Storage Systems

NPTEL Course
Jan 2012
(Lecture 08)

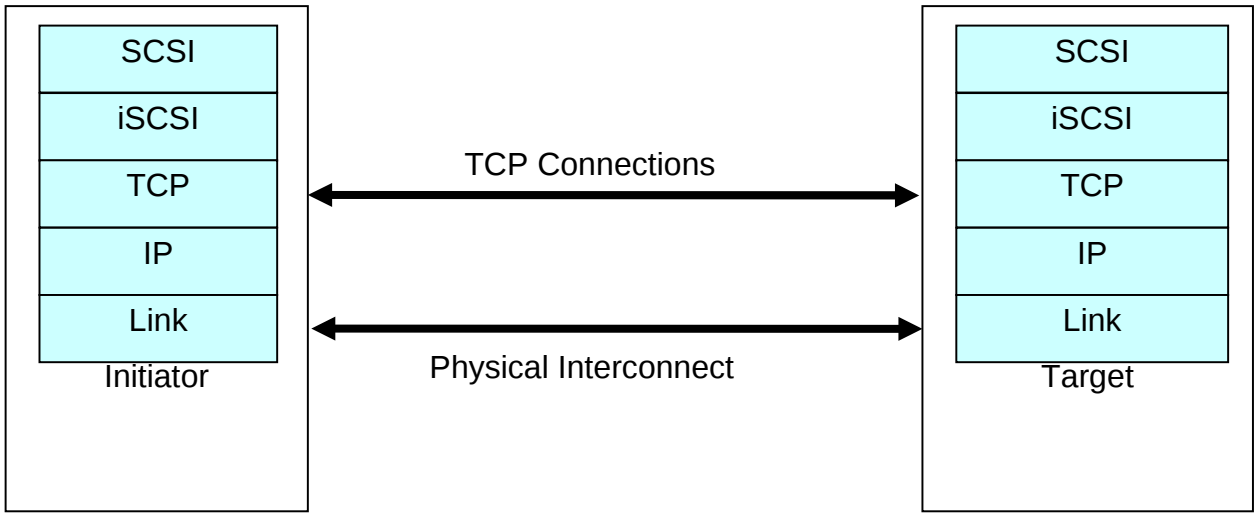
K. Gopinath
Indian Institute of Science

FC vs 10GEth

- Port cost of FC historically higher than Ethernet
- Many expect 10GEth to become dominant
  - Much lower costs
- Encapsulate FC frames in ethernet pkts
  - FCoE
  - FC based software does not change much
  - Increasing usage
- Or, drop FC completely and move to TCP as transport for SCSI

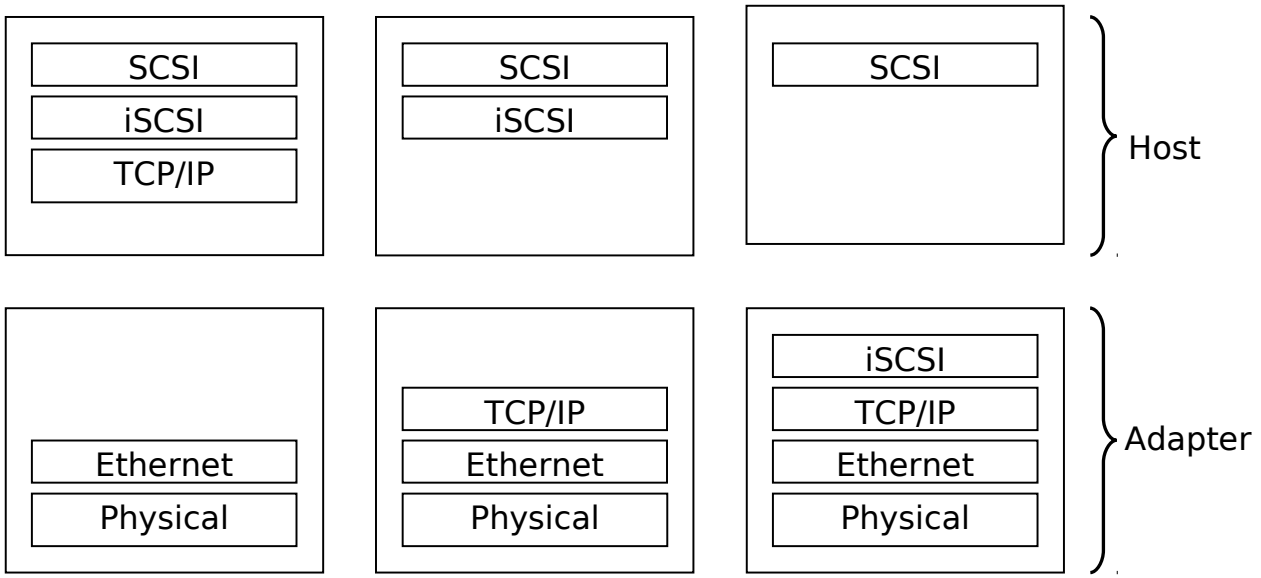
iSCSI (Internet SCSI)

