TCP Congestion Control in Wired and Wireless Networks

ENCS 835 Course Project
Spring 2008
April 7, 2008

Presented by: Mohamadreza Najiminaini
Professor: Ljiljana Trajkovic

Roadmap

- Introduction
- Related Work
- Slow-Start and Congestion Avoidance
- Fast Retransmit and Fast Recovery
- Various TCP Congestion Control Algorithms
- Simulation Results
- Conclusion

Introduction

- TCP mechanism for congestion control is implemented at the sender
- The congestion window size at the sender is set as:

  Send Window = MIN (flow control window, congestion window)

  - flow control window is advertised by the receiver
  - congestion window is set based on feedback from the network

Introduction

- Two significant parameters of TCP congestion control are:
  - Congestion Window size(cwnd)
  - Slow-start threshhold Value (ssthresh)
- Congestion control works in two phases:
  - slow start: cwnd < ssthresh
  - congestion avoidance: cwnd ≥ ssthresh

Related Work

- Comparison of TCP Tahoe, Reno, NewReno, SACK using ns-2:
  - K. Fall and S. Floyd, “Simulation based comparisons of Tahoe, Reno and SACK TCP,” *Computer Communications Review*, vol. 26, no. 3, July 1996, pp. 5-21
- Modified Tahoe and comparison with other TCP congestion control algorithms using Opnet Modeler:
  - M. N. Akhtar, M. A. O. Barry and H. S. Al-Raweshidy “Modified Tahoe TCP for wireless networks using OPNET simulator,” *Proc of the London Communications Symposium (LCS2003),* London, Sept 2003

Slow Start

- Initial value: cwnd = 1(MSS)
- When the sender receives an ACK, the congestion window is increased by 1 segment:

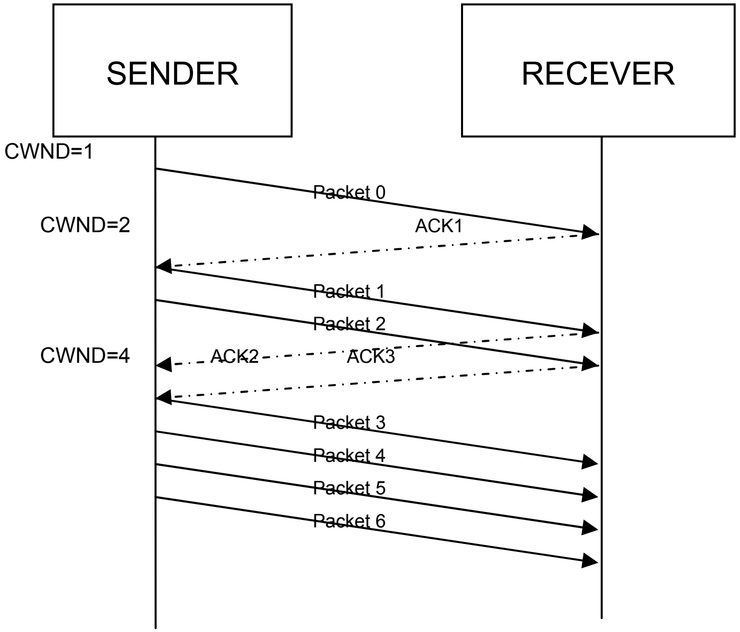
  cwnd = cwnd + 1

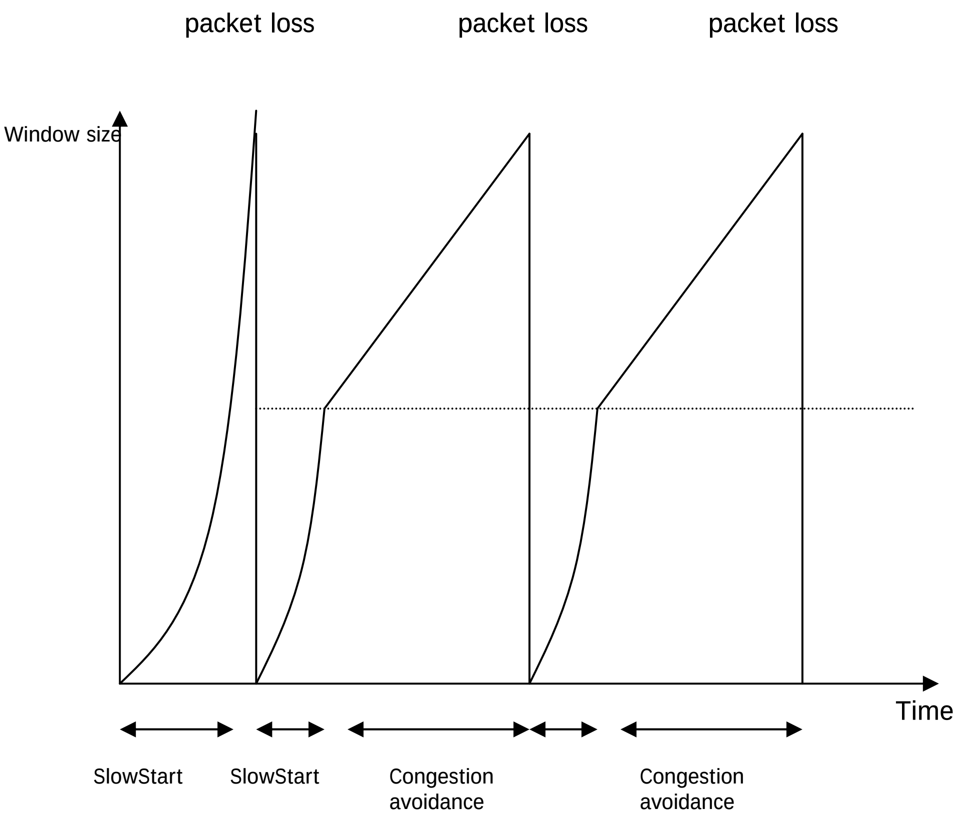
  - an ACK for two segments, cwnd is incremented by only 1 segment
  - an ACK for a segment that is smaller than MSS bytes, cwnd is incremented by 1

