Evaluation of Advanced TCP stacks on Fast Long-Distance production Networks

Prepared by Les Cottrell & Hadrien Bullot, SLAC & EPFL, for the

PFLDnet workshop, ANL

February, 2003

www.slac.stanford.edu/grp/scs/net/talk03/pfld-feb04.ppt

Partially funded by DOE/MICS Field Work Proposal on Internet End-to-end Performance Monitoring (IEPM), also supported by IUPAP

Project goals

- Test new advanced TCP stacks, see how they perform on short and long-distance **real production** WAN links
- Compare & contrast: ease of configuration, throughput, convergence, fairness, stability etc.
- For different RTTs, windows, txqueuelen
- Recommend “optimum” stacks for data intensive science (BaBar) transfers using bbftp, bbcp, GridFTP
- Validate simulator & emulator findings & provide feedback

Protocol selection

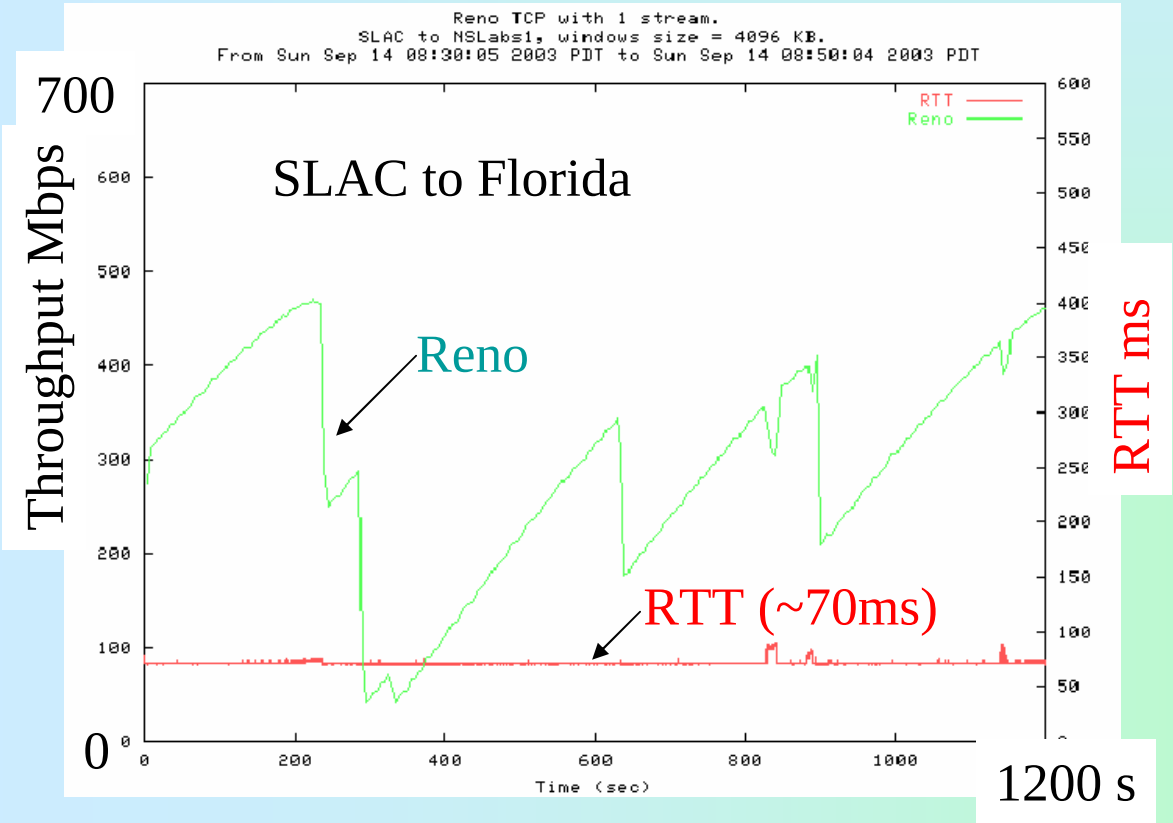
- TCP only
  - No Rate based transport protocols (e.g. SABUL, UDT, RBUDP) at the moment
  - No iSCSI or FC over IP
- Sender mods only, HENP model is few big senders, lots of smaller receivers
  - Simplifies deployment, only a few hosts at a few sending sites
  - No DRS
- Runs on production nets
  - No router mods (XCP/ECN), no jumbos,

Protocols Evaluated

- Linux 2.4 New Reno with SACK: single and parallel streams (P-TCP)
- Scalable TCP (S-TCP)
- Fast TCP
- HighSpeed TCP (HS-TCP)
- HighSpeed TCP Low Priority (HSTCP-LP)
- Binary Increase Control TCP (Bic-TCP)
- Hamilton TCP (H-TCP)

Reno single stream

- Low performance on fast long distance paths
  - AIMD (add *a=1* pkt to *cwnd* / RTT, decrease *cwnd* by factor *b=0.5* in congestion)



P-TCP

- TCP Reno with 16 streams
  - Parallel streams heavily used in HENP & elsewhere to achieve needed performance, so it is today's de facto baseline
  - However, **hard to optimize both the window size AND number of streams since optimal values can vary due to network capacity, routes or utilization changes**

S-TCP

- Uses exponential increase everywhere (in slow start and congestion avoidance)
- Multiplicative decrease factor *b = 0.125*
- Introduced by Tom Kelly of Cambridge

Fast TCP

- Based on TCP Vegas
- Uses both queuing delay and packet losses as congestion measures
- Developed at Caltech by Steven Low and collaborators

HS-TCP

- Behaves like Reno for small values of *cwnd*
- Above a chosen value of *cwnd* (default 38) a more aggressive function is used
- Uses a table to indicate by how much to increase *cwnd* when an ACK is received
- Introduced by Sally Floyd

HSTCP-LP

- Mixture of HS-TCP with TCP-LP (Low Priority)
- Backs off early in face of congestion by looking at RTT
- Idea is to give scavengers service without router modifications
- From Rice University