- SCSI cmds encapsulated within a TCP connection
- Uses existing netw infrastructure for accessing storage
  - Only one network for storage and data
  - Rides on rapid growth of ethernet 1GEth/10GEth
    - Lower cost per port compared to FC
- Supports authentication protocols and IPSEC
- Responses may encounter differing delays



iSCSI Configurations

- Software iSCSI
- Software iSCSI with TOE (TCP Offload Engines)
- Hardware iSCSI

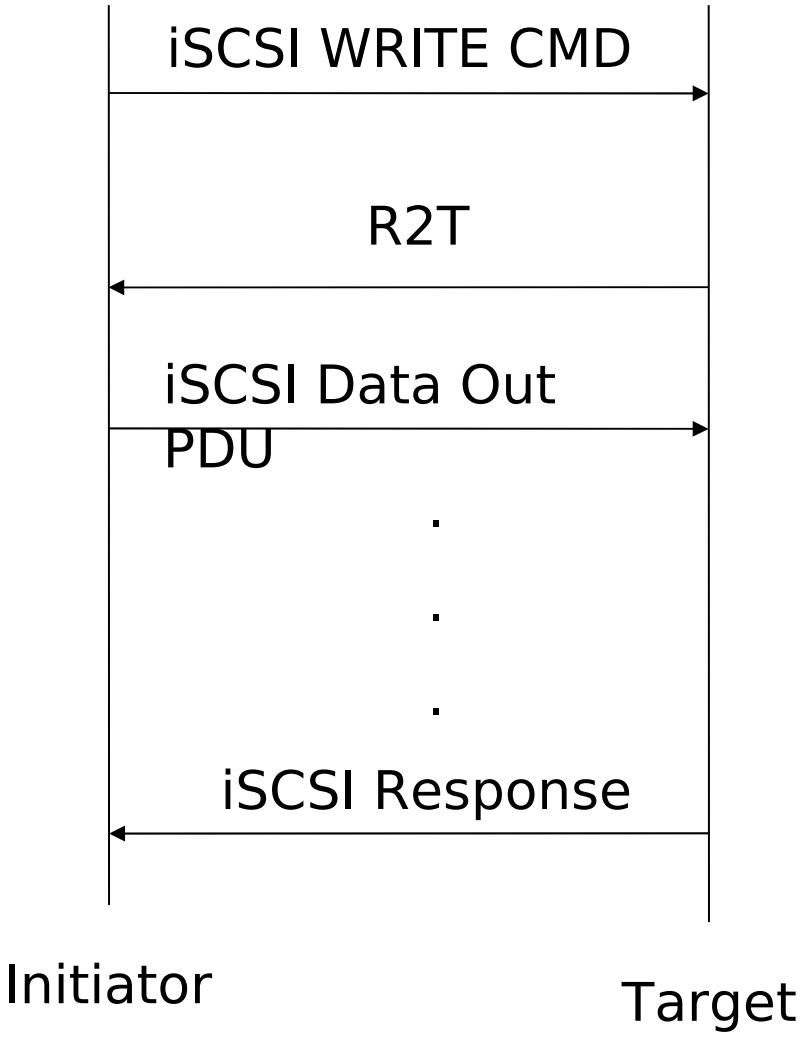


iSCSI

- Maps the SCSI block oriented storage data over TCP/IP
- Establishes an iSCSI session between SCSI Initiator and Target
  - session: group of TCP connections linking an initiator with a target identified by ConnectionID
- Supports ordered command delivery within a session
- iSCSI throughput governed by TCP Congestion Control Alg
  - End2End flow control
- An iSCSI parameter MaxBurstLength also determines maximum amount of data that can be sent out in one burst