Congestion Avoidance

- Congestion avoidance phase initiates after cwnd reaches the slow-start threshold value
- If $cwnd \geq ssthresh$ then every time an ACK reaches to the sender, increase cwnd as:
  - $cwnd = cwnd + 1/cwnd$

Slow-Start Algorithm with Congestion Avoidance





Fast Retransmit

- Three and more duplicate ACKs indicate a lost segment
- Then TCP performs a retransmission of the lost segment, without waiting for a timeout to happen
- Enter slow start:

  ssthresh = cwnd/2

  cwnd = 1

Fast Recovery

- Fast recovery avoids slow start after a fast retransmit
- After three duplicate ACKs set:
  - Retransmit the lost segment
  - ssthresh = cwnd/2
  - cwnd = ssthresh+3
  - Increase cwnd by one for each additional duplicate ACK
- When it receives an ACK for a new packet

  cwnd=ssthresh

  enter congestion avoidance

Various TCP Congestion Control Algorithms

- TCP Tahoe (1988, FreeBSD 4.3 Tahoe)
  - Slow start
  - Congestion avoidance
  - Fast retransmit
- TCP Reno (1990, FreeBSD 4.3 Reno)
  - Fast recovery
- New Reno (1996)
- SACK (1996)

TCP Reno

- Duplicate ACKs:
  - Fast retransmit
  - Fast recovery

    Fast recovery avoids slow start
- Timeout:
  - retransmit
  - slow start
- TCP Reno performs better than TCP Tahoe when a single packet is dropped within a round-trip time.

TCP New Reno

- When multiple packets drops, Reno can not handle well
- Partial ACK:
  - happens if multiple packets are lost
  - A partial ACK does not acknowledge all packets that are outstanding at the beginning of a fast recovery (this takes sender out of fast recovery)

    Sender must wait until timeout occurs
- New Reno:
  - Partial ACK does not take sender out of fast recovery
  - New Reno can handle multiple lost segments without entering slow start

Selective Acknowledgement (SACK)

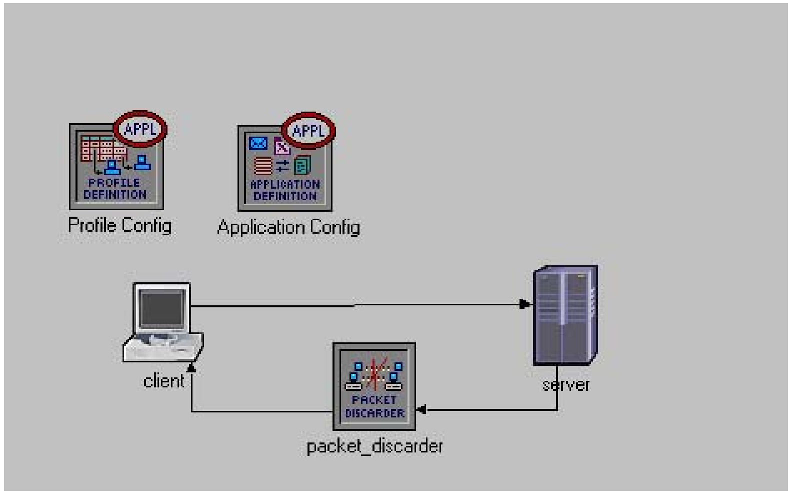
- At most 1 lost segment can be retransmitted in Reno and NewReno per round trip time
- Selective acknowledgments: acknowledges non-continuous blocks of data
- Multiple blocks can be transmitted in a single segment
- TCP SACK:
  - Initiate fast recovery upon 3 duplicate ACKs
  - Sender keeps records of SACKs and understand if segments are lost. Sender retransmits the subsequently segment from the list of the lost segments

