Testing TCP in the WAN

Yee-Ting Li
Stanford Linear Accelerator Center

@

Microsoft TCP Workshop - February 2007

Background

- SLAC focuses on high energy physics and light sciences
- Latest large scale projects are LHC (particle physics) and LCLS (photon sciences)
- Vast amounts of data collected about particle collisions, high speed images etc.
- Projects are highly collaborative with scientists all over the world

Why WAN?

- Controlled simulation & emulation critical for understanding
- BUT ALSO need to verify; results may different than expected
- Testing of TCP implementations over the WAN
  - Involves entire TCP stack - NOT just the TCP algorithm
  - algorithms may be coded incorrectly
- Interaction with existing production traffic; more realistic cross traffic patterns

Outline

- Initial WAN tests performed in 2005
  - by R. Les Cottrell, Saad Ansari, Parakram Khandpur, Ruchi Gupta, Richard Hughes-Jones, Michael Chen, Larry McIntosh, Frank Leers
  - Presented at PFLDnet 2005
- More recent WAN tests with Microsoft in June 2006
  - by Yee-Ting Li & Microsoft
  - Focus on Fairness issues with many flows and impact against bulk Reno flows
  - Testing the evolution of the CTCP algorithm

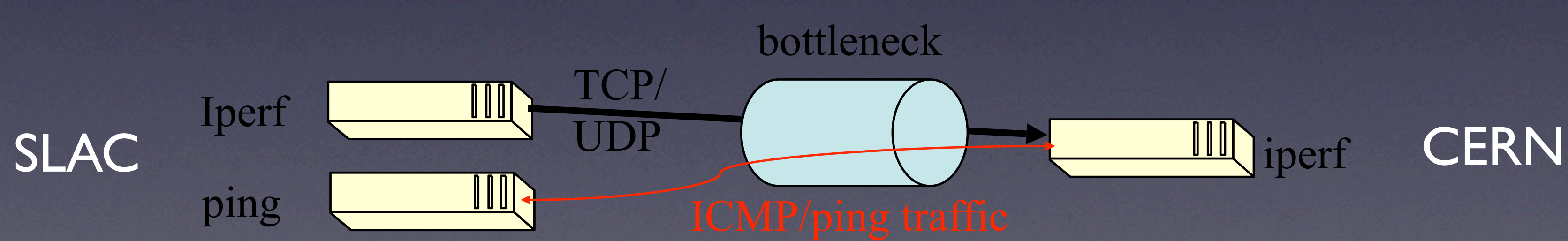
PFLDnet2005

Goals

- Evaluate various techniques for achieving high bulk-throughput on fast long-distance real production WAN links
  - Compare & contrast: throughput, fairness, stability etc.
- Recommend "optimum" techniques for data intensive science transfers using bulk transfer tools
- Validate simulator & emulator findings & provide feedback

Test Setup

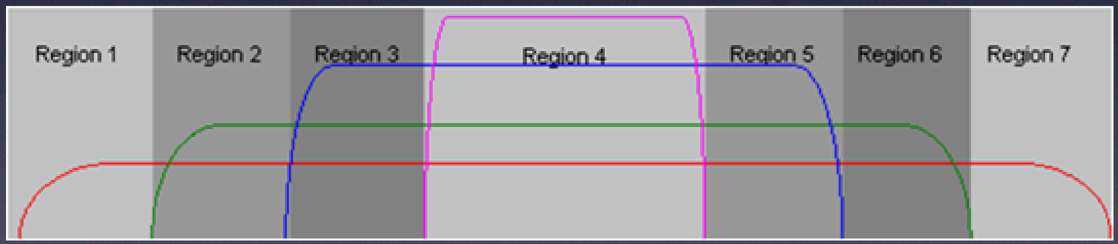
- From SLAC to CERN (~180ms)
- Production network: through ESnet/GEANT
- Used iperf/TCP generate traffic
- Single host to single host
- Also measured ping latencies during tests



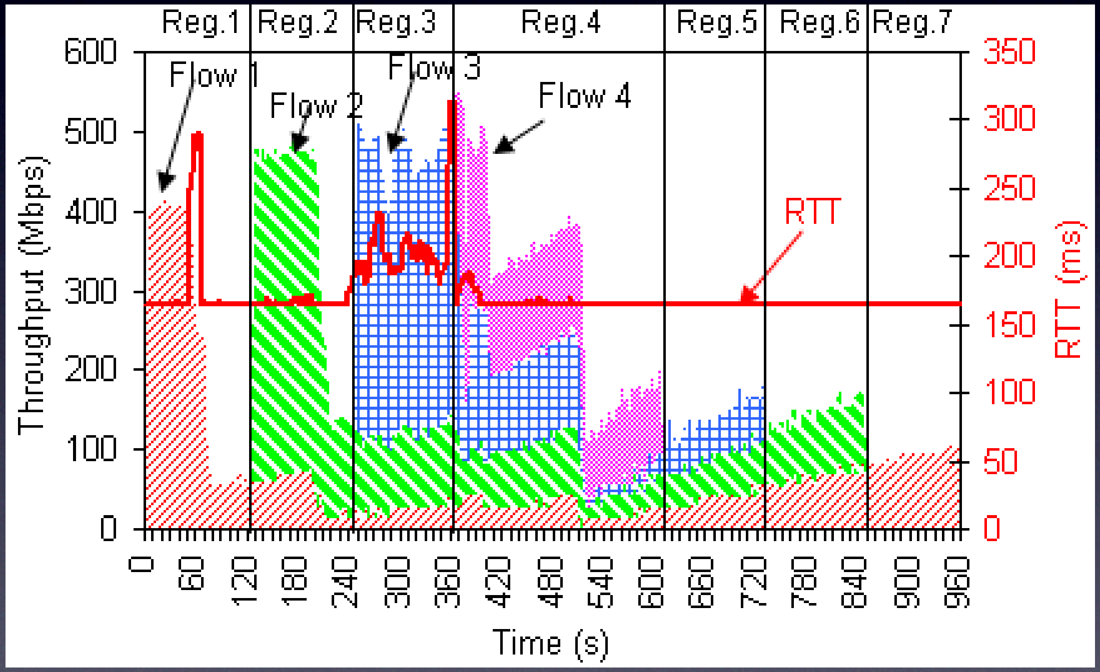
Tests were also conducted to Caltech (10ms), Univ. Florida (80ms) but not shown

