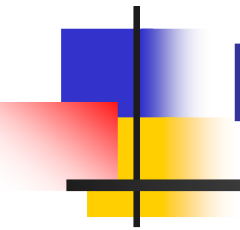
Evaluation of Advanced TCP Stacks in the iSCSI Environment

Girish Motwani and K. Gopinath
Department of Computer Science and Automation,
Indian Institute of Science, Bangalore

Agenda

- Motivation
- Background
- TCP Variants
- Setup
- Results and Analysis
- Conclusions



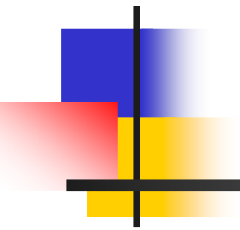
Motivation

Motivation

- iSCSI protocol is increasingly being used to access block storage over standard ethernet based TCP/IP Networks.
- iSCSI protocol could be used over high speed networks with long bw-delay products

Motivation

- Over high speed networks with large RTT, TCP Reno underutilizes network bandwidth
- To overcome this drawback, variants of the TCP Congestion Control Algorithm such as Scalable TCP, FAST, HSTCP, HTCP, BIC TCP have been proposed
- Our work compares the performance of these TCP Variants in an iSCSI environment.

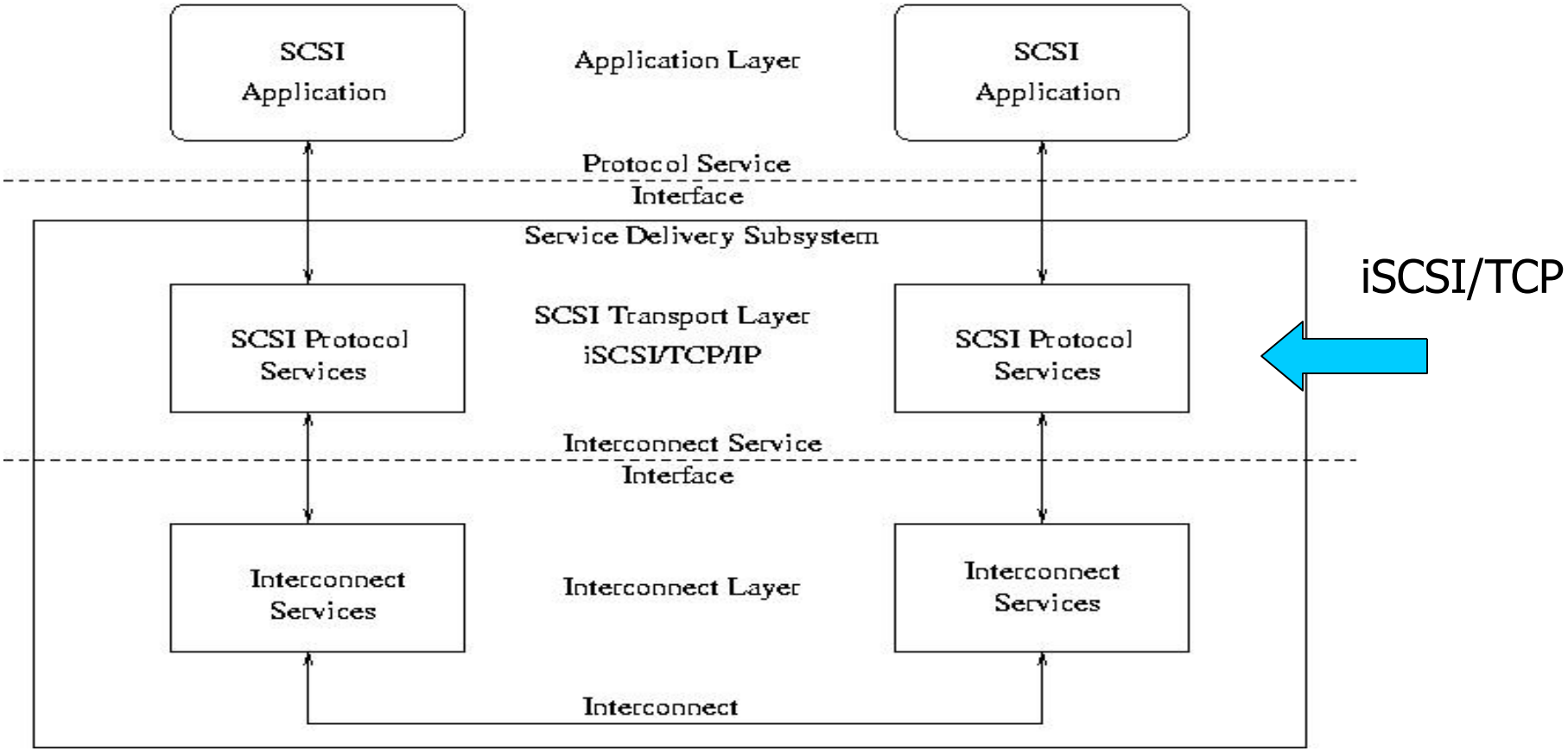


Background

SCSI (Small Computer System Interface)

- Standard interface and communication protocol for computer peripherals
- Based on the Client – Server Architecture
- Clients called Initiators issue requests and the Severs called Targets respond to initiator requests.
- SCSI commands are sent in units called Command Descriptor Blocks (CDB)

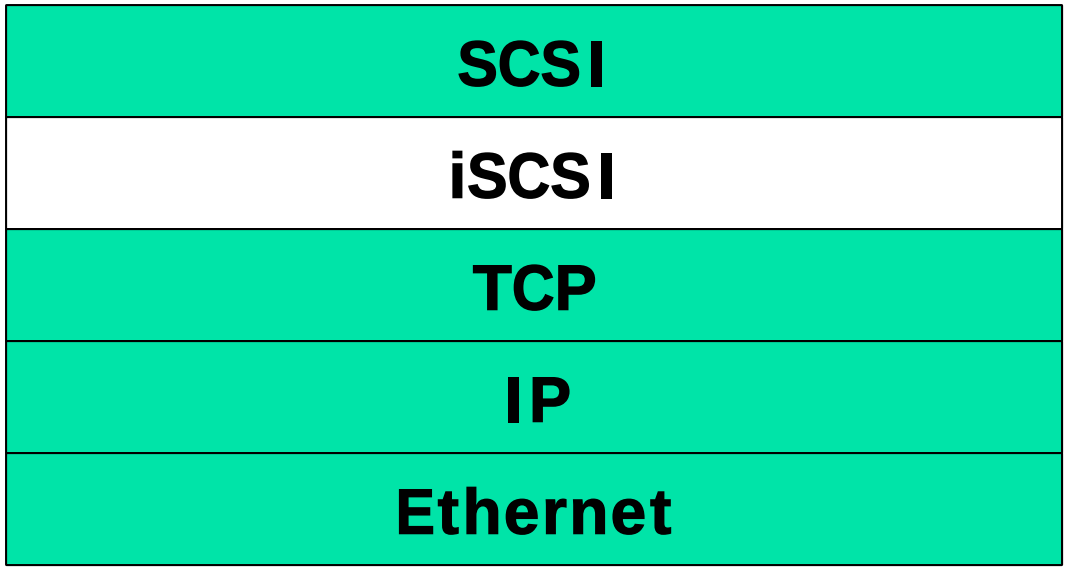
SCSI Protocol Stack



Internet SCSI (iSCSI)