Simulation

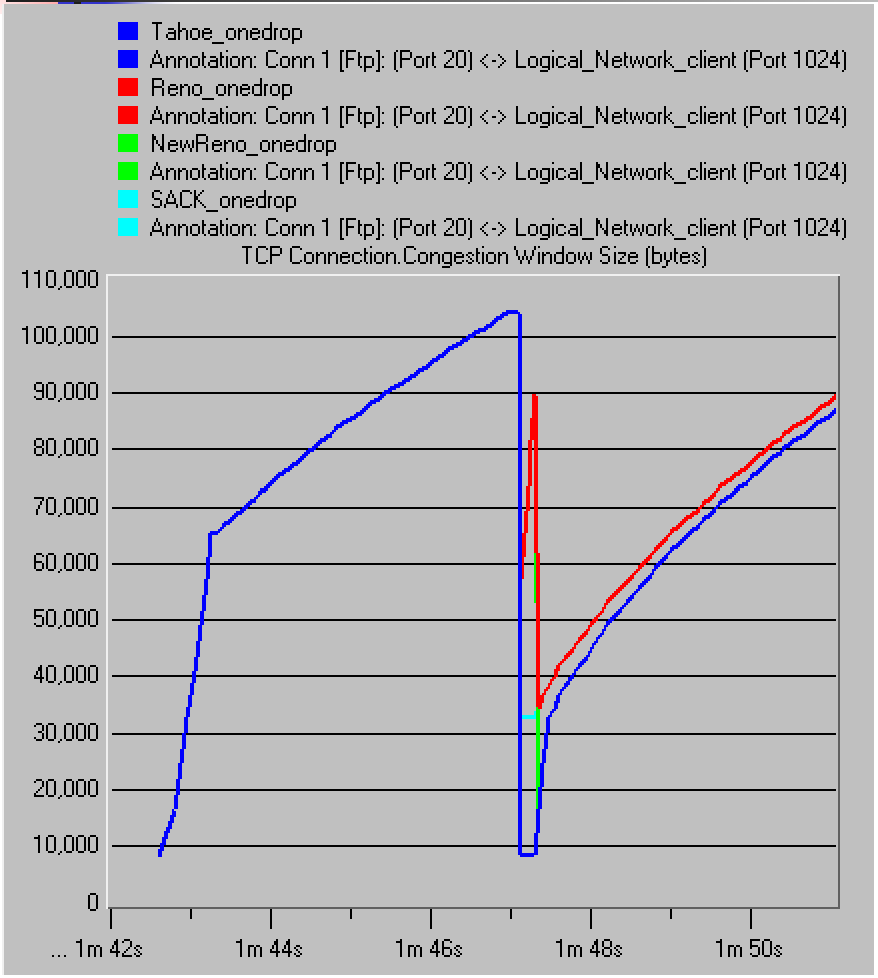
- Opnet Modeler and IT Guru
- Simulation results:
  - congestion window size (CWND)
  - sent segment sequence number (SSSN)
- Simulation scenarios:
  - Point to point client and server connection (PPP)
    - single packet drop
    - two packet drops
  - WAN topology: 0.05% packet drops and 0.001 sec packet latency
  - Wireless topology: using trajectory for packet drops

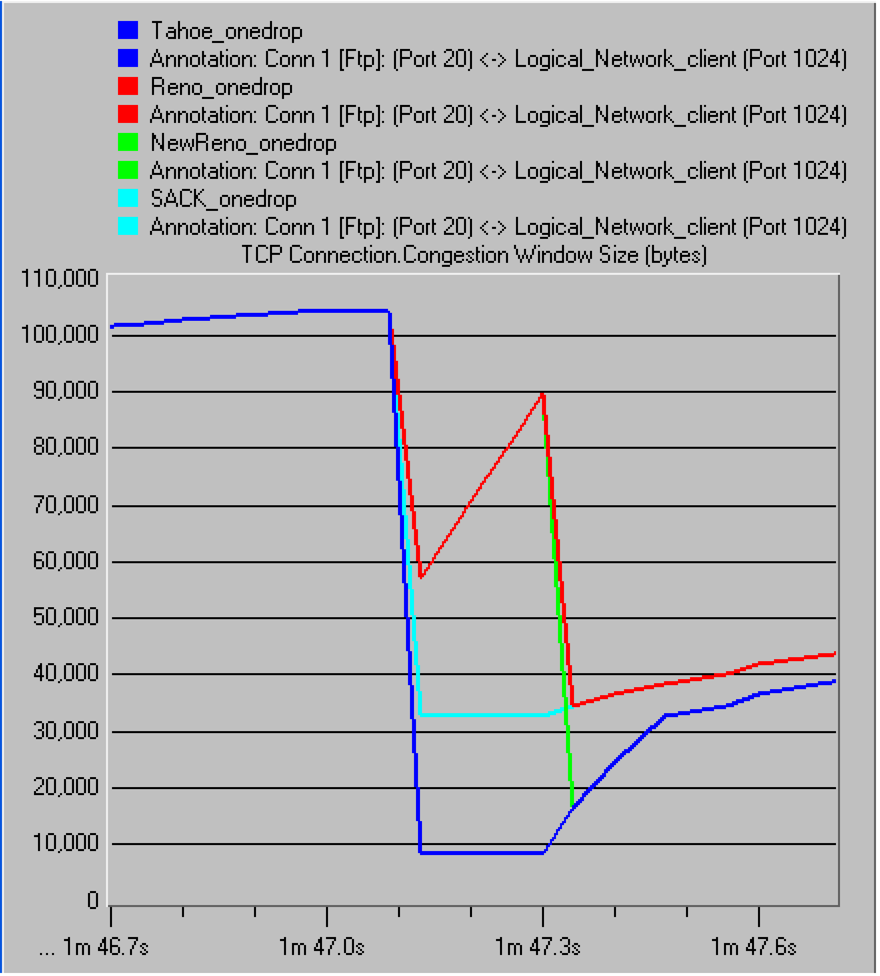
Point-to-Point Client Server connection (PPP)

- Application
  - FTP
  - constant file size
  - Constant inter-request time
- Profile
- Client
- Server: FTP server
- Packet discarder
  - drop packets