Test Setup

- Run 4 TCP flows
- Sufficient time between flows to allow algorithms to 'stabilise'

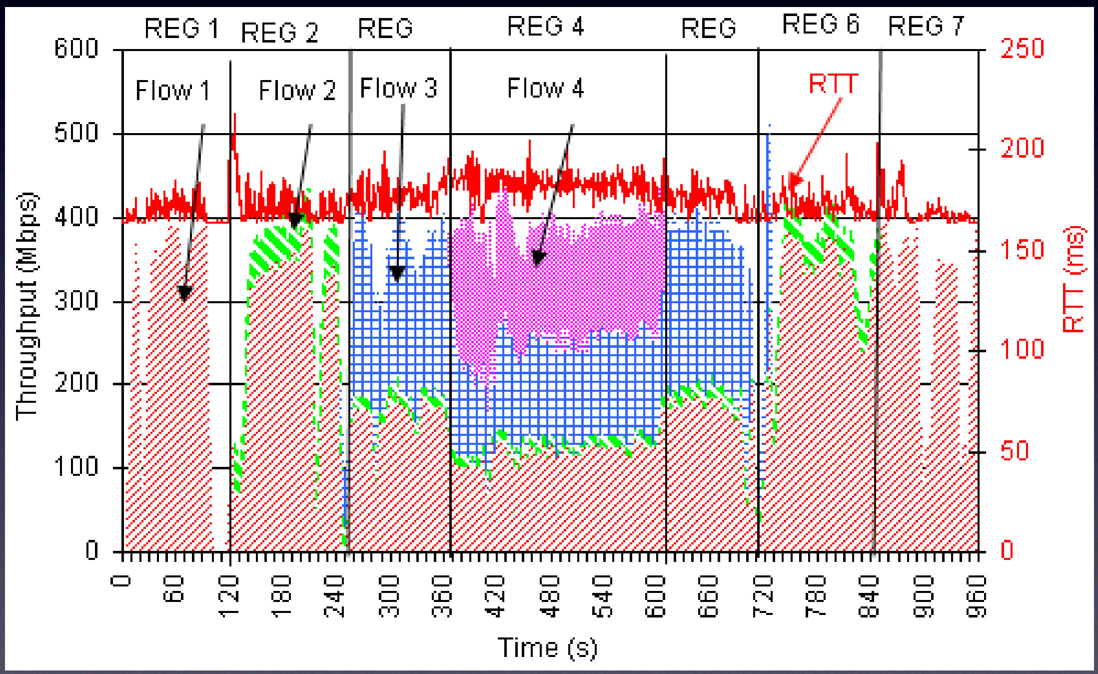


Reno



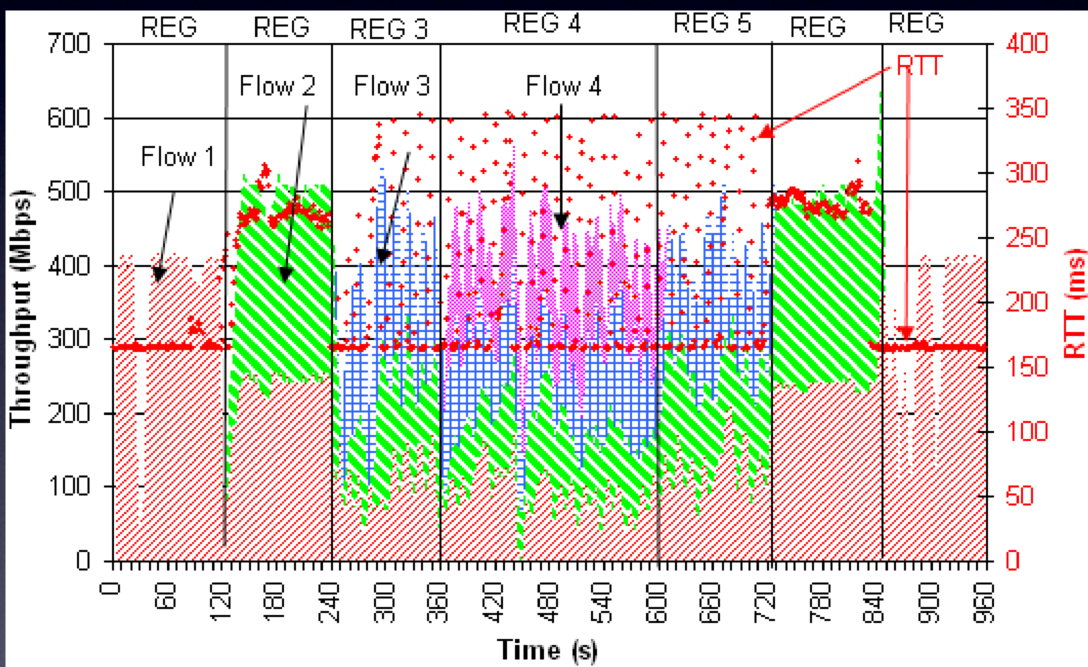
- Initial slow start allows high throughput
- Congestion has dramatic effect
  - low throughput
  - slow to recover
- Fairness between flows depends on when you measure it
- Growth rates between flows are similar

