Compound TCP
draft-sridharan-tcpm-ctcp-00.txt

Murari Sridharan
Windows TCP/IP Networking, Microsoft Corp.

(Collaborators: Kun Tan, Jingmin Song, MSRA & Qian Zhang, HKUST)

Design goals

- Efficiency
  - Improve throughput by efficiently using the spare capacity in the network
- RTT fairness
  - Intra-protocol fairness when competing with flows that have different RTTs
- TCP fairness
  - Must not impact performance of regular TCP flows sharing the same bottleneck
- Stability

The Compound TCP approach

- Synergy between loss and delay based approaches
  - Using delay to sense network congestion
  - Adaptively adjust aggressiveness based on network congestion level
- One flow, two components
  - Loss based component: *cwnd* (standard TCP Reno)
  - Scalable delay-based component: *dwnd*
  - TCP send window is controlled by

    $win = cwnd + dwnd$

CTCP congestion control

- Vegas-like early congestion detector
  - Estimate the backlogged packets (diff) and compare it to a threshold, γ
- Binomial increase when no congestion

$$win(t+1) = win(t) + \alpha \cdot win(t)^k$$

- Multiplicative decrease when loss

$$win(t+1) = win(t) \cdot (1 - \beta)$$

- On detecting incipient congestion
  - Decrease *dwnd* and yield to competing flows