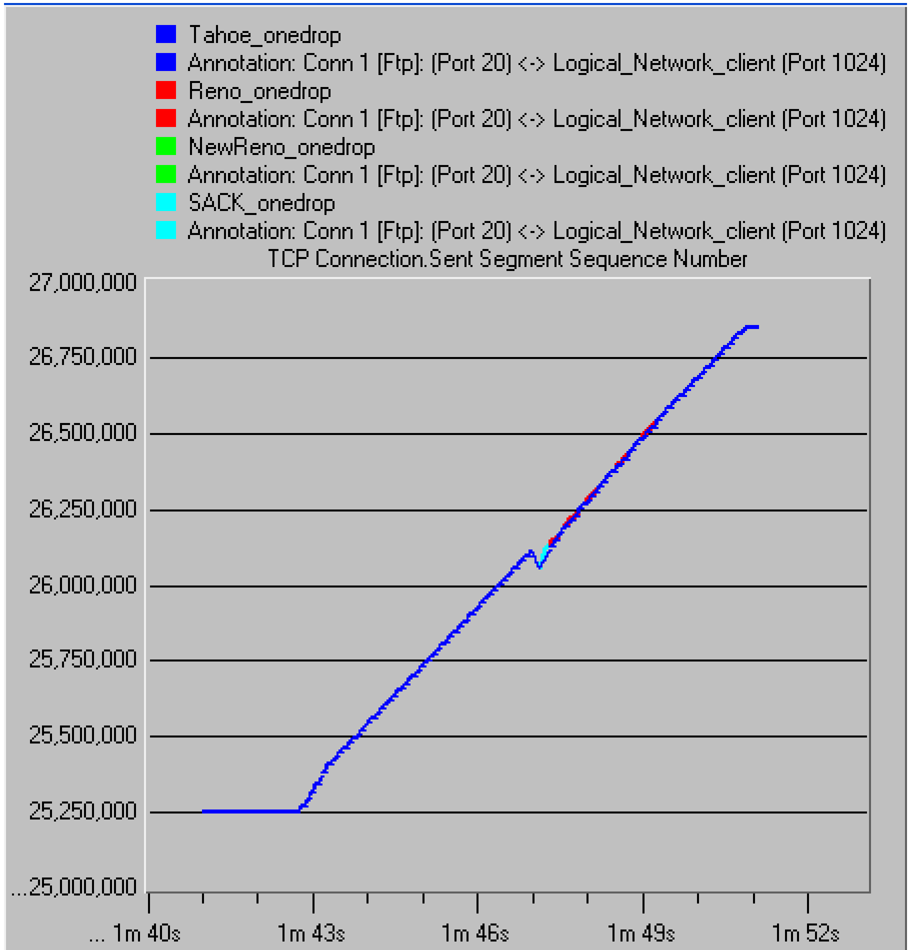


CWND in PPP Server-Client: one drop

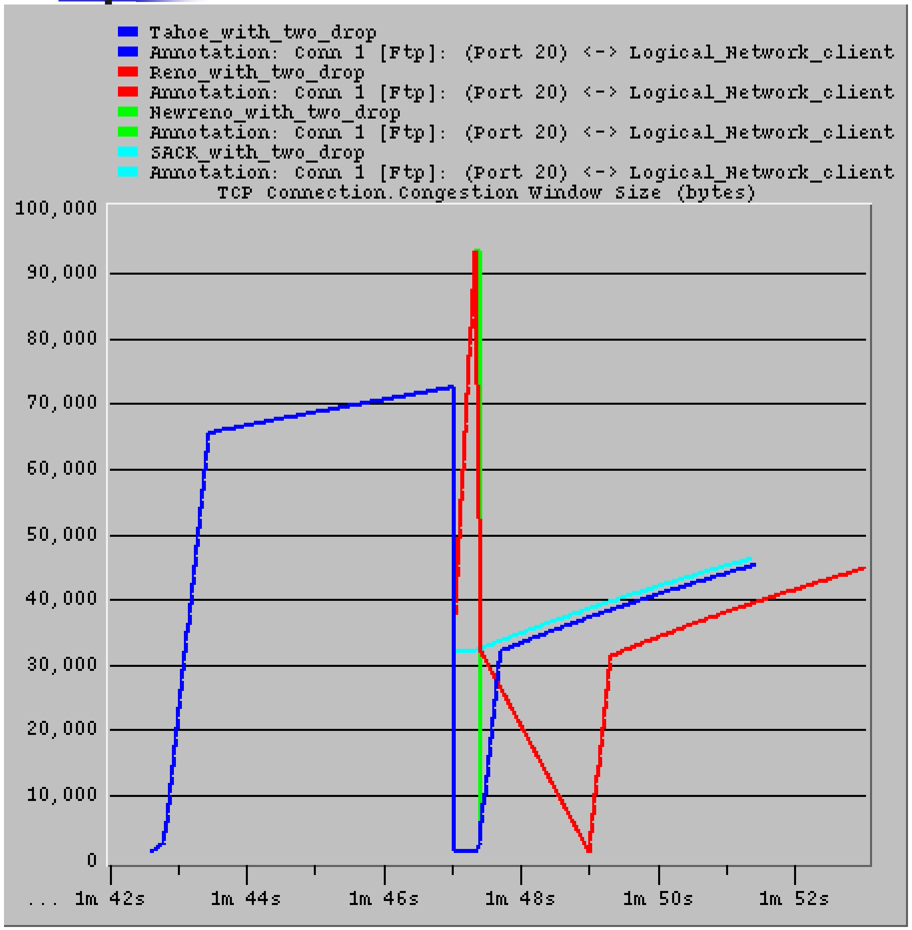


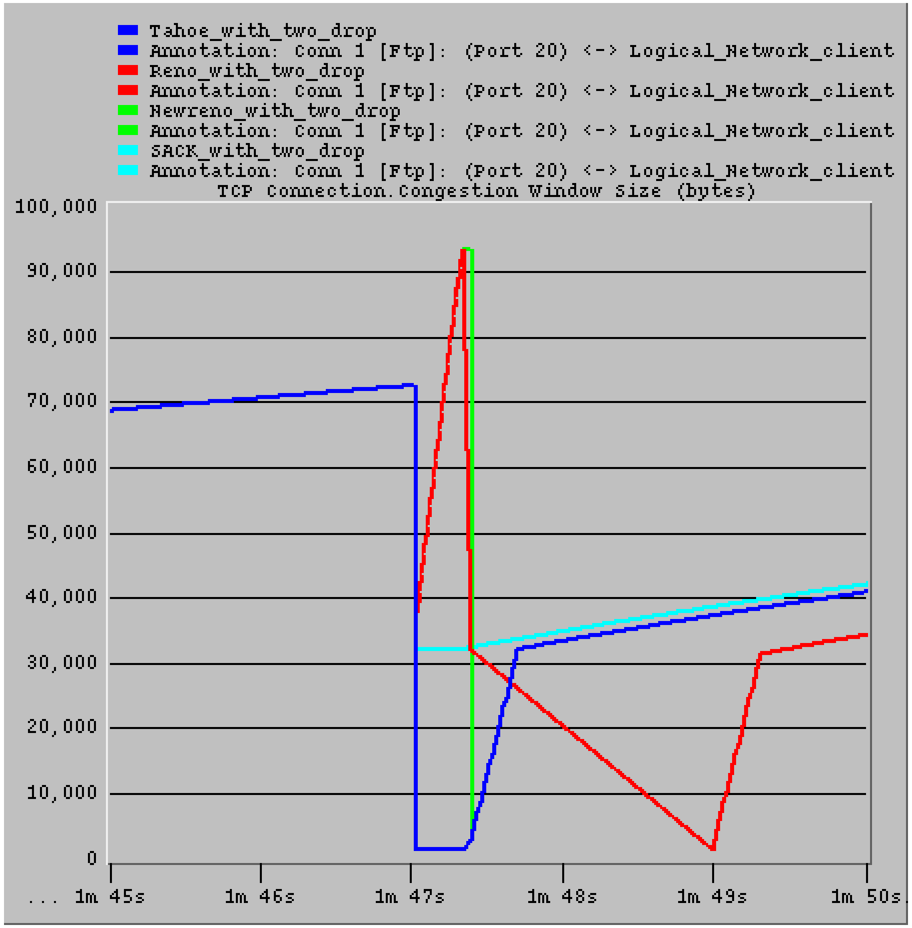