Bic-TCP

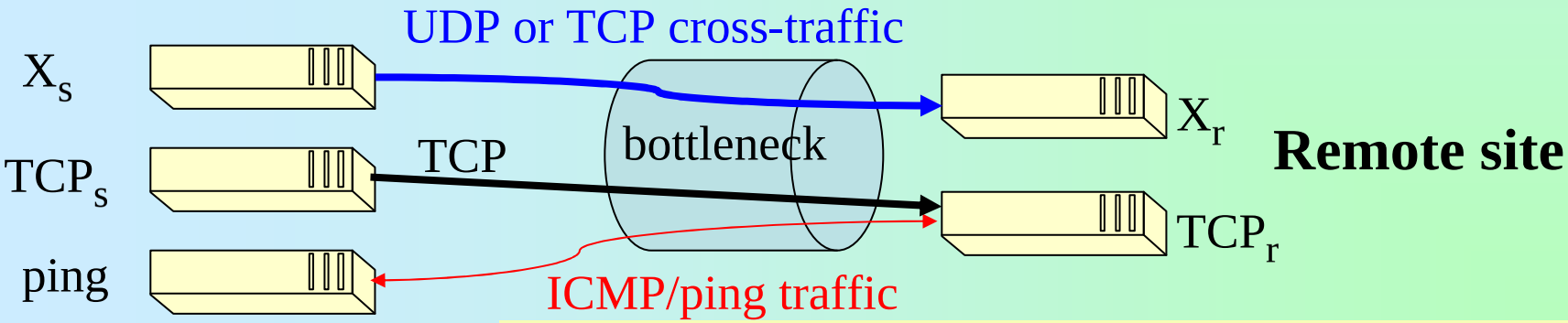
- Combine:
  - An additive increase used for large *cwnd*
  - A binary search increase used for small *cwnd*
  - Developed Injong Rhee at NC State University

H-TCP

- Similar to HS-TCP in switching to aggressive mode after threshold
- Uses an heterogeneous AIMD algorithm
- Developed at Hamilton U Ireland

Measurements

- 20 minute tests, long enough to see stable patterns
- Iperf reports incremental and cumulative throughputs at 5 second intervals
- Ping interval about 100ms
- At sender use: 1 for iperf/TCP, 2nd for cross-traffic (UDP or TCP), 3rd for ping
- At receiver: use 1 machine for ping (echo) and TCP, 2nd for cross-traffic



Ping traffic goes to TCP_r when also running cross-traffic

Otherwise goes to X_r

Networks

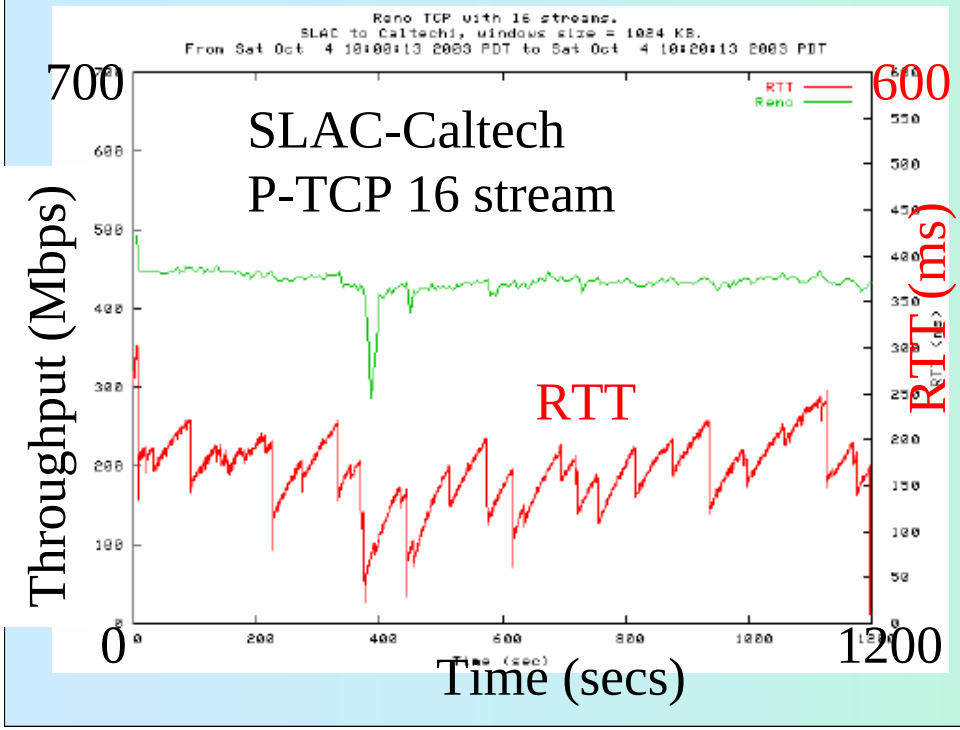
- 3 main network paths
  - Short distance: SLAC-Caltech (RTT~10ms)
  - Middle distance: UFL and DataTAG Chicago (RTT~70ms)
  - Long distance: CERN and University of Manchester (RTT ~ 170ms)
  - Tests during nights and weekends to avoid unacceptable impacts on production traffic