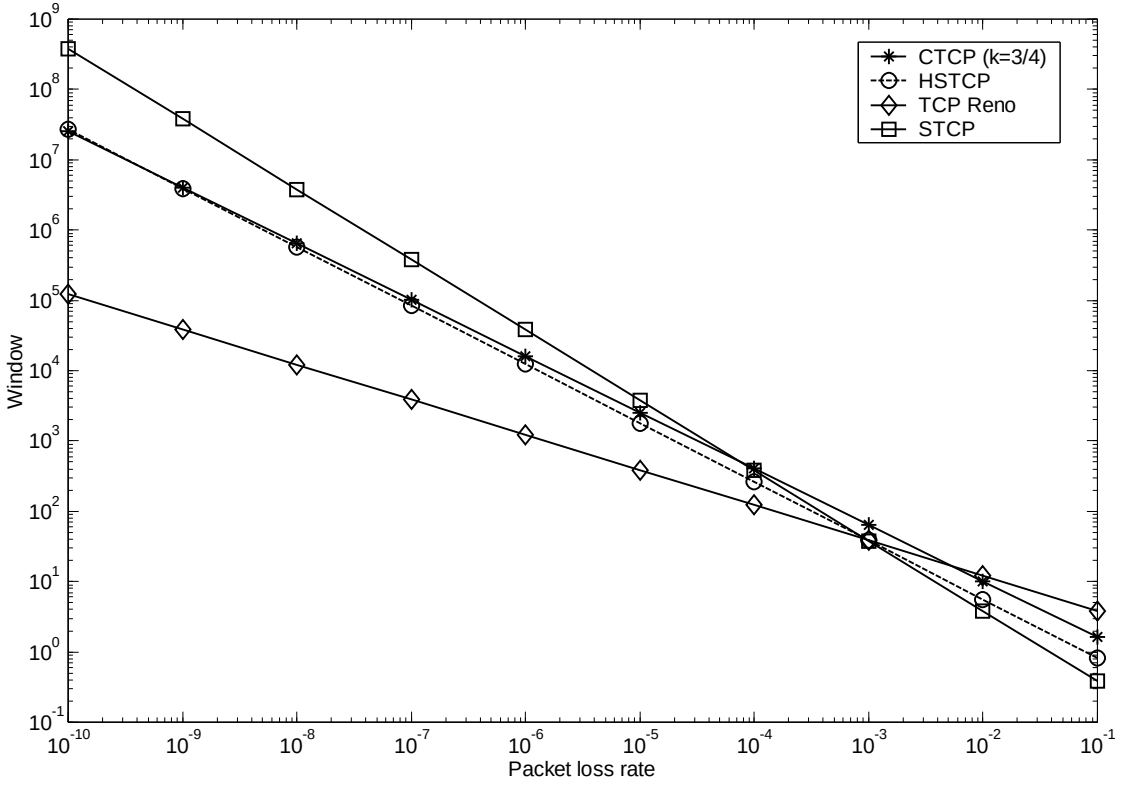
CTCP congestion control

- *cwnd* is updated as TCP Reno
- *dwnd* control law

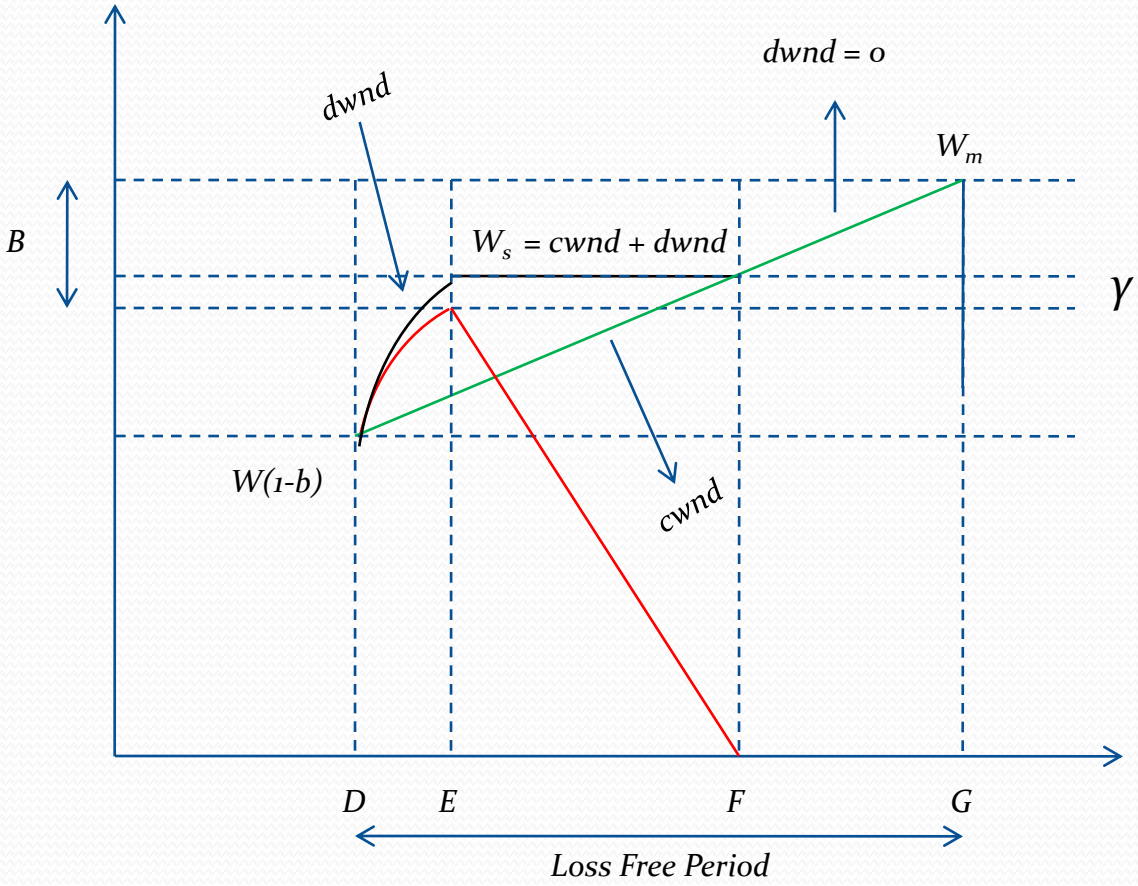
$$dwnd(t+1) = \begin{cases} dwnd(t) + (\alpha \cdot win(t)^k - 1)^+, \text{if } diff < \gamma \\ (dwnd(t) - \zeta \cdot diff)^+, \text{if } diff \geq \gamma \\ (win(t) \cdot (1-\beta) - cwnd/2)^+, \text{if loss is detected} \end{cases}$$

- The above control law kicks in only when the flow is in congestion avoidance and *cwnd* >= 38 packets. No changes to slow start phase.

Response function



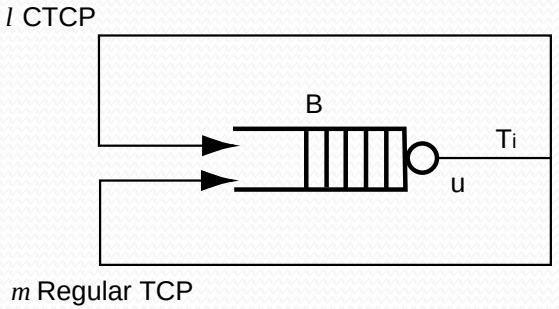
CTCP window evolution



Convergence and RTT fairness

- *Theorem 1:* Two CTCP flows with same round trip delay converge to fair share.
- *Theorem 2*: Let *Th1* and *Th2* present the throughput of two CTCP flows with round trip times *R1* and *R2*, respectively. Then, the following inequality satisfied