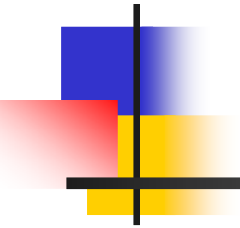
- Maps the SCSI block oriented storage data over TCP/IP
- Establishes an iSCSI session between SCSI Initiator and Target
- Session can comprise of Multiple TCP Connections identified by ConnectionID

iSCSI Layering



iSCSI

- iSCSI uses TCP as the Transport Protocol
- iSCSI Data Flow governed by TCP Congestion Control Algorithm
- iSCSI protocol messages are exchanged using structures called Protocol Data Unit (PDU)
- iSCSI parameter MaxBurstLength determines the Maximum amount of data that can be sent out in one sequence

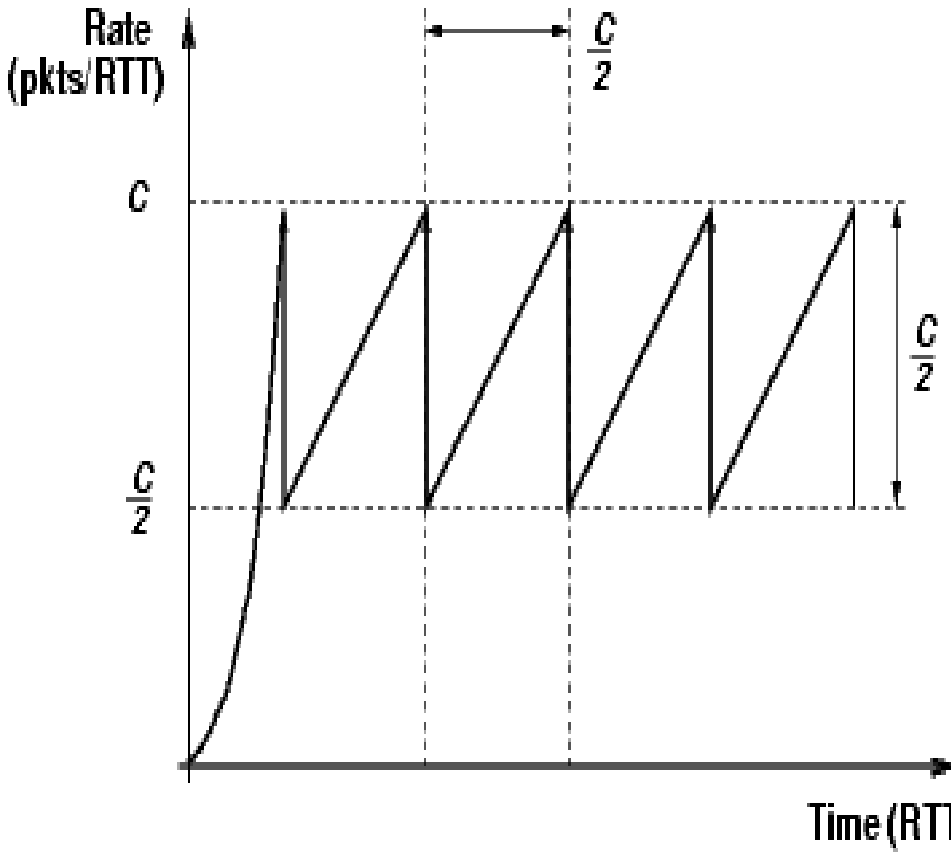


TCP Variants

TCP Reno

- Uses Additive Increase Multiplicative Decrease (AIMD) Congestion Control Approach
  - ACK : $new_{cwnd} = old_{cwnd} + \alpha / old_{cwnd}$
  - LOSS: $new_{cwnd} = \beta * old_{cwnd}$
  - For Reno, $\alpha = 1$, $\beta = 0.5$
- With a 1 Gbps link, 1500 bytes packet size, 100 ms RTT, Reno takes 14 minutes to achieve full utilization following a loss event

TCP Reno



- Over high bw-delay prod links, results in underutilization of the link bandwidth.
- Several TCP variants to overcome this drawback have been proposed.

FAST TCP

- Based on TCP Vegas
- Uses both queuing delay and packet loss to detect congestion
- Uses an equation based window control approach unlike TCP Reno's AIMD approach

Reference : S. Low et al, FAST TCP Motivation, Architecture Algorithms

Binary Increase Congestion (BIC) TCP

- Modification of TCP Reno's congestion control algorithm
- Consists of two parts
  - Binary Search Increase
  - Additive Increase

Reference : I Rhee, Binary Increase Congestion Control for fast long distance networks

Binary Search Increase

- Given the minimum and maximum window sizes set the target window to midway between the two
- If no losses are detected, the current window size becomes the new Minimum and a new target is calculated
- If losses occur, then current window size is the new Maximum and reduced window size is the new minimum
- More aggressive initially, gets less aggressive as the window size approaches the target.

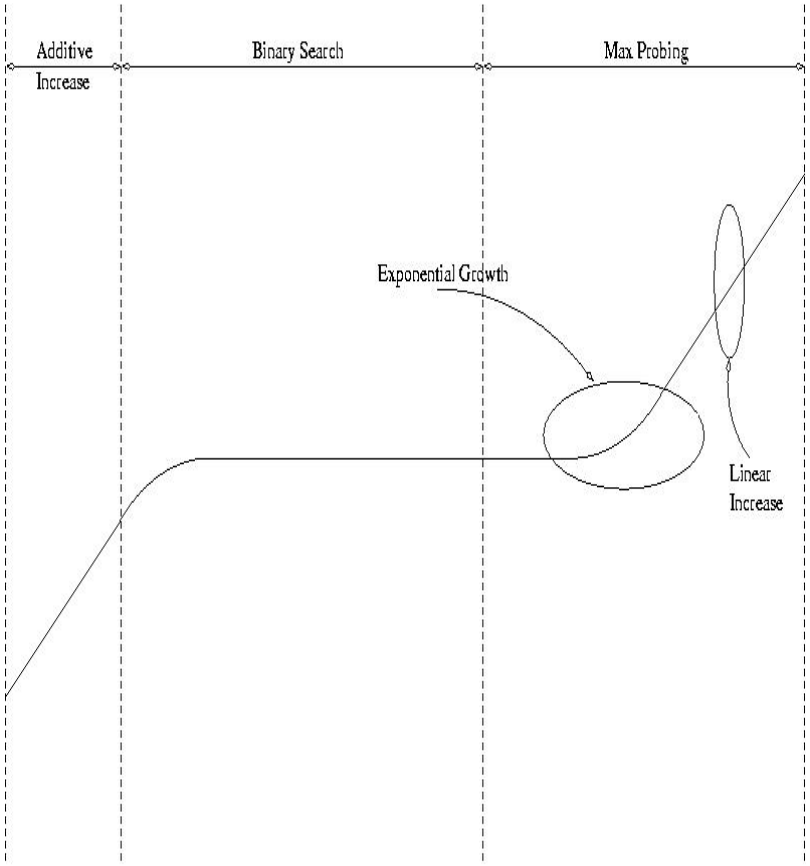
BIC – TCP Example

- Let cwnd = 128. If packet loss is detected at this stage
  - Min_Window=128/2= 64,
  - Target = (Max_Window + Min_Window)/2
- Congestion window changes as:

64 → 96 → 112 → 120 → 124→ 126→ 127

Additive Increase

- If difference between current window and target is large, then direct increase to target may stress the network
- Define a threshold, S_{Max}. If difference is greater than S_{Max} increase by S_{Max} until difference reduces to less than S_{Max}.



H-TCP

- Modification of TCP Reno's congestion control algorithm.
- Employs a different algorithm to increment the congestion window.
- Multiplicative Decrease Factor β is made adaptive

Reference : D Leith, HTCP: TCP for High speed and Long distance networks

H-TCP

- Additive Increment α varies according to the following relation,

  $\alpha = \alpha^L$ if $\Delta \leq \Delta^L$

  $\alpha = \alpha^H(\Delta)$ if $\Delta > \Delta^L$

  Where

  α^L is additive increment in low speed mode

  α^H is additive increment in high speed mode

  Δ is the time elapsed since last congestion event

  Δ^L is the Threshold

Scalable TCP

- Modification of TCP Reno Congestion Control Algorithm, uses MIMD Approach
  - ACK : $new_{cwnd} = old_{cwnd} + \alpha$
  - LOSS: $new_{cwnd} = \beta * old_{cwnd}$
  - Here, $\alpha = 0.01$, $\beta = 0.875$

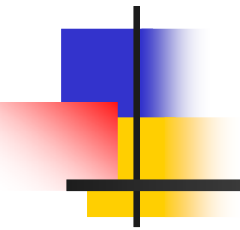
Reference : Tom Kelly, Scalable TCP : Improving Perf. In highspeed Wide Area Networks

HSTCP

- Uses the Reno Congestion Control Algorithm for Window Sizes less than Low_Window (38 segments)

- For Window Sizes greater than Low_Window, the TCP Increment Factor α and Decrement Factor β vary as a function of the current window size.

