Improving TCP Performance Over Satellite Channels

May, 1997

Mark Allman

mallman@cs.ohiou.edu

Ohio University School of Electrical Engineering and Computer Science

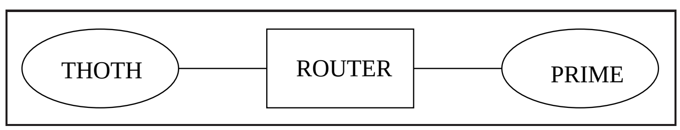
http://ace.cs.ohiou.edu/~mallman

Overview

- Sample networks
- Brief Overview of TCP
- Problems with TCP in the Satellite Environment
- Application-level Solution
  - Lessons that may help TCP
- Slow Start Modifications
- Results and Future Work

Sample Local Network

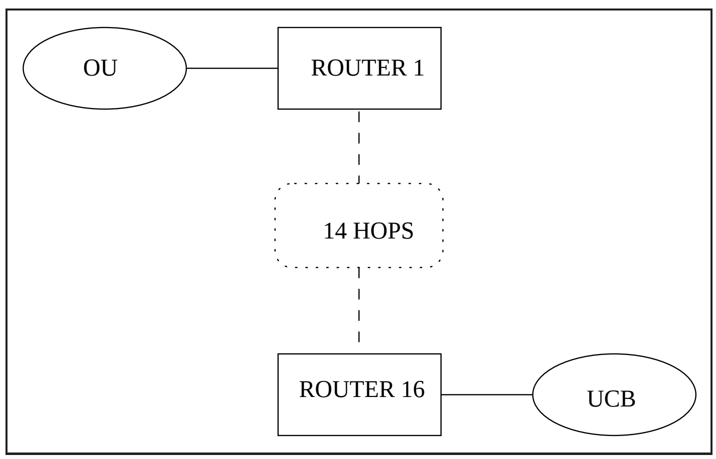
- Sample Local Network:
  - thoth to prime



- Bandwidth: 10 Mbits/second
- End-to-end Delay: $\approx$ 1 ms
- Round-trip Time: $\approx$ 2 ms

Sample Terrestrial Network

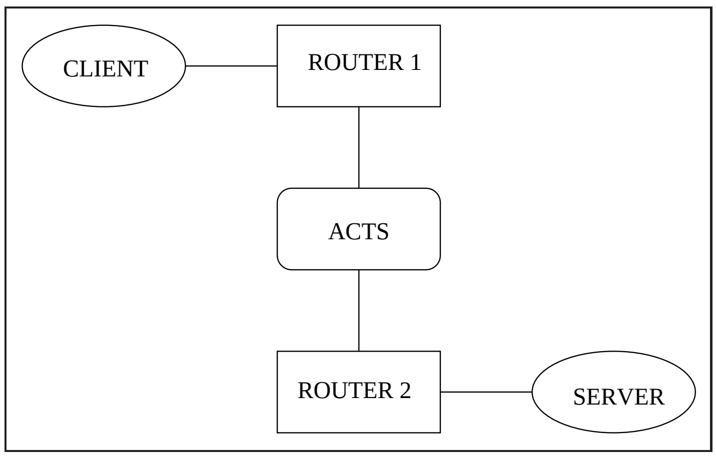
- Sample Terrestrial Network:
  - OU to UC Berkeley



- Bandwidth: ??? bytes/second
- End-to-end Delay: $\approx$ 40 ms
- Round-trip Time: $\approx$ 80 ms

NASA ACTS Satellite Environment

- NASA ACTS Satellite System:



- Bandwidth: 1.536 Mbits/second

- End-to-end Delay: $\approx$ 280 ms

- Round-trip Time: $\approx$ 560 ms

TCP Overview

TCP Overview

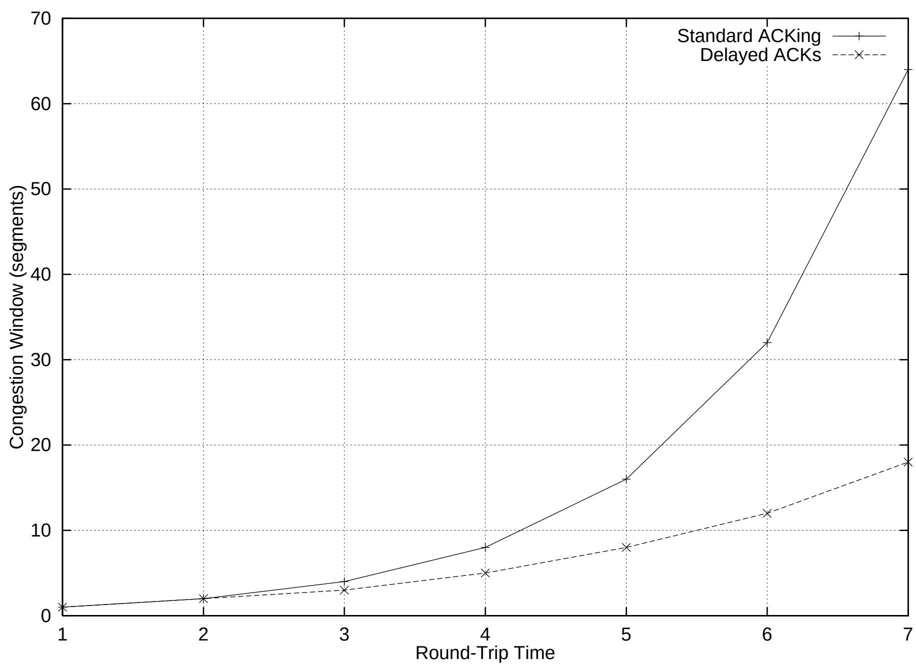
- TCP uses acknowledgments (ACKs) to guarantee delivery
- TCP is a sliding window protocol
  - Receiver's advertised window is an upper bound on unacknowledged data
- TCP uses several *congestion control* algorithms to facilitate sharing
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
  - Fast Recovery
- Congestion control algorithms alter the congestion window (*cwnd*) which is the real size of the sliding window.
  - *cwnd* $\leq$ advertised window

Slow Start and Delayed ACKs

- TCP uses *slow start* to gradually increase the size of the *cwnd*
  - Initializes *cwnd* to 1 segment
  - Increases *cwnd* by 1 segment for each acknowledgment received
  - Ends when *cwnd* reaches the advertised window size or loss is detected
- Delayed ACKs
  - TCP receivers are only required to ACK every second segment
  - If a second segment does not arrive within a given timeout the receiver must send an ACK
    - The timeout must be $\leq$ 500 ms

Slow Start and Delayed ACKs (cont.)

- Standard ACKing vs. Delayed ACKing:



TCP Problems in the Satellite Environment

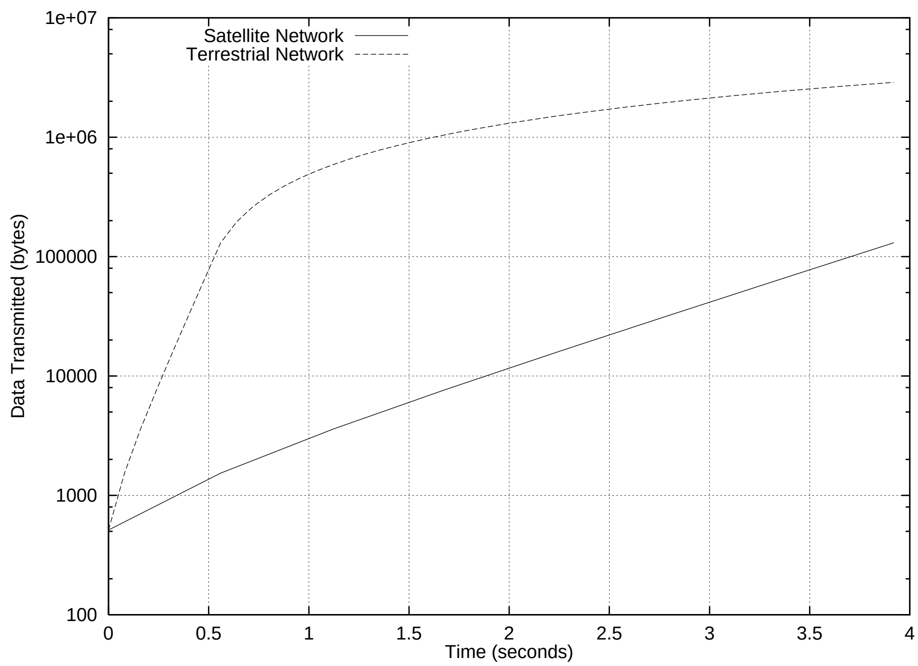
TCP Problems Over Satellite Channels

- Slow Start Problem:
  - Increasing the *cwnd* is always a response to an incoming ACK.
  - The long delay imposed by satellite links increases the amount of time required to open *cwnd*.
  - The time spent opening *cwnd* often represents wasted capacity.
  - Slow start takes *T* seconds to reach an advertised window of W_A segments on a network with a RTT of *R* seconds:

$$T = Rlog_2 W_A$$

TCP Problems Over Satellite Channels (cont.)

- Comparison of data transmitted over ACTS and terrestrial network during slow start:
  - 128 segment maximum window
  - 512 byte segments



TCP Problems Over Satellite Channels (cont.)

- Small Window Problem:
  - TCP's window size is limited to 65,535 bytes.
  - TCP's maximum throughput is limited by the following equation:

$$\begin{aligned} \textit{max throughput} &= \frac{\textit{max window size}}{RTT} \\ &= \frac{\textit{65,535 bytes}}{\textit{560 ms}} \\ &\approx \textit{117,027 bytes/second} \end{aligned}$$

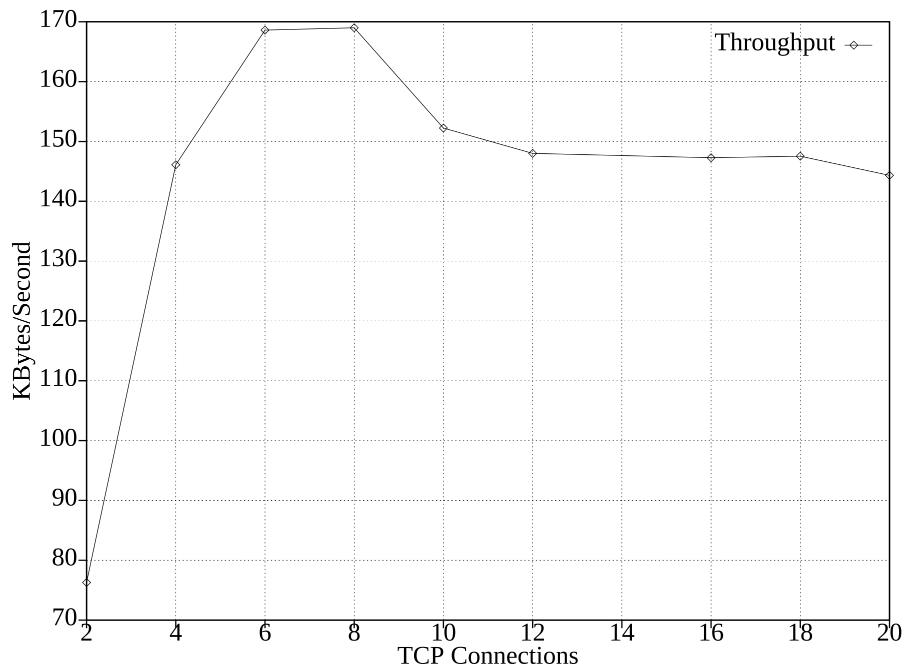
  - The maximum throughput possible is less than provided by a T1 channel (192,000 bytes/second).

Application-Level Solution

Application-Level Solution

- Modified version of FTP that uses parallel TCP connections to transfer a single file (called *XFTP*)

- XFTP Performance:



- When using 6–8 parallel connections XFTP is able to utilize 98% of the available capacity (when overhead is taken into account).

XFTP Lessons

- XFTP illustrates the need for TCP modifications:
  - Bigger windows
  - More aggressive slow start
  - More aggressive congestion avoidance
  - Selective acknowledgments
  - Estimating an appropriate point to end slow start

Slow Start Modifications

- Modifications to slow start:
  - Larger initial window
    - Eliminates a few unproductive round-trip times from the transfer
  - New window increase algorithm
    - Provides the same window increase rate regardless of whether delayed ACKs are used by the receiver

Slow Start Modifications (cont.)

