Issues Related to TCP Performance in a Satellite Environment and Discussion of Possible Protocol-Oriented Mitigations

TCP-over-Satellite Working Group
39th IETF
Munich, Germany
August 11, 1997

Mark Allman/NASA Lewis Research (mallman@lerc.nasa.gov
Robert Durst/The MITRE Corporation (durst@mitre.org)
Eric Travis/OMS (e.j.travis@ieee.org)

General Overview

- **Briefing contains:**
  - **Distillation (possibly lossy) of inputs from tcp-over-satellite mailing list**
  - **Input from current NASA and DOD research**
    - **Joint NASA/DOD Space Communications Protocol Standards (SCPS)**
    - **NASA Lewis Research Center TCP over Satellite activities**
- **The content of this briefing tracks the content of Section 4: Transport Issues) of the TCP Over Satellite Working Group's immanent Internet Draft (I-D)**
-

Note: throughout this briefing, the following designations are used to describe the current status of ongoing activities:

[ST] = Standards Track [R] = Research [BCP] = Best Common Practice

Disclaimer:

Any assertions and/or recommendations contained in this briefing are for discussion purposes only

TCP Over Satellite: Distant Travelers, Islands of Connectivity and Some Really Cool Science

- The satellite community is a big, diverse place
- The technology is far from monolithic,
  - LEOs, GEOs and HEOs… Oh my!
  - Constellations
  - Manned Vehicles (Space Shuttle, International Space Station, etc.)
- The missions are varied:
  - Providing connectivity to remote and developing regions
  - Telescience, telemedicine and scientific explorations
  - The Corporate World: Intranets, Extranets and Virtual LANs
  - "Car 54: Where are you?": Anytime, anywhere connectivity
  - Military operations (strategic and tactical)
- The broad spectrum of technology, missions and cost constraints conspire to confront us with a wide range of environmental factors to consider

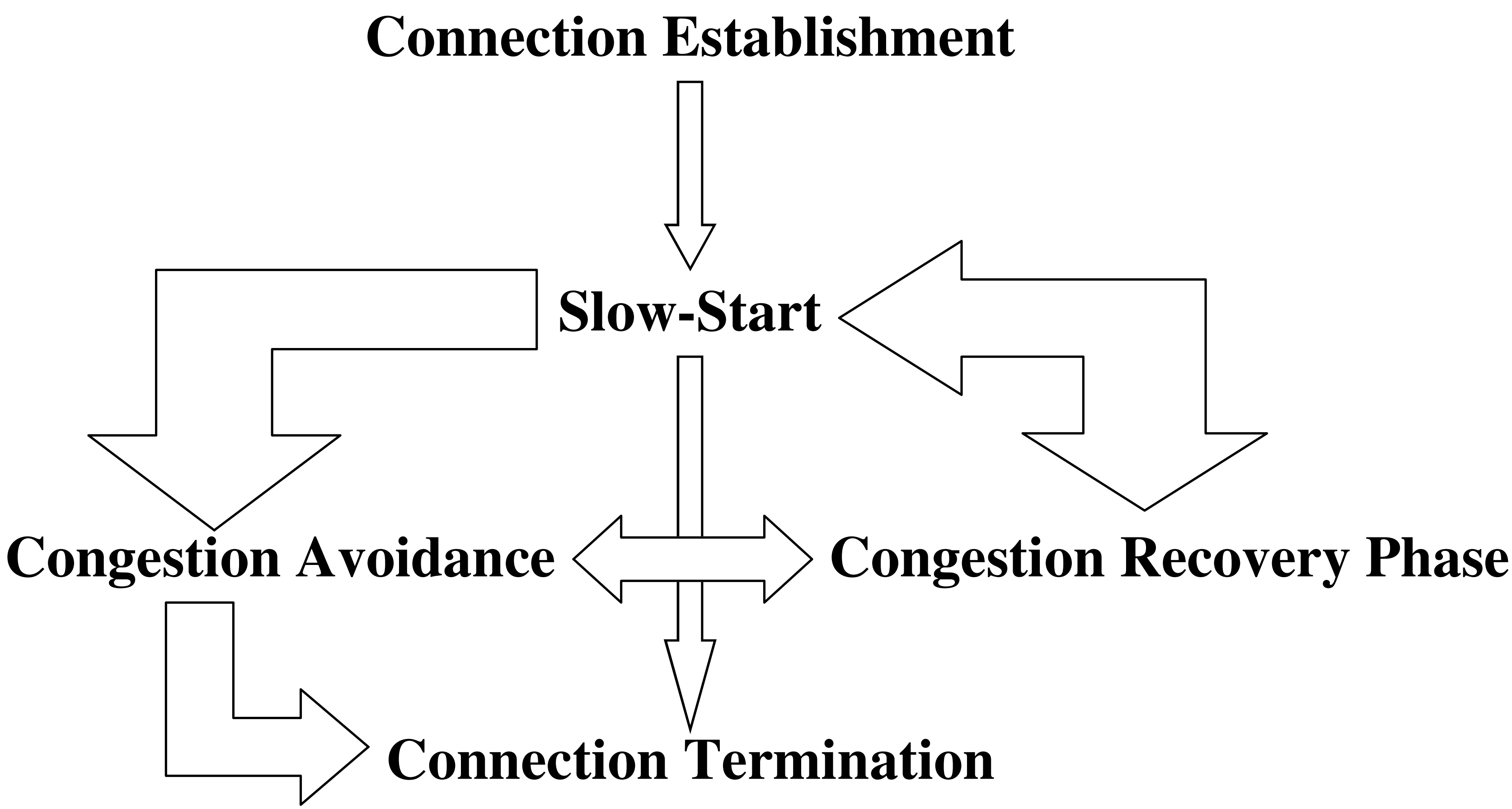
Satellite Specific Environmental Factors