$$\frac{Th_1}{Th_2} < \left(\frac{R_2}{R_1}\right)^2$$

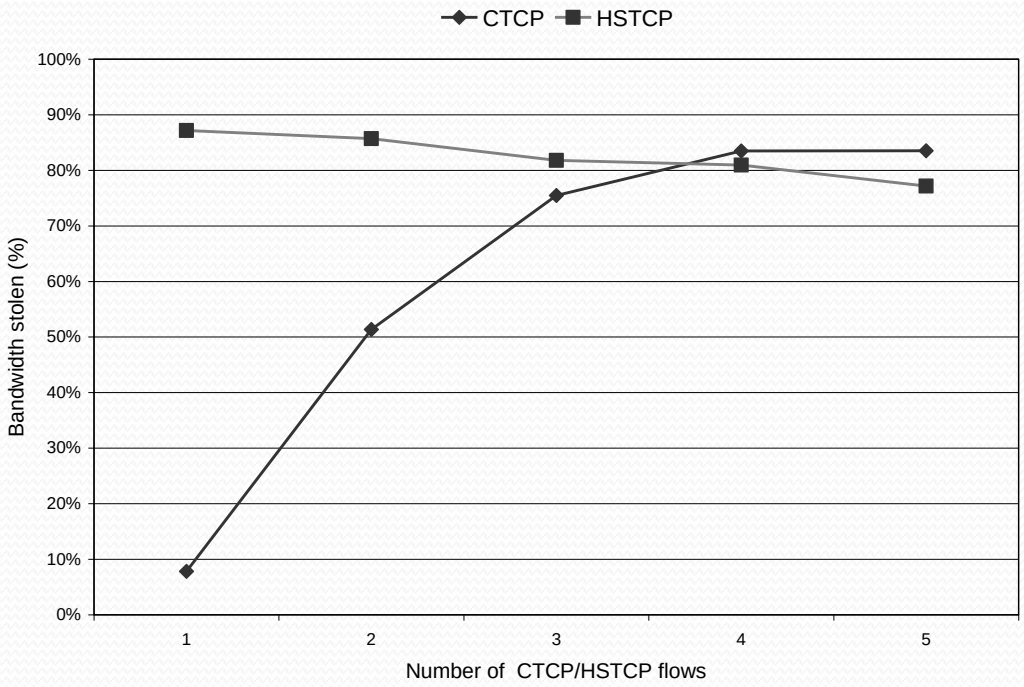


TCP fairness

- Bandwidth stolen
  - Let P be the aggregated throughput of m regular TCP flows when they compete with l regular flows. Let Q be the aggregated throughput when competing with high-speed flows. The bandwidth stolen by high-speed protocol flows from regular TCP flows is $B_{stolen} = \frac{P - Q}{P}$
- *Theorem 3*: CTCP is fair and will not steal bandwidth from competing flows when $\frac{B}{m+l} > \gamma$, where B is the bottleneck buffer size.

Effect of *Gamma*

- γ fixed at 30 packets. This works well on most scenarios
- Delay component loses ability to detect early congestion
  - Average buffer allocated for each flow is < γ



Gamma tuning by emulation

- Loss based component of CTCP emulates the behavior of regular TCP. The *cwnd's* of competing flows converge and should be the same before hitting a packet loss.
- At the end of every round, compute backlogged packets (*Diff_reno)* purely based on *cwnd* the loss based component.
- On a packet loss, choose $\gamma = 3/4 * Diff\_reno$. Update γ using an exponential moving average $\gamma = (1-\lambda)\gamma + \lambda \cdot \gamma^*$
- Ensure $\gamma_{low} <= \gamma <= \gamma_{high}$. Experimentally we have determined $\gamma_{low} = 5$, $\gamma_{high} = 30$

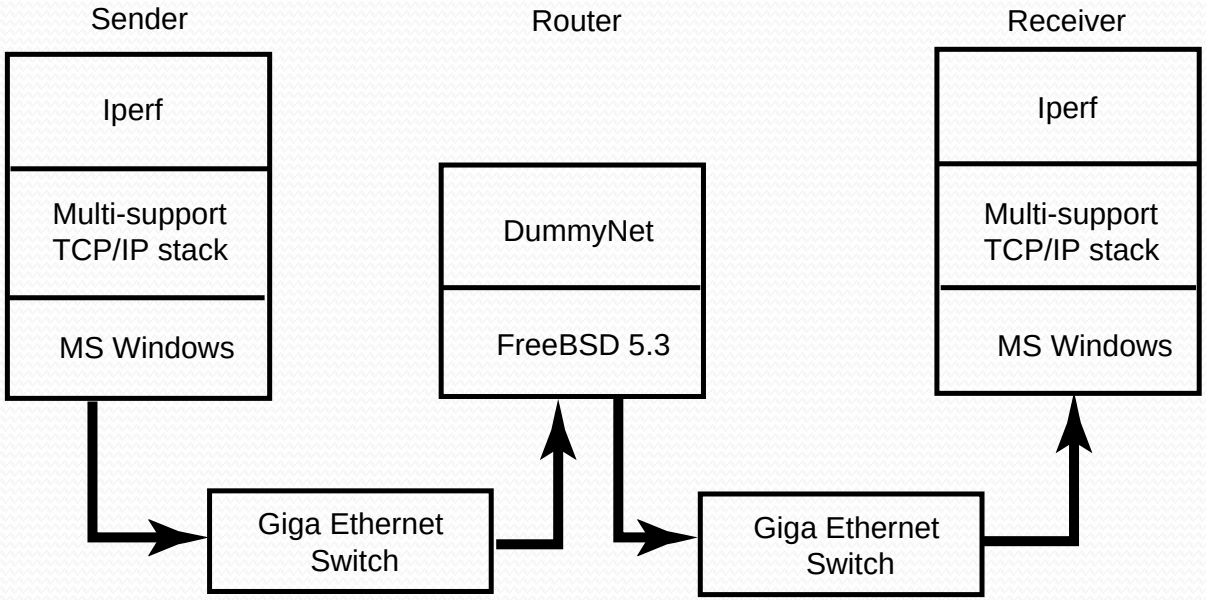
Summary

- CTCP is a promising approach that achieves good efficiency, RTT fairness and TCP fairness.
- Implemented on Windows platform and verified the above properties in a range of environments.
  - Validated on test-beds, Microsoft IT high-speed links, Microsoft internal deployments, SLAC/Internet2/ESNet production links.
- We believe CTCP is safe for Internet deployment
- Experimental RFC on Compound TCP http://research.microsoft.com/users/dthaler/draft-sridharan-tcpm-ctcp-00.txt

