framework <- next project I want to tackle

Further Information

- Papers
  - Lawrence Stewart, Grenville Armitage, Alana Huebner, "*Collateral Damage: The Impact of Optimised TCP Variants On Real-time Traffic Latency in Consumer Broadband Environments*", IFIP/TC6 NETWORKING 2009, Aachen, Germany, 11-15 May 2009.
  - Grenville Armitage, Lawrence Stewart, Michael Welzl, James Healy, "*An independent H-TCP implementation under FreeBSD 7.0 - description and observed behaviour*", ACM SIGCOMM Computer Communication Review, vol. 38 no. 3 pp. 29-38, July 2008.
- Links
  - `http://caia.swin.edu.au/urp/newtcp/`
  - `http://caia.swin.edu.au/freebsd/etcp09/`
  - `http://people.freebsd.org/~lstewart/`
  - `http://lists.freebsd.org/pipermail/freebsd-net/`

Acknowledgements

- Cisco Systems

- The FreeBSD Foundation

The End

tp->t_state = TCPS_QUESTIONS