- Satellite environments may be characterized by one or more of the following:
  - Long-delay paths
  - Large bandwidth-delay products
  - Large windows
  - Limited link capacities
  - Asymmetry
    - Bandwidth asymmetry
    - Path asymmetry - being addressed by UDLR Working Group
  - Varying round-trip-times
  - Transmission errors
  - Intermittent connectivity (handoffs and link-outages)
- Satellites and Terrestrial Mobile & Wireless: Separated at birth?

Mitigations: An Overview

- Above all else, mitigations must be fair to the entire Internet community
- Mitigations are generally not transparent to users (or end-systems)
  - May require modifications to protocol implementations
  - May require enhancements to protocol capabilities (protocol options)
- Mitigations are divided into three broad categories:
  - One-Sided Mitigations
    - Require deployment at either at the sending or receiving entity
    - Users generally need to know that a mitigation is required
    - Typically requires implementation upgrade to deploy
  - Two-Sided Mitigations
    - Requires deployment at both ends of a connection
    - Depending on the nature of the mitigation, the user *potentially* needs to have a clue
    - Generally need to be ubiquitous to be effective
    - Requires coordination in order to be used
  - Mitigations involving intermediate entities
    - Requires the assistance of an intermediate entity
    - Depending on the nature and location of the mitigation, at least one side of the connection is generally cognizant of the intermediate entity, and may even directly participate in the mitigation