- Larger Initial Window
  - Instead of initializing the window to 1 segment, initialize it to W_I segments ($W_I > 1$).
  - This modified version of slow start takes *T* seconds to increase *cwnd* from an initial window of W_I segments to an advertised window of W_A segments on a network with a RTT of *R* seconds:

$$T = R(log_2 W_A - log_2 W_I)$$

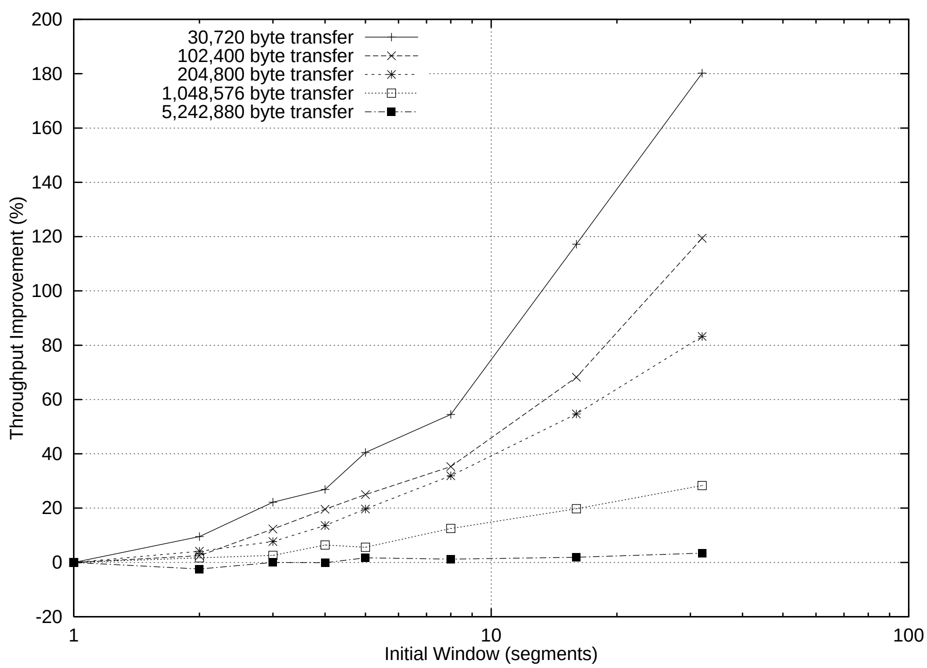
  - The time required to open the window is reduced by the time it takes slow start to increase the window to W_I segments.

Slow Start Modifications (cont.)

- New Window Increase Algorithm:
  - Instead of increasing the window by 1 segment for each ACK received, increase the window by the number of new segments covered by each incoming ACK.
  - Provides the same increase rate regardless of whether delayed ACKs are used.