FAST



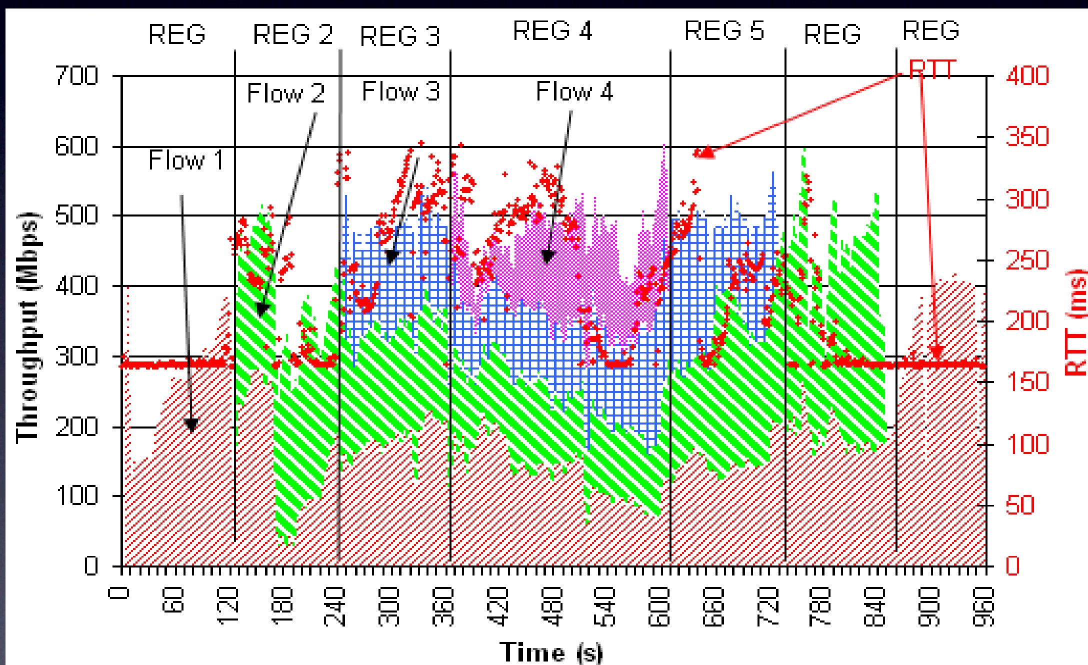
- 2nd flow never gets fair share (green)
- Big drops in throughput (stack issues?)

HTCP



- Gets more throughput with >1 flow
- Fair sharing of throughput
- Very variable throughput and RTT with >2 flows
    - bursty cross traffic?
    - TCP stack?
    - Host issues?

BicTCP



- Needs more than 1 flow for best throughput
- Not very stable throughput during test

Summary

Protocol	Avg thru (Mbps)	***S*** (σ/μ)	min (***F***)	σ (RTT)	MHz/ Mbps
HSTCP	255±187	0.73	0.79	25	0.9
Fast	335±110	0.33	0.58	9	0.66
Scalable	423±115	0.27	0.83	22	0.64
HTCP	402±113	0.28	0.99	57	0.65
BIC	412±117	0.28	0.98	55	0.71
Reno	248±163	0.66	0.6	22	0.63

± over entire single test

calculated with two flows

- Scalable has high throughput, but poor fairness (trace not shown)
- BicTCP and HTCP are about the same in terms of the metrics (even though results look different)
- FAST has low variance on the RTT and good stability, but low average throughput

Issues

- Using the same machine for all flows may have un-desirable effects; CPU contention, host based queuing etc.
- Fairness not considered for many flows; only for two flows
- Statistically not very thorough (tests only performed once)
- No/difficult to validate that the algorithm is functioning correctly (cwnd etc) compared to what we see with throughput (especially at many seconds resolution)
- Metrics do not appear to capture the differences in the throughput profiles

Summary

- Need a more visual way of determining performance
  - Many flows fairness?
  - Relation between fairness and convergence?
  - Stability:
    - Reno not stable because of large changes in cwnd
    - BicTCP and HTCP show similar values, but throughput profiles are very different

CTCP Tests

Tests with Microsoft

- Expand on PFLDnet2005 results
  - Fairness: analyse area where all flows are competing - define for multiple flows
  - Friendliness/Impact: Look at how a single TCP flow interact against Reno
- Focus on CTCP
- Again, start flows at different start times: important for RTT differences of different flows (see FAST).

Aggregate Throughput

Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	521 ± 15	114 ± 5	62 ± 19
1 Flow	CTCP	619 ± 21	252 ± 13	146 ± 37
1 Flow	HSTCP	605 ± 21	125 ± 7	135 ± 20
2 Flows	StandardTCP	574 ± 14	203 ± 19	167 ± 35
2 Flows	CTCP	640 ± 11	347 ± 3	190 ± 22
2 Flows	HSTCP	504 ± 136	161 ± 7	216 ± 42
4 Flows	StandardTCP	645 ± 26	280 ± 3	223 ± 42
4 Flows	CTCP	653 ± 33	379 ± 1	258 ± 42
4 Flows	HSTCP	668 ± 32	263 ± 51	253 ± 160
8 Flows	StandardTCP	732 ± 74	331 ± 1	431 ± 75
8 Flows	CTCP	672 ± 23	389 ± 2	435 ± 53
8 Flows	HSTCP	636 ± 37	392 ± 3	497 ± 29
16 Reverse TCP	StandardTCP	96 ± 9	33 ± 1	128 ± 14
16 Reverse TCP	CTCP	108 ± 7	45 ± 2	121 ± 11
16 Reverse TCP	HSTCP	87 ± 12	114 ± 4	133 ± 10

± taken from multiple repeated measurements

- All throughputs tend towards same value with more flows
- With 1-2 flows, CTCP achieves 2x throughput of StandardTCP
- Strange behaviour with presence of reverse traffic: not a host issue - still unknown

Fairness Metrics

- σ_f - overall fairness: define the magnitude of the differences between the throughputs of each flow. (standard deviation of average throughputs)