Simplified Phases of a TCP Connection: A Sending Side's Perspective



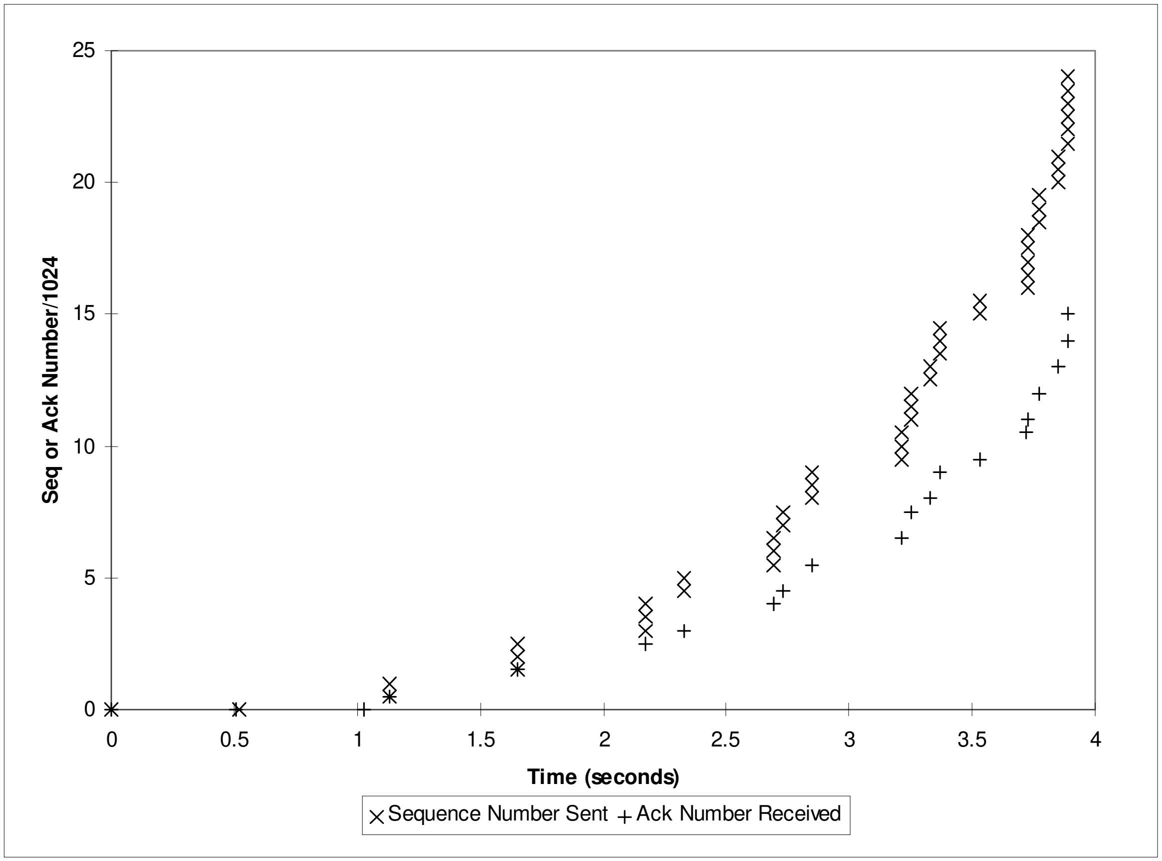
Connection Establishment

- Exchange of SYNs (and options)
  - Cost of exchange is 1 RTT (1.5 if passive side is sender)
  - Initiator needs to resolve address prior to doing anything (costs delay of one DNS query)
- Mitigations For Recurring Connections
  - Persistent connections (e.g. HTTP 1.1)
  - [ST] TCP for Transactions (T/TCP) (RFC 1644)
  - [R] Local caching of resolutions (avoid the DNS)
  - [BCP] Make sure your DNS entity is placed in a reasonable location with respect to the long-delay path

Slow-Start

- The exponential growth of cwnd:
  - cwnd is increased by one segment for each ACK received
  - Under *ideal* conditions, this would yield a doubling of cwnd per RTT
- Slow-start is initiated whenever (cwnd <= ssthresh)
  - Connection establishment, after a retransmission timeout, or after an extended idle period
- The initial slow-start period is probably the most important factor for a TCP connection's overall performance
  - ssthresh will never again be as large as the initial estimate, so after initial slow-start period, all future cwnd growth relative to this initial plateau will be in linear-mode (at *best* 1 segment/RTT)
  - Slow-start terminates upon any loss event
  - Premature termination of slow-start should be avoided, if possible
  - Tardy termination of slow-start should also be avoided

Illustration of Slow-Start Behavior



Collective Impacts on Slow-Start Behavior

- The delayed ACK algorithm slows the cwnd growth rate by attempting to ACK every 2nd full-sized segment (out-of-sequence segments should generate an immediate duplicate ACK)
- Long-Delay paths increase the growth period
  - In early stages, cwnd increases approximately once per RTT
  - When (RTT > delayed_ACK_timer) detrimental impact of the delayed ACK algorithm is reduced
- Large bandwidth-delay products and long delays make ssthresh estimate critical
  - ssthresh too small causes majority of cwnd growth to be linear (slow!)
  - ssthesh too large increases likelihood of multiple-losses within a window
    - Leads to a retransmission timeout and loss of hard-earned cwnd
    - When slow-start resumes, the new ssthresh (old_cwnd/2) will cause the bulk of future cwnd growth to be linear

Collective Impacts on Slow-Start Behavior (cont)

- Highly asymmetric channels may retard the responder's ability to properly pace the transmission of ACKs, further hampering the growth of cwnd
- Any non-congestion losses will cause slow-start to prematurely terminate and may have a significant impact on throughput for the remainder of the connection

Slow-Start Mitigations: Single-Sided [R] Proposal to Increase Initial cwnd

- Current Internet Draft (draft-floyd-incr-init-win-00.txt)
  - Allow the initial cwnd for a TCP connection to be increased from a single segment to ~4K bytes:

    initial cwnd = min (4*MSS, max (2*MSS, 4380 bytes))
  - Increase applies only to the connection's initial window, or to connections resuming data transmission after an extended idle period
  - Following a retransmission timeout, cwnd will still be set to a single segment
- Benefits to satellite environment
  - Allows small transactions (4K or less) to complete in
    - 1 RTT after connection establishment, or 1 RTT total with T/TCP
  - Increases momentum of slow-start by eliminating up to 3 RTT
- Risks exist and should be discussed...

Slow-Start Mitigations:Single Sided

- [R] Byte-Counting versus ACK Counting (only during slow-start)
  - Increase cwnd by an amount equal to the size of acknowledged data
- [R] Disable delayed ACKs on responder
  - Attempt to ACK every packet, yielding optimal cwnd growth
  - May cause increased burstiness, aggravate asymmetry problems
- [R] Better estimation of initial ssthresh value
  - Make use of cacheable information (srtt, cwnd, etc.)
    - cached information in routing structure
    - T/TCP (RFC 1644)
    - TCP Control Block Interdependence (RFC 2140)

Slow-Start Mitigations: Single Sided

- **[R] Better estimation of initial ssthresh value (continued)**
  - **Compute ssthresh based on initial ACK spacing (J.C. Hoe Dissertation)**
    - **May be sensitive to delayed ACKs and fluctuations in intermediate queue lengths**
    - **Adds implementation complexity (unless you are doing something TCP-Vegas like already)**
- **[R] TCP Vegas (Brakmo, Peterson)**
  - **Virtues**
    - **Doesn’t depend on loss as a signal of congestion**
      - Uses changes in instantaneous throughput as indications of network loading
    - **Doesn’t depend on recv_window as upper-bound on cwnd growth**
      - recv_window is the absolute upprer-bound
  - **Drawbacks**
    - **Sensitive to variations in RTT**
    - **Potential problems if network already congested at connection establishment**
      - Mitigated by RED (RED w/ECN if all loss is not due to congestion)