Sent Segment Sequence Number PPP Server-Client: one drop

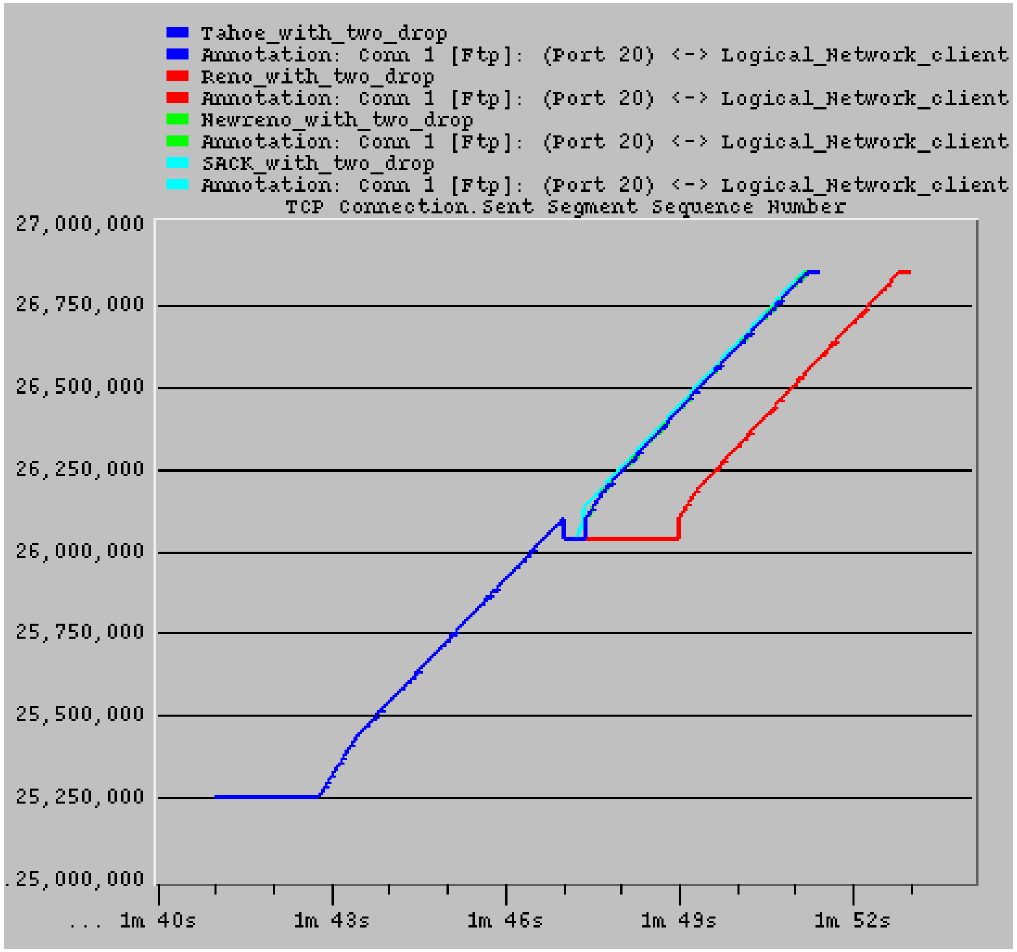


CWND in PPP Server-Client: two drops

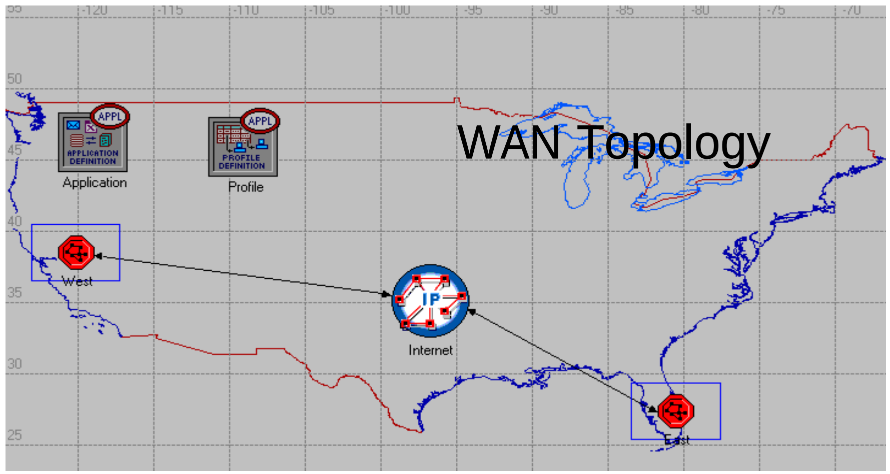