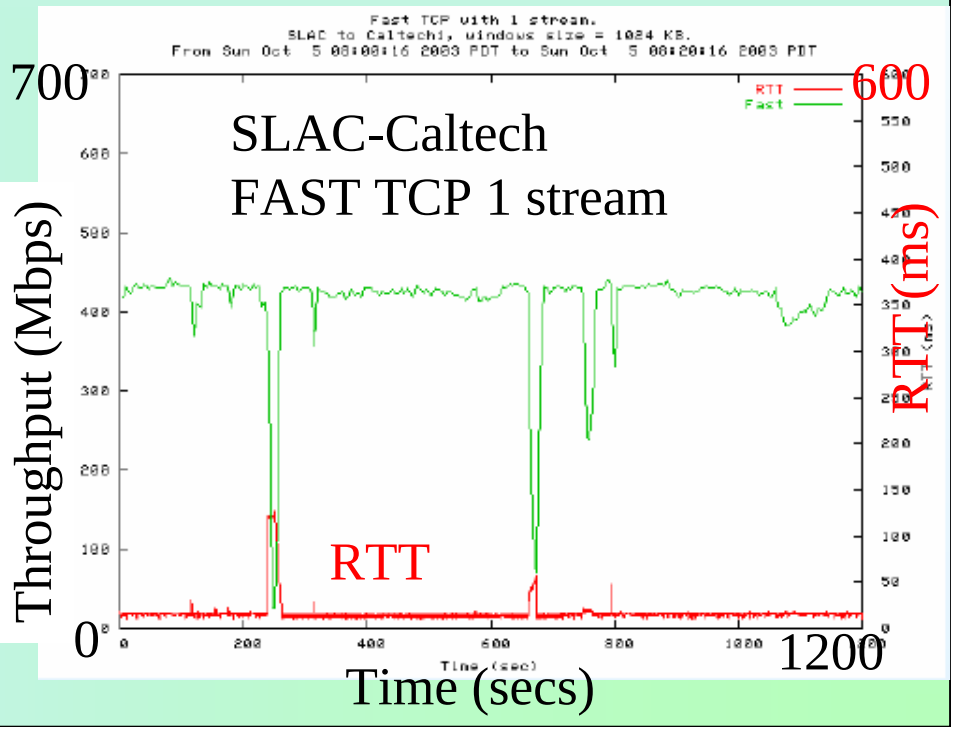
Windows

- Set large maximum windows (typically 32MB) on all hosts
- Used 3 different windows with iperf:
  - Small window size, factor 2-4 below optimal
  - Roughly optimal window size (~BDP)
  - Oversized window

RTT

- Only P-TCP appears to dramatically affect the RTT
  - E.g. increases by RTT by 200ms (factor 20 for short distances)





txqueuelen

- Regulates the size of the queue between the IP layer and the Ethernet layer
- May increase the throughput if we find optimal values
- But may increase duplicate ACKs (Y. T Li)

Txqueuelen vs TCP for UFl 4MB window	Reno 16	S-TCP	Fast	HS	Bic	H TCP	HS LP	avg
tqueuelen=100	428	301	340	431	387	348	383	374
tqueuelen=2000	434	437	400	224	396	310	380	368.71
tqueuelen=10000	429	281	385	243	407	337	386	352.57
Avg	430.33	339.67	375	299.33	396.67	331.67	383	

- All stacks except S-TCP use txqueuelen=100 as default
- S-TCP uses txqueuelen=2000 by default
- Tests showed these were reasonable choices

Throughput (Mbps)

Windows too small (worse for longer distance)

Throughput SLAC to Remote	Reno 16	Sc	Bic	Fast	HS LP	H	HS	Reno 1	Avg
Caltech 256 KB	395	226	238	233	236	233	225	239	253
UFl 1 MB	451	110	133	136	141	140	136	129	172
Caltech 512 KB	413	377	372	408	374	339	307	362	369
UFl 4 MB	428	437	387	340	383	348	431	294	381
Caltech 1 MB	434	429	382	413	381	374	284	374	384
UFl 8 MB	442	383	404	348	357	351	387	278	369
Average	427	327	319	313	312	298	295	279	321
Rank	1	2	2	2	2	4	4	4	

Poor performance
Reasonable performance
Best performance

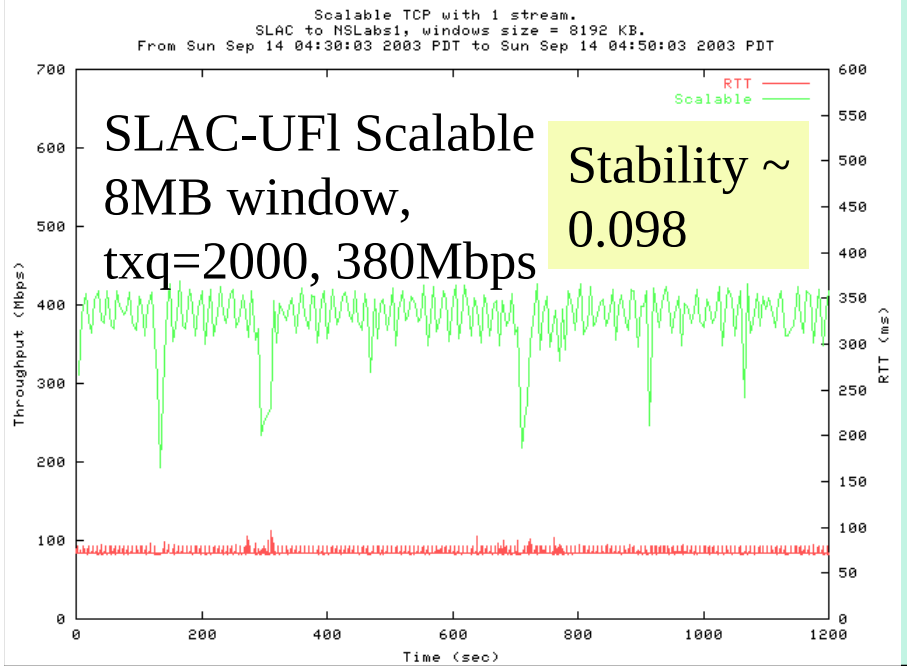
Reno with 1 stream has problems on Medium distance link (70ms)

