SNIA

STORAGE NETWORKING INDUSTRY ASSOCIATION

EDUCATION

TCP/IP Optimization for Wide Area Storage Networks

Dr. Joseph L White, Decru (NetApp)

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA.
- Member companies and individuals may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced without modification
  - The SNIA must be acknowledged as source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.

Abstract

TCP/IP Optimization for Wide Area Storage Networks

This session provides an overview of TCP/IP as it applies to block storage protocols (iFCP, FCIP, and iSCSI) and bulk data transfers. With all of the interest in using IP storage for servers, business continuance, and disaster recovery, the various IP storage protocols such as iSCSI, iFCP, and FCIP have gained increasing attention from storage networking professionals and administrators. TCP/IP provides the transport for these upper level protocols. Knowledge of TCP/IP and how it applies to block storage and bulk data transfers is critical for optimal deployment of IP storage solutions. This session will provide an overview of TCP/IP and its use in transporting block storage. The primary focus will be on the behavior of the TCP/IP layer itself. The session will review the effects of latency, packet drops, congestion, and high bandwidth links, as well as provide a discussion of block data transport optimizations and TCP improvements to mitigate these effects. The session will also cover common acceleration techniques for disk and tape operations over distance.

Agenda

- Block Storage across WANs overview
- TCP Details
  - TCP Connections
  - TCP Flow Control
  - Long Fat Networks
  - Retransmissions and Packet Drop Recovery
- TCP Modifications for Block Storage
  - Compression
  - Packet Drop Recovery Modifications
  - SACK
  - Network Reordering
- SCSI Level Optimizations
  - Write Acceleration
  - Tape Pipelining

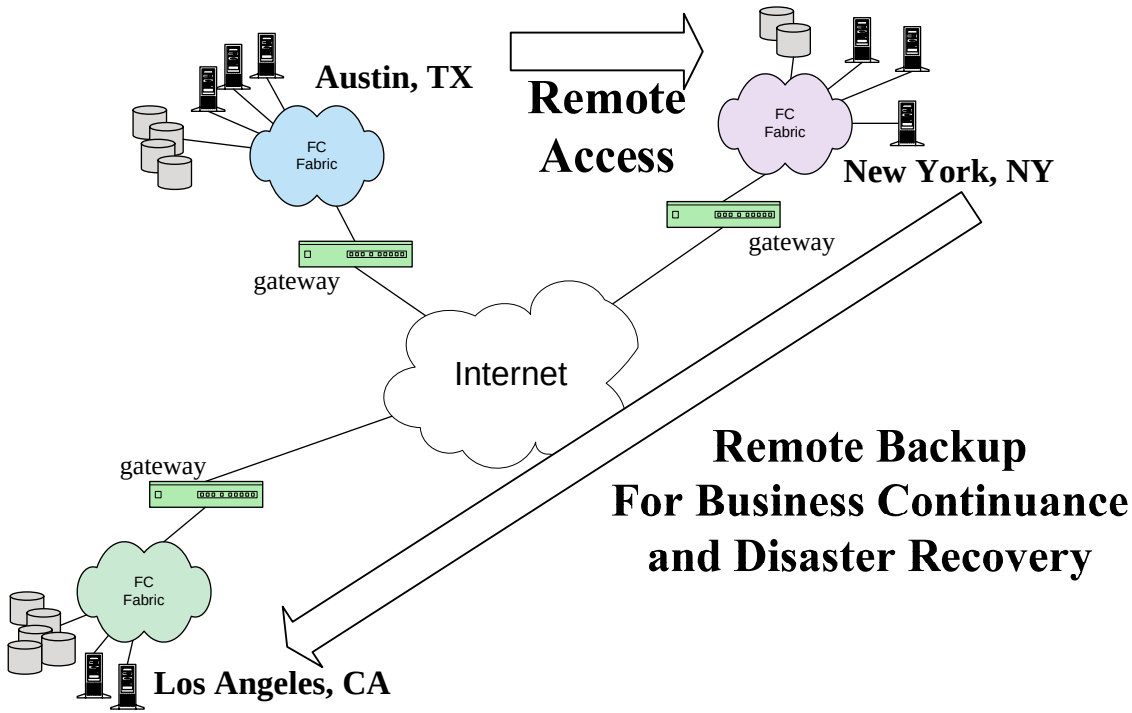
Characteristics of Storage Traffic

- Demanding Requirements
  - High Throughput
  - Low Latency
  - Wide Scalability
  - Robustness
  - High Availability
- Technology pushed by mission critical applications
  - Deployments have matured into multi-Terabit networks
  - Critical to the Data Center
- Must meet these requirements across wide area networks

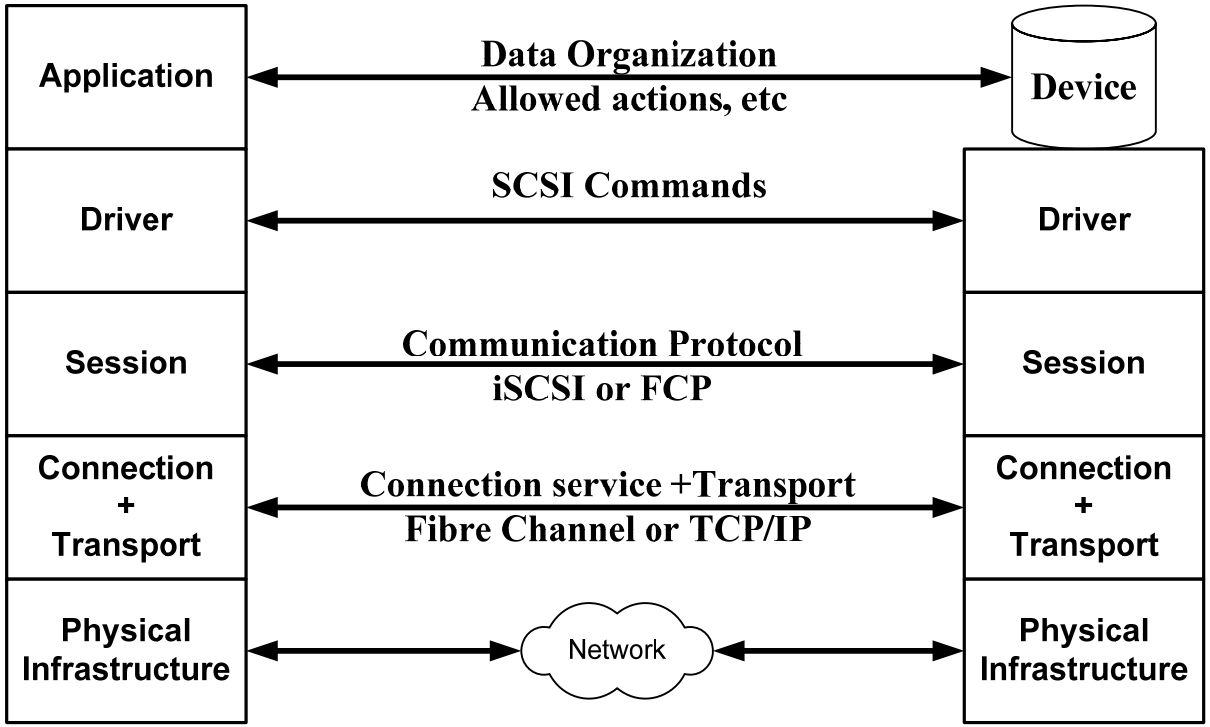
TCP Based Storage Protocols

- Block Storage Protocols based on TCP/IP
  - FCIP
  - iFCP
  - iSCSI
- WAFS
  - makes use of many of the same techniques
- TCP WAN Acceleration
  - makes use of many of the same techniques
- There do exist other ways to interconnect FC Fabrics:
  - Direct Optical Connection
  - “Channel Extension”

Distributed Storage Area Network



Communication Layers



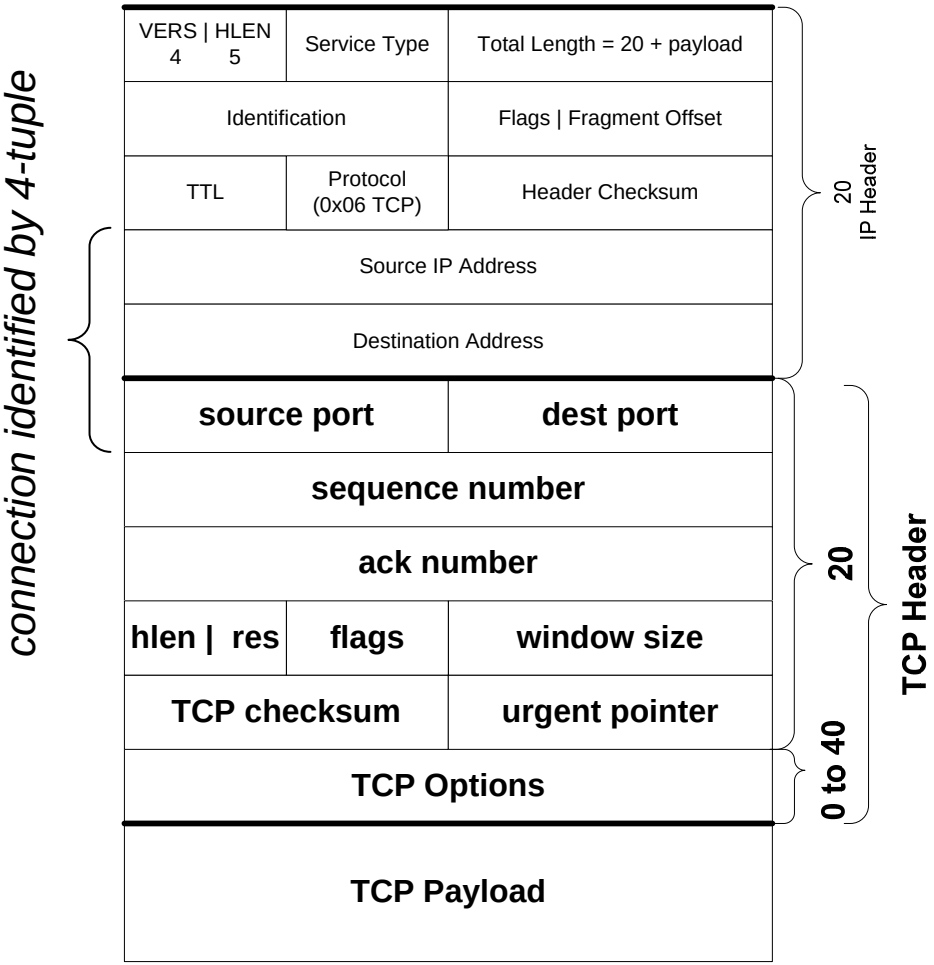
- Optimizations can occur at any of the layers
  - Reduce application chattiness (e.g. 'WAFS' accelerators)
  - Optimize session upper level protocol (e.g. write command acceleration)
  - Improve Transport behavior (e.g. TCP optimizations)
- Value of optimizations can depend upon latency and bandwidth available
- Gateway devices are often used to intercept and optimize traffic

