Xingkun Yin

U44255956

TODO: compilation instruction

TODO: description of trace file header

TODO, highlight trace

==============================================

Stop & Wait (window = 1) Working correctly

Score is out of 10 pts

==============================================

For stop and wait, we tested several cases by setting the window size to 1 in SR.

works for no loss and no corruption

* Our code work for this one and the trace is in trace/sw/sw\_noloss\_nocorruption.txt.

# Graphical user interface, application Description automatically generated

works for loss and no corruption

* Our code show recovery from loss
* Trace can be found in trace/sw/sw\_loss\_nocorruption.txt
* recovery from DATA loss, error detection by timeout
  + Graphical user interface, application, Word

    Description automatically generated
  + The screen shot above shows a data packet from A is lost , it is detected by timeout and resent. The packet is then successfully collected by B.
* recovery from ACK loss, error detection by timeout
  + Graphical user interface, application, Word

    Description automatically generated
  + The above is a screen of a loss in ACK packet and was detected by timeout.
* A computer screen capture

  Description automatically generated with medium confidence
* The above is the result for loss but no corruption, stop and wait protocol. It can be found in trace/sw/sw\_loss\_nocorruption.txt.

works for corruption and no loss

* Our code show recovery from corruption
* Trace can be found in trace/sw/sw\_noloss\_corruption.txt
* recovery from DATA corruption, error detection by timeout
  + Graphical user interface, application, Word

    Description automatically generated
  + The screen shot above shows a data packet from A is corrupted and detected by B. The packet is resent and then successfully collected by B.
* recovery from ACK corruption , error detection by timeout
  + Graphical user interface, application, Word

    Description automatically generated
  + The above is a screen of a corruption in ACK packet and was detected by A.
* Graphical user interface, application

  Description automatically generated
* The above is the result for corruption but no lost, stop and wait protocol. It can be found in trace/sw/sw\_noloss\_corruption.txt.

works for both loss and corruption

* A computer screen capture

  Description automatically generated with medium confidence
* The above is the result for corruption andlost, stop and wait protocol. It can be found in trace/sw/sw\_loss\_corruption.txt.

==============================================

SR (window > 1)

Score is out of 90 pts

==============================================

C1: works for no loss + no corruption

* Text, letter

  Description automatically generated
* Our code work for this one and the trace is in trace/sr/ sr\_noloss\_nocorruption.txt

**For the following scenario, C2 to C5, the trace are in trace/sr/ sr\_loss\_corruption.txt**

C2: identify (on output trace) case where ack is lost/corrupted and a later cumulative ack moves sender window by more than 1

* Text

  Description automatically generated
* In the image above, the ACK packet is lost can cause the window in A to move from base at 10 to 12.

C3: identify (on output trace) case where when data packet is lost/corrupted, and data is retransmitted after RTO

* Text

  Description automatically generated
* In the screen shot above, the Data packet is dropped

C4: identify (on output trace) case where when data packet is lost/corrupted, and data is retransmitted after receiving duplicate ack

* Text

  Description automatically generated
* In the screen shot above, the data packet is lost and it is retransmitted after a duplicate ack is received.

C5: identify (on output trace) case where when data packet is lost/corrupted, and the retransmitted data is delivered and a cumulative ack moves the sender window by more than 1

* Text

  Description automatically generated
* In the screen shot above, a data packet is lost and cause the sender window to go from 12 to 14.

**Logistics**

* TODO, overall logic
* Our checksum related functions include addChecksum(), calculateChecksum() and evaluateChecksum()
  + The way the checksum is calculated is:
    - Checksum = seqnum + seqnum + Character.getNumericVale() for evey character in the payload.
* TODO, FSM
* Our code is readable as each state in the FSM is in a single clause of if else clauses. We have a lot of comments describe how the code works. The helper functions and attributes we added are in java regions, i.e.
  + //region helper function
  + //endregion
* TODO, how it works
* Possible tradeoff:
  + We are storing all packet from layer 5 to A, and related time in a array list which is the same length as the amount of packets A receive from layer 5.
* Extension
  + We may reduce the usage of memory by using a constant array to store the information and delete the packets in the past that we no longer need.
* Compilation instruction is at the top of this report.