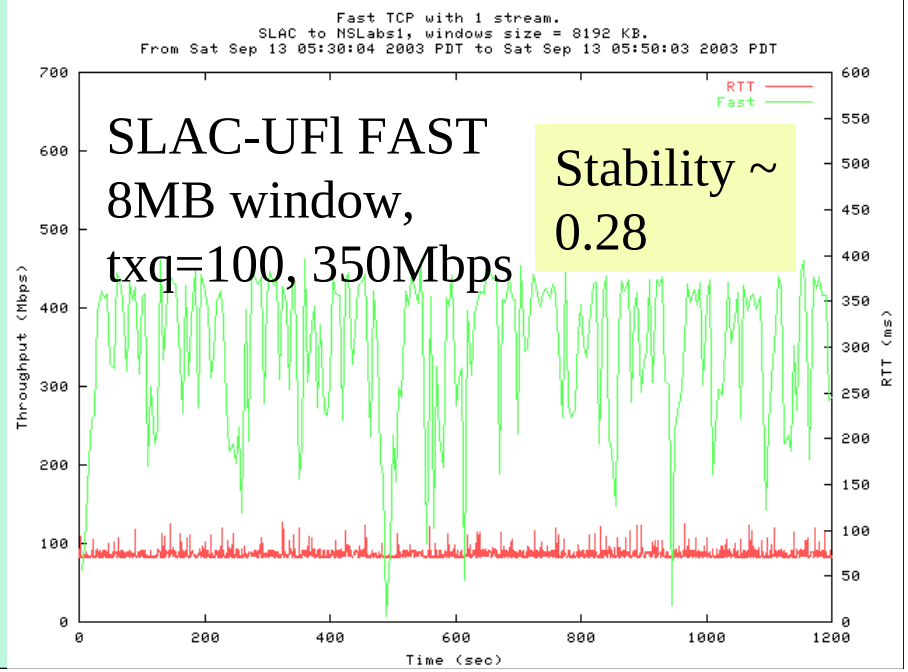
Stability

- Definition: standard deviation normalized by the average throughput

Stability for optimal txq vs window & stack for SLAC to Ufl	Reno TCP	Reno TCP 16	S-TCP	Fast TCP	HS-TCP	Bic-TCP	H TCP	HSTCP-LP
1 MB	0.2065	0.0713	0.0988	0.0897	0.1100	0.0955	0.0985	0.1288
4 MB	0.3754	0.1660	0.1167	0.2985	0.2115	0.1335	0.2181	0.3132
8 MB	0.4149	0.1179	0.0986	0.2772	0.2471	0.0850	0.1595	0.3333

  - At short RTT (10ms) stability is usually good (<=12%)
  - At medium RTT (70ms) P-TCP, Scalable & Bic-TCP and appear more stable than the other protocols





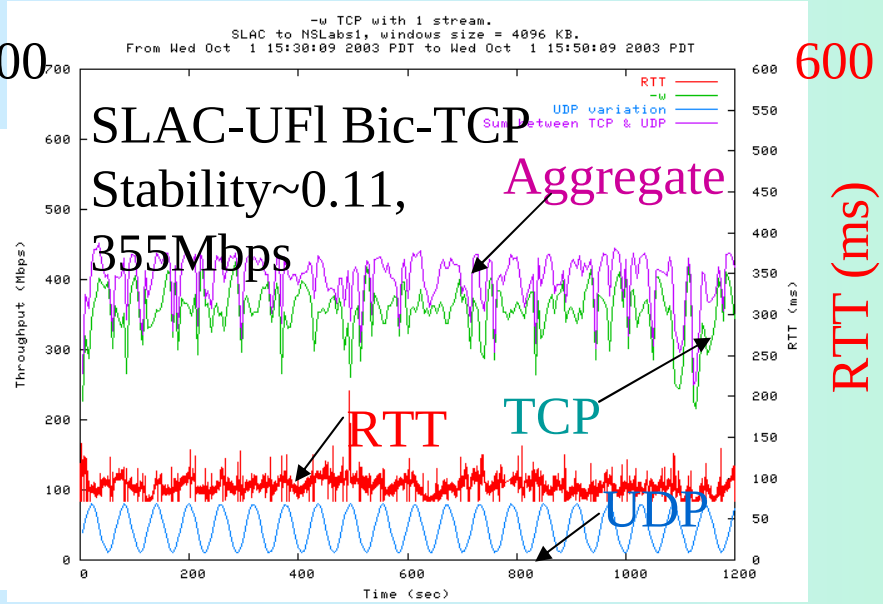
Sinusoidal UDP

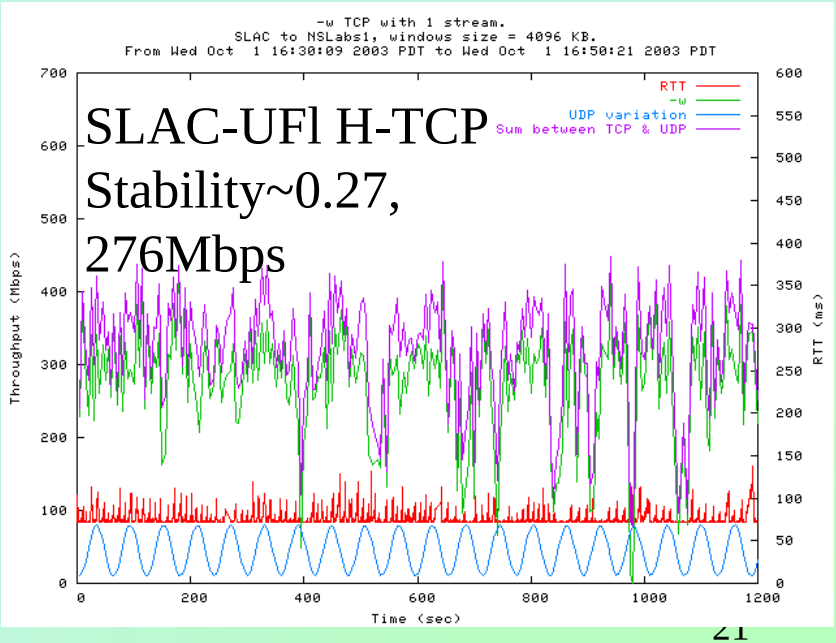
- UDP does not back off in face of congestion, it has a “stiff” behavior
- We modified iperf to allow it to create UDP traffic with a sinusoidal time behavior, following an idea from Tom Hacker
  - See how TCP responds to varying cross-traffic
- Used 2 periods of 30 and 60 seconds and amplitude varying from 20 to 80 Mbps
- Sent from 2nd sending host to 2nd receiving host while sending TCP from 1st sending host to 1st receiving host
- As long as the window size was large enough all protocols converged quickly and maintain a roughly constant aggregate throughput
- Especially for P-TCP & Bic-TCP