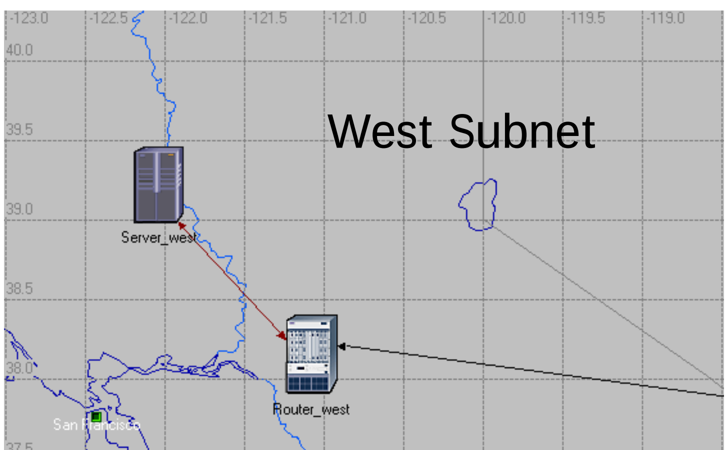


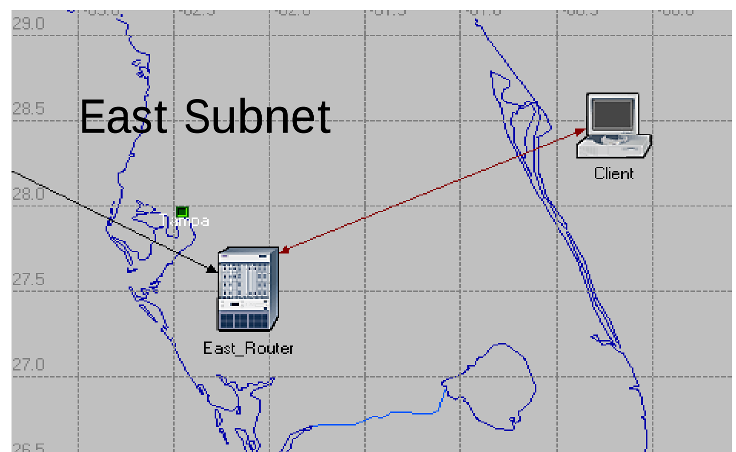
Sent Segment Sequence Number PPP Server-Client: two drops



