Comparing RTO Reduction Algorithms in TCP

Joshua Blanton
Ohio University
<jblanton@cs.ohiou.edu>

Mentors: Jim Griner and Mark Allman

Explanation of RTO

TCP (the most commonly used protocol on the internet) relies on acknowledgements sent by the receiving side of a connection to know that data is arriving.

- When data arrives in order, the sender sends new data
- When data arrives out of order, duplicate acknowledgements are sent that indicate the last data successfully received in-order so that more data can be sent or resent as necessary
- When data does not arrive, no acknowledgements are sent

Explanation of RTO (cont.)

If three duplicate acknowledgements arrive from the receiver, the sender re-sends data to fix the apparently missing data.

An RTO (retransmission time out) is caused by:

- Receiver does not send/sender does not receive three duplicate acknowledgements
- A calculated period of time passes
- Sender gives up and resends the last unacknowledged data

An RTO can massively reduce the speed at which a TCP transfer can run, and so should be avoided if possible.

Research

Several algorithms have been proposed to reduce the number of RTOs in a given TCP stream; we have chosen four to analyze. The four algorithms are:

- Larger initial window: Start a TCP connection by sending more packets than usual, to increase the number of acknowledgements sent back in an attempt to increase the window to four packets faster.
- Limited transmit: Send out another packet when a duplicate acknowledgement is received, to coax the receiver into sending more acknowledgements if necessary.
- Smart framing: Send smaller packets so there are always four packets "in flight" to increase the chance of getting three duplicate acknowledgements.
- Early retransmit: Reduce the number of duplicate acknowledgements required to re-send data, if there are not enough packets in flight to generate three duplicate acknowledgements.

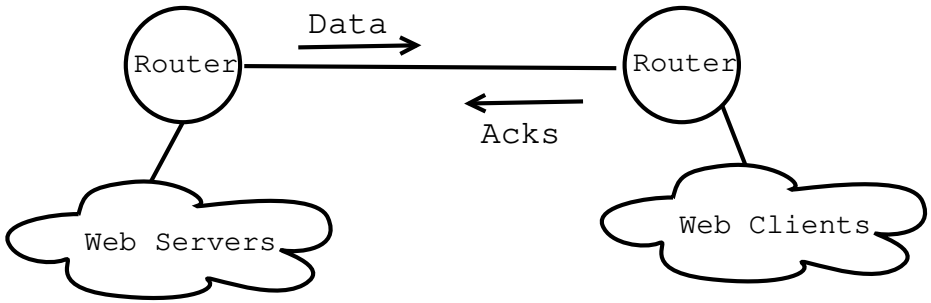
Research (cont.)

I have written code in C++ and TCL (two programming languages) to add these extensions to ns - the Network Simulator - so that analyses can be run.

Some difficulties have arisen when implementing these extensions, due to broken existing code; however, at this point I believe that the simulator is producing valid data.

Research (cont.)

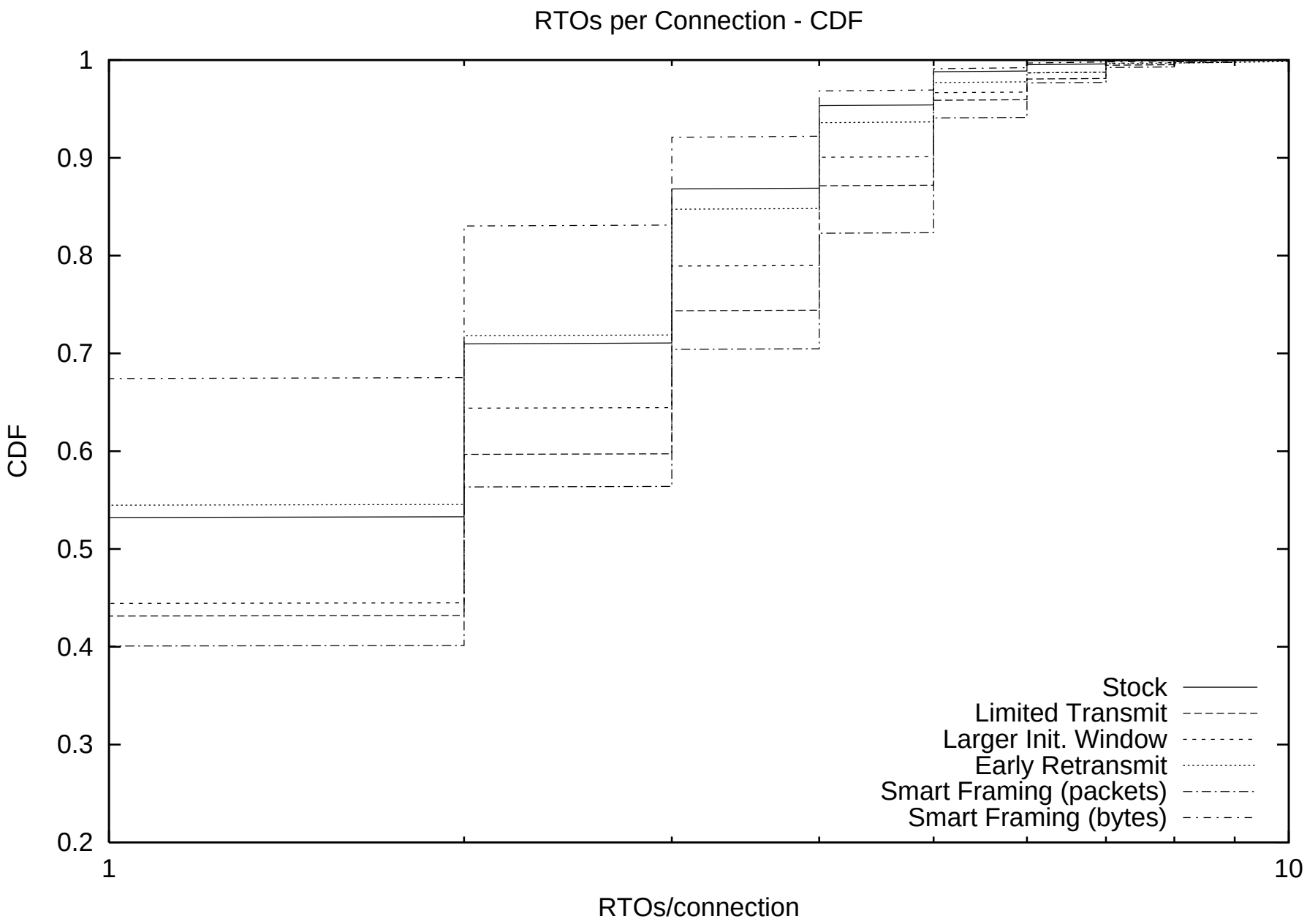
I am now running simulations using a simple link model to test each algorithm.



The number of web clients is varied to simulate an increase in traffic, to test what each algorithm does under different conditions.

All loss on the link is generated by contention between different web clients.

Preliminary Results



Additional RTO Research

This summer I have also assisted Mark Allman and Bob Dimond in designing some "trace-based" simulations to test RTO reduction algorithms.

- Trace-based simulations use actual network traffic as the basis for the simulation data and timing, rather than random number generation.
- These simulations are not necessarily more or less accurate than the ns simulations, but any validation or invalidation of the tests will help.

Future work

- Finish current simulations
- Find reasonable metrics with which to compare the different algorithms
- Write up results