TCP Convergence against UDP

- Stability better at short distances
- P-TCP & Bic more stable

Stability to UFl vs window & UDP freq.	Reno 16	Scal	Fast	HS	Bic	H	HS LP
UDP 60s + 1 MB	0.13	0.13	0.09	0.10	0.10	0.11	0.17
UDP 60s + 4 MB	0.12	0.26	0.35	0.18	0.11	0.27	0.25
UDP 60s + 8 MB	0.13	0.14	0.36	0.20	0.14	0.14	0.23
UDP 30s + 1 MB	0.12	0.11	0.07	0.11	0.09	0.21	0.17
UDP 30s + 4 MB	0.16	0.38	0.29	0.21	0.12	0.27	0.30
UDP 30s + 8 MB	0.13	0.11	0.26	0.25	0.11	0.19	0.42



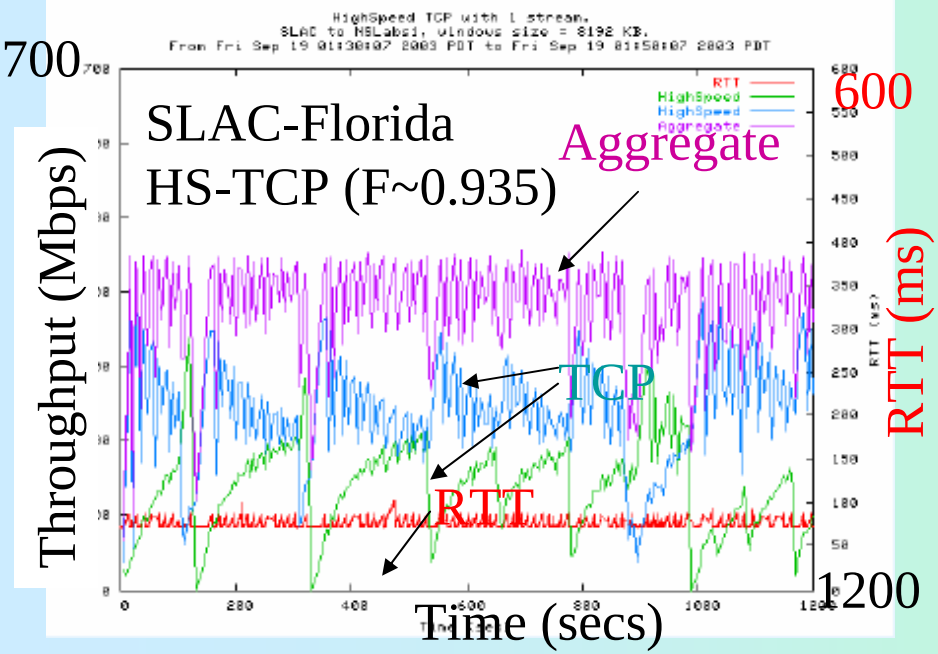


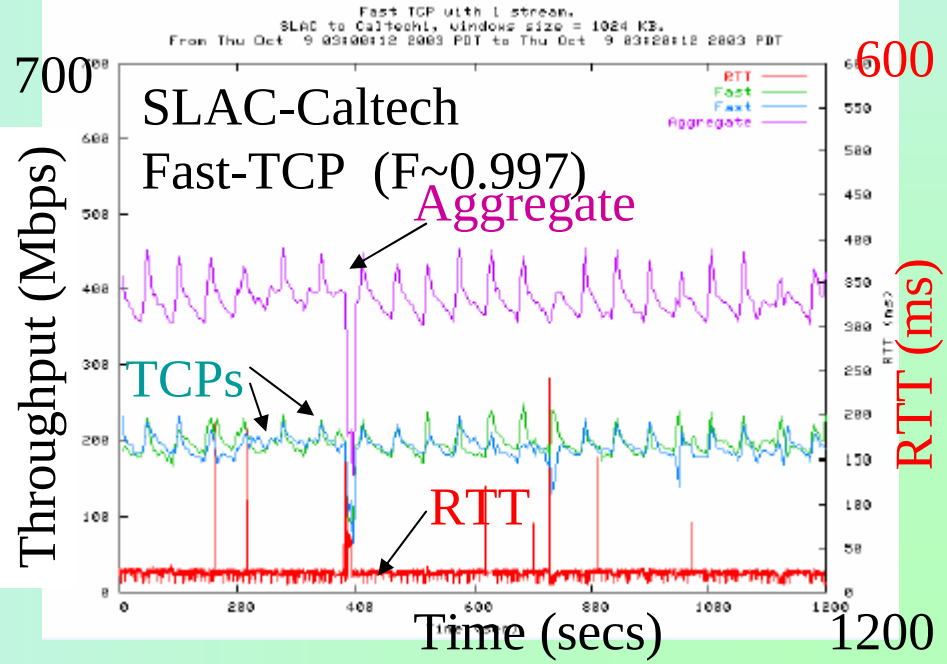
Cross TCP Traffic

- Important to understand how fair a protocol is
- For one protocol competing against the same protocol (**intra-protocol**) we define the fairness for a single bottleneck as:

$$F = \frac{(\sum_{i=1}^{n} \overline{x}_i)^2}{n \sum_{i=1}^{n} \overline{x}_i^2}$$

- All protocols have good intra-protocol Fairness (F>0.98)
- Except HS-TCP (F<0.94) when the window size > optimal