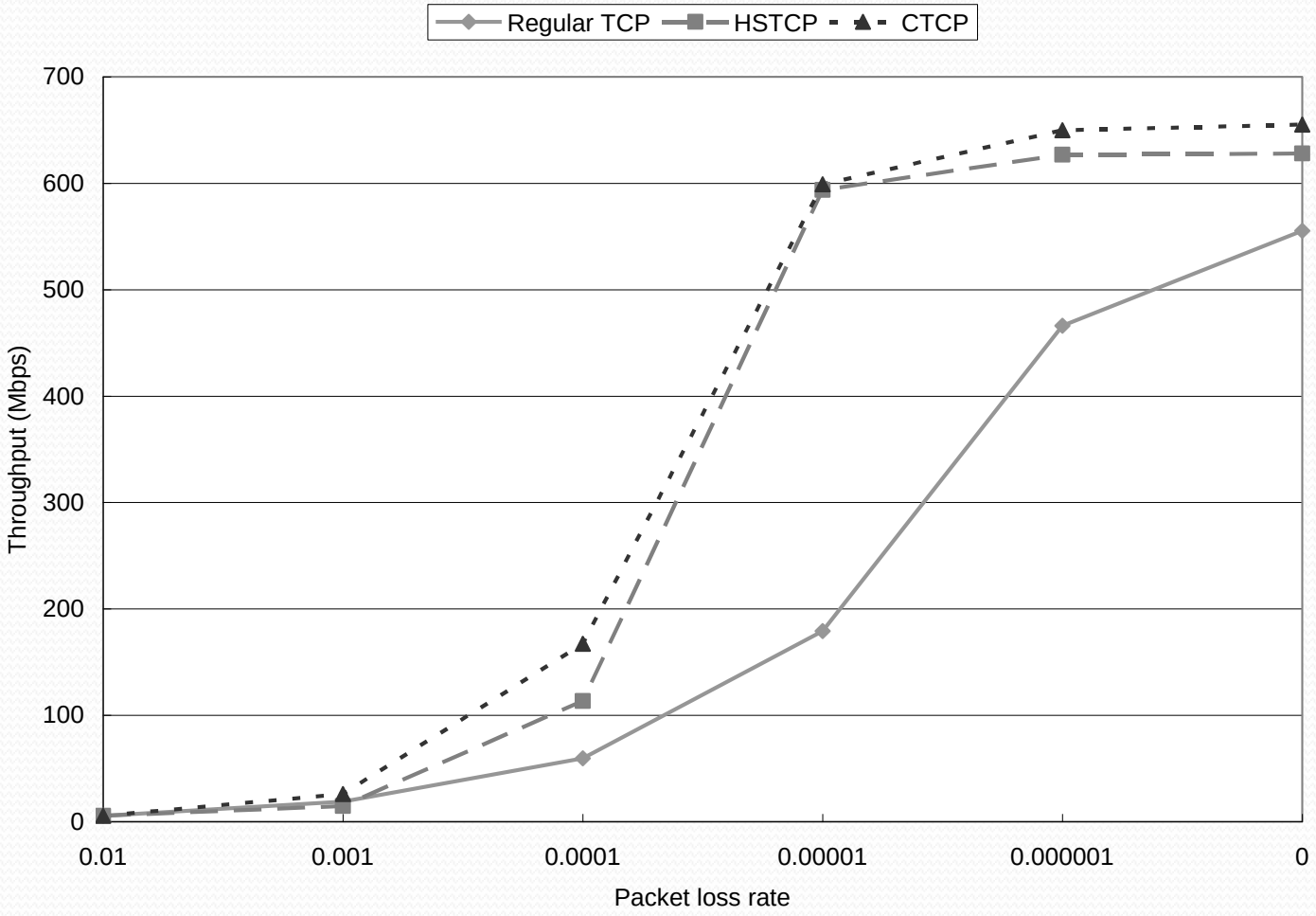
Results

Implementation & Evaluation

- Windows platform implementation
  - Microsecond resolution RTT timer
  - Dynamic memory management for sample buffers
- DummyNet-based Test-bed