Slow-Start Mitigations: Two-Sided

- Better handling of multiple packet losses in a single window
  - [ST] Selective Acknowledgements (SACK) (RFC2018)
    - Enables better handling of retransmission queue
  - [R] Explore refinements to TCP congestion control
    - Without SACK: New-Reno, J. Hoe’s Fast Recovery Period proposal
    - With SACK: FACK, SACK “Pipe-Algorithm”, ?
  - [R] Selective Negative Acknowledgements (SNACK) and (RFC 1106)
    - Explicit retransmission requests
    - Relation to SACK is TBD
    - I-D in progress for SNACK (Travis-Durst-Feighery)
- [R] End-To-End compression of TCP Headers
  - Tolerant to loss
  - Allows for end-to-end encryption of transport headers (if desired)
  - I-D in progress (Durst)

Slow-Start Mitigations: Intermediate Entities

- [ST?] RFC 1144 Header Compression for reducing effects of Asymmetry
  - Highly efficient compression of both TCP & IP headers
  - Operates at the link layer; use must be coordinated with link provider
  - Intolerant of loss
    - Losses require resynchronization of traffic (go-back-n behavior)
    - Impacts can be significant if congestion windows become large
  - Intermediate entity must be trusted to have access to TCP header
- [BCP] Forward Error Correction (FEC) on noisy links
  - Contributes to noise in RTT measurements due to interleaving, etc.

Congestion-Avoidance

- **Congestion avoidance provides linear growth of the congestion-window**
  - **When cwnd > ssthresh, window growth will be linear**
  - **Increase cwnd by *no more than* one segment/RTT**
  - **cwnd += (segment_size * segment_size/cwnd)   (all quantities in octets)**
- **Footnote on interactions between delayed-ACKs and congestion-avoidance**
  - **Growth rate of cwnd during congestion-avoidance be:**
    - **one segment every two round-trips**
    - **Say, MSS = 1460 octets, current cwnd = 29200 octets (20 segments)**
      - We send 20 (full-sized) segments this window, we get 10 Acks
      - cwnd += 10 * (1460 * 1460/29200) == 10 * 73
      - cwnd += 730 bytes (one-half segment)
  - **For geosynchronous delays, we'll grow our cwnd by one segment/second**

Combined Effects on Congestion Control

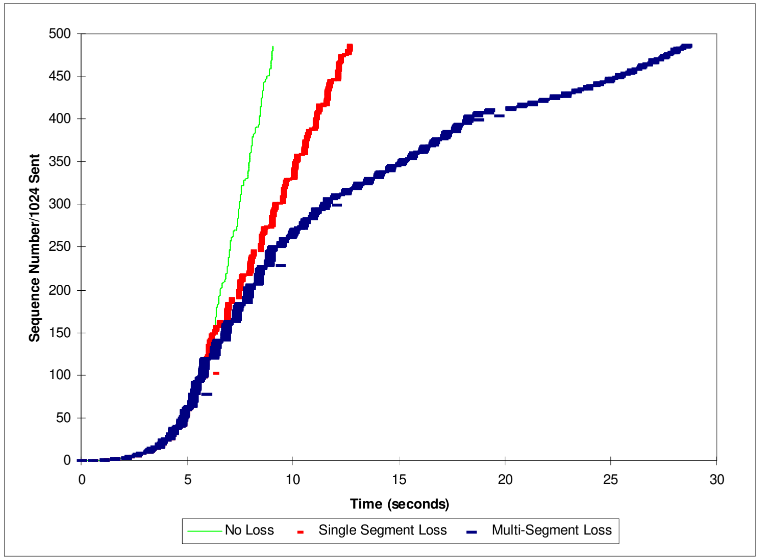
- **Long delay paths make the delayed ACK interaction more painful**
- **Because Vanilla TCP limits the maximum window size to 2^16 octets, bandwidth-delay products exceeding 65535 octets *may* become window-limited (but only if network congestion allows growth > 65535 octets)**
  - **TCP limits maximum window size to 2^16 octets**
  - **Potential for wrap in sequence number space**
- **Large Windows (in terms of segments, not necessarily octets) contribute:**
  - **Increased aliasing of the measured RTT**
    - **Situation is even worse with lossy paths (congestion or corruption)**
    - **Poor RTT estimation may trigger unnecessary retransmissions, and other undesirable effects**
  - **Increased probability of multiple losses within a single window**
    - **Connection will experience a retransmission timeout, and therefore loss of hard-earned cwnd growth**