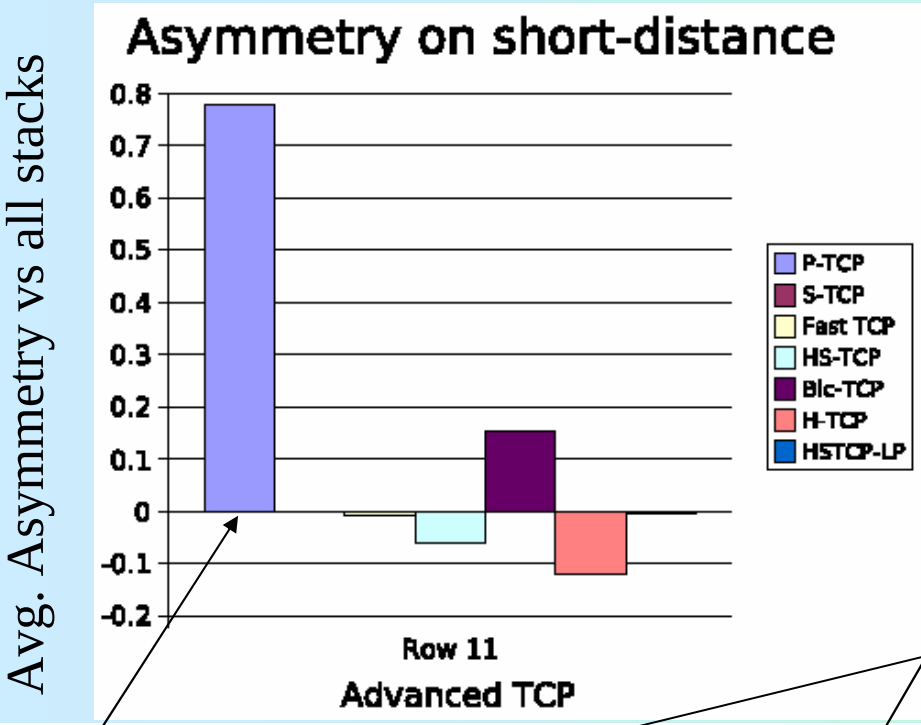
Fairness (F)

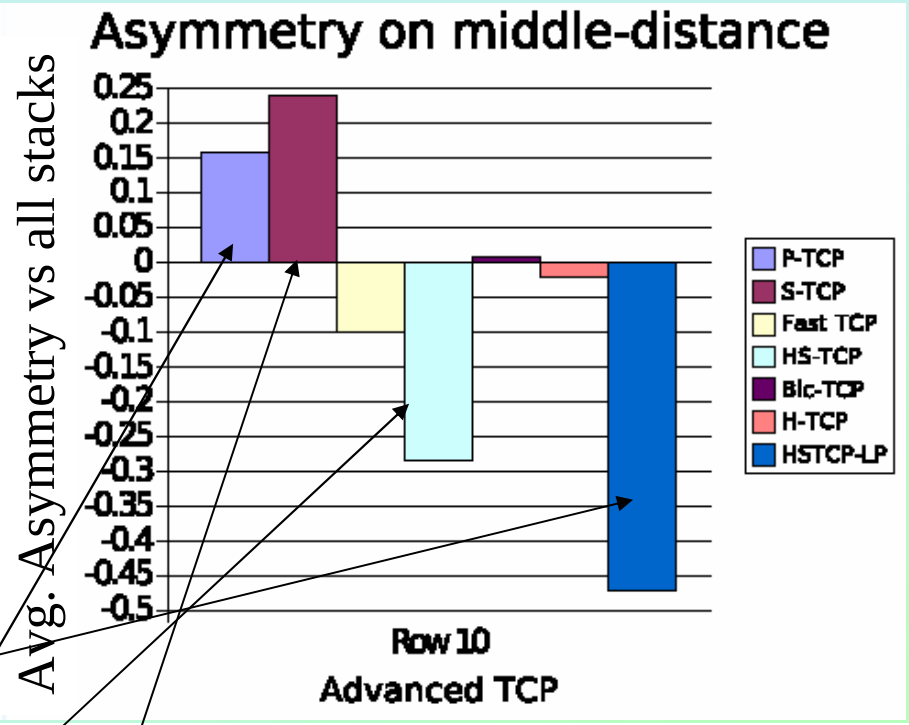
Avg Fairness from SLAC to UFl. Cross-traffic=> Source	Reno TCP 16	S-TCP	Fast TCP	HS-TCP	Bic-TCP	H TCP	HSTCP-LP	Avg
P-TCP	1.00	0.92	0.89	0.90	0.95	0.94	0.69	0.90
S-TCP	0.92	1.00	0.87	0.90	0.91	0.92	0.78	0.90
Fast TCP	0.89	0.87	1.00	0.92	0.93	0.99	0.78	0.91
HS-TCP	0.90	0.90	0.92	0.97	0.95	0.94	0.95	0.93
Bic-TCP	0.95	0.91	0.93	0.95	1.00	0.99	0.93	0.95
H-TCP	0.94	0.92	0.99	0.94	0.99	1.00	0.95	0.96
HSTCP-LP	0.69	0.78	0.78	0.95	0.93	0.95	1.00	0.87
Average	0.90	0.90	0.91	0.93	0.95	0.96	0.87	0.92

- Most have good intra-protocol fairness (diagonal elements), except HS-TCP
- Inter protocol Bic & H appear more fair against others
- Worst fairness are HSTCP-LP, P-TCP, S-TCP, Fast, HSTCP-LP
- But cannot tell who is aggressive and who is timid

Inter protocol Fairness

- For inter-protocol fairness we introduce the asymmetry between the two throughputs: $A = \frac{\overline{x}_1 - \overline{x}_2}{\overline{x}_1 + \overline{x}_2}$
  - Where x_1 and x_2 are the throughput averages of TCP stack 1 competing with TCP stack 2





Reno 16 v. aggressive at short RTT, Reno & Scalable aggressive at medium distance

HSTCP-LP very timid on medium RTT, HS-TCP also timid

Inter Fairness – UFI (A)