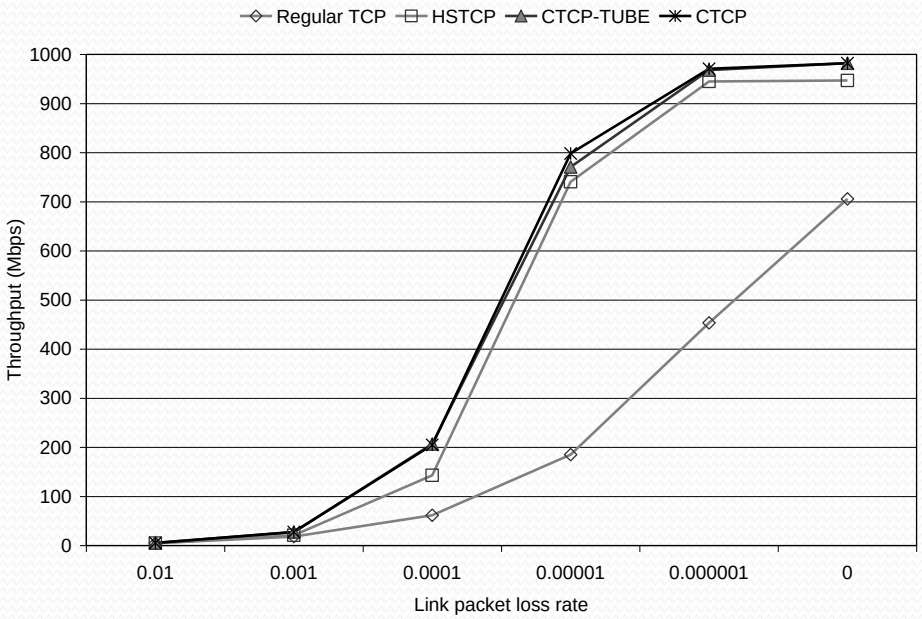
Efficiency

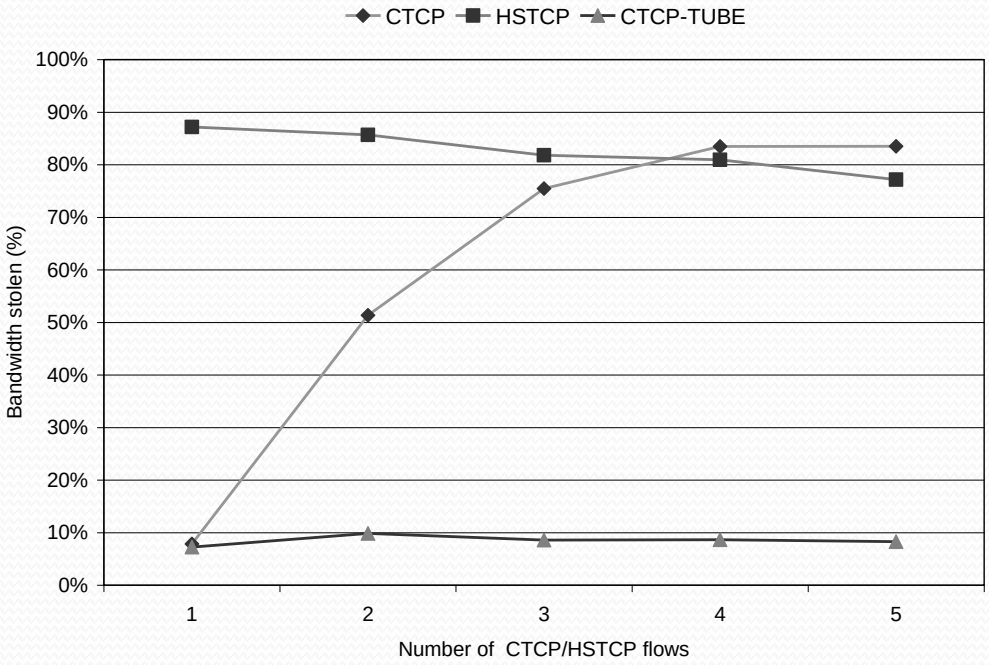

Regular TCP
HSTCP
CTCP
700
600
500
400
300
200
100
0
Throughput (Mbps)
0.01
0.001
0.0001
0.00001
0.000001
0
Packet loss rate

RTT fairness

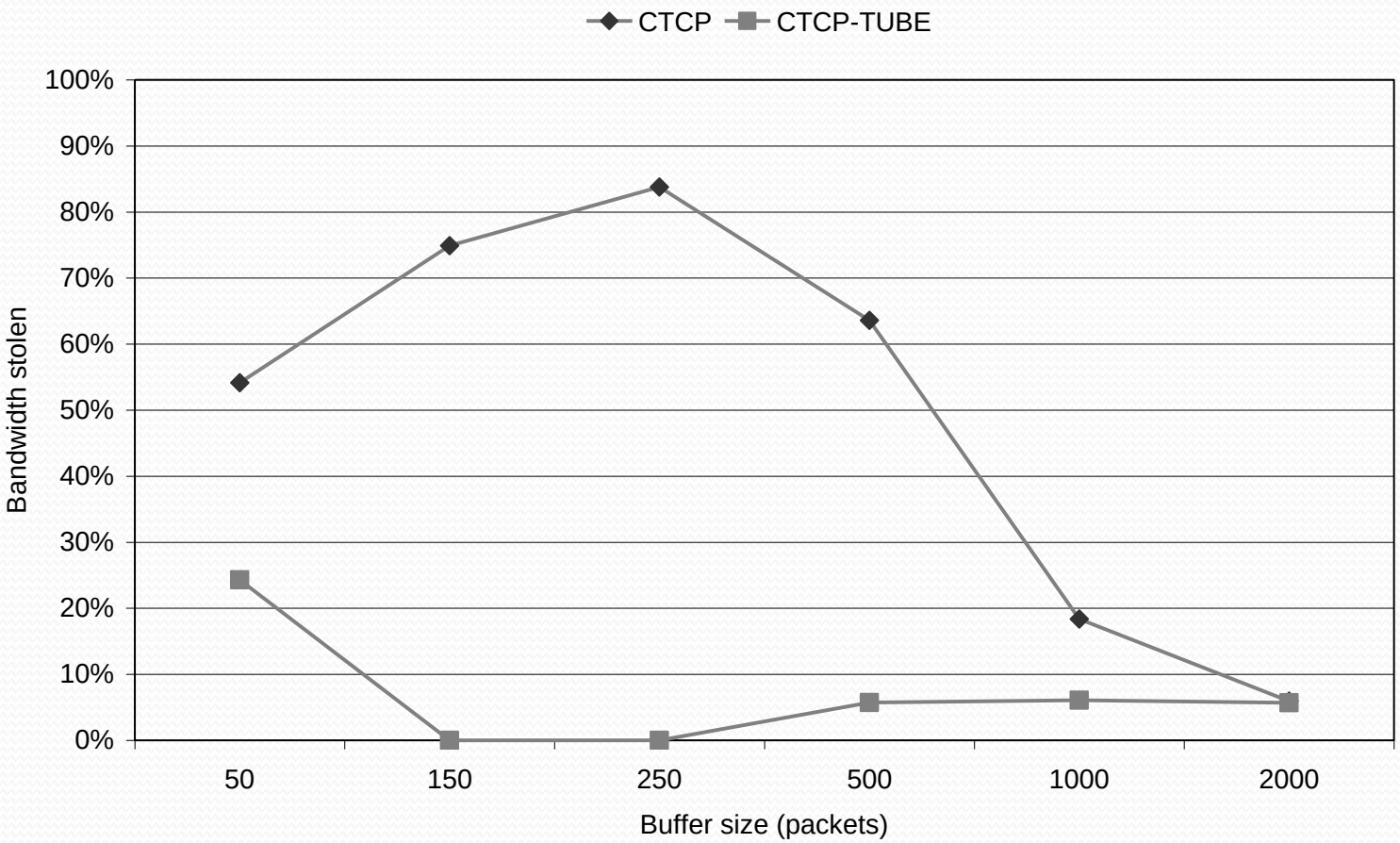
Inverse RTT ratio	1	2	3	6
Regular TCP	0.9	3.6	6.2	31.6
HSTCP	1	28.9	90.5	233.8
CTCP	1	2.2	4.1	9.5