Reference : S. Floyd, HSTCP RFC 3649

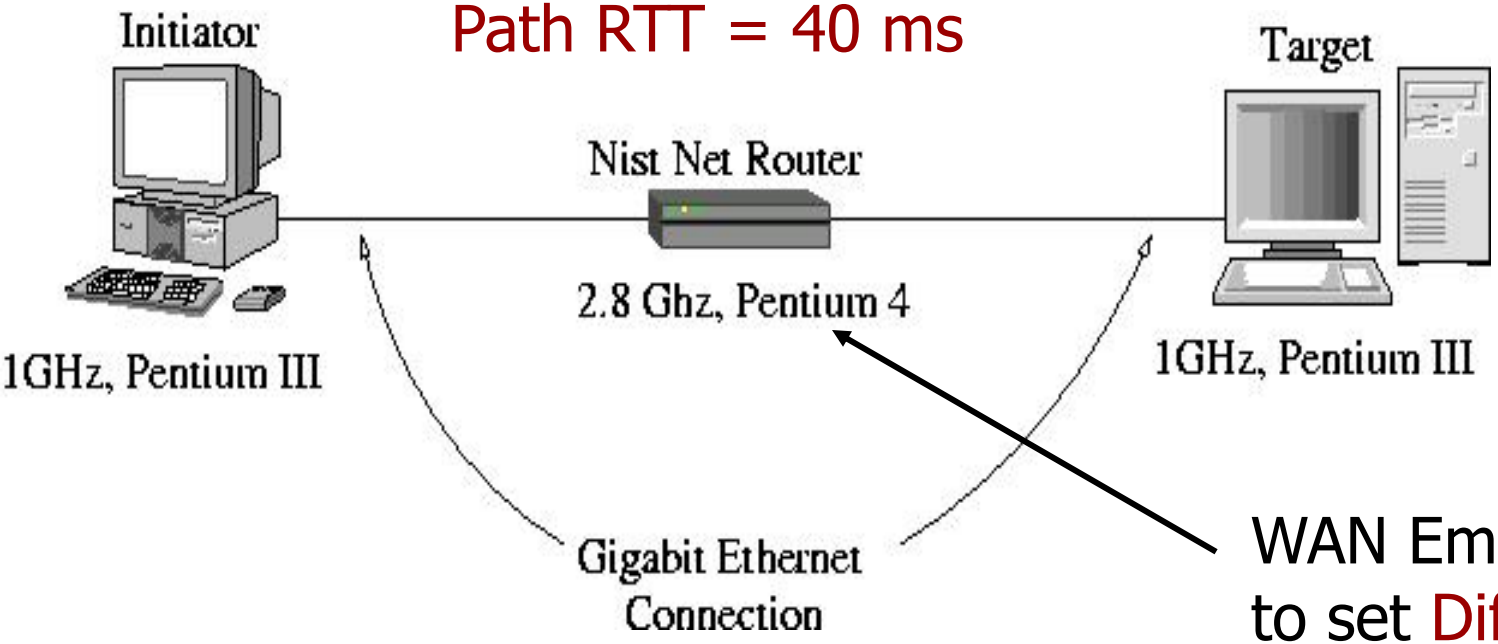


Setup

Experimental Setup

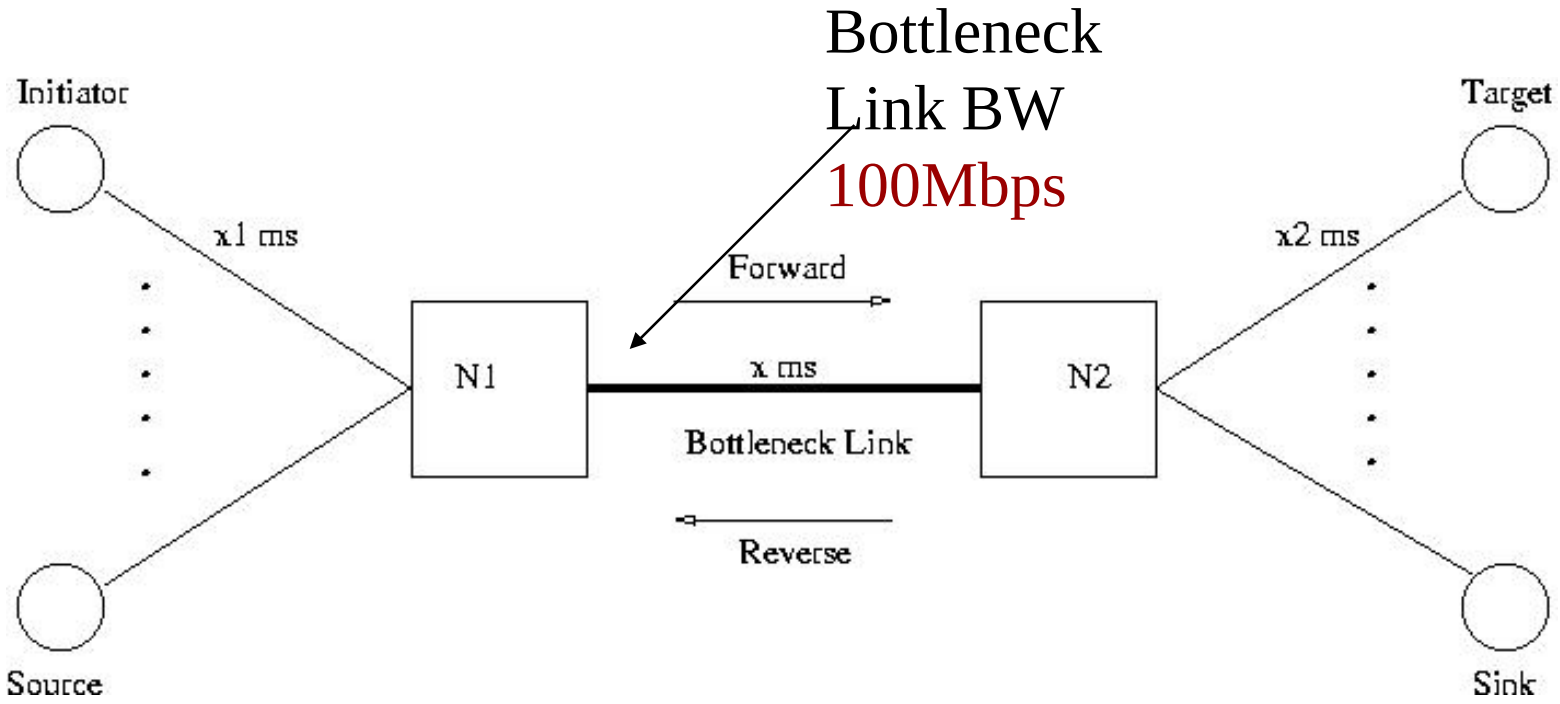
Link Capacity = 18MBytes/sec

Path RTT = 40 ms

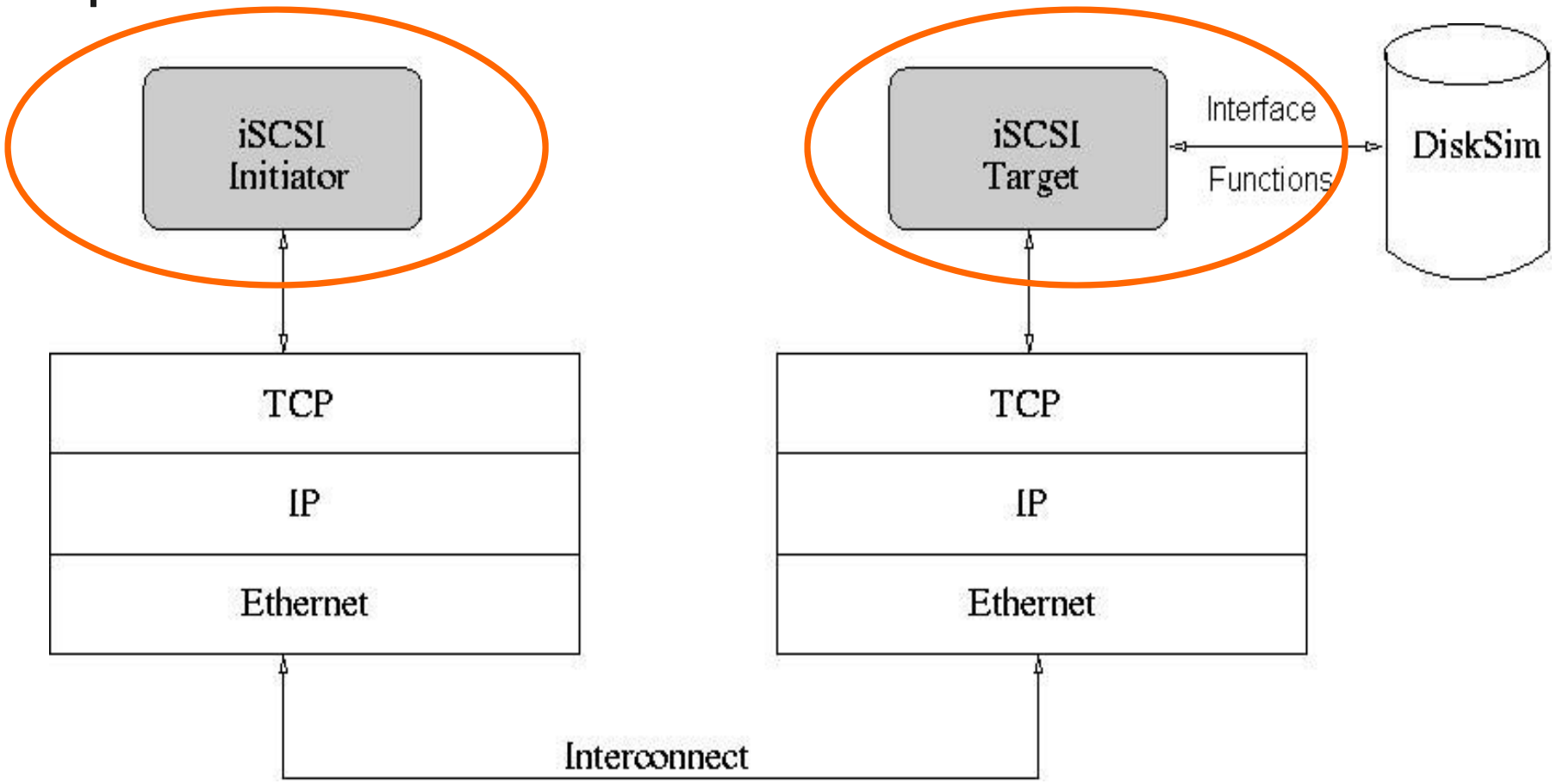


WAN Emulation tool to set Different Path Delays

Simulation Setup

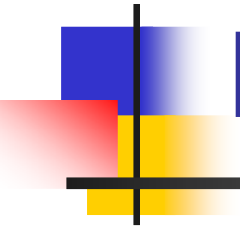


Simulation Setup



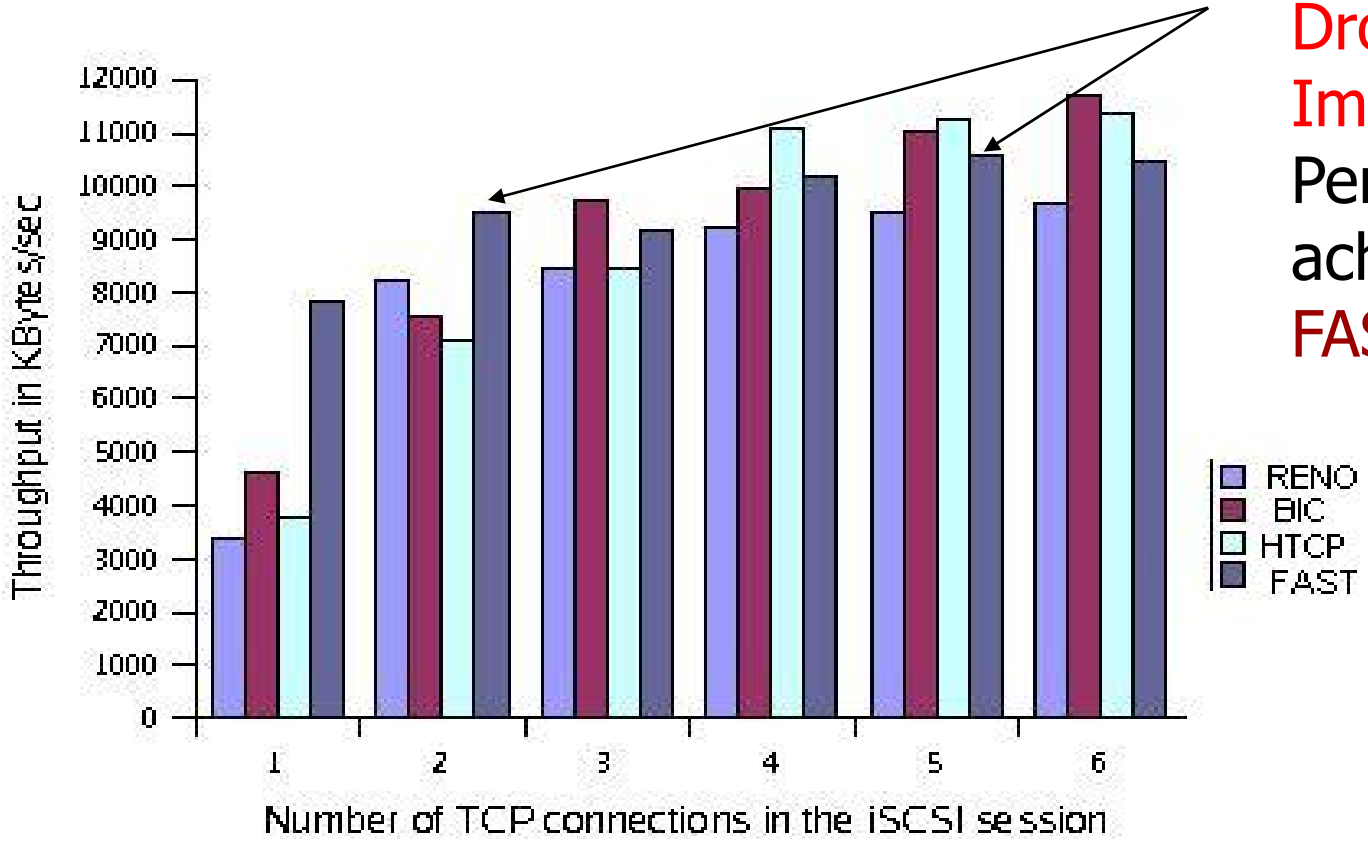
Simulation Setup

- Simulations using ns-2