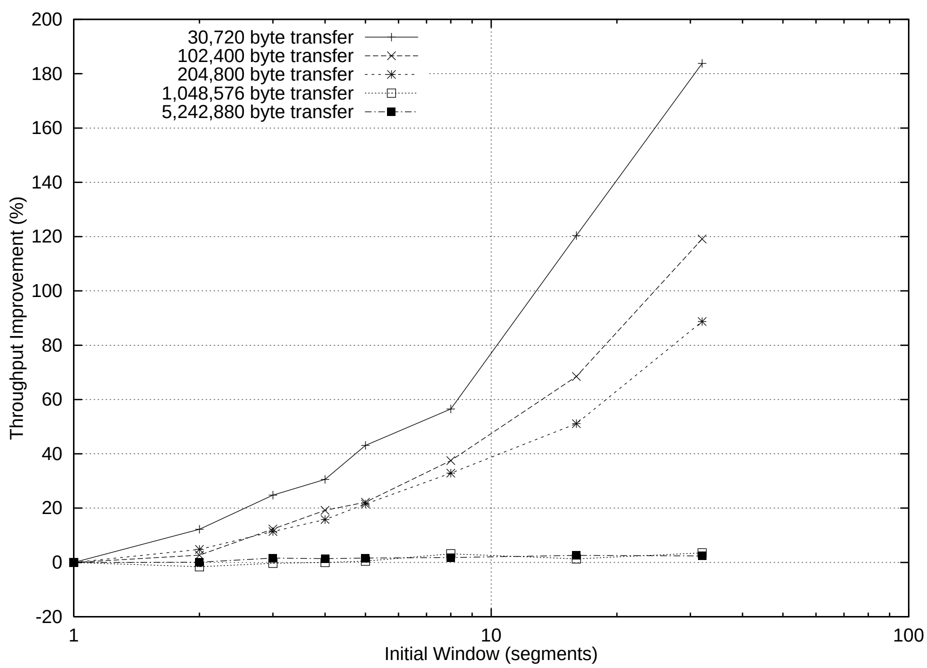
Slow Start With Larger Initial Windows

- Slow start employing larger initial windows (no congestion loss):



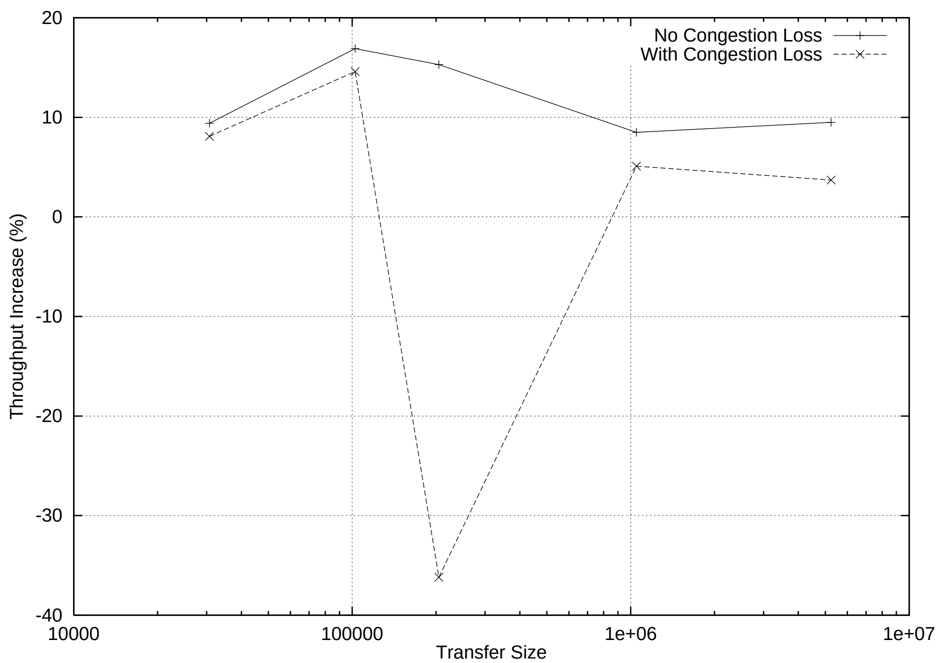
Slow Start With Larger Initial Windows (cont.)

- Slow start employing larger initial windows (with congestion loss):