Characteristics of TCP

- Connection Oriented Protocol
  - Full Duplex - Byte Stream
  - Port Numbers allow multiple connections between an IP address pair
  - Certain features negotiated at connection initialization
- Guaranteed In-Order Delivery
  - Segments carry sequence and acknowledgement information
  - Sender keeps all transmitted data until acknowledged by the receiver
  - Sender maintains send timers and attempts to resend any missing bytes in the sequence
- Flow Control and Congestion Avoidance
  - Sender maintains sending limit as a function of trips across the network and as a function of congestion events
  - Receiver maintains a sliding window that is advertised back to the sender

TCP Header

- Source Port Number
- Destination Port Number
- Sequence Number
- ACK Number
- Header Length
- Flags
  - SYN
  - FIN
  - RST
  - ACK
  - PSH
  - URG
- Window Size
- Checksum
- Urgent pointer

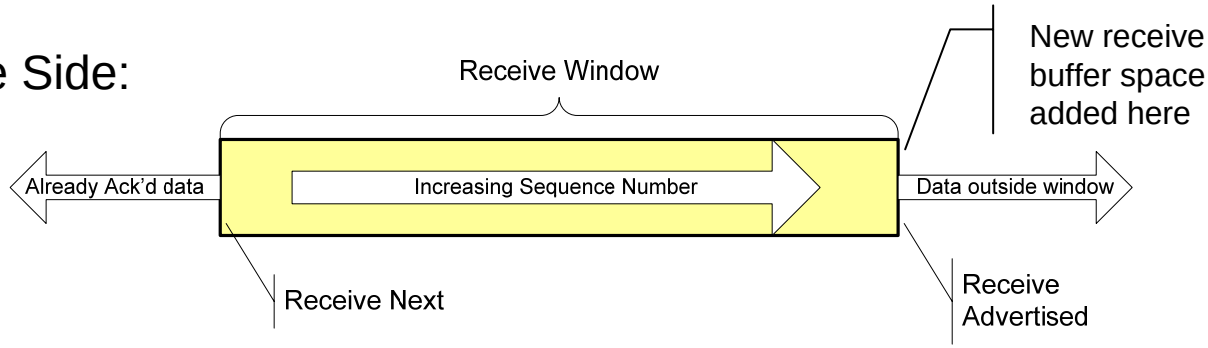


TCP Connections

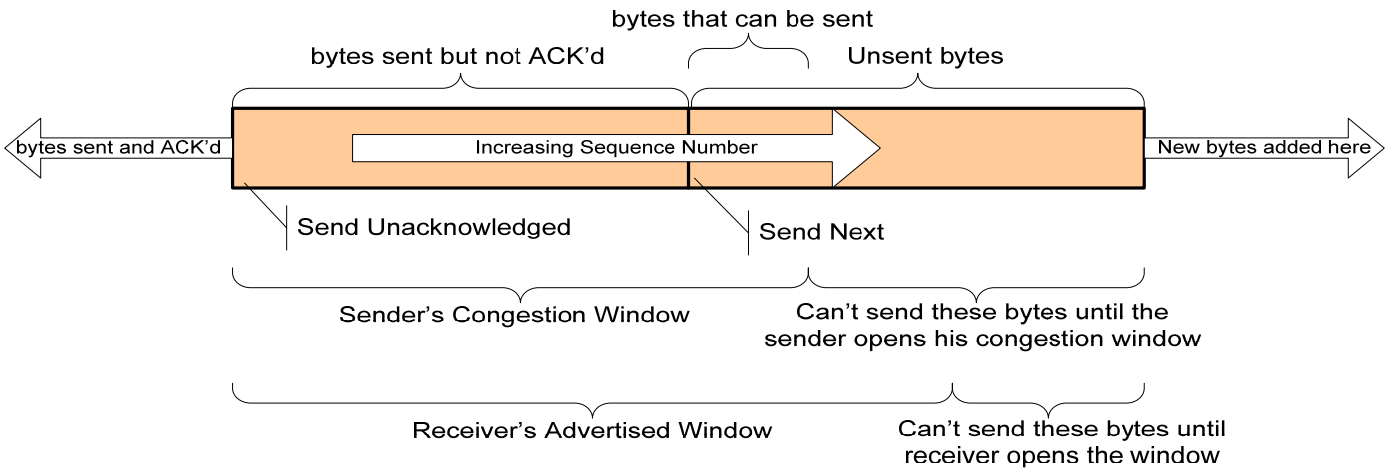
- Connection Setup
  - Peer to Peer
  - SRC_IP, DST_IP, SRC_PORT, DST_PORT
    - 4-Tuple is Unique identifier for the connection
    - Assignment of port numbers: well known vs. allocated
  - Triple handshake
    - SYN : SYN-ACK : ACK
    - TCP State Machine handles simultaneous open collisions
  - Operating Parameter Negotiation
    - Initial Sequence Number
    - TCP Options
- Connection Closure
  - Use FIN to indicate sender done transmitting data
    - TCP Half Closed connection
    - A FIN is acknowledged
  - Use RST to reset the connection and force it closed
  - TCP State Machine has close wait times built into it
- Keep-alive Timer
  - detects idle connections (2hours…)
- MTU Discovery

TCP Sliding Window and Flow Control

- Receive Side:



- Transmit (Sender) Side:



TCP Receiver Flow Control

- Sliding window protocol
  - Receiver advertises a window to the sender (rwnd)
  - Out of Order reassembly
  - Duplicate or out of window detection
  - Avoiding silly window syndrome
    - Smallest advertise size is minimum of one segment or ½ actual receive window
    - Advertise zero otherwise
  - Persist timer maintained by sender
    - Generates a window probe
    - If window is zero for 5-60 seconds, send 1 byte to get the other side to send back an ACK with a window update.
      - Follows same back-off scheme as the retransmission timer

TCP Sender Basic Flow Control

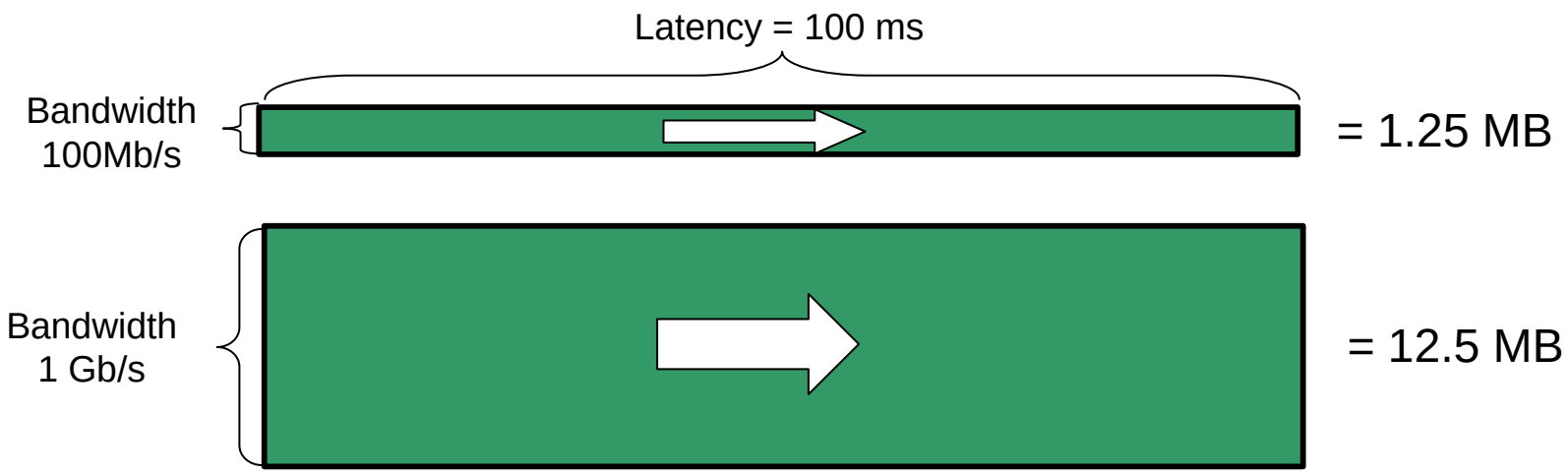
- Data Stream chopped into chunks (segments)
  - Sent as IP Datagrams
  - Segments protected by TCP checksum
- Sender maintains a Congestion Window (cwnd)
  - Highest sequence number that can be sent is last ACK + MIN (cwnd, rwnd)
- Slow Start
  - Rate of packet injection into the network is equal to rate at which ACKs are received.
  - Leads to exponential sender cwnd ramp until:
    - A congestion event occurs
    - The limit of the receiver's advertised window is reached
    - The limit of the sender's un-acknowledged data buffering is reached
    - The limit of the network to send data is reached (network saturation)
- Nagle Algorithm (RFC 896)
  - don't send less than one segment of data unless all sent data has been ACK'd
  - SCSI storage specific modifications:
    - Always push data when FCP End of Sequence encountered
    - Can instead push each FC/FCP frame (typically 2 segments per data frame)