- Uses our implementation of the iSCSI Protocol in ns-2
- Disk Simulation tool DiskSim integrated with ns-2 to account for disk behavior
- Simulator fed with HP Traces



Results and Analysis

Effect of Number of Connections per Session



Drop in Improvement of Performance achieved with FAST TCP

BIC TCP and HTCP result in improved performance with number of connections

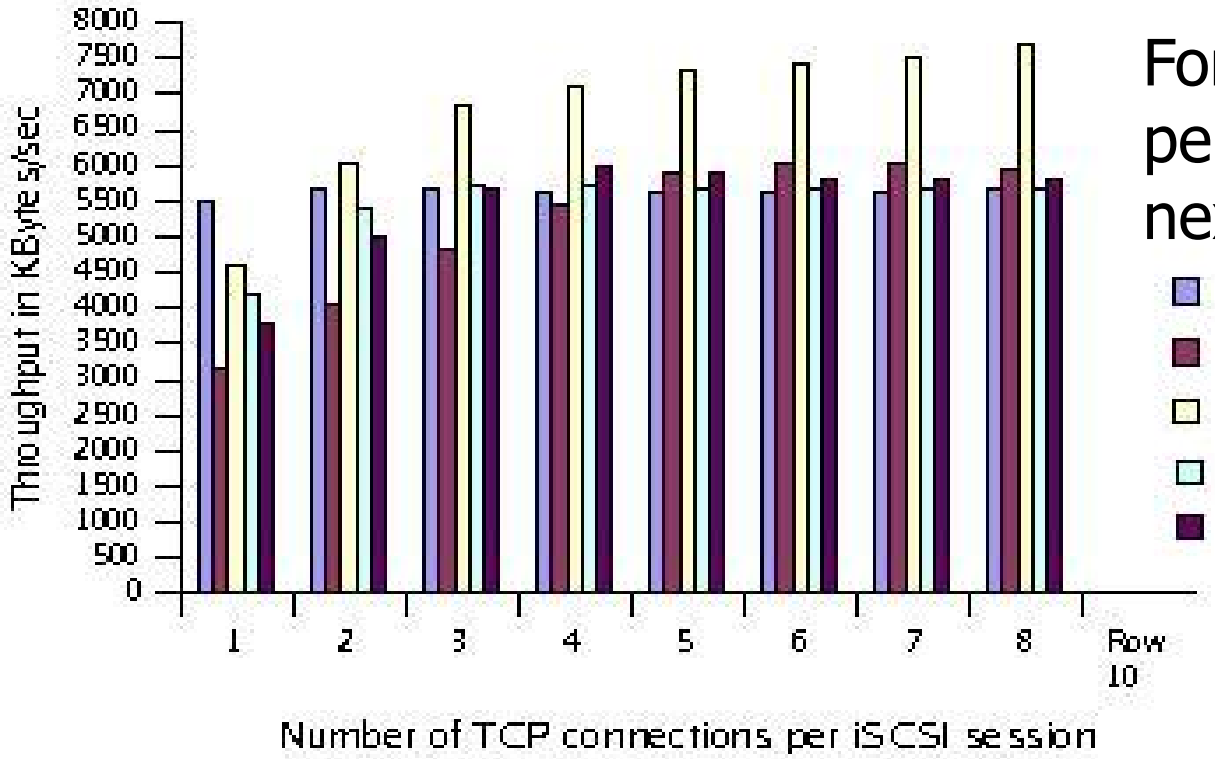
Reasons for drop in Improvement with FAST TCP