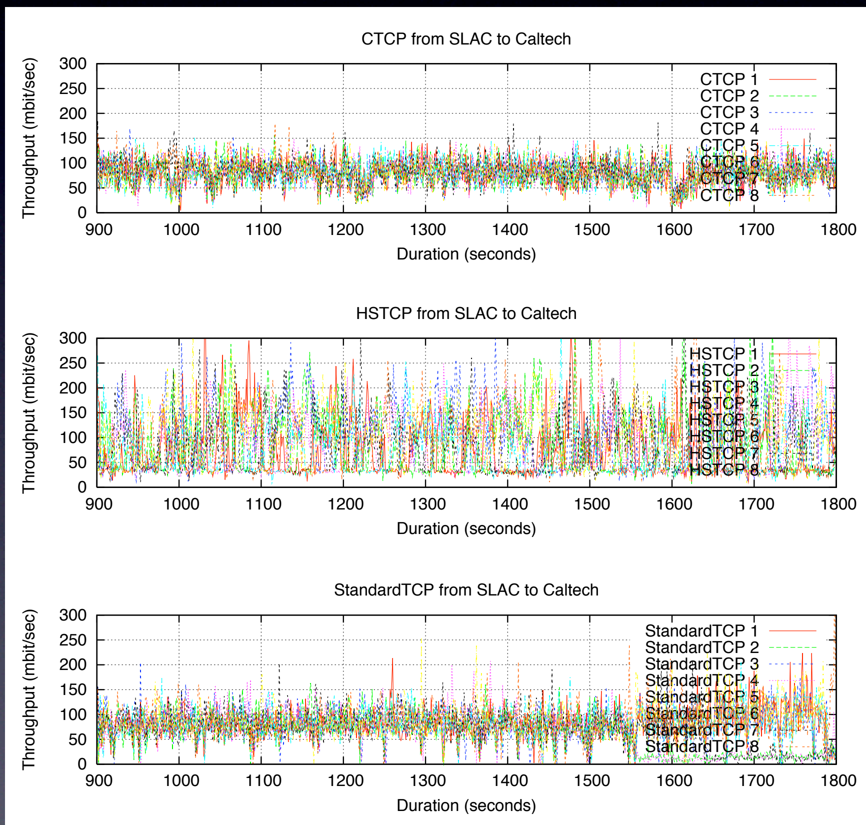
$$\sigma_f := \frac{1}{\bar{x}}\sqrt{\frac{\sum_{i=1}^{n}(\bar{x}_i - \bar{x})^2}{n}}$$

- ξ_f - instantaneous fairness: define the standard deviation of throughput of each flow through time. (standard deviation of standard deviations)

$$\xi_f := \frac{1}{\bar{x}}\sqrt{\frac{\sum_{t=1}^{T}(\bar{\sigma}_t - \bar{\sigma})^2}{T}}$$

SLAC to Caltech

8ms Baseline RTT

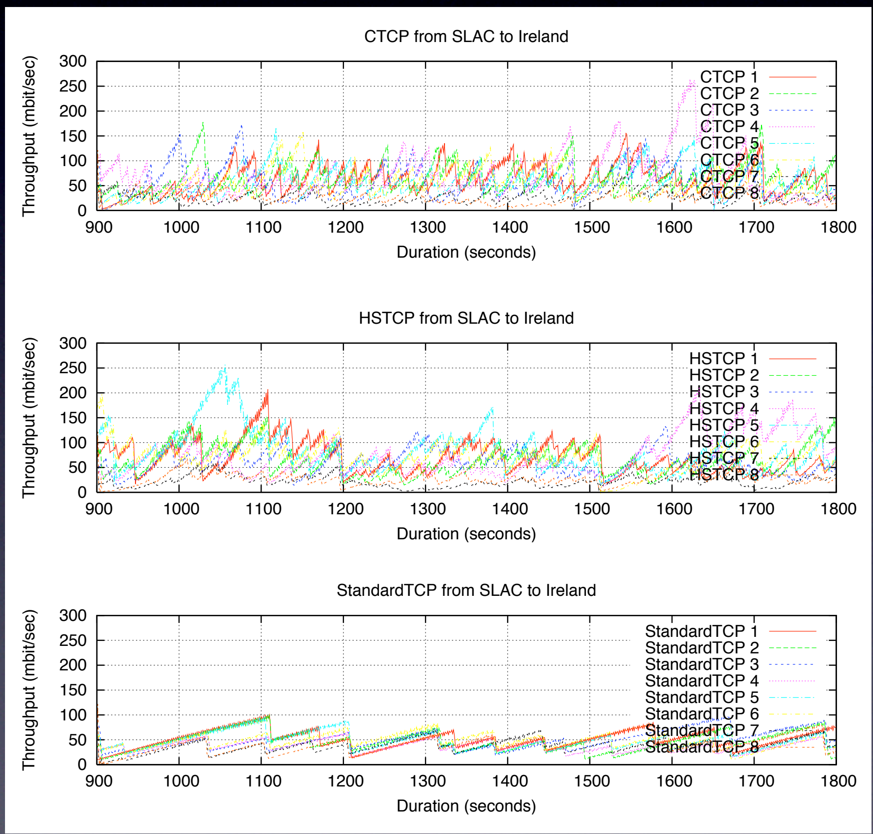


	σ_f	ξ_f
CTCP	0.013±0.006	0.236±0.018
HSTCP	0.090±0.013	0.677±0.005
Standard	0.114±0.048	0.326±0.037

- All stacks give approximately same aggregate throughput
- HSTCP highly variable - higher ξ_f, however σ_f not too different

SLAC to Ireland

150ms Baseline RTT



	σ_f	ξ_f
CTCP	0.345±0.016	0.570±0.029
HSTCP	0.386±0.029	0.597±0.058
Standard	0.169±0.048	0.317±0.032

- ξ_f and σ_f similar for CTCP and HSTCP
- σ_f for Standard TCP almost half that of HS/CTCP