iSCSI Phases

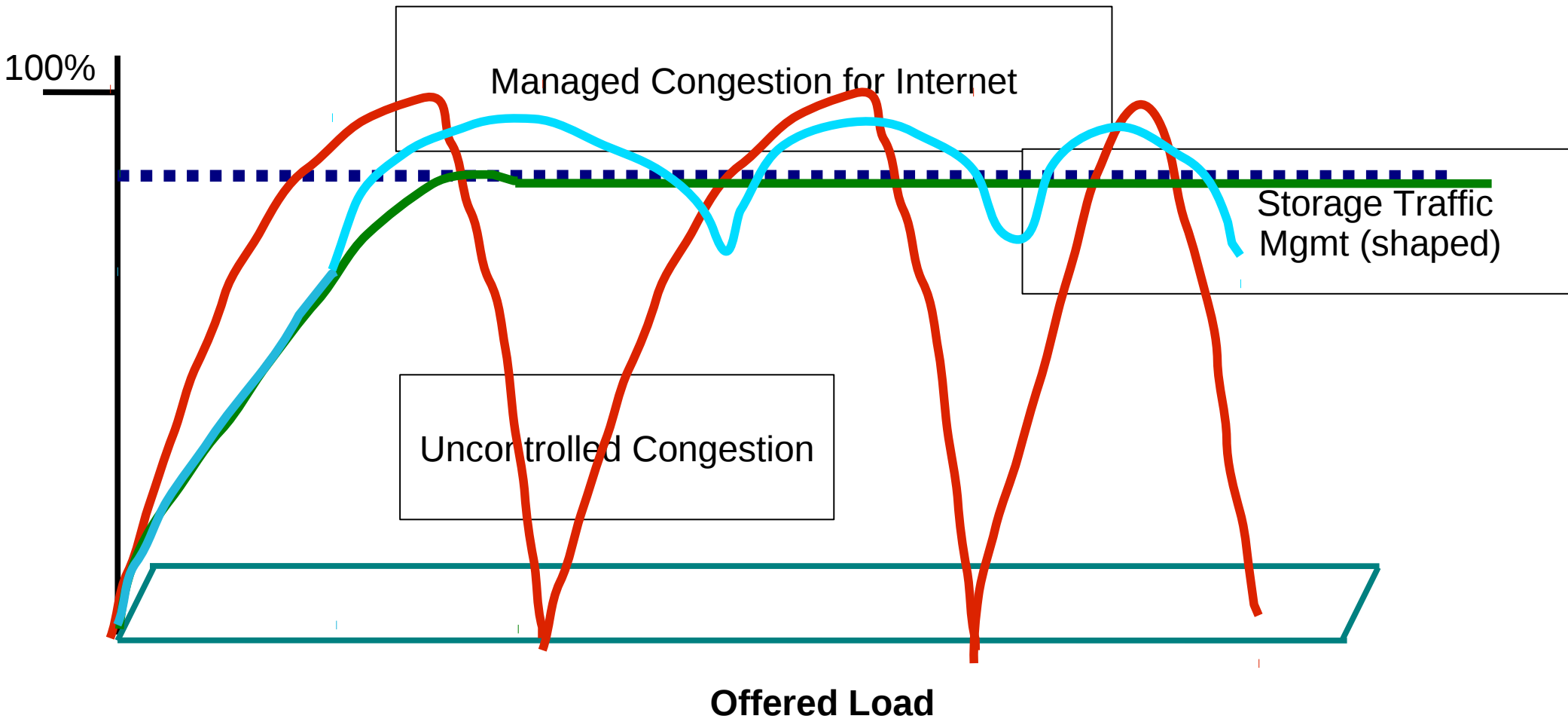
- Login Phase
  - Establish iSCSI session
  - Security Negotiation
  - Parameter Negotiation
- Full Feature Phase
  - Transfer of command and data



iSCSI Flow Control

- Depends on TCP
  - TCP uses sliding window for flow control
  - TCP also uses congestion avoidance alg
- Over High Speed Networks, TCP Reno underutilizes Network Bandwidth
- To overcome this drawback, variants of TCP Congestion Control Algorithm such as Scalable TCP, HTCP, BIC, CUBIC TCP developed
- Multiple TCP connections increase throughput
  - However, concurrent connections compete for bandwidth resulting in unfair sharing
    - On loss, all flows may reduce transmission
  - Congestion information needs to be shared among concurrent connections such as “Fair TCP”