- Reverse Traffic
  - Read data flows from Target to Initiator and acts as reverse traffic for the Write Data transfers from Initiator to Target.
  - Acks for the write transfers get Delayed or may get Lost
  - Results in increased RTT values

Reasons for drop in Improvement with FAST TCP (contd)

- FAST TCP uses queuing delay as congestion measure.
- Increased RTT values imply congestion in Forward Path
- With more connections sharing the link Reverse Traffic increases.

Effect of No. of Connections per Session- Using Simulations



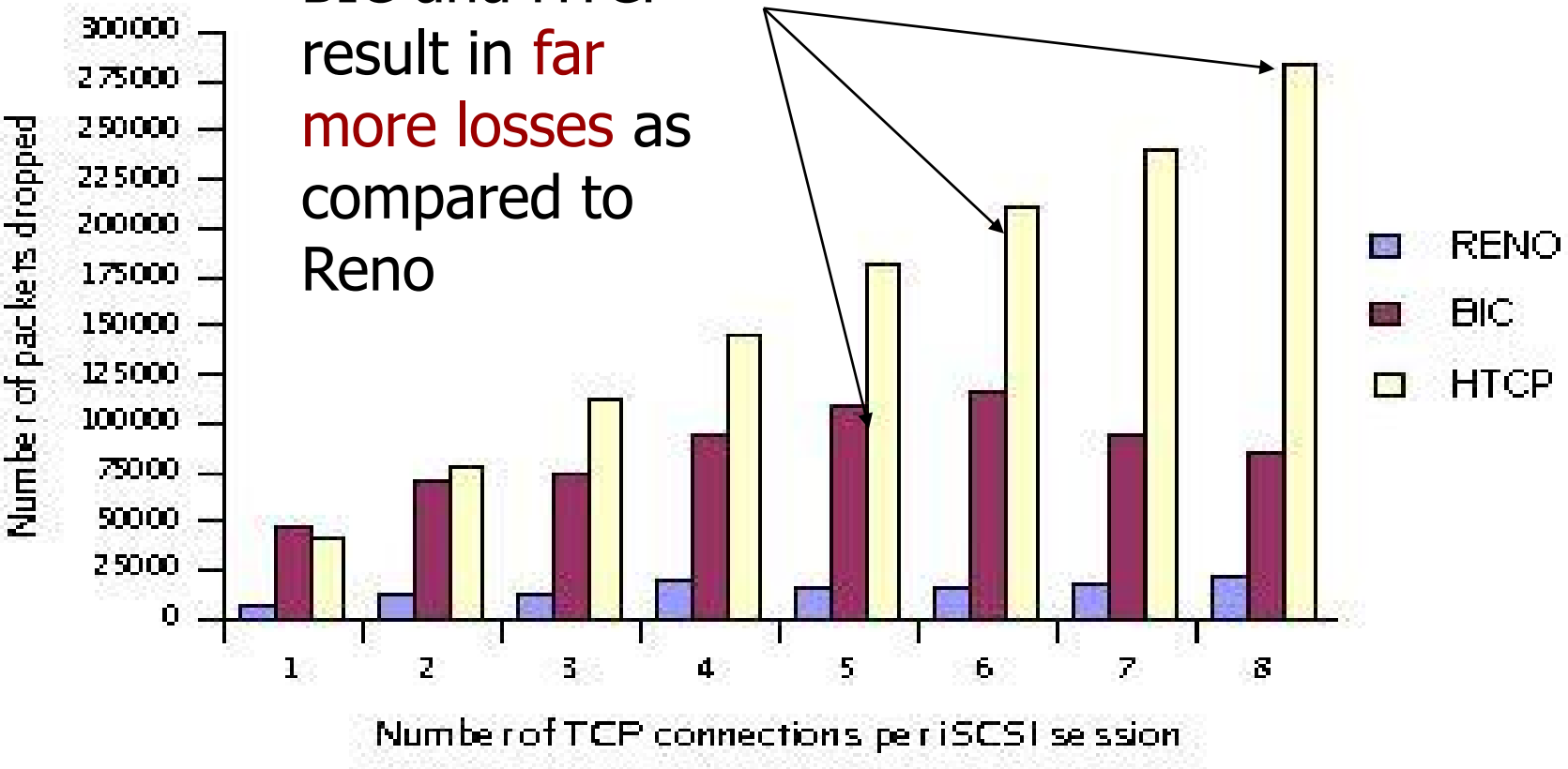
HTCP outperforms the other TCP variants

For NConnections > 4, performance with BIC is next only to HTCP

- RENO
- BIC
- HTCP
- HSTCP
- SCALABLE

Comparison of Packet Drops

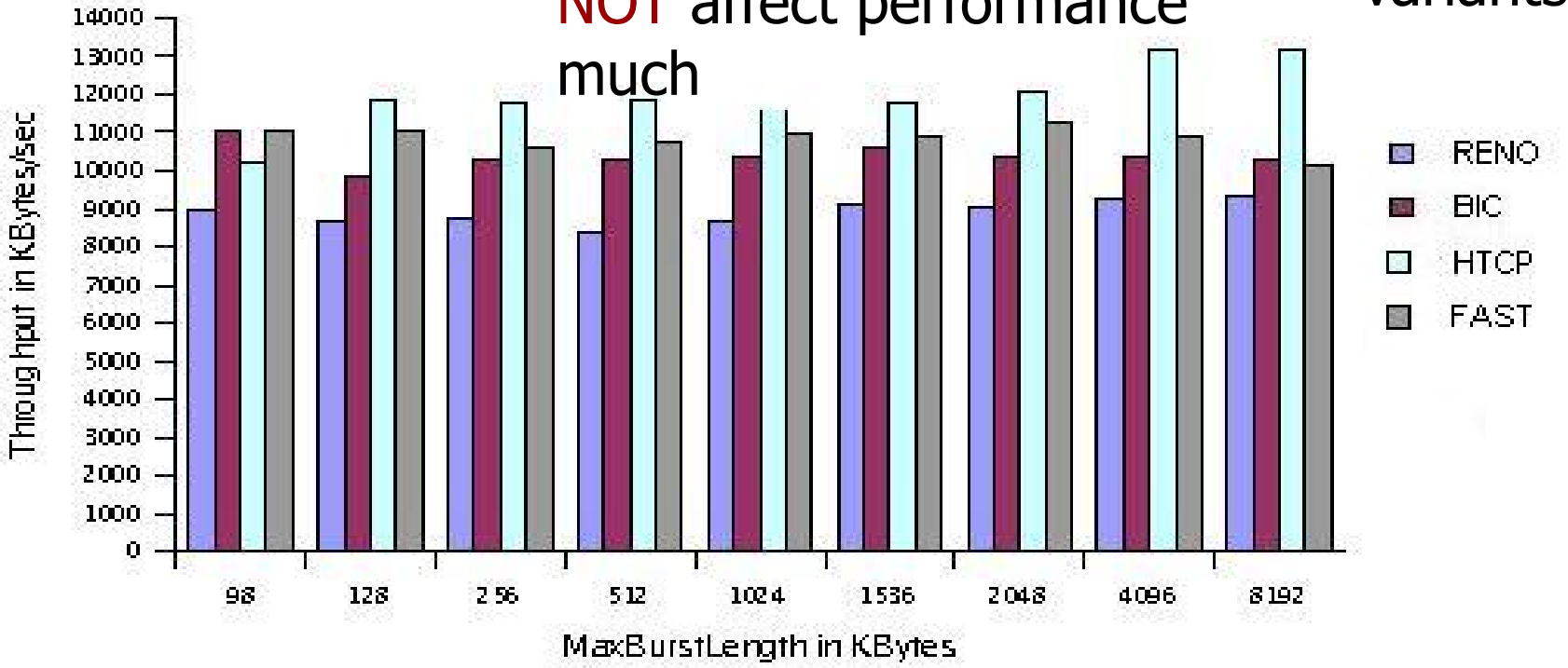
BIC and HTCP result in far more losses as compared to Reno



Effect of parameter MaxBurstLength

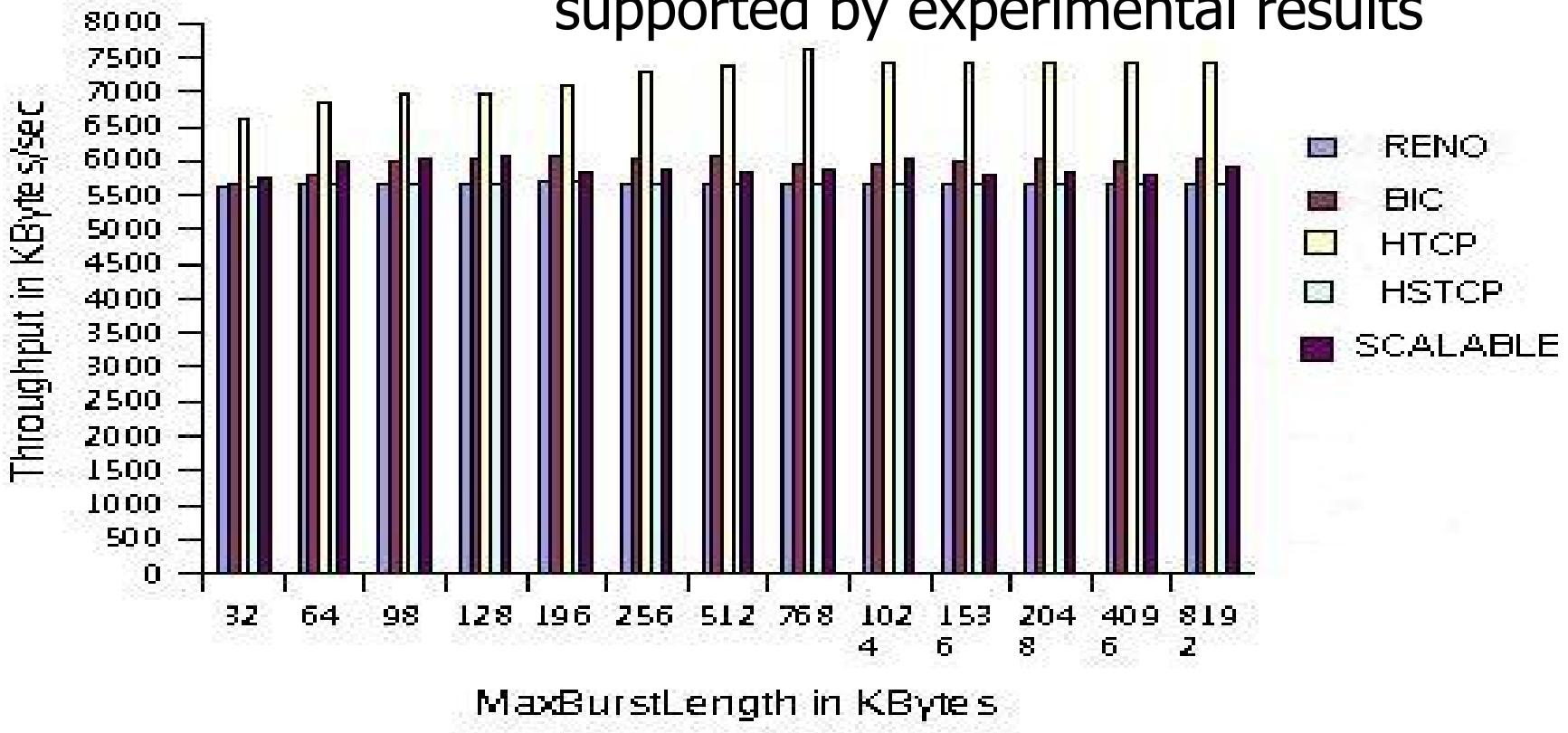
Parameter MaxBurstLength does NOT affect performance much

HTCP outperforms the other variants



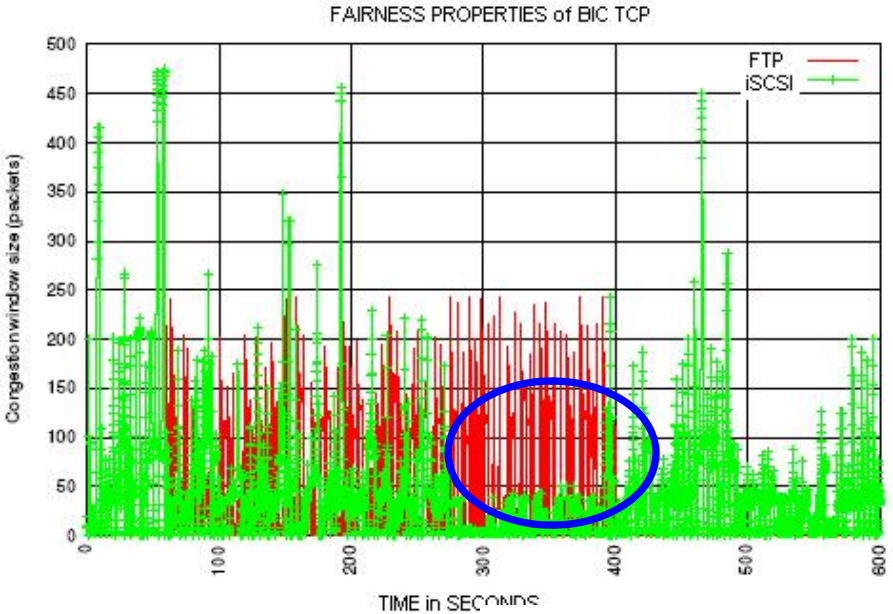
Effect of MaxBurstLength- using Simulations