TCP fairness – Effect of γ

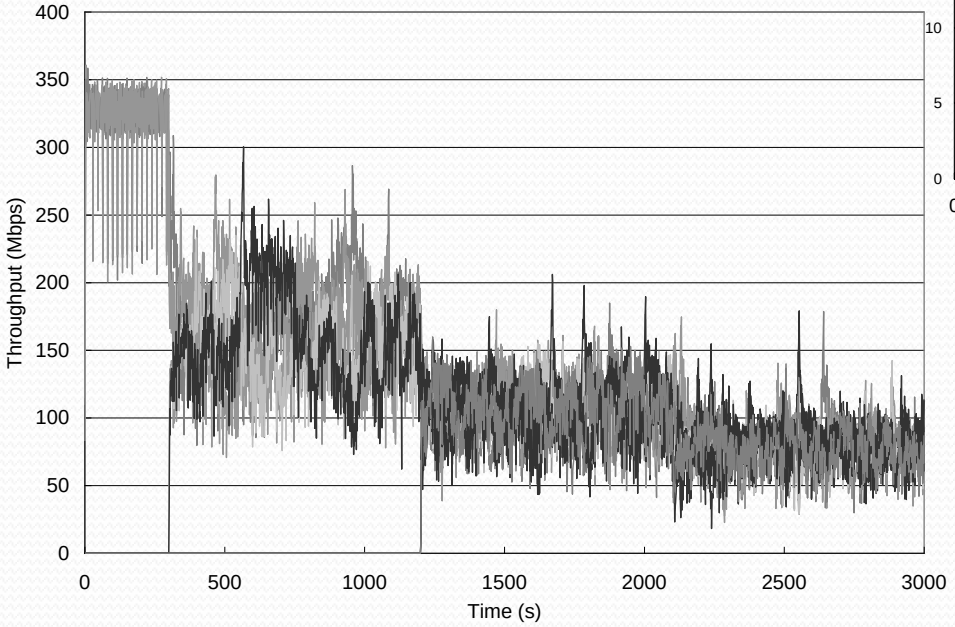


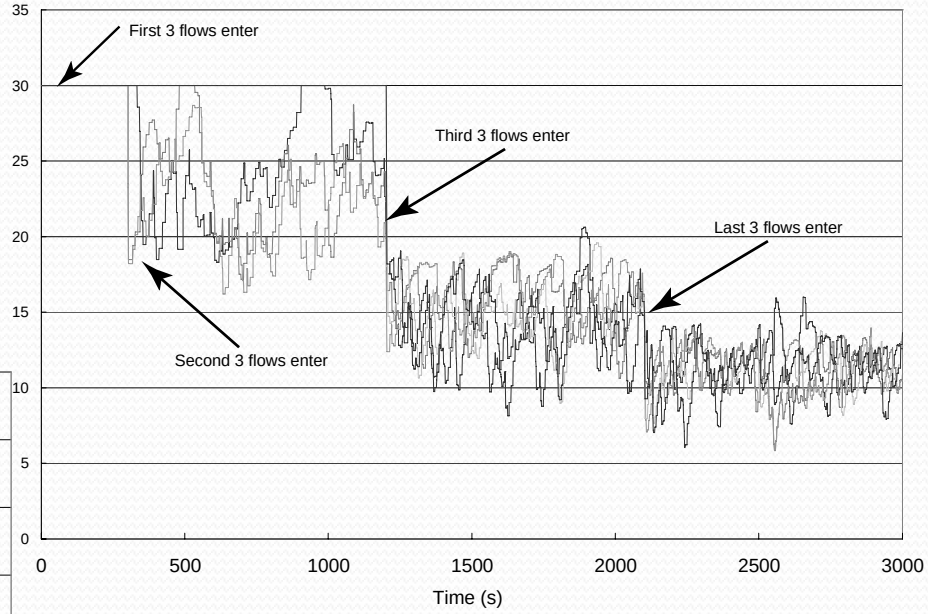


TCP fairness – Varying buffer sizes

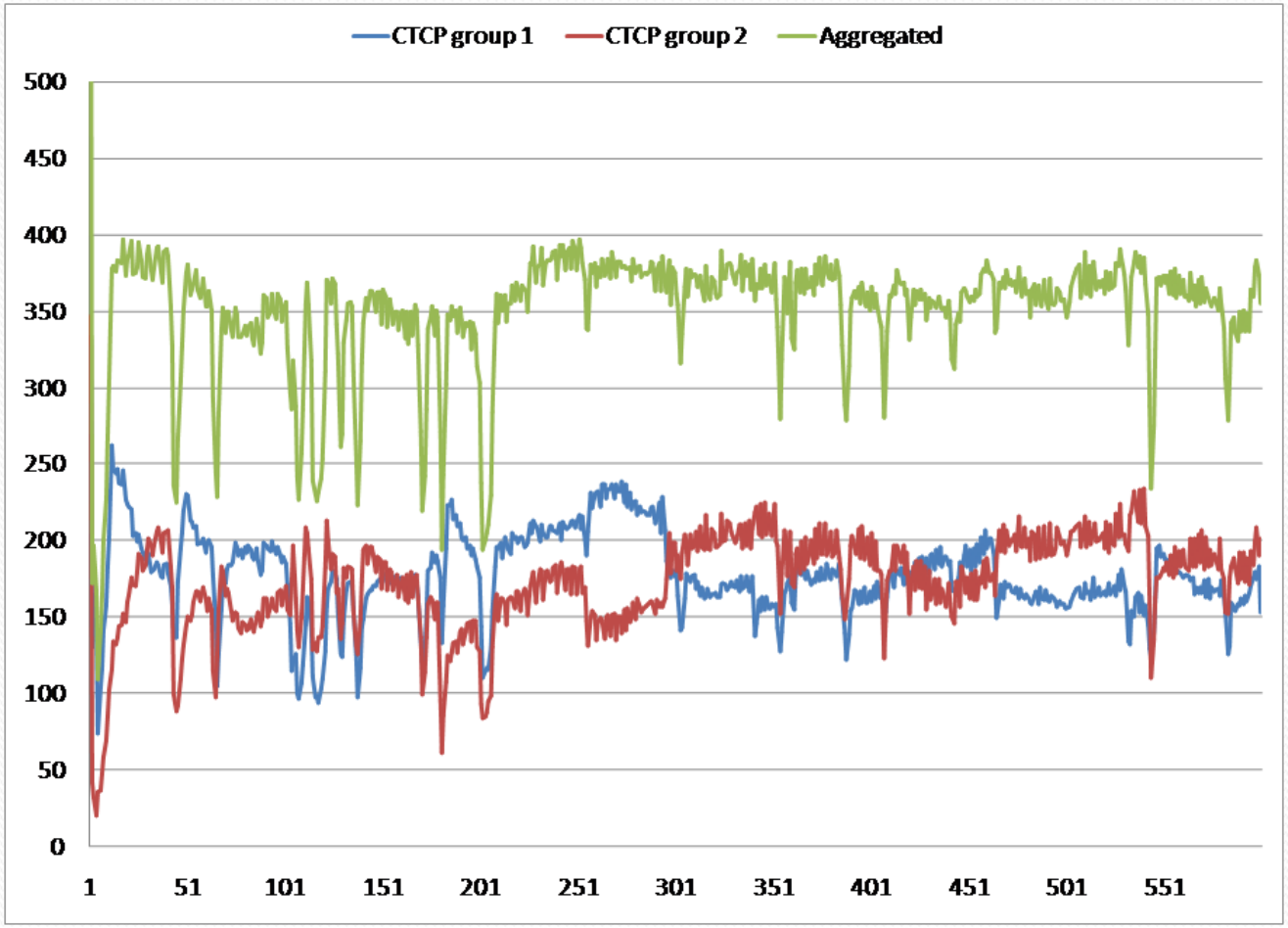


Intra-protocol fairness and convergence of γ