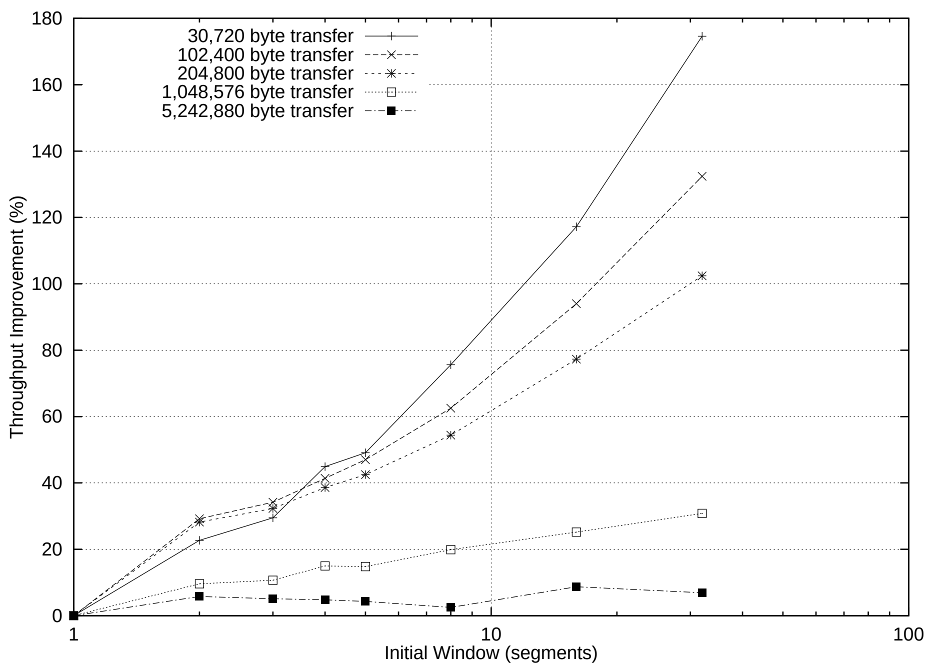
Slow Start Using a New Window Increase Algorithm

- Slow start employing a modified window increase algorithm:



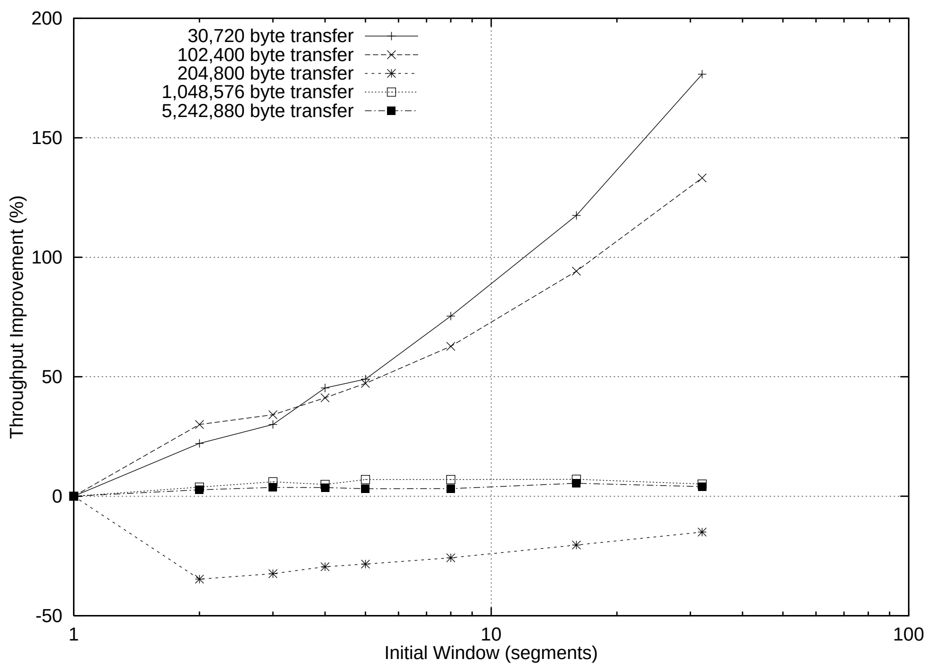
Slow Start Using Both Modifications

- Slow start using both larger initial windows and a new window increase algorithm (no congestion):



Slow Start Using Both Modifications (cont.)

- Slow start using both larger initial windows and a new window increase algorithm (with congestion):



Results

- TCP extensions our experiments have shown useful:
  - TCP Larger Window Extensions
  - TCP Selective Acknowledgment Extension
- Therefore, we recommend these TCP extensions.

Future Work

- Our research shows that the following TCP mechanisms require further study:
  - Slow start modifications
    - Further testing over the shared Internet is needed.
  - Congestion avoidance modifications.
  - Estimating an appropriate point to end slow start
    - Packet Pair (Keshav and Hoe)
    - Receiver-based Packet Pair (Paxson)
  - New loss recovery mechanisms
    - "New Reno" (Hoe)
    - Fast Recovery replacement (Floyd and Fall)
    - Forward Acknowledgments (Mathis and Mahdavi)