σ_f - overall fairness

Test	TCP Algorithm	Caltech (8ms)	Florida (70ms)	Ireland (150ms)
1 Flow	StandardTCP	-	-	-
1 Flow	CTCP	-	-	-
1 Flow	HSTCP	-	-	-
2 Flows	StandardTCP	0.023 ± 0.012	0.292 ± 0.148	0.118 ± 0.030
2 Flows	CTCP	0.006 ± 0.004	0.062 ± 0.020	0.305 ± 0.063
2 Flows	HSTCP	0.076 ± 0.035	0.496 ± 0.149	0.292 ± 0.075
4 Flows	StandardTCP	0.109 ± 0.082	0.024 ± 0.001	0.107 ± 0.031
4 Flows	CTCP	0.017 ± 0.008	0.032 ± 0.010	0.362 ± 0.061
4 Flows	HSTCP	0.091 ± 0.022	0.622 ± 0.311	0.410 ± 0.015
8 Flows	StandardTCP	0.114 ± 0.048	0.048 ± 0.000	0.169 ± 0.048
8 Flows	CTCP	0.013 ± 0.006	0.028 ± 0.005	0.345 ± 0.016
8 Flows	HSTCP	0.090 ± 0.013	0.245 ± 0.180	0.386 ± 0.029
16 Reverse TCP	StandardTCP	-	-	-
16 Reverse TCP	CTCP	-	-	-
16 Reverse TCP	HSTCP	-	-	-

- HSTCP has a relatively larger value of σ_f (bad) compared to both CTCP and StandardTCP
- CTCP good under short RTT paths but comparable to HSTCP under the Ireland link

ξ_f - instant. fairness

Test	TCP Algorithm	Caltech	Florida	Ireland
1 Flow	StandardTCP	-	-	-
1 Flow	CTCP	-	-	-
1 Flow	HSTCP	-	-	-
2 Flows	StandardTCP	0.214±0.007	0.554±0.202	0.289±0.040
2 Flows	CTCP	0.145±0.005	0.210±0.020	0.441±0.035
2 Flows	HSTCP	0.540±0.222	1.000±0.065	0.458±0.045
4 Flows	StandardTCP	0.294±0.067	0.236±0.003	0.256±0.031
4 Flows	CTCP	0.174±0.010	0.208±0.009	0.549±0.035
4 Flows	HSTCP	0.432±0.005	0.961±0.350	0.519±0.035
8 Flows	StandardTCP	0.326±0.037	0.285±0.022	0.317±0.032
8 Flows	CTCP	0.236±0.018	0.223±0.002	0.570±0.029
8 Flows	HSTCP	0.677±0.005	0.434±0.148	0.597±0.058
16 Reverse TCP	StandardTCP	-	-	-
16 Reverse TCP	CTCP	-	-	-
16 Reverse TCP	HSTCP	-	-	-

- Similar results to σ_f
- CTCP performs well under the low/medium latency Caltech/Florida link
- CTCP performs similarly to HSTCP under the long latency Ireland link
- StandardTCP is the most instantaneously fair for medium/long latency paths
- HSTCP performs badly over short/medium paths

Experience with CTCP

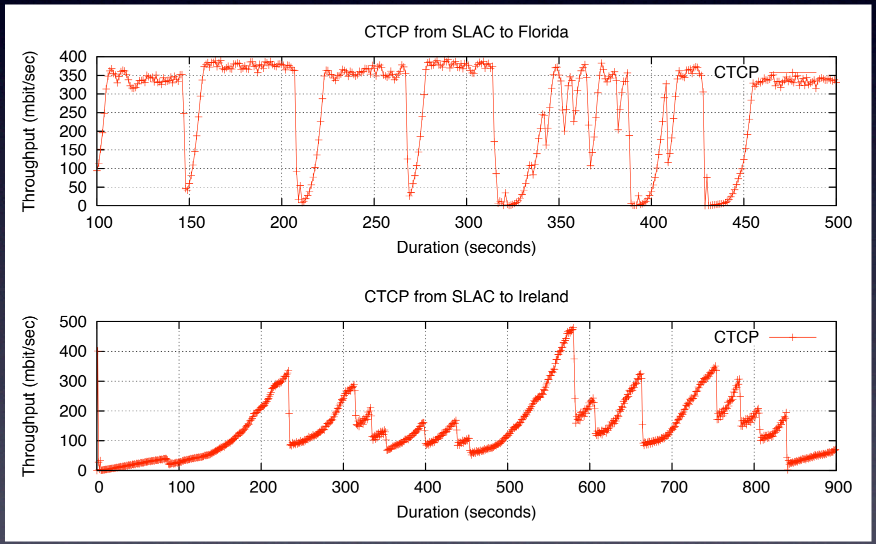
- Use performance metrics to help identify if ‘improvements’ can be made to CTCP
- Two mods introduced by Microsoft:
  - Burst control
  - γ auto-tuning

CTCP and γ

- Delay based TCP algorithms, like CTCP, need to gather sufficient delay information from network
- For equilibrium, maintain approximately γ number of packets per flow
- Different networks need different values of γ due to network sharing/buffering etc.

CTCP with Small Queues

SLAC-Florida: 375 packets buffer



SLAC-Ireland: 250 packets buffer

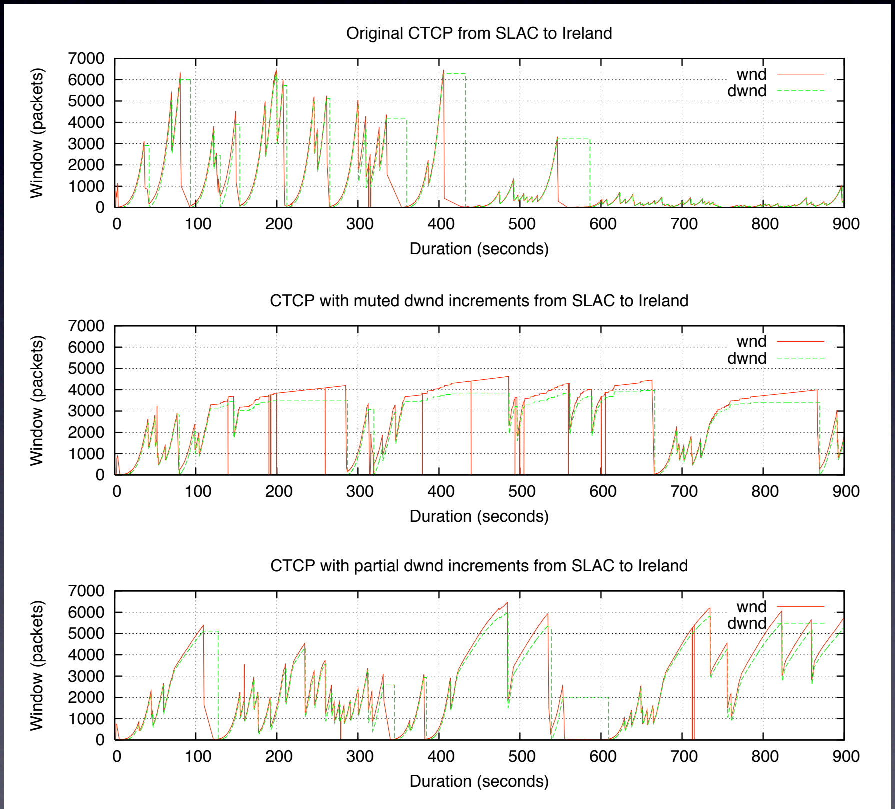
- CTCP response depends on select value of γ
- Default value of γ=30 packets
- Ireland:
  - diffWnd~3 pkts
  - diffWnd<γ
  - Insufficient for effective algorithm usage