TCP Acknowledgements

- ACK number is next expected sequence number
  - This is the last sequence number received + 1
- Segments must be in order to advance ACK Number
  - SACK allows additional information to be sent
- Duplicate ACK
  - Sent when segment is received out of order
  - May indicate a missing segment to the sender
- Delayed ACK
  - Do not send an ACK right away, wait a short time to see if Additional segments arrive which can also be ACK'd
- ACK value valid in a Data Segment
  - If no data, the segment is often called a 'Pure ACK'
- ACK/N (*one possible modification*)
  - Wait for several (N) segments to arrive before sending an ACK
  - Send anyway after a short time interval

Long Fat Networks (LFN)

- **LFNs are characterized by a large bandwidth-delay product**
  - Due to this, they place more data buffering demands on the TCP connection than Local or Metro Networks
- **The bandwidth-delay product is the amount of data that the network itself can hold from the sender to the receiver**
  - As the speed of the network goes up, more data is needed to fill the same latency



LFNs and the TCP Buffering

- **TCP Receive Window and Transmit Buffering limit LFN transfer rates**
  - On the receive side the buffering must be allocated to the connection
  - On the transmit side the data must be held until ACK'd
  - TCP can only move one window's worth of data per RTT
    - Data segment must travel to the receiver
    - ACK for that segment must travel back
  - For 1Gb Ethernet
    - Full rate GE gives about 125 MB/s throughput
    - 1 ms RTT requires 125 KB of buffering
    - 1 ms RTT is 100 Km separation
    - Trans-continental distances require several MB of buffering
      - 60 ms RTT requires about 8 MB, enough for traversing North America
    - As speeds decrease, less buffering is required for the same distance

TCP Window Scaling

- Basic limits
  - TCP header contains 16-bit window size in units of bytes
  - Allows a 64KB-1 maximum window size

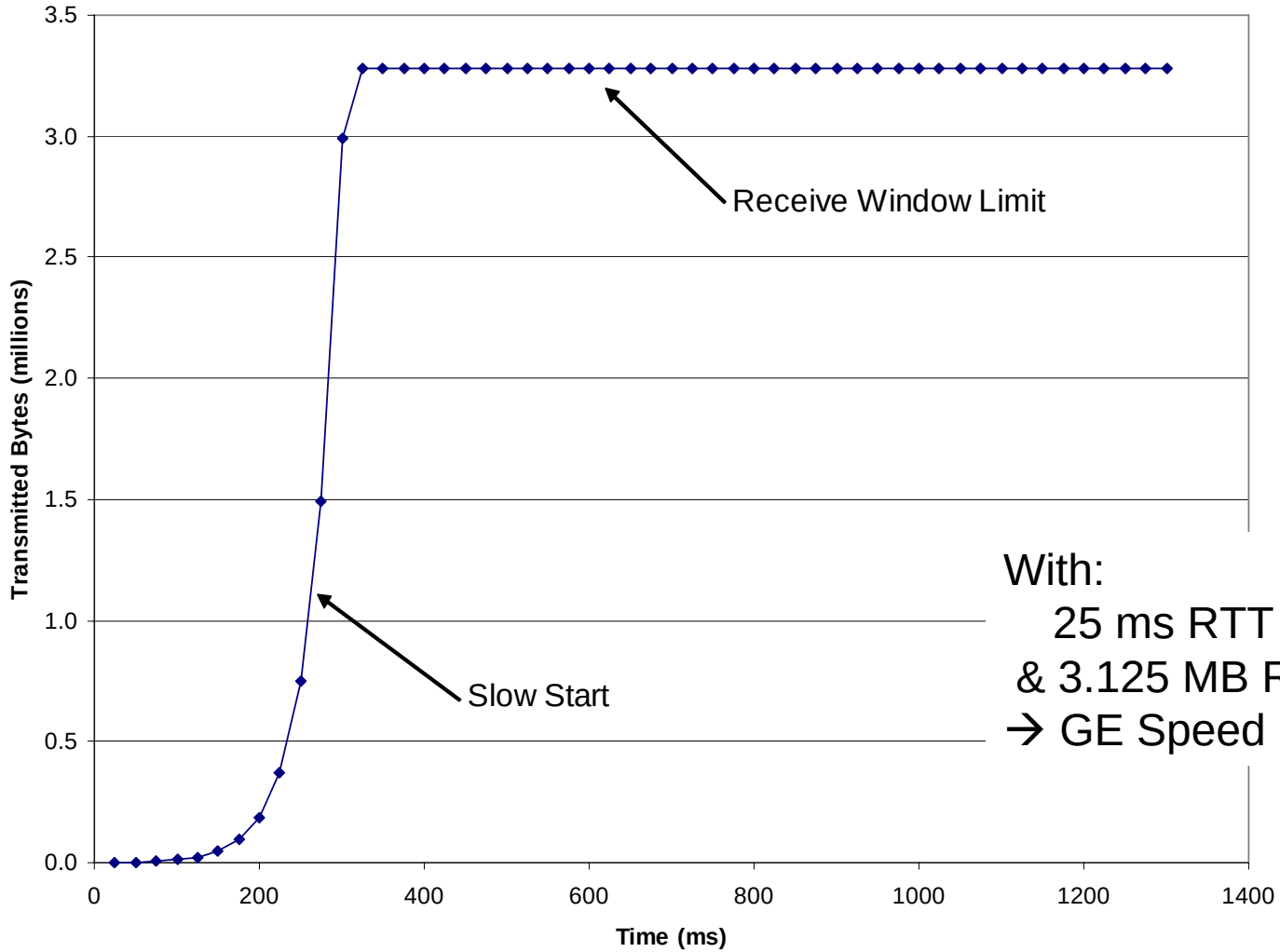
- Scaled Window TCP Option
  - During connection setup a scale factor is negotiated in each direction
    - The option is carried in the SYN segments
  - Each ‘count’ in the advertised window field is worth $2^{scaleFactor}$ bytes
    - For example if the scale factor were 10, the window field would carry the KBs of available window
    - The resolution for the window advertisement is reduced but in practice this is not important since the scale factor does not have to be very large to reach a significant maximum window advertised. The same scale factor of 10 allows a 64 MB window.

Timestamp and PAWS

- Timestamp Option
  - enables PAWS
  - allows RTT calculation on each ACK instead of once per window
  - produces better smoothed averages
  - Timer rate guidelines: 1 ms <= period <= 1 second

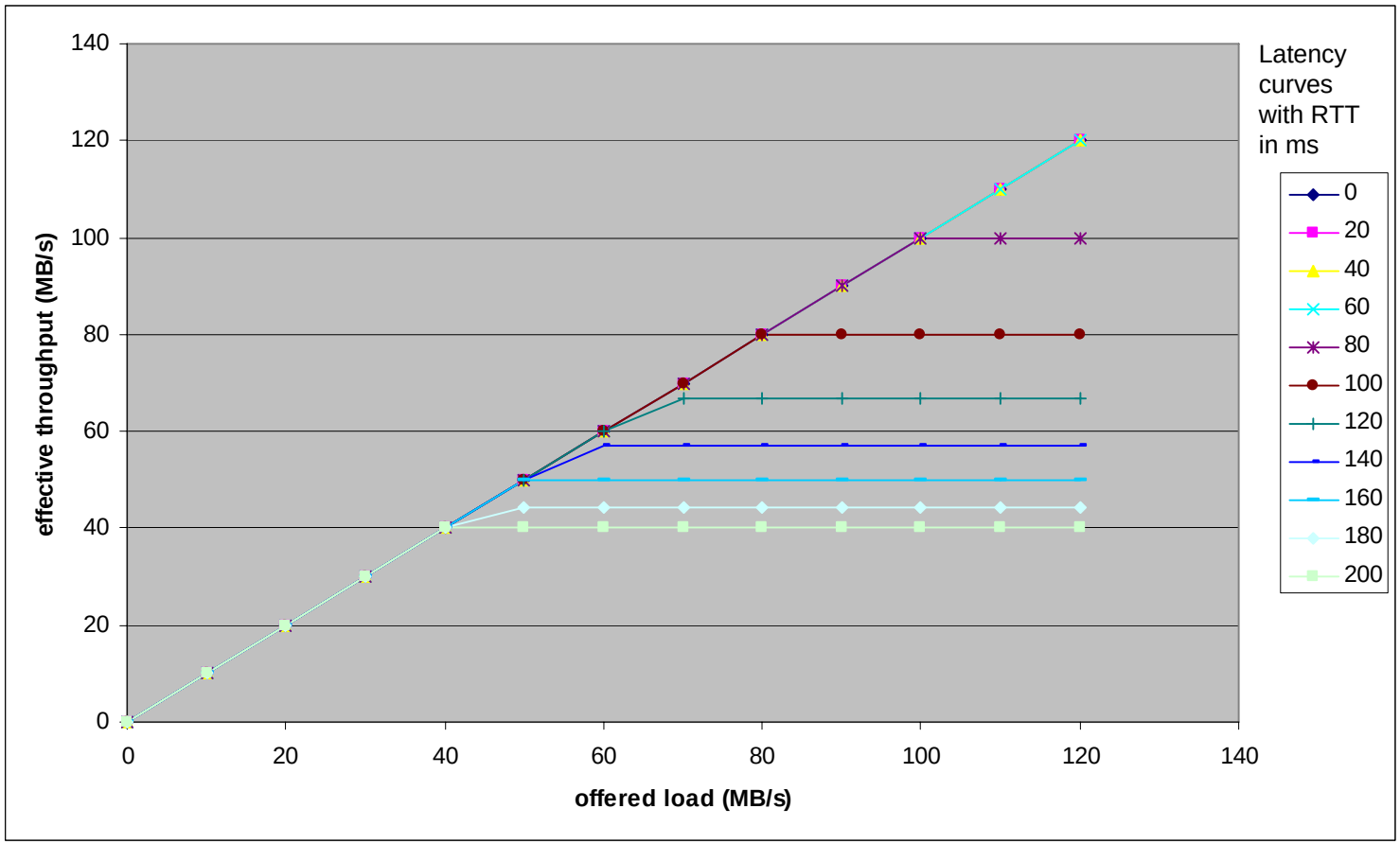
- PAWS: protection against wrapped sequences
  - In very fast networks where data can be held, protects against old sequence numbers accidentally appearing as though they are in the receiver's valid window
  - uses timestamp as 32-bit extension to the sequence number
  - requires that the timestamp increment at least once per window