Traffic Shaping or Rate Limiting Critical for iSCSI Throughput



TCP Background

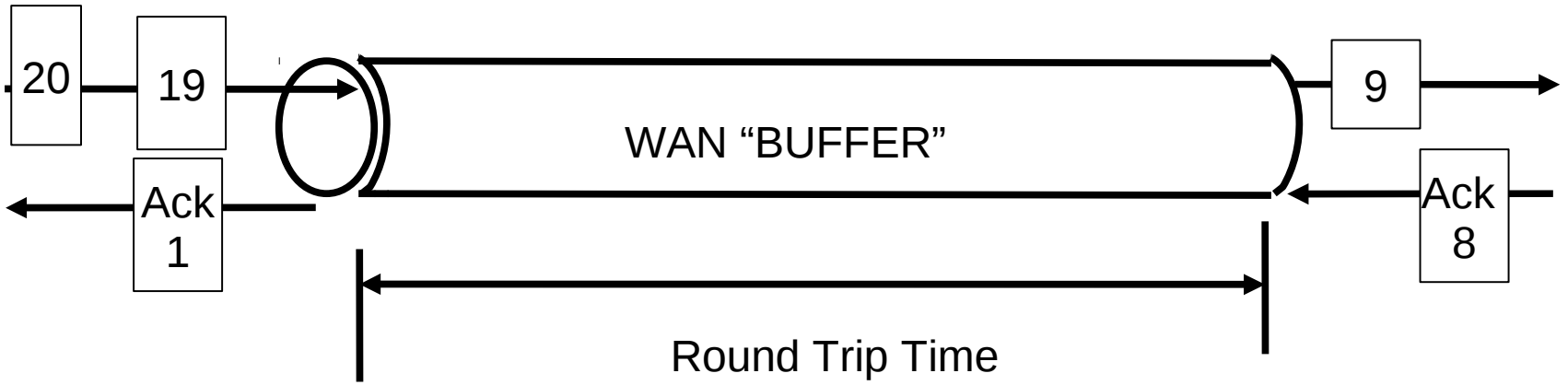
- Peer to peer protocol
  - Intelligent end devices and “dumb” network
- Has to estimate bandwidth avlbl
  - No setup before transmission
  - Control-theoretic model
    - Acks “clock” the protocol
    - Exponential increase till a threshold
    - Additive increase till loss, then multiplicative decrease
  - Losses indicate lack of capacity in network
    - Necessary for discovering capacity
  - For each connection, TCP maintains a congestion window, that limits total number of unack'ed packets that may be in transit end-to-end.

TCP Optimizations

"Self-clocking" protocols such as TCP need to keep pipeline full

- Many concurrent sessions
- Sending large data units (large TCP window size)

As number of sessions increases, buffer shifted from WAN to TCP sender's transmit buffers



WAN Buffer Size = RTT * bandwidth

TCP Throughput proportional to MSS/(RTT*sqrt(p))
where p: random pkt loss prob; MSS: max seg size