CTCP Modifications

- 1st Mod.) Burst Control: reduce the rate of cwnd increase when diffWnd is measured to be between γ_{low} and γ
  - CTCP with “muted dwnd increments”
  - CTCP with “partial dwnd increments”
- 2nd Mod.) γ Auto-Tuning: dynamic γ value
  - CTCP with “Diffwnd Based Fairness”
  - CTCP with “Loss Window Based Fairness”

CTCP Burst Control

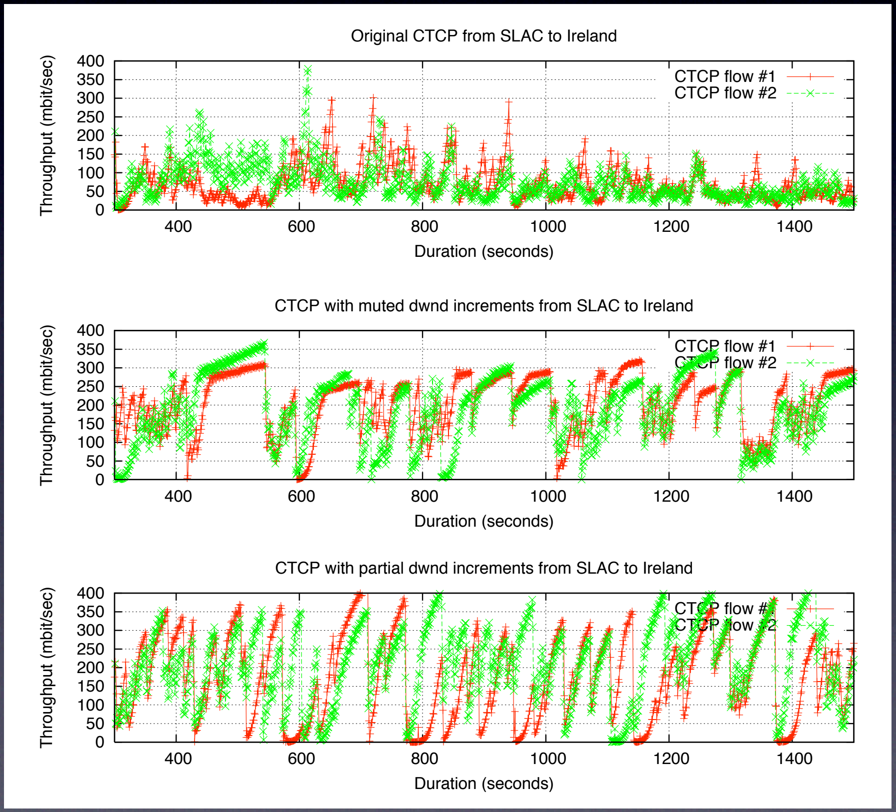
Single Flow



- To Ireland, both mods facilitate higher throughput
- muted dwnd increments shows more gradual cwnd increments
- aggressive cwnd increments of partial dwnd increments causes large losses and throughput variation

CTCP Burst Control

Two Flows



- More equal sharing of throughput with muted dwnd increments
- Burst control mods achieve ~ average throughput
    - “Partial dwnd” show larger fluctuations in throughput
    - “muted dwnd” shows slightly longer periods of unfairness

CTCP Burst Control

Flows	CTCP Algorithm	Samples	Throughput	σ_f	ξ_f
1	Original	29	116±7	-	-
1	*with muted dwnd increments*	35	222±8	-	-
1	*with partial dwnd increments*	36	177±11	-	-
2	Original	19	143±11	0.22±0.08	0.52±0.06
2	*with muted dwnd increments*	15	197±24	0.13±0.02	0.34±0.02
2	*with partial dwnd increments*	18	217±26	0.10±0.02	0.35±0.01
8	Original	10	384±15	0.35±0.05	0.87±0.04
8	*with muted dwnd increments*	9	474±35	0.19±0.02	0.52±0.03
8	*with partial dwnd increments*	8	470±23	0.27±0.03	0.61±0.02

- Throughput of both burst control mods similar
- With >1 flow, CTCP with muted dwnd increments shows better fairness characteristics

CTCP γ Auto-tuning

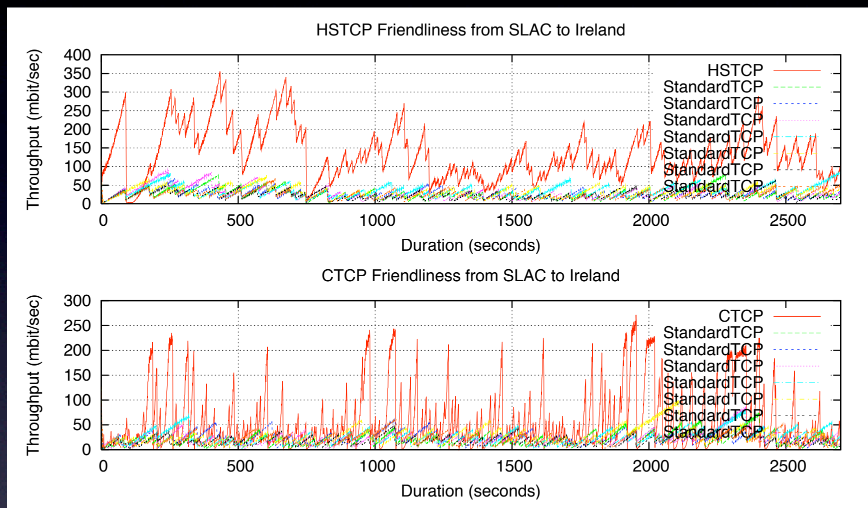
- ?