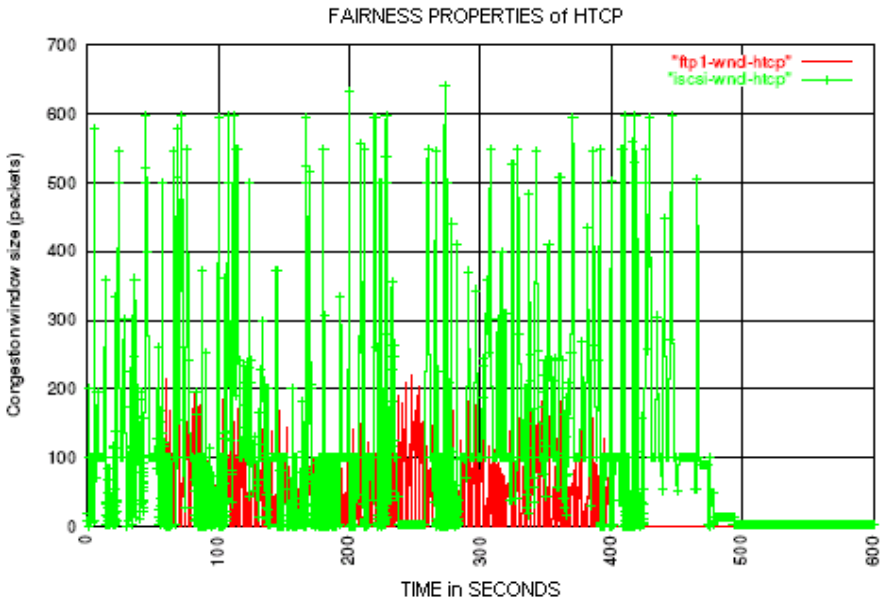
HTCP outperforms other TCP variants, as supported by experimental results

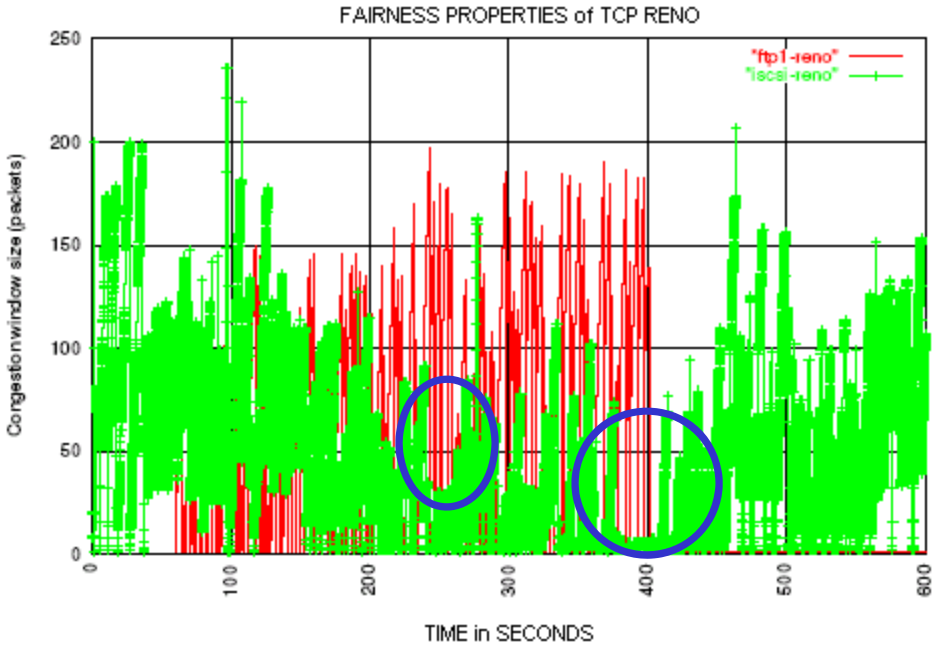


Fairness

- Two Flows are established
  - iSCSI Flow starts at 0 seconds and ends at 600 seconds
  - FTP flow begins at 60 seconds and ends at 400 seconds.
- Flows share bottleneck link of 100 Mbps


FAIRNESS PROPERTIES of BIC TCP
FTP
iSCSI
Congestion window size (packets)
TIME in SECONDS


FAIRNESS PROPERTIES of HTCP
"ftp1-wnd-htcp"
"iscsi-wnd-htcp"
Congestion window size (packets)
TIME in SECONDS


FAIRNESS PROPERTIES of TCP RENO
"ftp1-reno"
"iscsi-reno"
Congestion window size (packets)
TIME in SECONDS

BIC – Fairness

- Achieved Using Fast Convergence (FC)
- Applied when New Max is less than previous Max
- Fast Convergence
  - On Detecting Packet Loss,
    - FC_Max = (Max + Min)/2
    - FC_Target computed accordingly
    - Once FC_Max is Reached, use slow start technique.

Fast Convergence – Example

- Consider Flows F1 and F2 sharing link with $cwnd_{F1}$ = 128 and $cwnd_{F2}$ = 64 when they detect packet loss
- F1 : Min = 128/2 =64

  FC_Max = (128+64)/2 = 96 FC_Target = (96+64)/2 = 80

Switch to Slow start

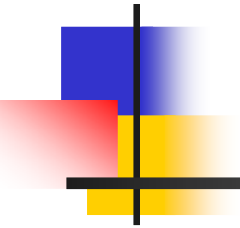
64 → 80 → 88 → 92 → 94 → 95 → 96 → 97 → 99 → 103 → 111

- F2 : Min = 64/2 =32

  FC_Max = (64+32)/2 = 48 FC_Target = (48+32)/2 = 40

32 → 40 → 44 → 46 → 47 → 48 → 49 → 51 → 55 → 63 → 79

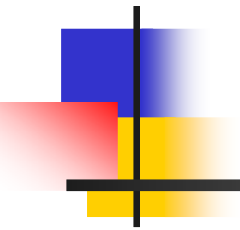
Switch to Slow start



Conclusions

Conclusions

- HTCP outperforms the other TCP variants considered.
- iSCSI parameter MaxBurstLength does not affect the performance much
- BIC TCP gives reasonably good performance and is fairer to other flows unlike HTCP.



Thank You ....