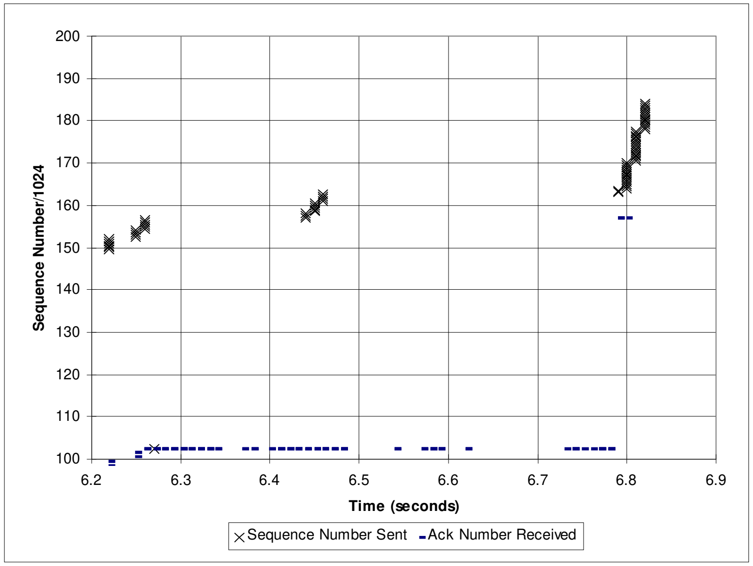
Combined Effects on Congestion Control

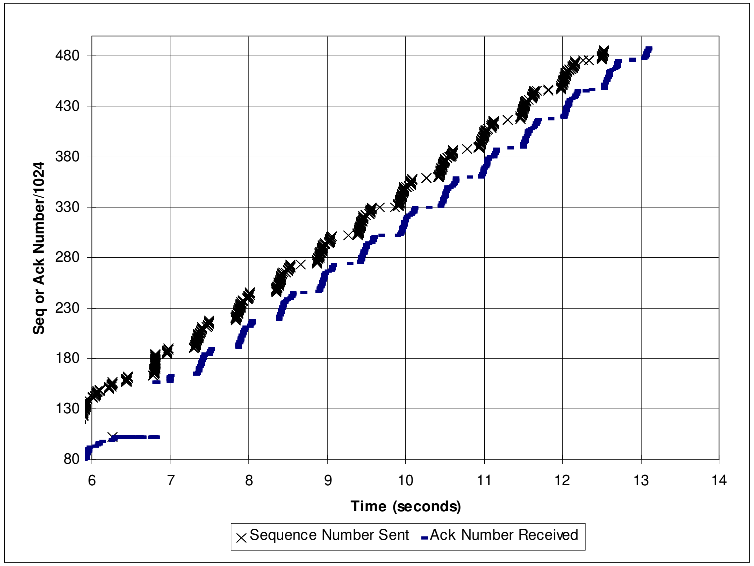
- **Highly asymmetric channels may impact TCP's linear cwnd growth:**
  - **Potential for ACK compression and ACK loss**
  - **Possible inability to provide ACKs at rate required to maintain TCPs self-clocking**
    - **Channel asymmetries of [1000+:1] are not uncommon in some segments of the scientfic satellite community**
      - 1Mbps downlink: 1Kbps uplink
      - Gbps downlink:Mbps (or less) uplink
    - **Even in less constrained environments, reverse channel capacity is needed to support traffic other than ACKs**
- **Any losses not due to congestion (I.e., corruption-based losses) will trigger false congestion-events**

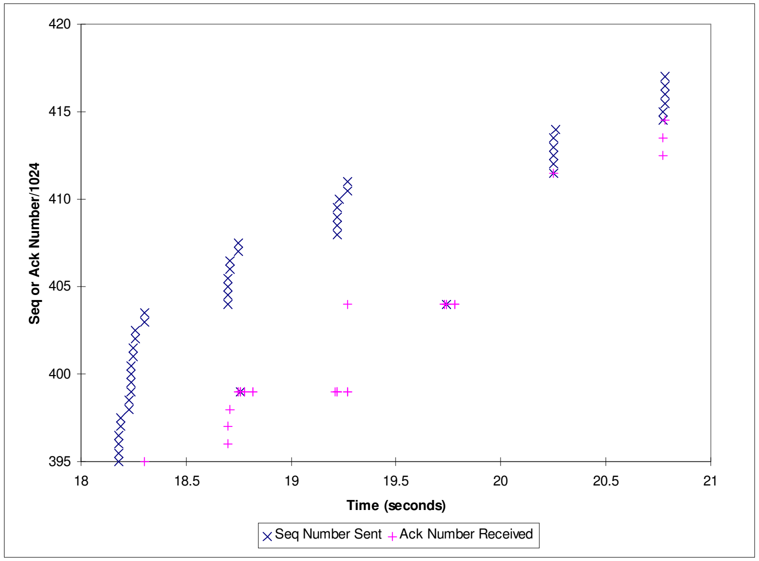
Space Mission Operations Standardization Program
National Aeronautics and Space Administration

Effects of Loss on TCP









SCPS

Congestion Avoidance: Implementation Note

- Most implementations calculate window increase using integer arithmetic
- This can clamp the value of cwnd, and possibly the throughput over a long-delay path
  - Delay = 0.5 sec, MSS = 256 octets, max sustainable throughput ~ 1Mbps
  - Delay = 0.5 sec, MSS = 512 octets, max sustainable throughput ~ 4 Mbps
  - Delay = 0.5 sec, MSS = 1500 octets, max sustainable throughput ~ 36 Mbps