Achieving High TCP Performance

Large window sizes

Jumbo Frames

- Matches Ethernet frame size with TCP seg size

SACK: Selective Acknowledgements

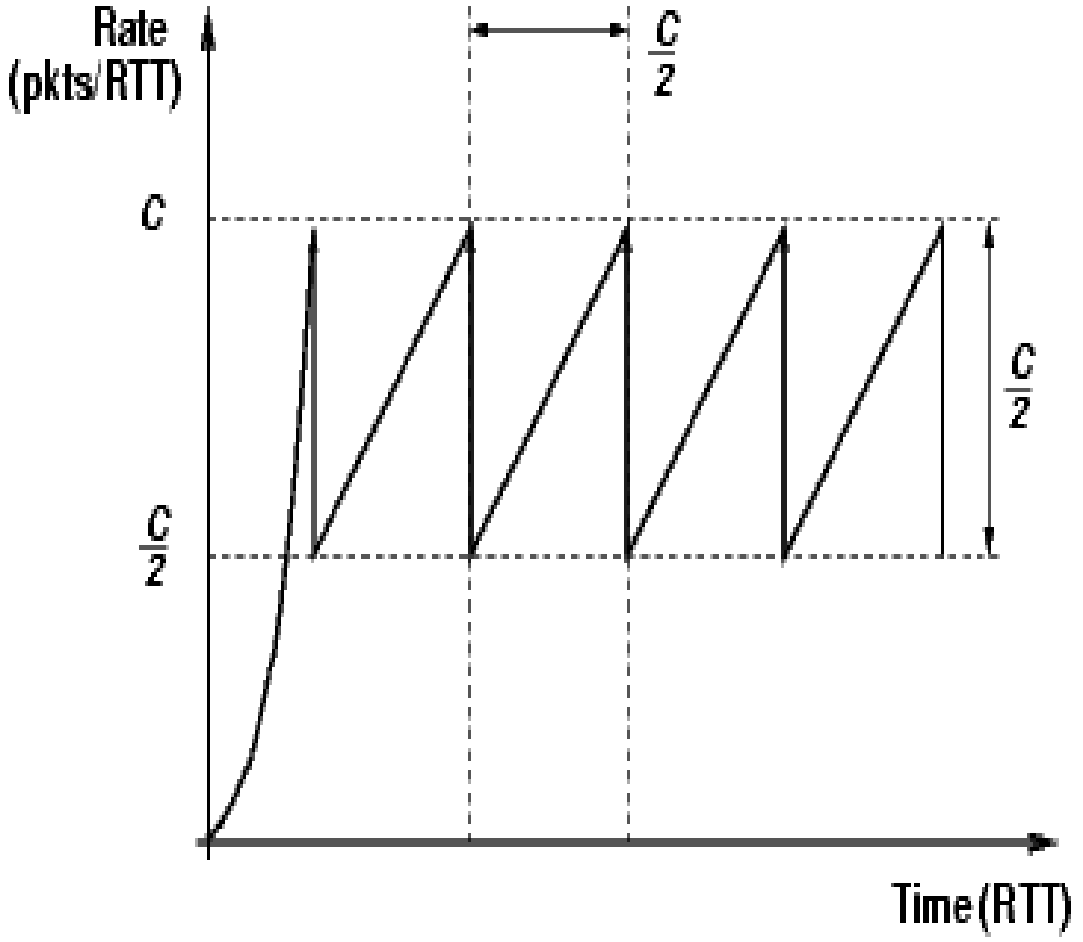
- Useful when medium is dropping many packets

Traffic shaping or rate limiting:

- Critical to avoid packet drops
  - Takes a long time to recover in high RTT env
- TCP retransmit mechanisms should not be relied upon for PERSISTENT packet drops

TCP Window Scaling and Seq wrapping: RFC 1323

TCP Reno



- Thruput determined by min of
  - Congestion window
  - Receiver window
  - Sender window
- Assume last 2 large

Over high bandwidth Links, changes in cwnd results in under utilization of link bandwidth.

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

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

HTCP

- Modification of TCP Reno
- Employs different alg to incr congestion window
  - Multiplicative Decrease Factor β adaptive
  - Also, Additive Increment α:

$\alpha = \alpha^L$ 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

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

Binary Increase Congestion (BIC) TCP

Modification of TCP Reno (in Linux kernel 2.6.8 -.18)

- Consists of two parts
  - Binary Search Increase
  - Additive 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.

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

Additive Increase

- If difference between current window and target large, then direct increase to target may stress 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}.

TCP CUBIC

- a less aggressive and more systematic derivative of BIC
- window size a cubic function of time since last loss event
- performs well in wired networks with large bandwidth-delay product.
- in Linux kernel since 2.6.19

RDMA

- RDMA over TCP: zero copy
  - Standardised as iWARP (Internet Wide Area RDMA Protocol
  - TCP Offload Engines (ToE): zero copy arch mostly proprietary
- iSCSI Extension for RDMA (iSER)
  - Eliminates TCP processing overhead from RDMA-capable NICs
- SCSI over RDMA (SRP)

Infiniband

- Proposed as system to system interconnect
- Widely used in HPC due to high speed
  - 40Gbps common
- Has multiple lanes in each connection with QoS
  - Also, failover
- Used as storage interconnect
  - Can use RDMA over Infiniband
- No std API in specification:
  - only a set of "verbs": functions that must exist.

Summary

- iSCSI becoming popular but 10GEth has yet not become widespread
  - FCoE evolutionary path
- RDMA stds critical
- Infiniband strong in HPC environments and being used for storage also