Receive Window Plateau



With:
25 ms RTT
& 3.125 MB Receive window
→ GE Speed is reached

Effect of TCP Receive Window Size on Maximum Throughput

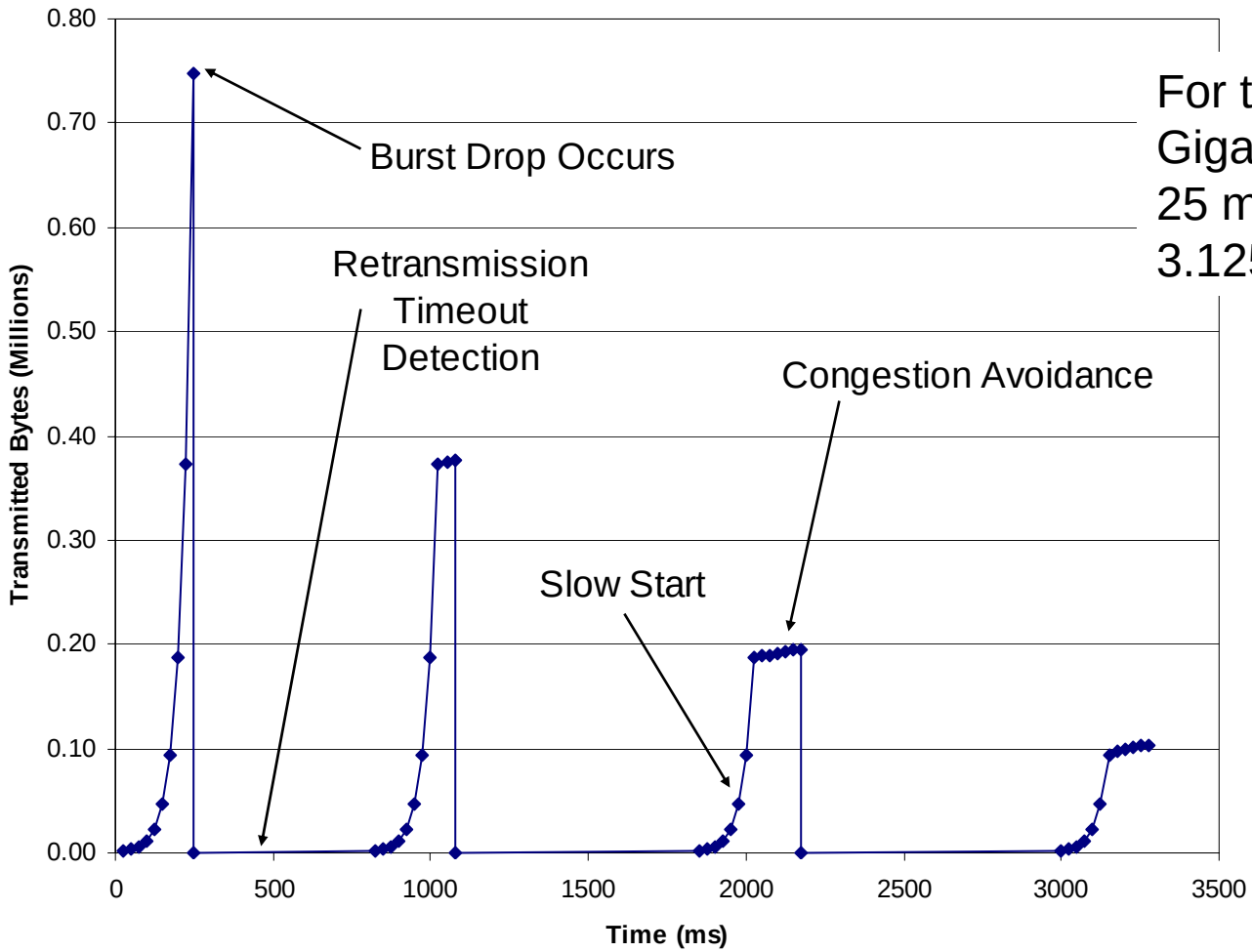


Steady state TCP Throughput for an 8 MB Receive Window at Various Latencies across a GE network

Effect of Packet Loss

- Packet (segment) loss can occur for several reasons
  - Congestion
  - Ethernet/IP proactive flow control schemes (RED)
  - Faulty equipment
  - Uncorrectable bit errors
- When packet loss does occur it can be extremely detrimental to the throughput of the TCP connection
  - Extent of disruption determined by the pattern of drops
  - There are TCP features and techniques which mitigate effects of packet loss
- The effects of packet loss and the resultant TCP behavior is discussed in the following slides

TCP Retransmission Timeouts



For this example:
Gigabit Ethernet Speed
25 ms RTT
3.125 MB Receive window

Retransmission Timer

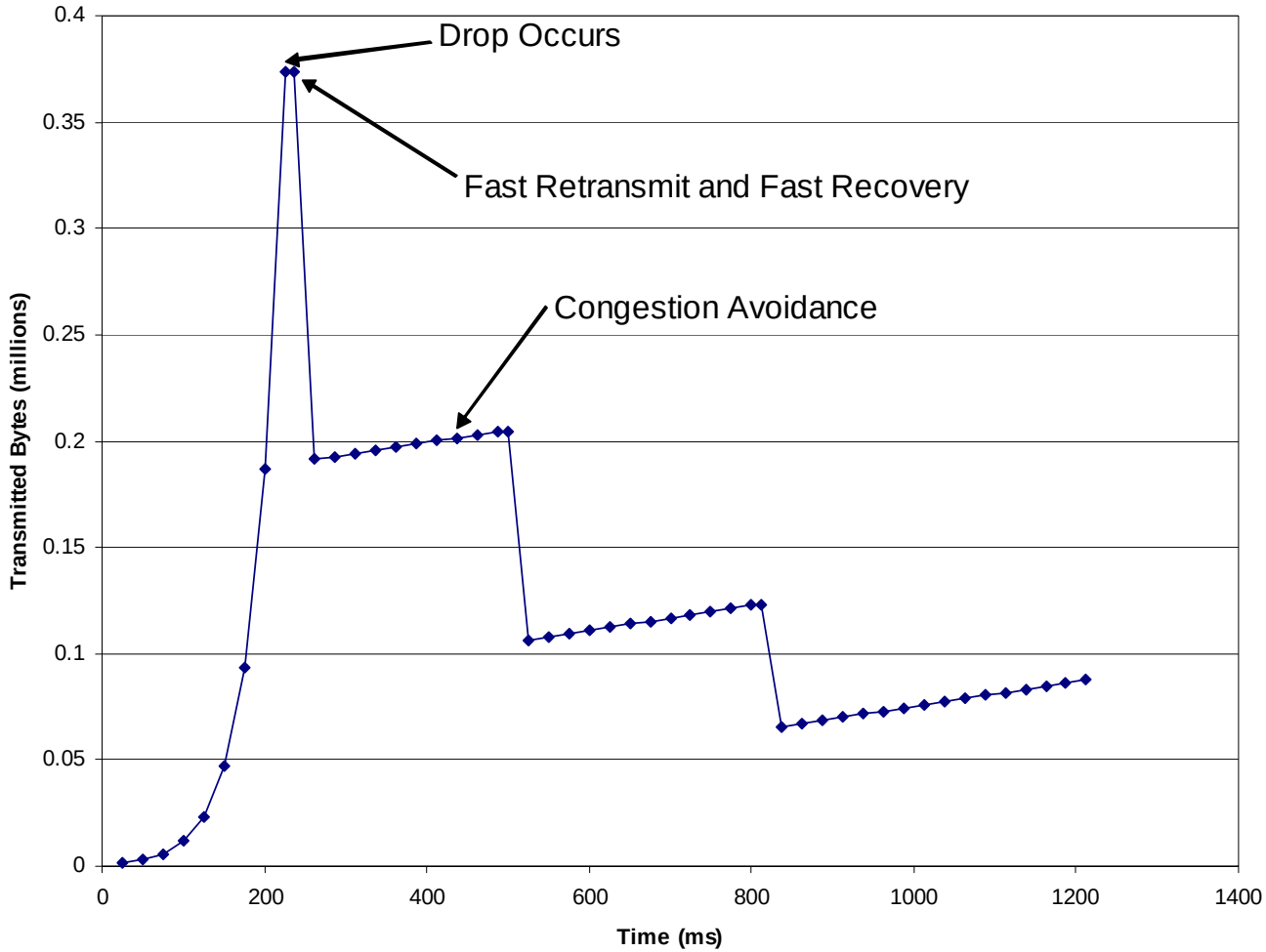
- Retransmission timer:
  - times oldest sent, unacknowledged data
  - cancelled and restarted on non-duplicate ACK
  - used to detect packet loss
- requires RTT estimation for connection
  - smoothed estimator updated once per round trip
  - timestamp option allows RTT estimate for each segment
  - RTT estimate used to set the timeout expiration value
  - Karn's algorithm
    - cancel RTT estimation during retransmission timeout
  - Can use better resolution timer (1ms, 10 ms) than normal TCP implementation's 500ms
- Multiple Retransmissions and connection closure
  - After some number of retransmission attempts (usually 13) the TCP connection is closed
  - Usually the timeout value increases with successive timeouts
  - Many different patterns used in the various TCP stacks
    - Some close in a few seconds, Some in several minutes