Congestion Avoidance Mitigations: Two Sided

- **[ST] Window-Scaling (RFC-1323)**
  - **Enables increase of TCP's maximum supportable window-size to 2^20**
  - **Must be "specified" at Connection-Establishment**
    - **can *not* be enabled later in the connection**
    - **scaling factor can *not* be changed once it is initialized**
    - **Scaling does *not* imply or guarantee large-windows will be available**
    - **No conformance testing on available implementations**
      - Can exhibit behavior other that what is expected
  - **Note that an arbitrary server can not know the appropriate window size at the time of connection establishment**
- **[ST] Round-Trip-Timing Measurements (RFC-1323)**
  - **Must be "negotiated" at Connection-Establishment**
  - **Adds twelve additional octets of overhead per segment (two 4 octet timestamps) - may be a problem for severely bandlimited channels**
- **[ST] Protection Against Wrapped Sequence Numbers (RFC-1323)**
- **[ST] Selective Acknowledgements (SACK) RFC-2018**

Congestion Avoidance Mitigations: Intermediate Entities

- [R] Explicit Notification - distinguishing between corruption and congestion-based losses; optionally also provide notification to hosts of any extended link-outage periods
- Congestion response *must* be the default response to losses if traffic is passing through the Internet (or any other shared paths that you do not own)
  - Desire for source-quench controversial, but
    - Should be "cheaper" in a RED world
    - Reduces the length of a congestion period (if the bottleneck is on the near end of the long-delay path from the data source)
  - Corruption-detection is done at the link-layer and triggers notification via an ICMP message
    - Requires definition of a new ICMP message(s)
  - I-D in progress [Durst-Feighery-Travis]

Congestion Control: TCP's Response to Loss

- **The myth of Go-Back-N Behavior**
  - **Modern TCPs don't Go-Back-N (well… Tahoe TCP sort of does)**
- **Retransmission Time-Out**
  - **ssthresh = cwnd/2; cwnd = 1;**
- **Fast-Retransmit**
  - **Trigger a retransmission after receiving 3 duplicate ACKs**
  - **Receiver needs to generate an immediate ACK for each out-of-sequence segment received to help make this happen (required to make Fast Recovery work)**
  - **Fast Recovery**
  - **ssthresh = max((cwnd/2),1); cwnd = ssthresh + 3;**
  - **Retransmit lost segment**
  - **Inflate cwnd by 1 MSS for each duplicate ACK received**
  - **Transmit new segments while there is available cwnd**
  - **Upon receipt of new ACK, cwnd = ssthresh**

Combined Effects on Congestion Control

- Long delay paths don't seem to hurt (or help) much on their own but:
  - cwnd losses will take a long time to recover
    - At least (prev_cwnd/2*MSS) * RTT in response to a single-loss if receiver is ACKing every packet
  - May be more prone to multiple losses in a window (but then your window gets smaller in response)
  - The decreased fidelity of the round-trip-time measurements mentioned earlier
- All effects of asymmetric channels and corruption-based losses apply

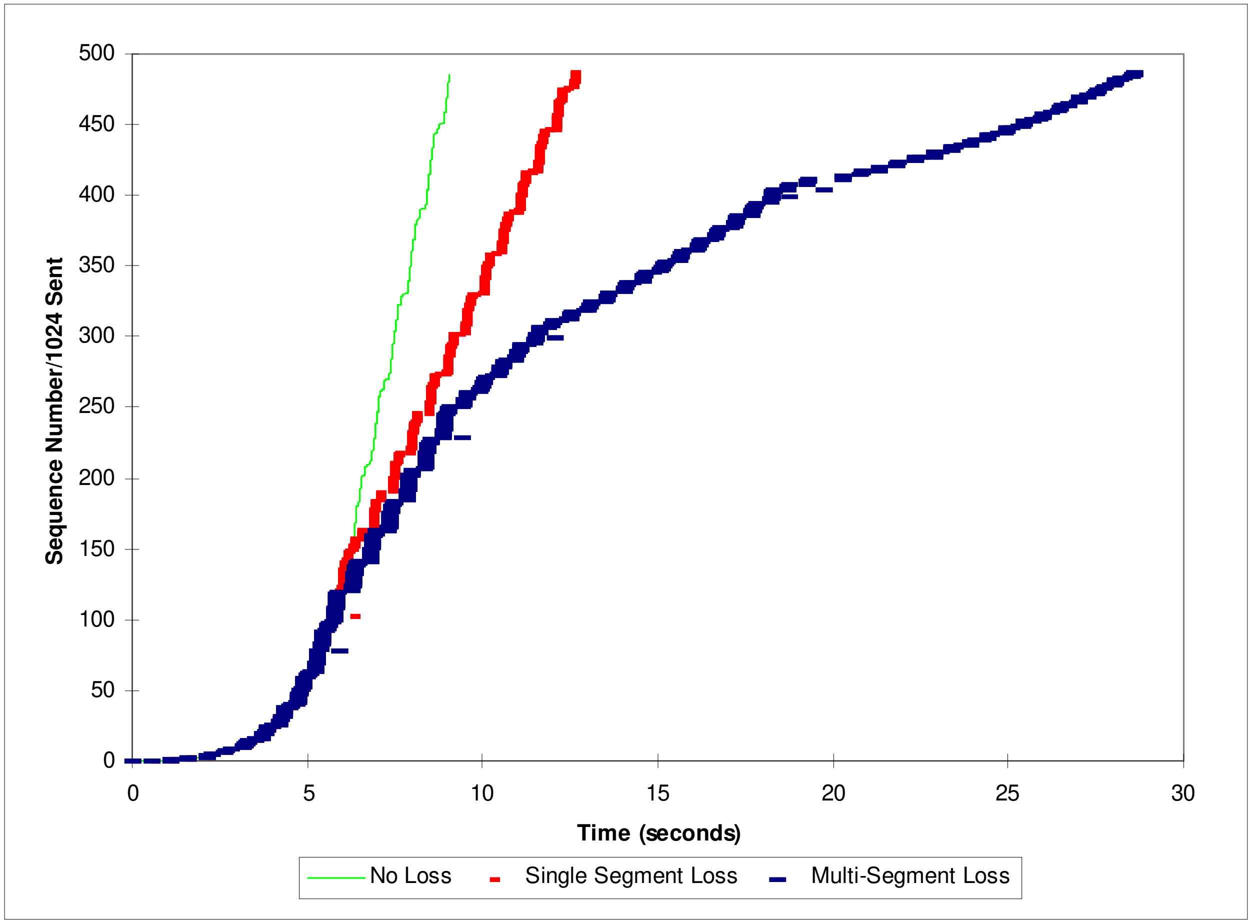
Congestion Control Mitigations: Two-Sided