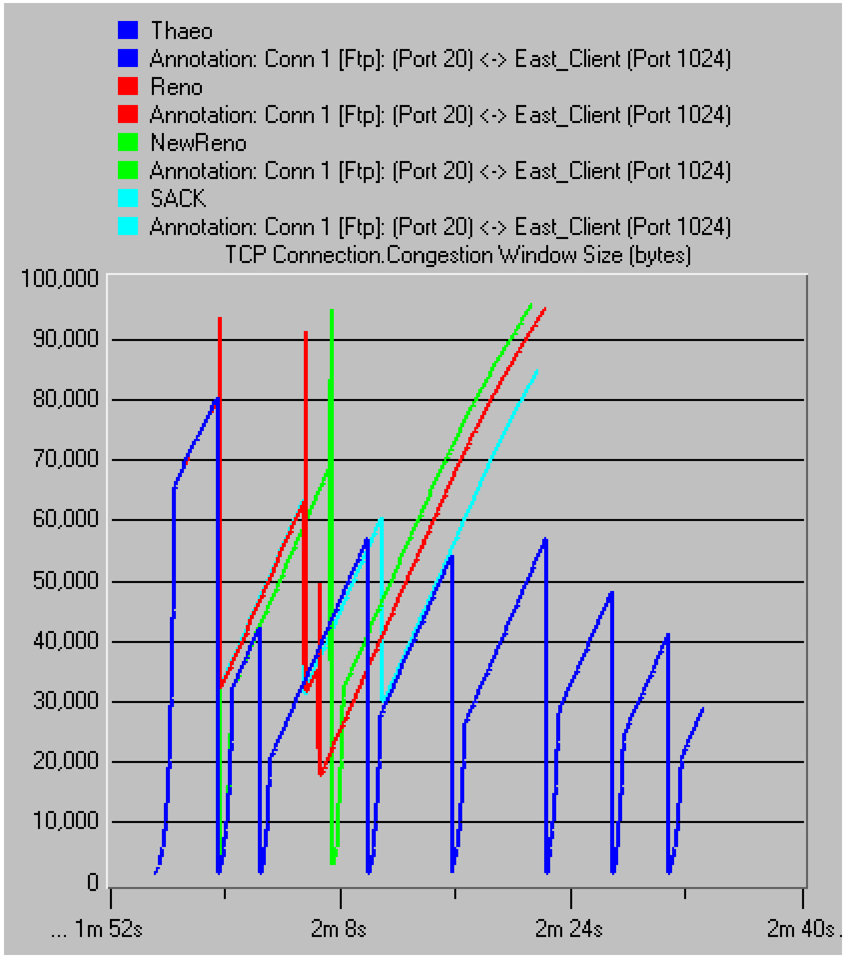
WAN Topology



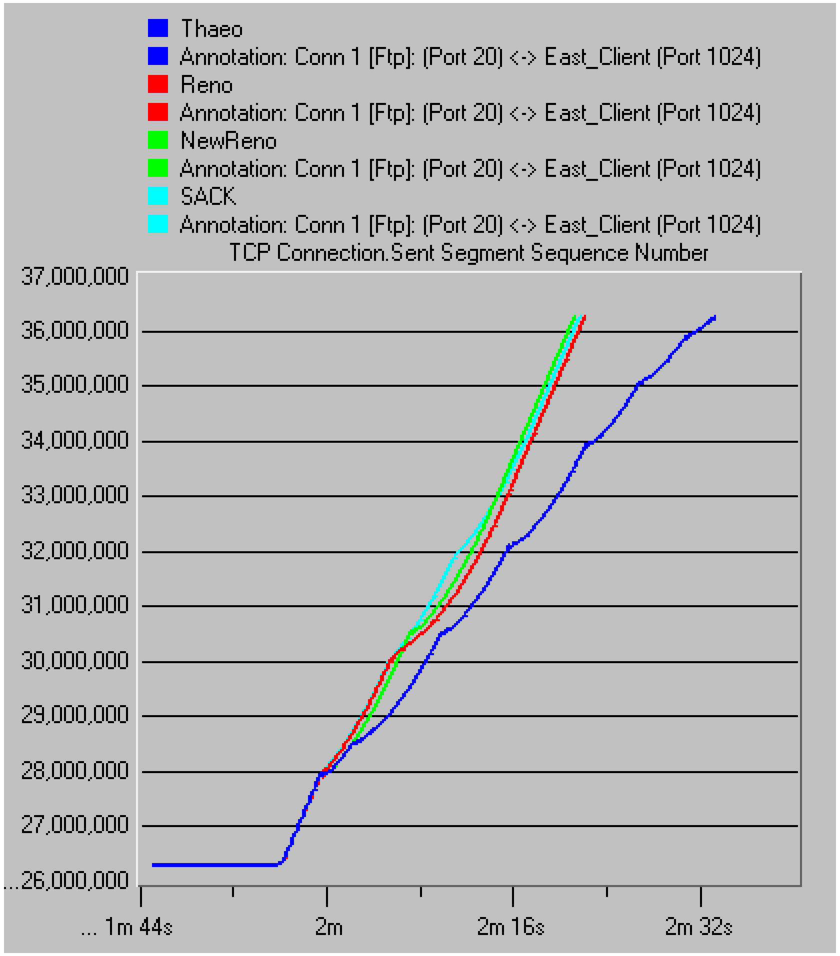




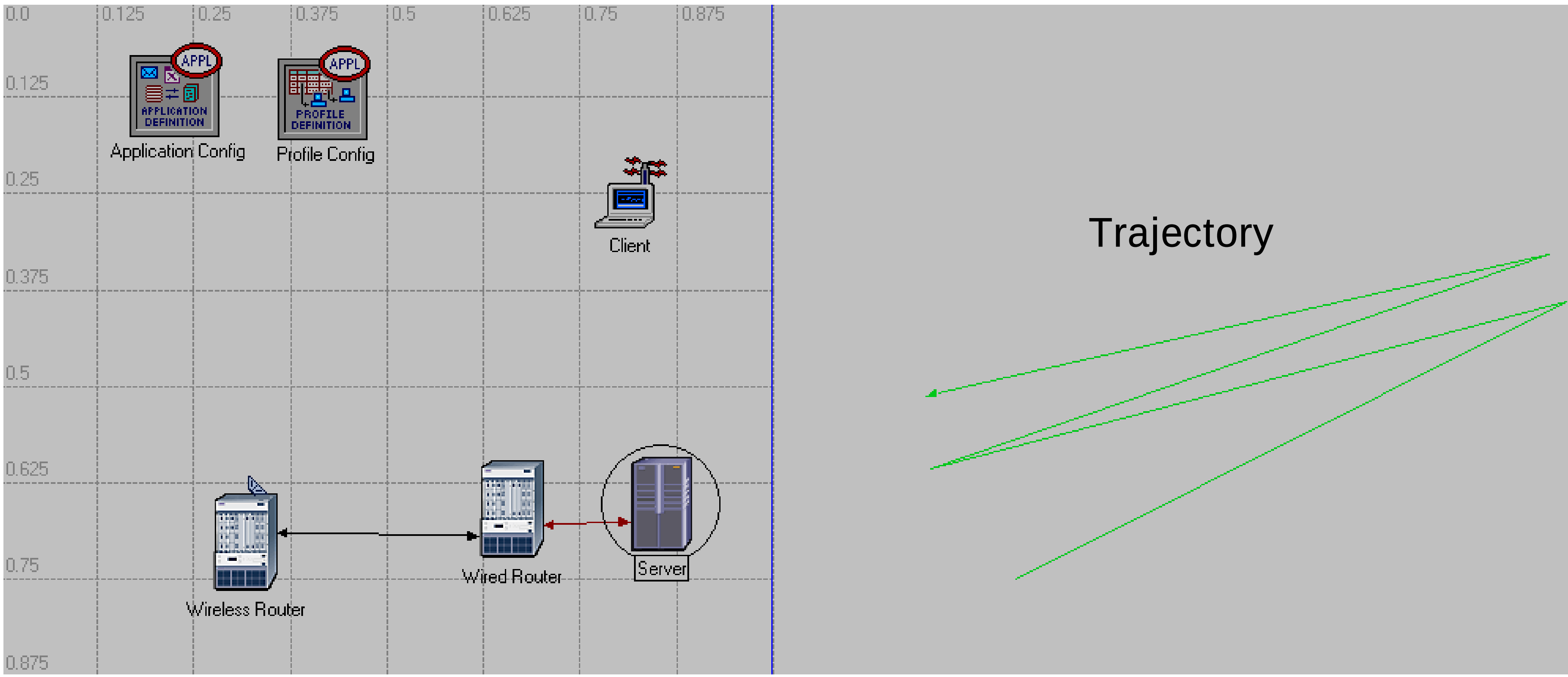
CWND in WAN: packet drop (0.05%) packet latency (0.001)