Congestion Avoidance

- Congestion event
  - .. is any dropped or reordered frames resulting in
    - fast recovery
    - Retransmission timeout
      - often referred to as going back to slow start

- Once in congestion avoidance the sender will ramp linearly above the most recent fast recovery cwnd threshold (slow start threshold)
  - Increment the sender's cwnd by 1/cwnd per ACK instead of by 1 segment per ACK
  - segment size/8 often used instead of 1/cwnd as approximation
  - below this threshold still exponential growth of the sender's cwnd

Fast Retransmit, Fast Recovery



Fast Retransmit, Fast Recovery

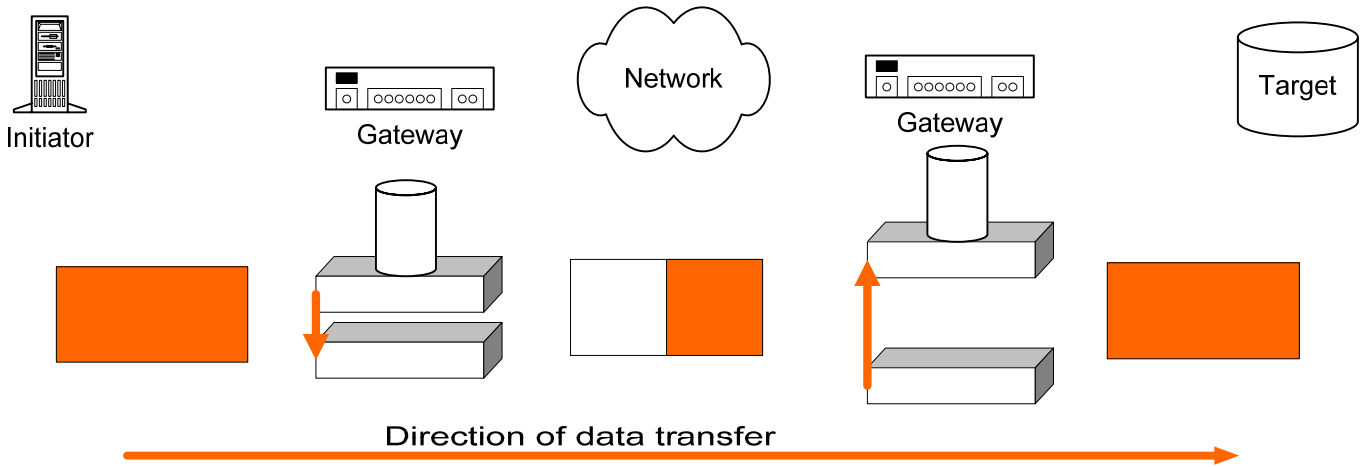
- Dropped frames detected by looking for duplicate ACKs
- On 3 duplicate ACKs received by the sending TCP:
  - **Fast Retransmit**
    - Send the oldest unacknowledged segment again
    - Does not stop data transmission for new data
  - **Fast Recovery**
    - The cwnd is reduced to ½ its value + 3 segments
    - ½ the cwnd value is marked as the slow start threshold
    - Does not stop segment sending
    - Once new data acknowledged
      - Pull the sender's cwnd back to the slow start threshold
    - If new data not acknowledged then a retransmission timeout will eventually occur

- *Networks which reorder extensively can cause TCP to react even though no packets are lost!*

TCP/IP for Block Storage

- Must use scaled receive windows
- Implement Selective Acknowledgement (SACK)
- Detect retransmission timeouts faster
  - and with less back-off for successive timeouts
- Modify Congestion Controls
  - Different ramp up or starting value for slow start (aka QuickStart)
  - Less reduction during fast recovery
  - Ignore or reduce the effects of congestion avoidance
- Modify Fast retransmit and Fast Recovery detection scheme
- Reduce the amount of data transferred
  - Compression (Also used by TCP WAN Accelerators)
- BUT, Don’t want to break TCP’s fundamental congestion behavior

Compression



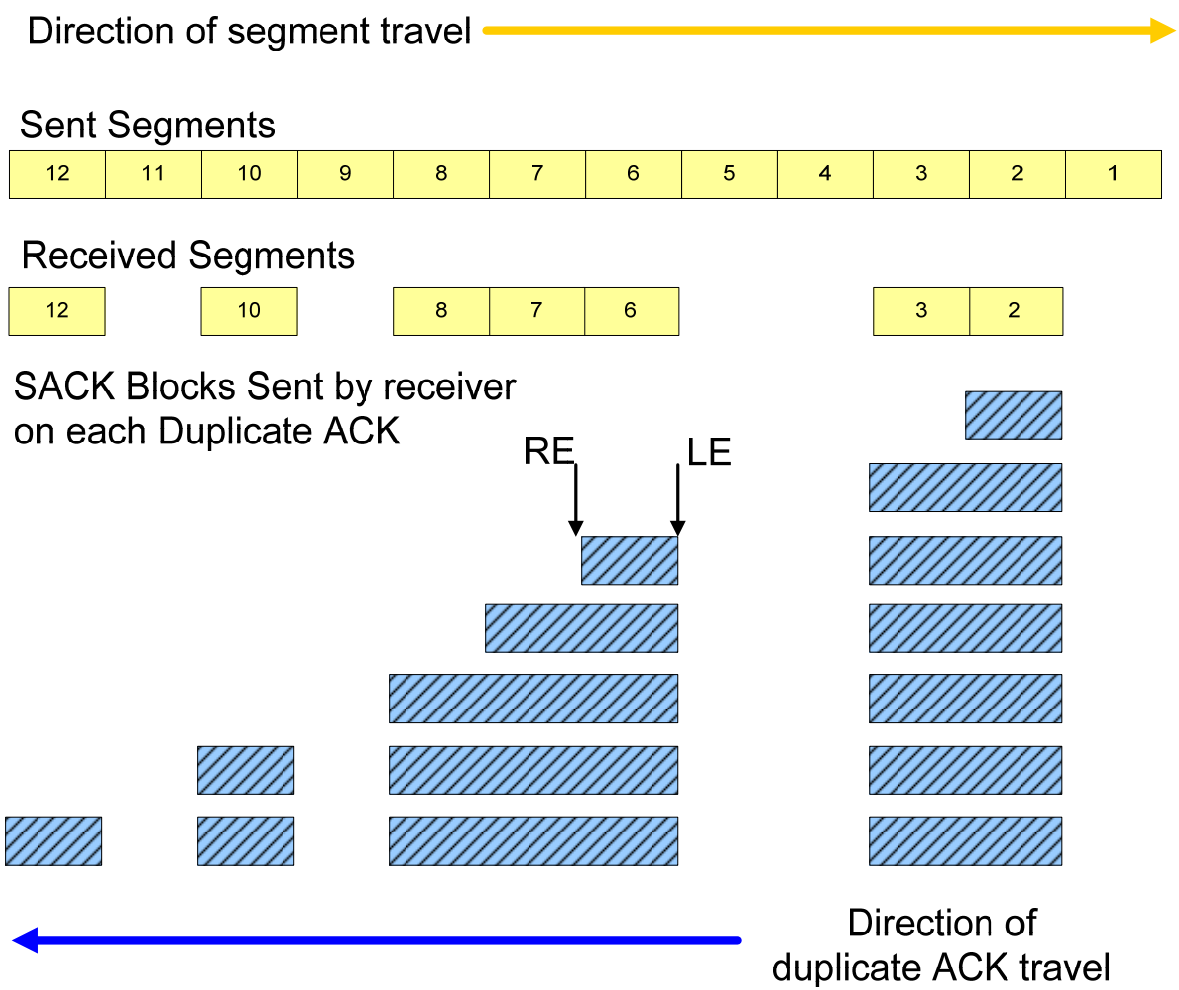
- Compression Ratio
  - The size of the incoming data divided by the outgoing data
  - Determined by the data pattern and algorithm
  - History buffers help the compression ratio since they retain more data for potential matches
- Compression Rate
  - How fast the incoming data is processed
  - Depending upon implementation, the algorithm may affect the rate