- [ST] Fast Retransmit & Fast Recovery (RFC 2001)
  - More of a given than a mitigation
  - Prevents retransmission timeouts for *single* losses within a window
- [ST] Selective Acknowledgments (RFC 2018)
  - Allows for more expedient handling of multiple losses within a window (when coupled with enhancements to TCP congestion control mechanism which are SACK-aware)

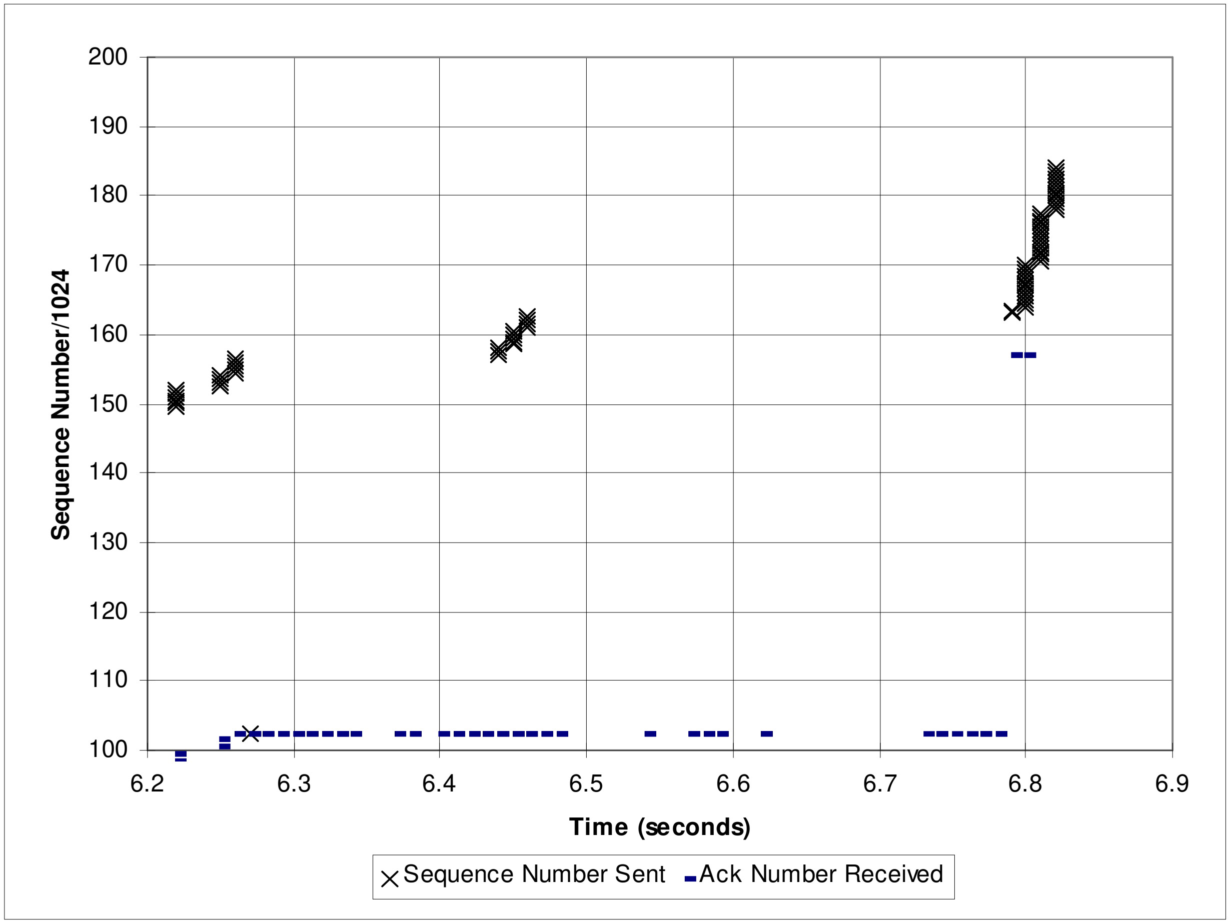
Summary of Issues and Potential Mitigations