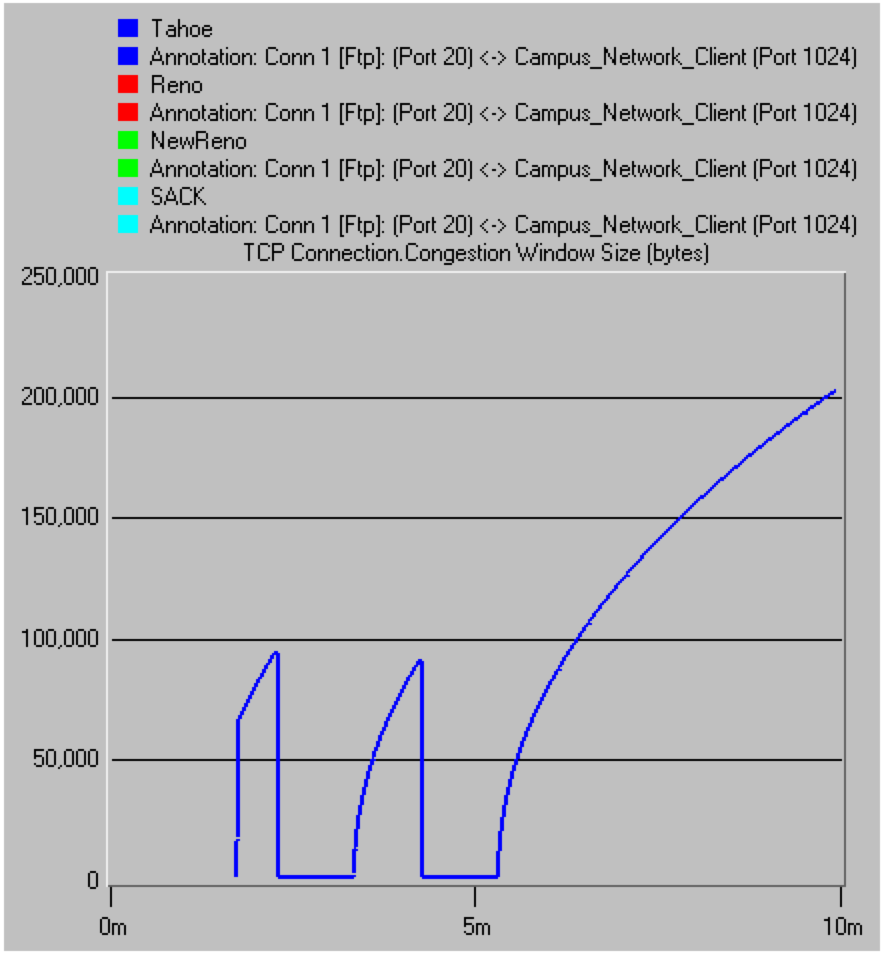
SSSN in WAN, packet drop (0.05%)-packet latency (0.001)



Wireless Topology



CWND in Wireless Topology



Conclusion

- Reno performs well only if no loss or one packet drop within a window
- NewReno can deal with multiple lost segments without entering slow start
- SACK (selective acknowledgement and selective retransmit)
- TCP congestion control algorithms do not perform satisfactory in wireless network due to signal attenuation in wireless environment.

References

- K. Fall and S. Floyd, “Simulation based comparisons of Tahoe, Reno and SACK TCP,” *Computer Communications Review*, vol. 26, no. 3, July 1996, pp. 5-21.
- M. N. Akhtar, M. A. O. Barry and H. S. Al-Raweshidy “Modified Tahoe TCP for wireless networks using OPNET simulator,” *Proc of the London Communications Symposium (LCS2003),* London, Sept 2003.
- W. Stevens, “TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms,”. Network Working Group, RFC2001, Jan 1997.
- M. Omueti and Lj. Trajkovic, “M-TCP+: using disconnection feedback to improve performance of TCP in wired/wireless networks,*” Proc. SPECTS 2007, San Diego*, CA, USA, July 2007, pp. 443-450.
- A. S. Tanenbaum, “Computer Networks,” The Transport Layer, 4rd Edition.