Selective Acknowledgments

- Generally called SACK

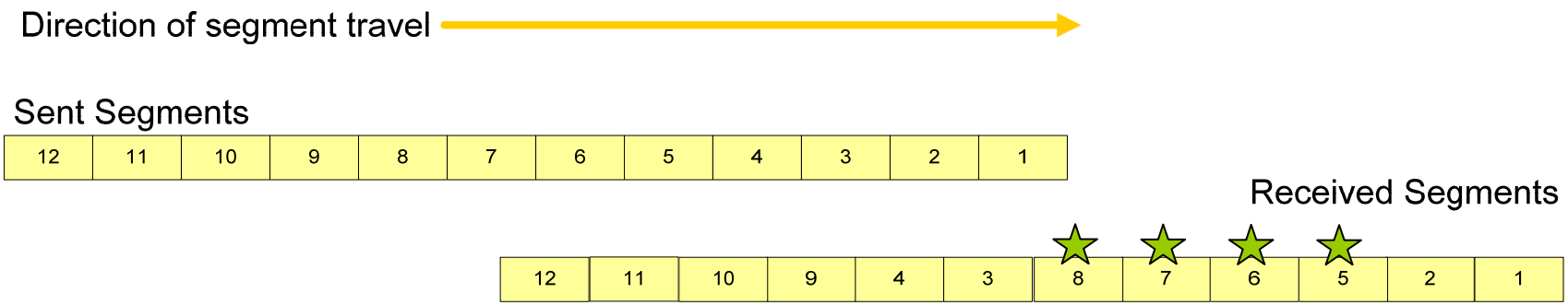
- SACK Blocks
  - TCP option carries 1 to 4 Left Edge (LE) – Right Edge (RE) Pairs
  - For each block
    - All the bytes from the LE to the RE -1 have been received
    - The same SACK block may received repeatedly
- Receiver
  - Maintains list of most recently received contiguous blocks
  - Whenever Out Of Order received,
    - send dup ACK as normal
    - Plus 3-4 most recent SACK blocks
- Sender
  - Maintains map of acknowledged contiguous blocks
  - Retransmit un-ACK'd data blocks to try and fill in multi-segment holes

SACK Illustration

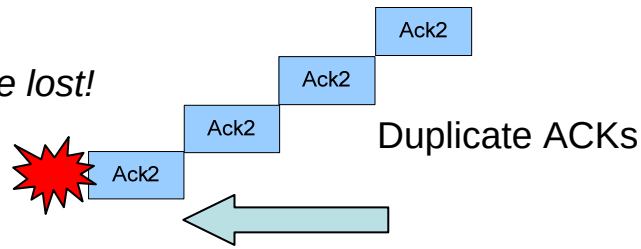


Network Reordering

- Networks which dynamically load balance cause excessive false congestion events due to extensive reordering and TCP normally only uses a value of 3 for the duplicate ACK threshold

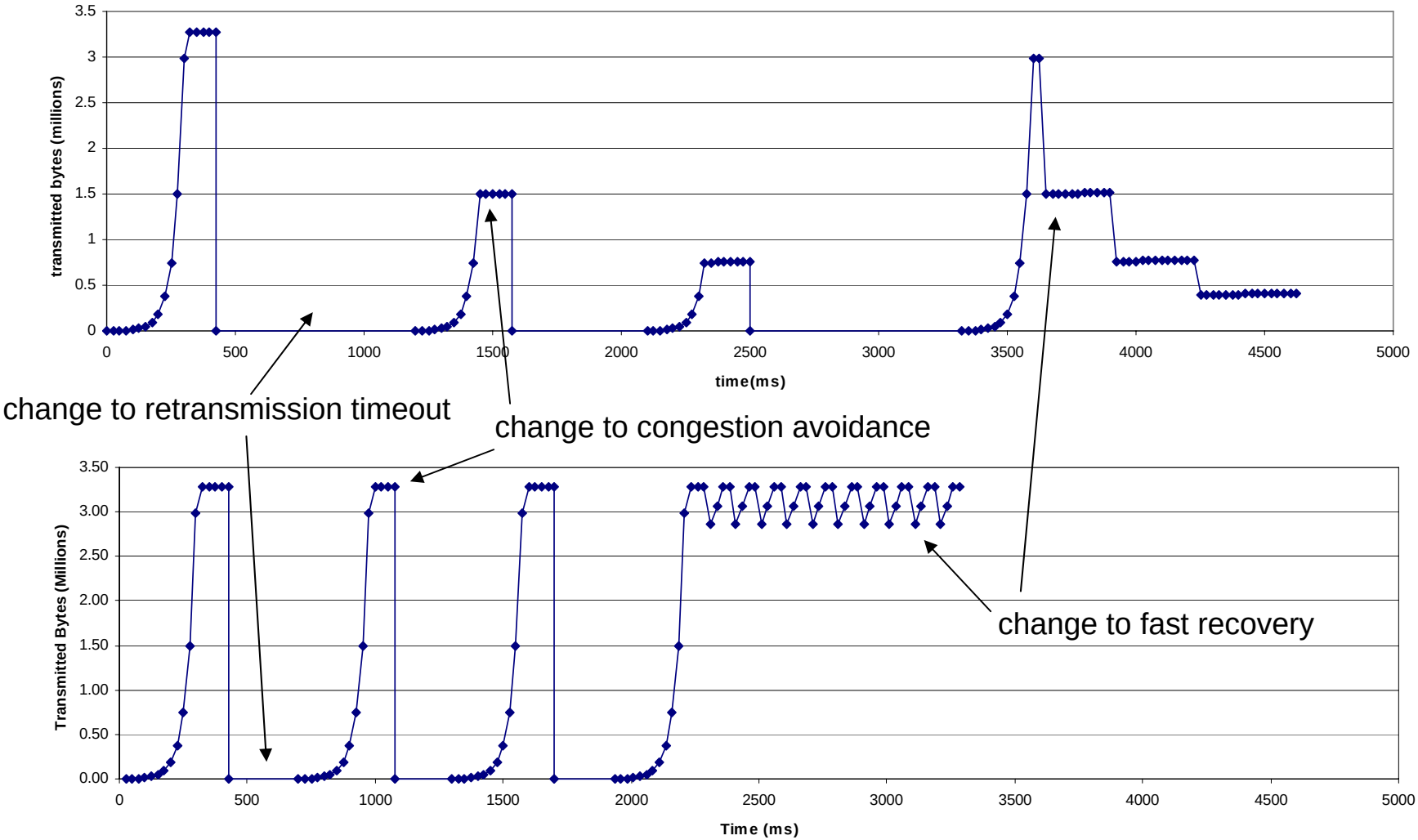


Causes Fast Retransmit and Fast Recovery
by the sender even though no segments were lost!



- Can be helped by ignoring more duplicate ACKs before Fast Retransmit/Recovery
- Have to be careful not to miss a retransmit that should have gone out
- Several ways to implement this

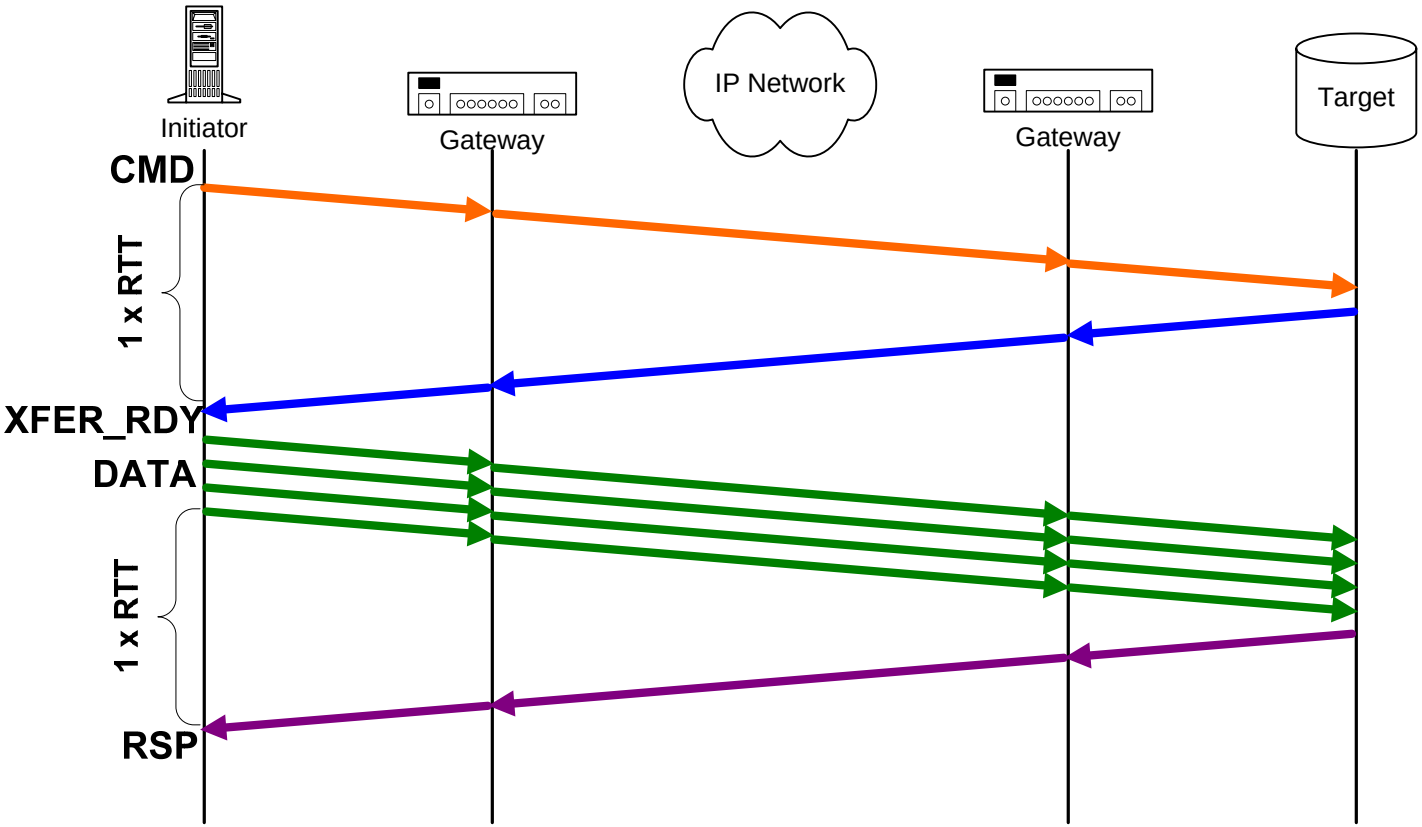
TCP Block Storage Modifications -- Effects