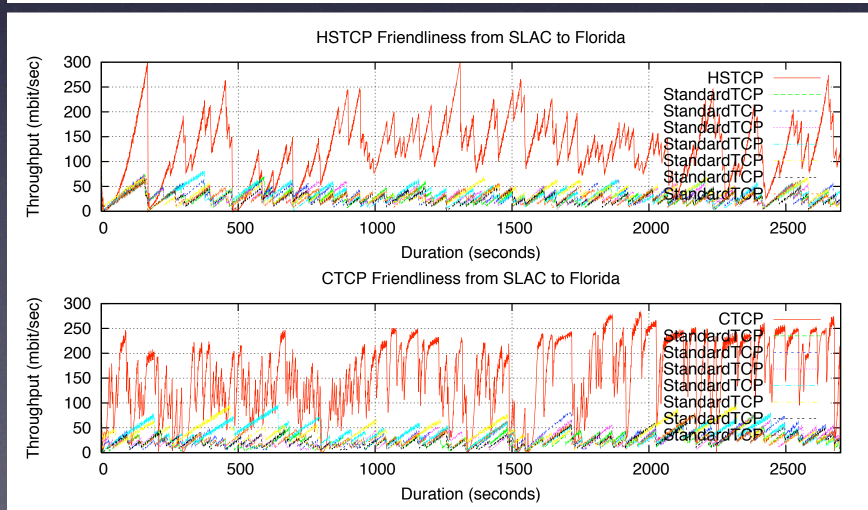
γ Auto-Tuning

Flows	CTCP Algorithm	Samples	Throughput	σ_f	ξ_f
1	*with muted dwnd increments*	5	119±19	-	-
1	*with DiffwndBasedFairness*	8	131±9	-	-
1	*with LossWindowBasedFairness*	2	103±33	-	-
2	*with muted dwnd increments*	19	157±12	0.08±0.02	0.33±0.01
2	*with DiffwndBasedFairness*	15	200±21	0.12±0.03	0.28±0.03
2	*with LossWindowBasedFairness*	18	182±17	0.15±0.02	0.30±0.02
8	*with muted dwnd increments*	10	377±17	0.17±0.02	0.51±0.01
8	*with DiffwndBasedFairness*	9	406±10	0.17±0.02	0.49±0.01
8	*with LossWindowBasedFairness*	8	318±87	0.13±0.03	0.47±0.04

- Implemented with "muted dwnd increment" burst control algorithm
- Both perform better in terms of throughput
- Fairness performs similar/better
- γ Auto-Tuning: statistically comparable fairness, but with higher throughput

Friendliness & Impact





- Interaction between one New TCP flow against 7 StandardTCP flows
- High impact: reduces mean throughput of Standard TCP flows
- Low/No impact: New TCP affects does not affect mean throughput of Standard TCP flows
- Assumes we're not at full capacity

Friendliness & Impact

	Ireland (Samples)	Florida
8 StandardTCP	310.68±21.44	340.80±4.23
Total	310.68±21.44 (6)	340.80±4.23 (9)
7 StandardTCP	182.91±16.41	290.65±2.48
1 CTCP	125.47±17.72	79.32±1.81
Total	312.31±33.36 (7)	371.83±0.65 (4)
7 StandardTCP	206.10±7.67	222.24±2.07
1 HSTCP	122.89±8.16	88.21±1.29
Total	332.29±17.33 (7)	310.97±0.88 (4)

		Throughput per StandardTCP flow		
Destination	New-TCP	Without New-TCP	With New-TCP	Change
Ireland	CTCP	38.83±2.68	26.13±2.34	-32%±7%
Ireland	HSTCP	38.83±2.68	29.44±1.16	-24%±6%
Florida	CTCP	42.60±5.29	41.52±0.35	-2%±1%
Florida	HSTCP	42.60±5.29	31.75±0.29	-25%±1%

- Original CTCP algorithm with no mods
- CTCP has higher impact on Ireland link
- CTCP has nominal effect of path to Florida
- In both cases: HSTCP has similar impact on both network paths

γ Auto-Tuning Impact
on Ireland link

	Ireland (Samples)
8 StandardTCP	186.903±11.013
Total	186.903±11.013 (7)
7 StandardTCP	152.688±16.422
1 CTCP *with muted dwnd increments*	82.758±8.574
Total	240.311±24.473 (9)
7 StandardTCP	167.908±15.638
1 CTCP *with DiffwndBasedFairness*	84.326±5.628
Total	252.361±20.471 (10)
7 StandardTCP	144.714±6.812
1 CTCP *with LossWindowBasedFairness*	75.610±4.637
Total	227.440±11.130 (10)

- Both versions give better throughput
- "diffwnd based" has no noticeable impact upon StandardTCP
- "loss based" actually have higher impact than the muted dwnd increments

Destination	CTCP Algorithm	Throughput per StandardTCP flow Without New-TCP	Throughput per StandardTCP flow With New-TCP	Change
Ireland	*muted dwnd increments*	23.522±1.378	21.813±2.346	-7%±11%
Ireland	*DiffwndBasedFairness*	23.522±1.378	23.987±2.234	+2%±2%
Ireland	*LossWindowBasedFairness*	23.522±1.378	20.673±0.973	-12%±1%