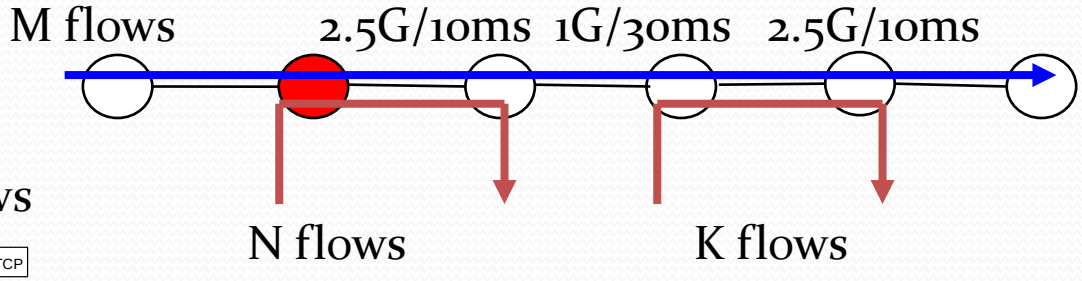




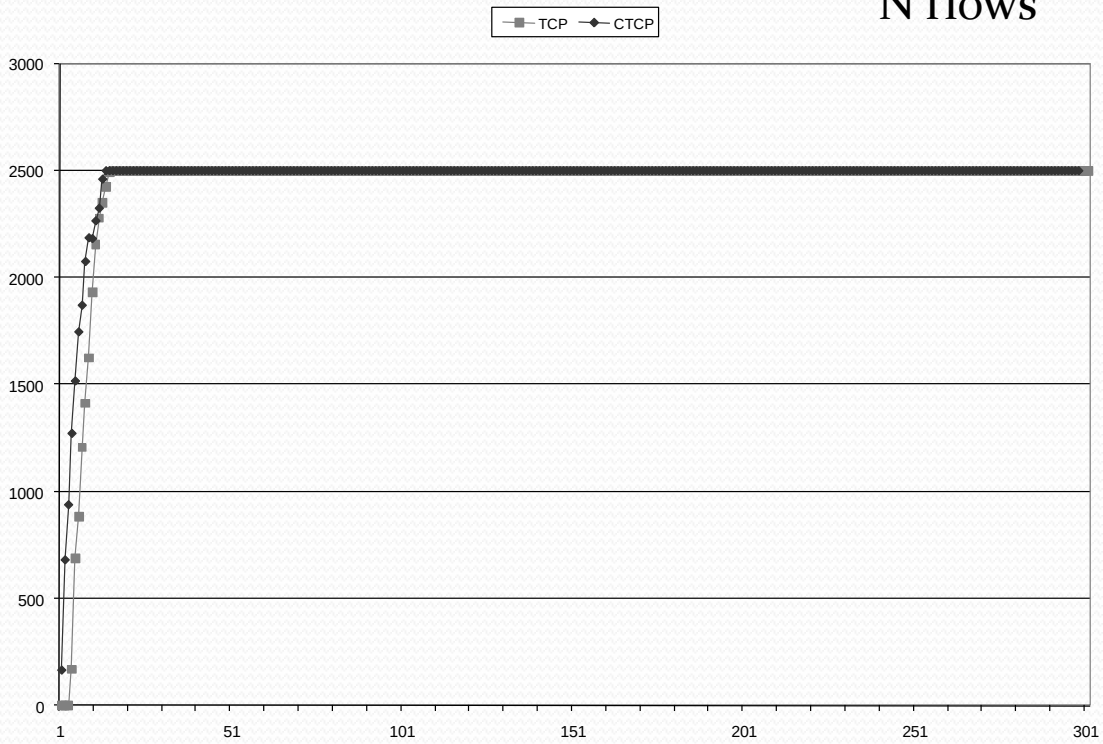
Stability


CTCP group 1
CTCP group 2
Aggregated
500
450
400
350
300
250
200
150
100
50
0
1
51
101
151
201
251
301
351
401
451
501