WAN Optimizations for SCSI based Block Storage

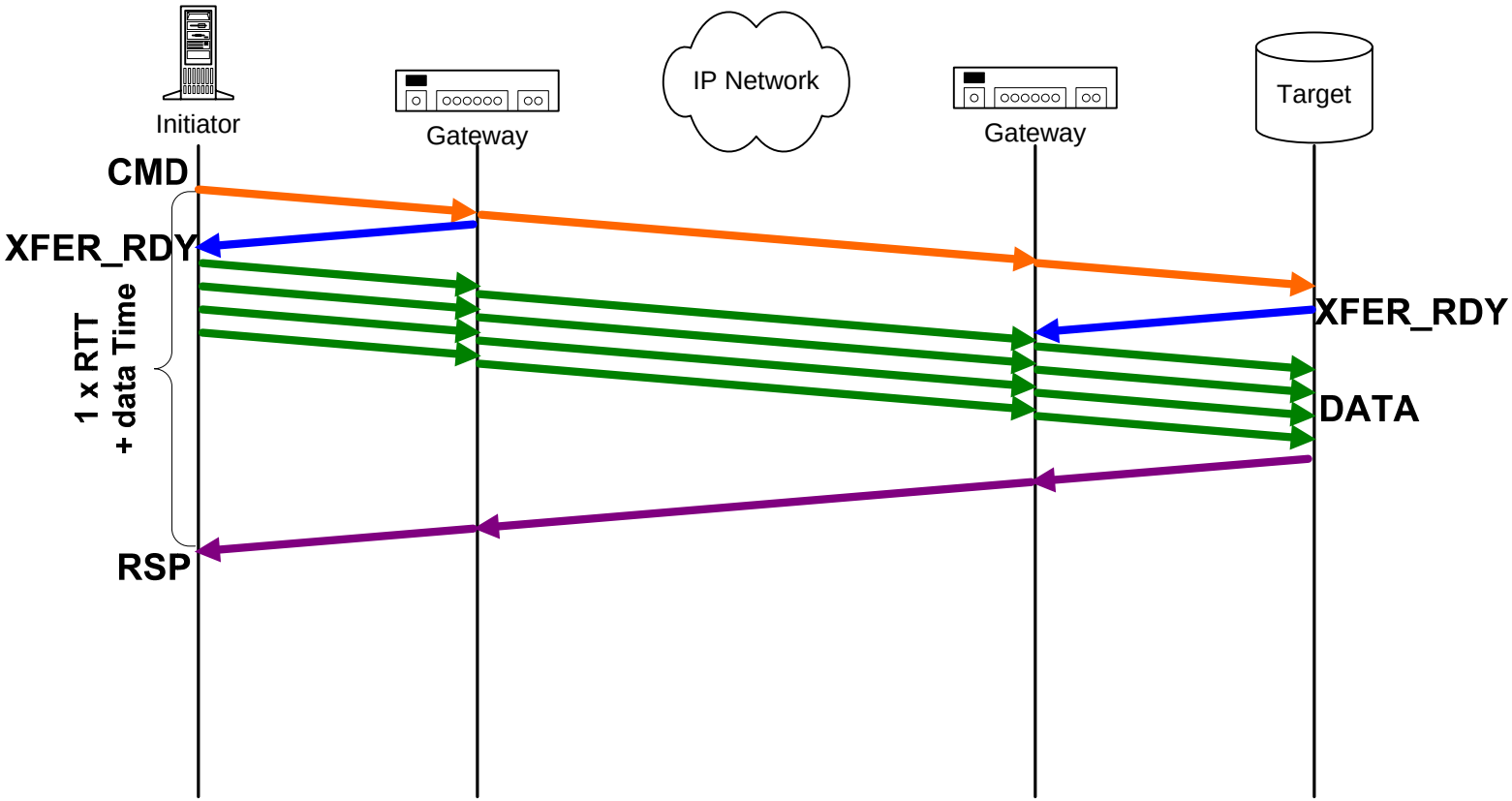
- Generally requires a pair of cooperating Gateways or Accelerators
  - protocol might be modified FCIP, modified iFCP, or proprietary
- Acceleration For Write Commands
  - Called Fast Write or Write Acceleration
  - iSCSI has immediate data and unsolicited data
- Acceleration For Tape Devices
  - Called Tape Pipelining or Tape Acceleration
    - Write Command Acceleration
    - Read Command Acceleration

Normal Write Command



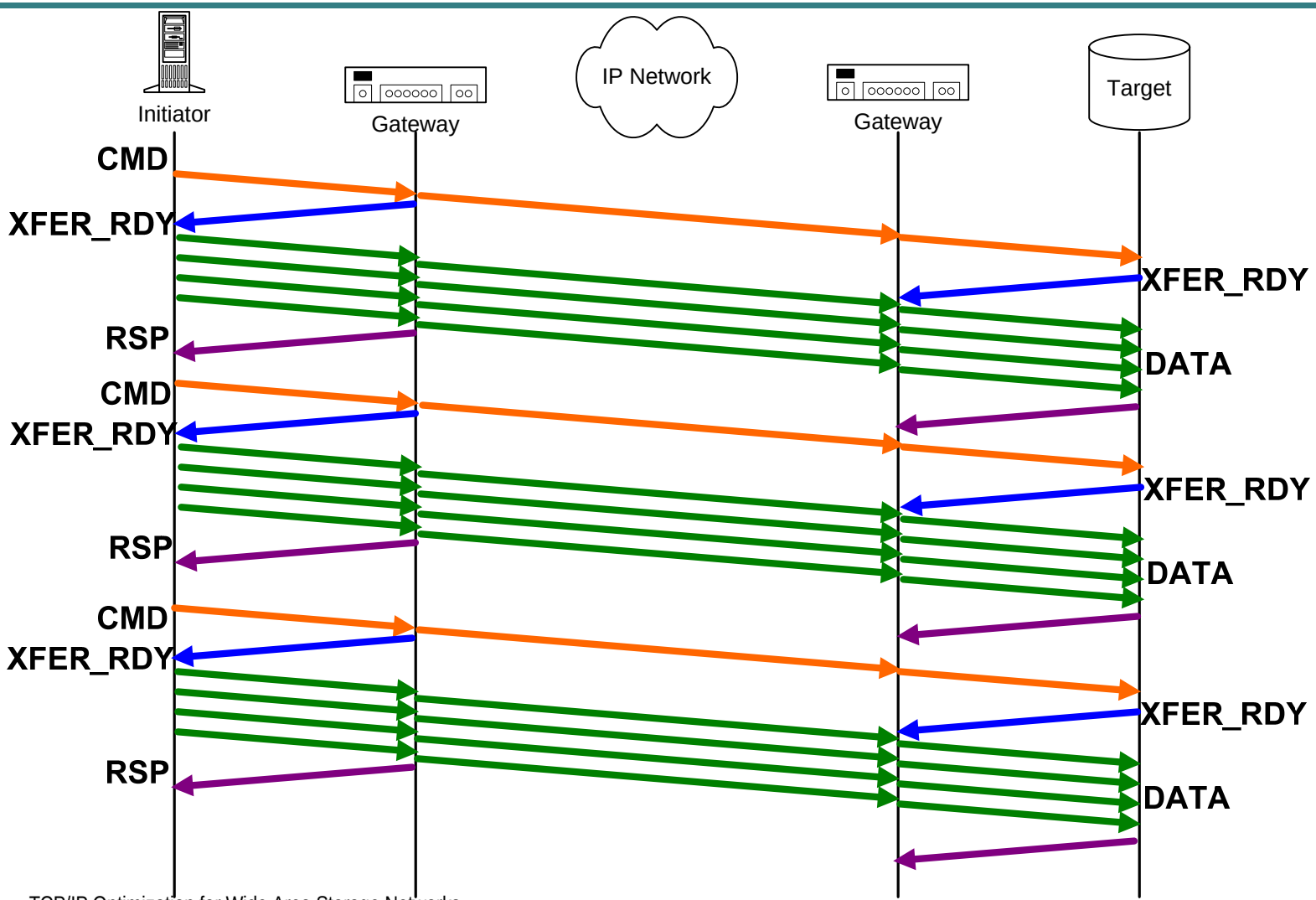
- A write command normally takes 2 x RTT + the data transmission and device response time to complete