$A=(x_m-x_c)/(x_m+x_c)$

Aggressive

Fair

Timid

Cross traffic=> Major source	Reno 16	Sca	Fast	HS	Bic	H	HS LP	Avg
Reno 16 + 4 MB	0.00	0.38	0.26	0.45	0.05	0.12	0.66	0.27
Reno 16 + 8 MB	0.00	-0.16	0.25	0.35	0.10	0.09	0.61	0.18
S-TCP + 4 MB	-0.38	0.00	0.33	0.07	0.19	0.12	0.65	0.14
S-TCP + 8 MB	0.16	0.00	0.63	0.65	0.56	0.54	0.70	0.46
Fast TCP + 4 MB	-0.26	-0.33	0.00	0.26	-0.29	0.11	0.68	0.03
Fast TCP + 8 MB	-0.25	-0.63	0.00	0.48	-0.38	0.11	0.68	0.00
HS-TCP + 4 MB	-0.45	-0.07	-0.26	0.00	-0.25	-0.17	0.37	-0.12
HS-TCP + 8 MB	-0.35	-0.65	-0.48	0.00	-0.33	-0.41	0.13	-0.30
Bic-TCP + 4 MB	-0.05	-0.19	0.29	0.25	0.00	-0.10	0.29	0.07
Bic-TCP + 8 MB	-0.10	-0.56	0.38	0.33	0.00	-0.15	0.31	0.03
H TCP + 4 MB	-0.12	-0.12	-0.11	0.17	0.10	0.00	0.19	0.01
H TCP + 8 MB	-0.09	-0.54	-0.11	0.41	0.15	0.00	0.37	0.03
Average	-0.16	-0.24	0.10	0.28	-0.01	0.02	0.47	0.07

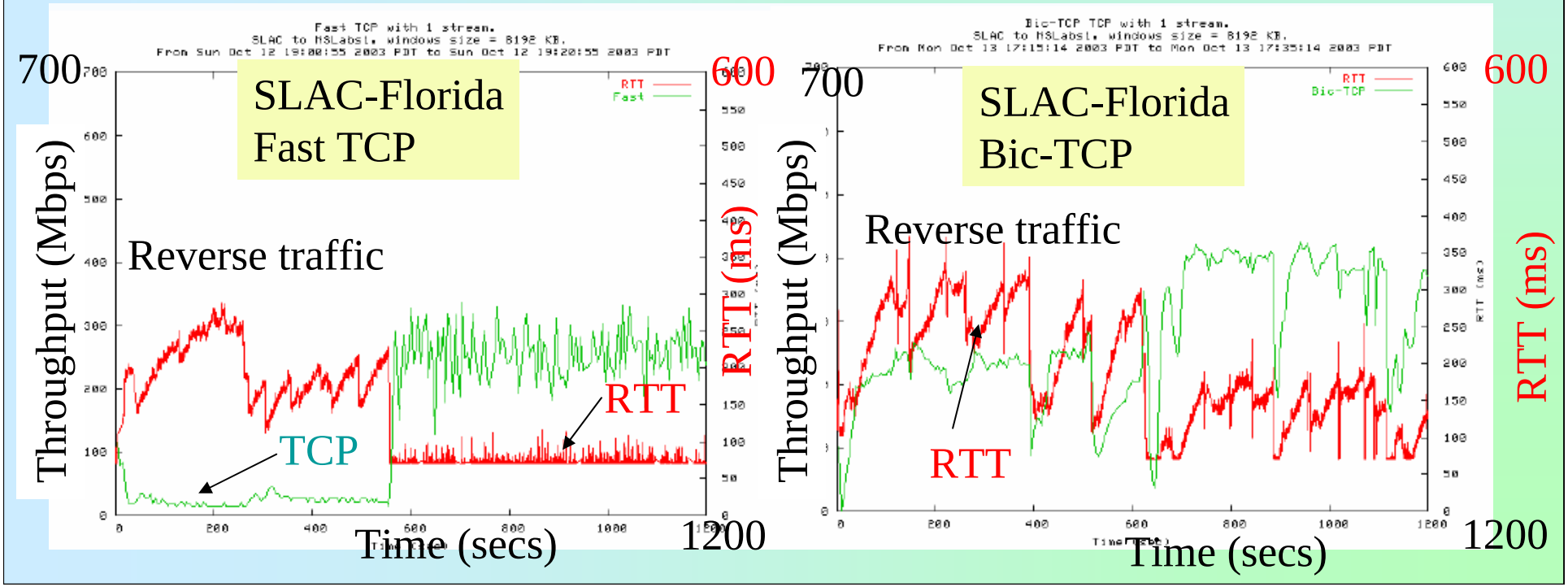
Diagonal = 0 by definition
Symmetric off diagonal
Down how does X traffic behave

Scalable & Reno 16 streams are aggressive
HS LP is very timid

Fast more aggressive than HS & H
HS is timid

Reverse Traffic

- Cause queuing on reverse path by using P-TCP 16 streams
- ACKs are lost or come back in bursts (compressed ACKs)
- Fast TCP throughput is 4 to 8 times less than the other TCPs.



Future work

- Finish measurements to Manchester/CERN
- More analysis
- Work with Caltech to correlate with simulation
- Compare with other people's measurements
- Test Westwood+
- Tests with different RTTs on the same link
- Try on 10Gbps links
- More tests with multiple streams
- Look at performance of rate based protocols