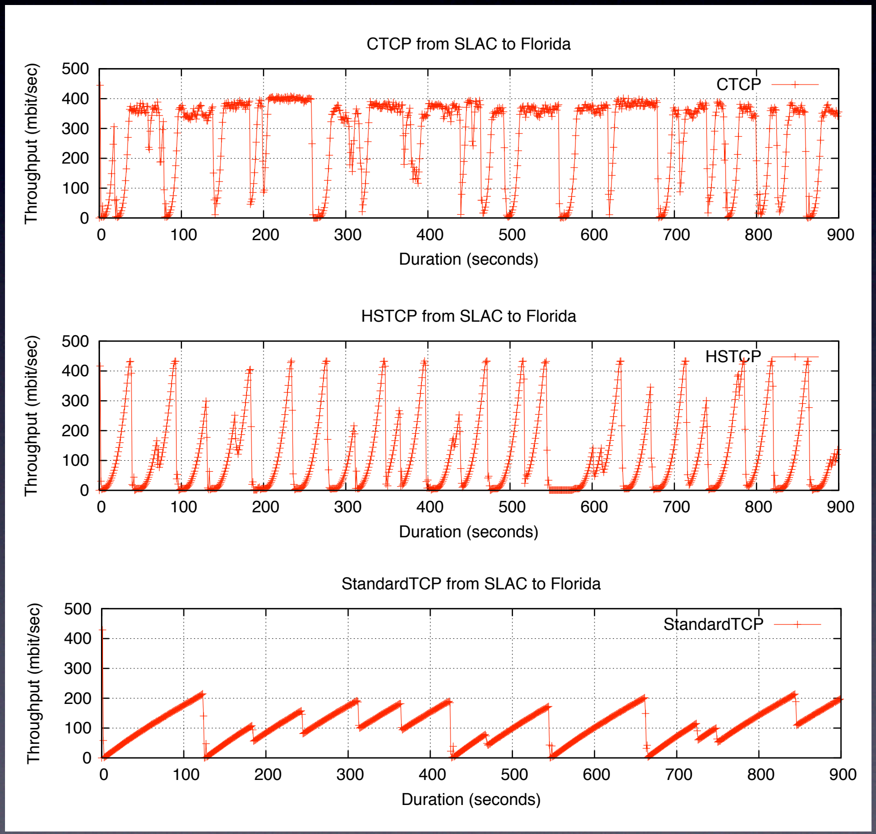
Summary

- (unsurprisingly) CTCP performance appears to be related to the queue provisioning on the network path
  - Good σ_f and ξ_f fairness on well provisioned networks (eg Caltech and Florida)
  - On Ireland link fairness and throughput performance is comparable to HSTCP
- Two mods for CTCP tested:
  - burst control: improves both throughput and fairness metrics compared to original CTCP
  - "diffwnd based" γ auto-tuning: facilitates higher throughput but also maintains low/no impact for the Ireland link

Issues

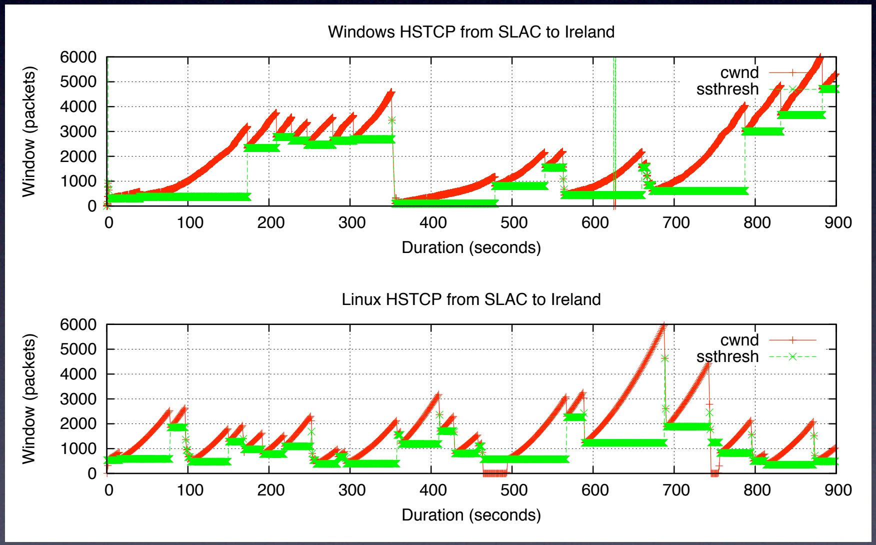
- Instantaneous fairness only considers 1 second intervals
  - Better to analyse as a number of (base) RTTs to give a better indication of the variation of fairness
- All TCP tests exhibit performance problems related to the multiple consecutive drops.
  - due to aggressive re-transmission strategies?
  - TCP stack implementation issues such as SACK processing?
- Analysis is of therefore of stack rather than algorithm

Drops Experienced



- 1-2 flows show very variable throughput
    - Not so apparent with many flows
    - high cwnd values?
- Host issues
    - SACK?
    - Aggressive retransmits?
- Network
    - Cross traffic?

Stack Differences



- Windows vs. Linux
- Temporal difference in tests make direct comparison difficult
- Windows stack appears to push ssthresh to very low values - prevents effective slow start

Conclusion

- Real life tests on real life networks may be overwhelmed by stack differences rather than just TCP congestion control
  - Variations in bandwidth unknown: aggressive retransmissions, SACK deficiencies, cross traffic?
- Fairness very important
  - Defined two fairness metrics; each give a different perspective of the relative performance (intra protocol)
  - Defined impact parameter to determine the effect on existing bulk transport (inter protocol fairness)
- Used to determine how effectiveness modifications to the CTCP stack were

Papers

- "Characterization and Evaluation of TCP and UDP-based Transport on Real Networks", Les Cottrell et al, PFLDnet2005
- "Evaluation of TCP Congestion Control Algorithms on the Windows Vista Platform", Yee-Ting Li
    - http://www.slac.stanford.edu/pubs/slactns/tn04/slac-tn-06-005.pdf