Write Command Acceleration

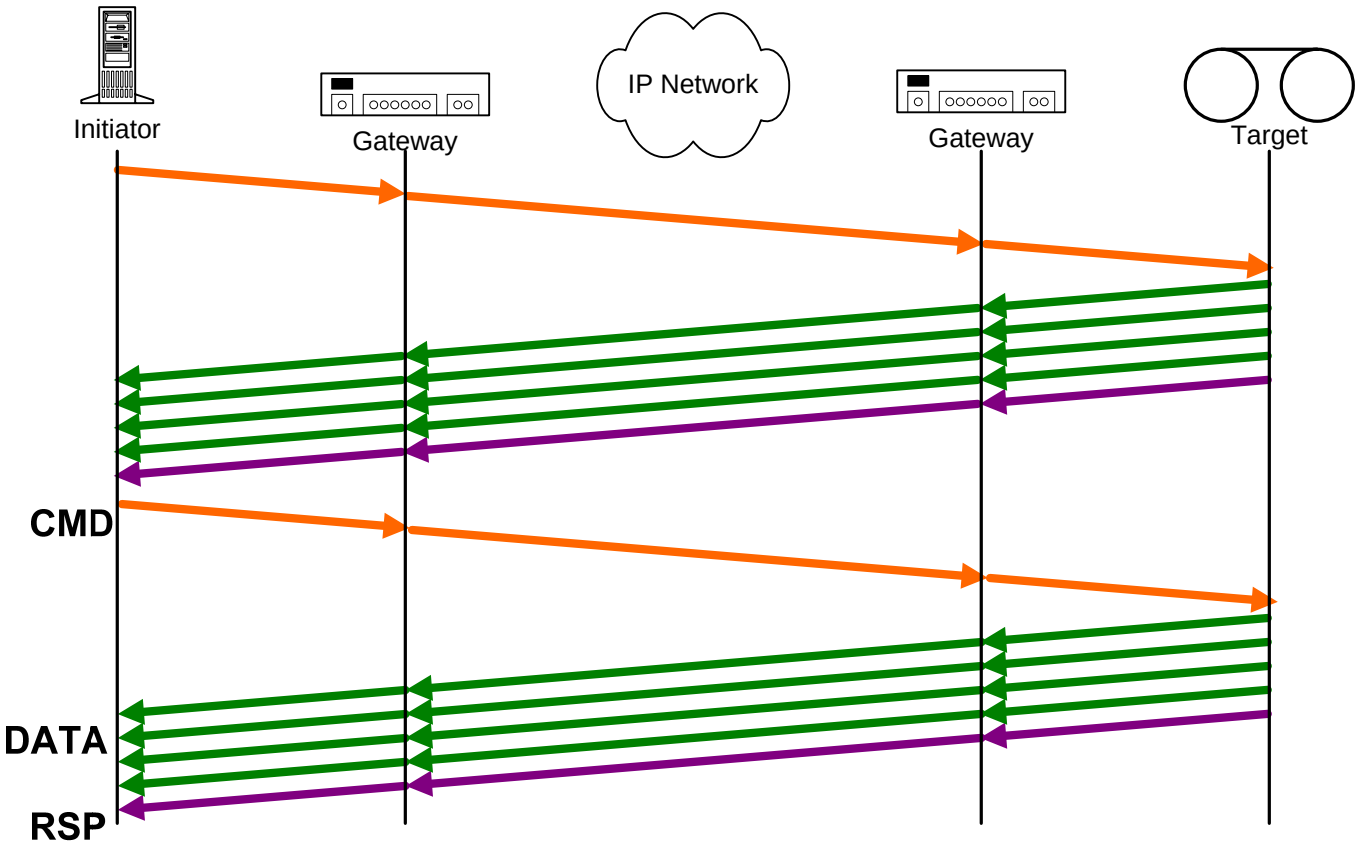


- An accelerated write command takes 1 x RTT + the data transmission time

Accelerated Tape Write Commands

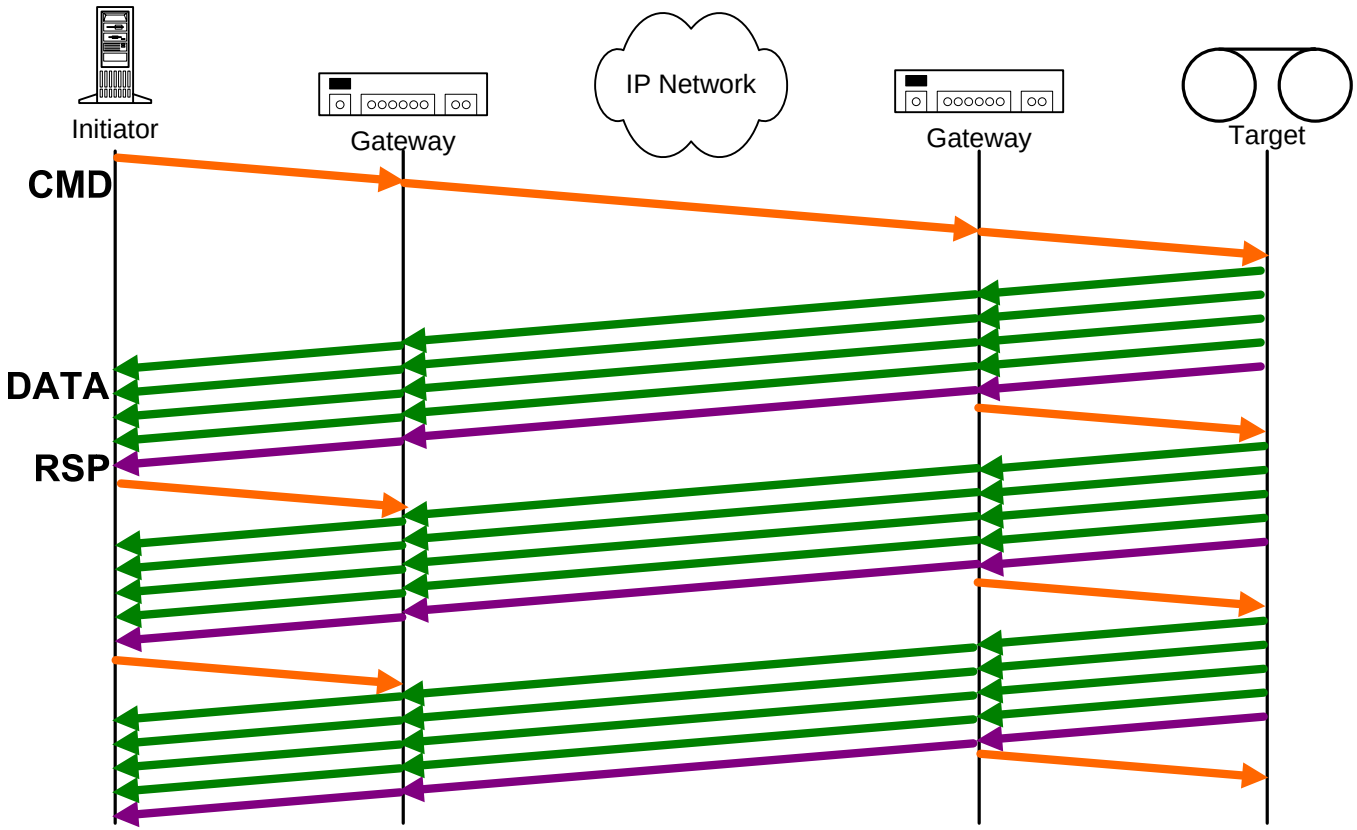


Normal Tape Read Commands



- Tape devices only allow 1 outstanding command for read as well

Accelerated Tape Read Commands



- Remote Gateway reads ahead by issuing commands itself
  - Data sent to and buffered by Local gateway until command received
  - Works well because a tape is a sequential access device

Q&A / Feedback

- Please send any questions or comments on this presentation to SNIA: tracknetworking@snia.org

Many thanks to the following individuals for their contributions to this tutorial.

SNIA Education Committee

Joseph L White

Appendix: Relevant Internet RFCs

SNIA
Storage Networking Industry Association
EDUCATION

- RFC 793 – Transmission Control Protocol
- RFC 896 – Congestion control in IP/TCP internetworks
- RFC 1122 – Requirements for Internet Hosts - Communication Layers
- RFC 1323 – TCP Extensions for High Performance
- RFC 2018 – TCP Selective Acknowledgment Options
- RFC 2140 – TCP Control Block Interdependence
- RFC 2581 – TCP Congestion Control
- RFC 2861 – TCP Congestion Window Validation
- RFC 2883 – An Extension to the Selective Acknowledgement (SACK) Option for TCP
- RFC 2988 – Computing TCP's Retransmission Timer
- RFC 3042 – Enhancing TCP's Loss Recovery Using Limited Transmit
- RFC 3124 – The Congestion Manager
- RFC 3155 – End-to-end Performance Implications of Links with Errors
- RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP
- RFC 3390 – Increasing TCP's Initial Window
- RFC 3449 – TCP Performance Implications of Network Path Asymmetry
- RFC 3465 – TCP Congestion Control with Appropriate Byte Counting (ABC)
- RFC 3517 – A Conservative Selective Acknowledgment based Loss Recovery Algorithm for TCP
- RFC 3522 – The Eifel Detection Algorithm for TCP
- RFC 3649 – HighSpeed TCP for Large Congestion Windows
- RFC 3742 – Limited Slow-Start for TCP with Large Congestion Windows
- RFC 3782 – The NewReno Modification to TCP's Fast Recovery Algorithm
- RFC 4015 – The Eifel Response Algorithm for TCP
- RFC 4138 – Forward RTO-Recovery (F-RTO)
- RFC 4653 – Improving the Robustness of TCP to Non-Congestion Events
- RFC 4782 – Quick-Start for TCP and IP

Appendix: TCP Options

- End of option list

Kind	Len	Values...

0

- No operation

Options are usually 4 byte aligned with leading NOPs

1

- Maximum (*Receive*) Segment Size [SYN only]

2	4	MSS

- Window Scale Factor [SYN only]

NOP	3	3	shift

- Timestamp

NOP	NOP	8	10	Timestamp Value	Timestamp Echo Reply

- Selective ACK Permitted [SYN packet only]

NOP	NOP	4	2

- Selective ACK block

NOP	NOP	5	L	Left Edge	Right Edge

...

L = 2 + N * 8, N is number of left-right pairs