551

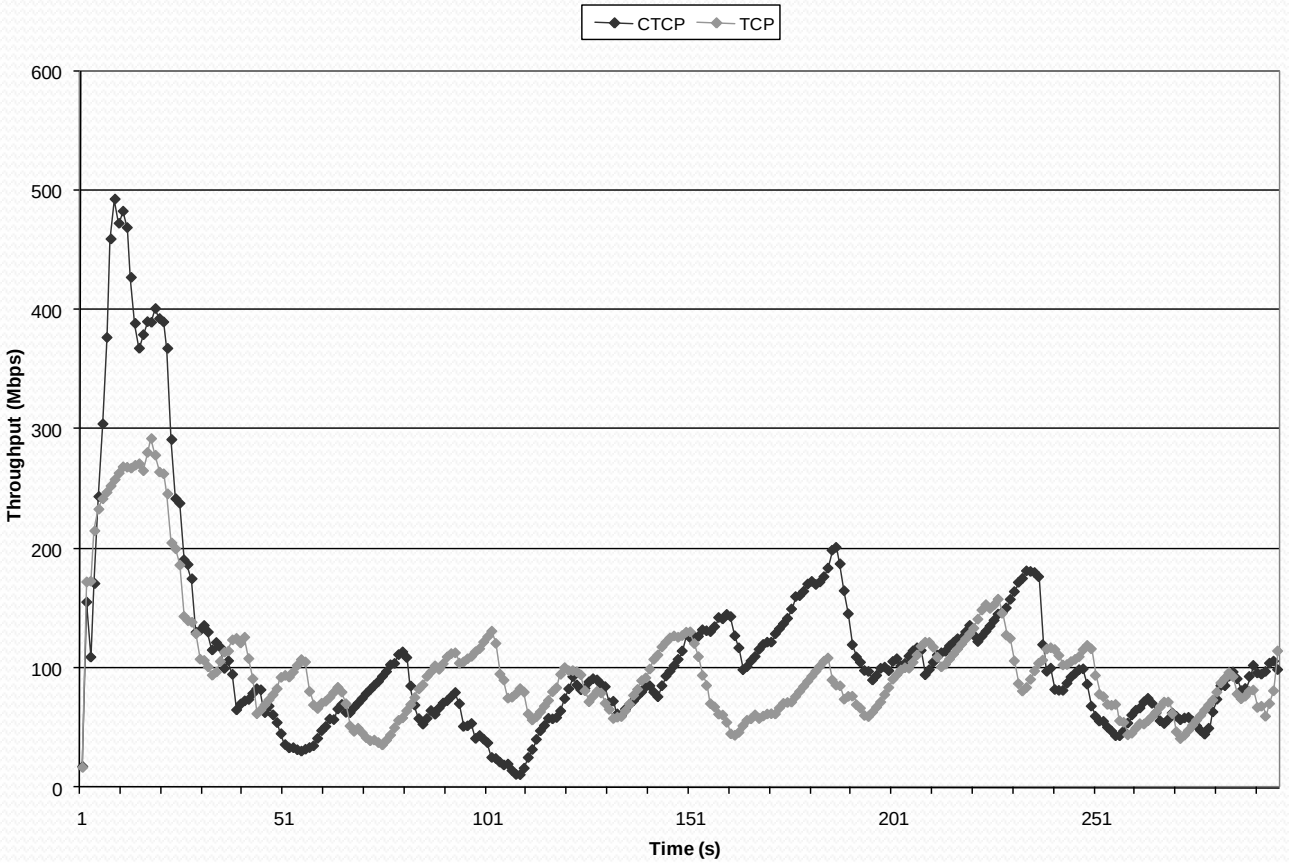
Multiple bottlenecks – Utilization



N=400 TCP, K=50 TCP
M is either 8 CTCP or TCP flows



Multiple bottlenecks – Throughput



N=400 TCP, K=50 TCP
M is either 8 CTCP or TCP flows