- Issues:
  - Not yet, maybe next time
- Standards Track Mitigations
  - Window-Scaling (RFC 1323)
  - Round-Trip-Timing-Measurement (RFC 1323)
  - Protection Against Wrapped Sequence Numbers (RFC 1323)
  - Selective Acknowledgements (RFC 2018)
  - Link layer header compression (RFC 1144)
- Currently Identified Mitigations Undergoing Research
  - Use of cached information
  - ssthresh estimation
  - Selective Negative Acknowledgements
  - End-to-End Header Compression
  - Modifications to generation and interpretation of Acknowledgements
  - Explicit Notification

Impact of Loss on Overall Throughput



Fast Retransmit of Single Segment Loss



Linear cwnd Increase after Single Loss

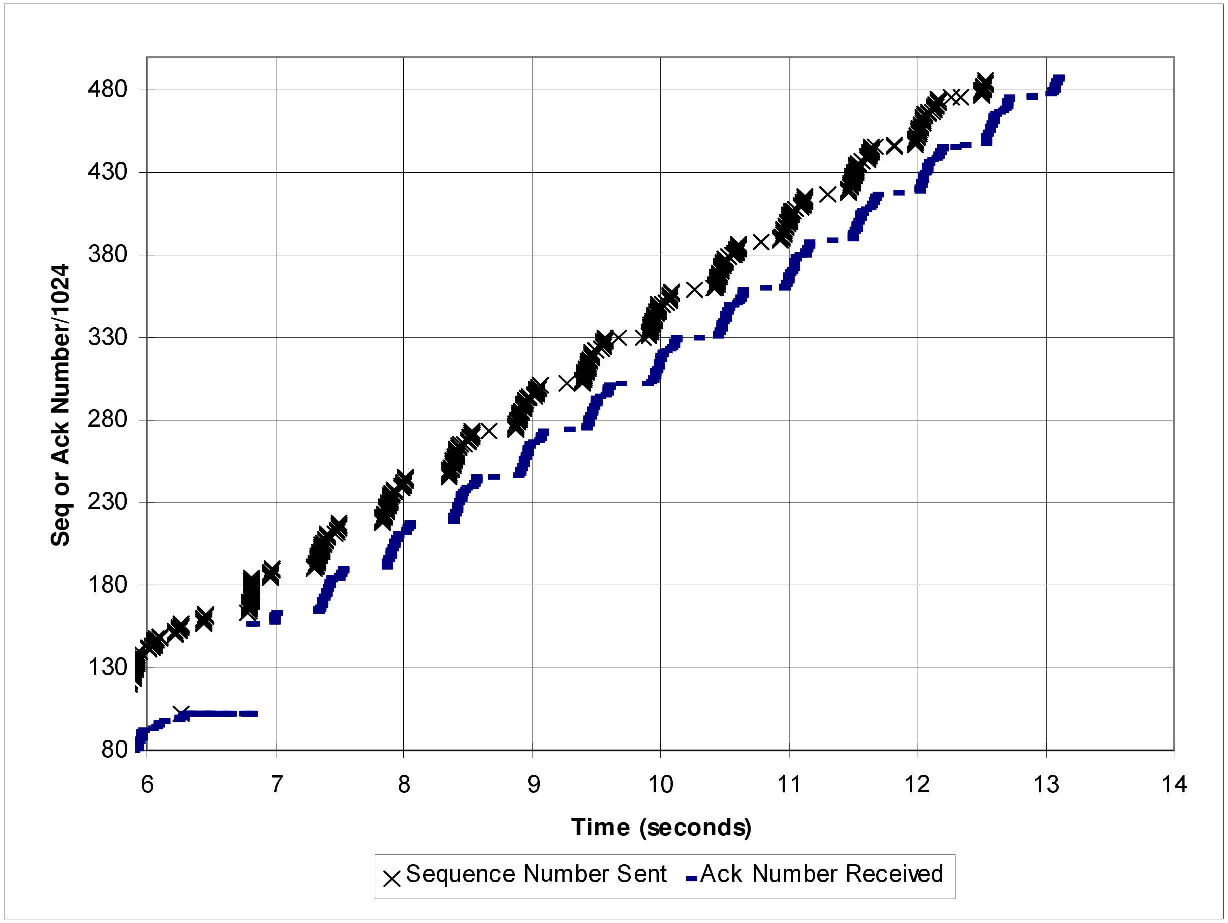


Illustration of Multiple Packet Losses