Preliminary Conclusions

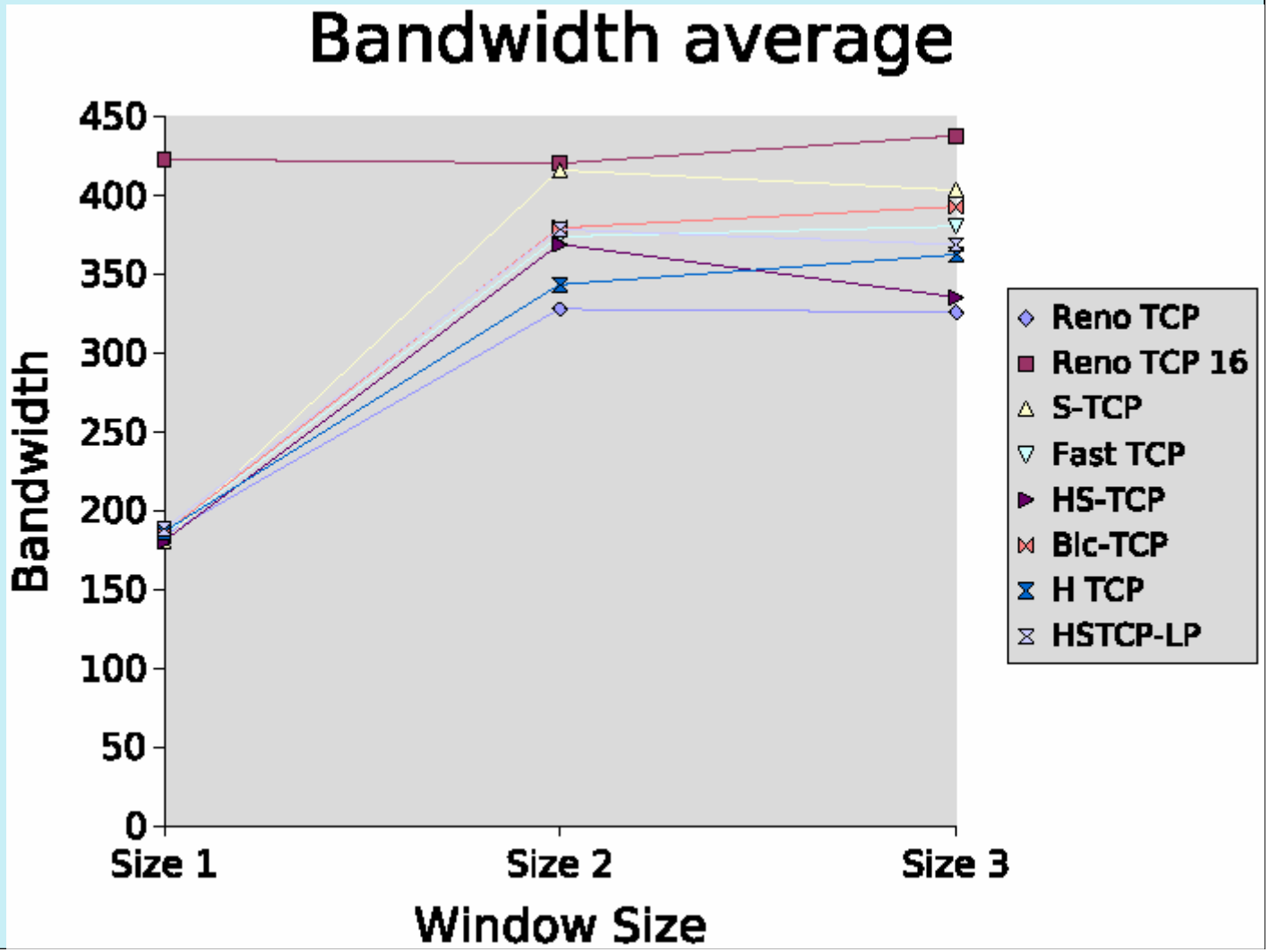
- Advanced stacks behave like TCP-Reno single stream on short distances for up to Gbits/s paths, especially if window size limited
- TCP Reno single stream has low performance and is unstable on long distances
- P-TCP is very aggressive and impacts the RTT badly
- HSTCP-LP is too gentle, **this can be important for providing scavenger service without router modifications**. By design it backs off quickly, otherwise performs well
- Fast TCP is very handicapped by reverse traffic
- S-TCP is very aggressive on long distances
- HS-TCP is very gentle, like H-TCP has lower throughput than other protocols
- Bic-TCP performs very well in almost all cases

More Information

- TCP Stacks Evaluation:
  - www-iepm.slac.stanford.edu/bw/tcp-eval/

Throughput

- With optimal window all stacks within ~20% of one another, except Reno 1 stream on medium and long distances
- P-TCP & S-TCP get best throughput



Inter Fair Caltech

$$A = \frac{(x_1 - x_2)}{(x_1 + x_2)}$$

Aggressive

Fair

Timid

Cross traffic=> Major source	Scal	Fast	HS	Bic	H	HS LP	Avg
Reno 16 + 512 KB	0.81	0.91	0.88	0.76	0.83	0.88	0.84
Reno 16 + 1 MB	0.69	0.89	0.88	0.57	0.34	0.88	0.71
S-TCP + 512 KB	0.00	0.00	-0.07	-0.12	-0.06	-0.06	-0.05
S-TCP + 1 MB	0.00	0.38	0.00	-0.13	0.10	-0.06	0.05
Fast TCP + 512 KB	0.00	0.00	0.09	-0.16	-0.02	-0.05	-0.02
Fast TCP + 1 MB	-0.38	0.00	0.40	-0.42	0.29	0.15	0.01
HS-TCP + 512 KB	0.07	-0.09	0.00	-0.04	-0.02	-0.03	-0.02
HS-TCP + 1 MB	0.00	-0.40	0.00	-0.32	0.19	-0.07	-0.10
Bic-TCP + 512 KB	0.12	0.16	0.04	0.00	0.28	-0.02	0.10
Bic-TCP + 1 MB	0.13	0.42	0.32	0.00	0.32	0.10	0.21
H TCP + 512 KB	0.06	0.02	0.02	-0.28	0.00	-0.04	-0.04
H TCP + 1 MB	-0.10	-0.29	-0.19	-0.32	0.00	-0.29	-0.20
HSTCP-LP + 512 KB	0.06	0.05	-0.03	0.02		0.00	0.02
HSTCP-LP + 1 MB	0.06	-0.15	0.07	-0.10		0.00	-0.02
Avg	0.11	0.14	0.17	-0.04	0.19	0.10	0.11

Everyone timid in presence of Reno 16 streams (even Scalable)

Less inter protocol differences than for UFL (10ms vs 70ms)