

Shawn McManus

Intro to Computer Networking

Project RDT

- Where did the bulk of logic occur?

The bulk of the logic took place in the `processReceiveAndSendResponse` function. This is where we handled any issues with the packets (out of order, loss packets, corrupted packets, etc.). This required first checking all the items in the receive channel satisfied the checksum property. Then, we partitioned into two lists depending whether the packets were ACKs or DATA. Then, the necessary actions were carried out to adjust the sliding window (if ACKs were received) or correctly read and increment out last-byte-received counter (if DATA were received).

- How were the timeouts resolved? What happened if timeouts have been resolved? (3+ 2 pts)

Each sent segment is tracked in `send_buffer` with a `last_sent` timestamp (the iteration it was last transmitted). The sender periodically checks all outstanding segments and if `currentIteration - last_seng >= TIMEOUT` for a given segment, the sender retransmits that segment and updates the `last_sent` to the current iteration. Then, the `countSegmentTimeouts` is incremented for record keeping.

When the timeouts have been resolved, the sender delivers the missing segment and sends a cumulative ACK covering that data, the sender removes all `send_buffer` entries whose bytes are less than or equal to the ACK value. The sender then advances the base, which is the left edge of the buffer window, to the smallest outstanding sequence number or `next_seq` if there are no outstanding segments. This frees window slots continuously. In summation, we have timeouts - > retransmit -> remove entries and slide window when ACK'ed -> pipeline resumes.

- How was the packet dropping handled? (2 pts)

The receiver verifies each incoming Segment with `seg.checkChecksum()`. Corrupted segments are dropped and counted (`countDroppedDataPackets` or `countDroppedAckPackets` as applicable). The receiver still sends a cumulative ACK for the last in-order byte it successfully received (`ack = expected_seq - 1`), so the sender knows exactly what is missing.

- The retransmission policy used both timeout-based and duplicate-ACK-based recovery. Each sent segment is tracked in `send_buffer` with its send time, and if no ACK arrives within the timeout window, that segment is rebuilt and retransmitted, ensuring eventual delivery. The sender also counts duplicate ACKs and if the same ACK is received three times, it immediately retransmits the earliest unacknowledged segment (fast retransmit) without waiting for the timer to expire. This allows quicker recovery from isolated losses. All retransmitted packets update their `last_sent` timestamp and are tracked through cumulative ACKs to prevent repeated or unnecessary resends.

```
rdt_layer.py 9+  rdt_main.py 9+  unreliable.py 9+  segment.py 9+
rdt_main.py > ...
PROBLEMS 235 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [ ] [ ] [ ] [ ] [ ] [ ]

Press enter to continue...

-----
Time (iterations) = 3
Client-----
Timeout: retransmitting seg 0 at iter 3 (timeouts = 5)
Timeout: retransmitting seg 4 at iter 3 (timeouts = 6)
Timeout: retransmitting seg 8 at iter 3 (timeouts = 7)
Timeout: retransmitting seg 12 at iter 3 (timeouts = 8)
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 15, data:
Main-----
DataReceivedFromClient: We have a workin
Press enter to continue...

-----
Time (iterations) = 4
Client-----
Sending segment: seq: 16, ack: -1, data: g rd
Sending segment: seq: 20, ack: -1, data: t pr
Sending segment: seq: 24, ack: -1, data: otoc
Sending segment: seq: 28, ack: -1, data: oll!
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Delivered chunk at seq 16: 'g rd'
Delivered chunk at seq 20: 't pr'
Delivered chunk at seq 24: 'otoc'
Delivered chunk at seq 28: 'oll!'
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 30, data:
Main-----
DataReceivedFromClient: We have a working rdt protocol!
$$$$$$$ ALL DATA RECEIVED $$$$$$$
countTotalDataPackets: 16
countSentPackets: 20
countChecksumErrorPackets: 1
countOutOfOrderPackets: 0
countDelayedPackets: 3
```

Screenshots:

(smaller input)

```

DataReceivedFromClient: The quick brown fox jump
-----
Time (iterations) = 12
Client-----
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Length of Receive Unacked Packets List: 2
Sending ack:  seq: -1, ack: 27, data:
Main-----
DataReceivedFromClient: The quick brown fox jumped o
-----
Time (iterations) = 13
Client-----
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Length of Receive Unacked Packets List: 2
Sending ack:  seq: -1, ack: 27, data:
Main-----
DataReceivedFromClient: The quick brown fox jumped o
-----
Time (iterations) = 14
Client-----
Sending segment: seq: 40, ack: -1, data: dog
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 43, data:
Main-----
DataReceivedFromClient: The quick brown fox jumped over the lazy dog
$$$$$$$ ALL DATA RECEIVED $$$$$$$
countTotalDataPackets: 29
countSentPackets: 52
countChecksumErrorPackets: 4
countOutOfOrderPackets: 3
countDelayedPackets: 3
countDroppedDataPackets: 3
countAckPackets: 14
countDroppedAckPackets: 3
# segment timeouts: 19
TOTAL ITERATIONS: 14
PS C:\Users\Shawn\Documents\Oregon State\2025\Fall 2025\Intro to Computer Networking\Project RDT>

```

(larger input)

e of new metal alloys, some of which have not yet been invented, capable of standing heat and stresses several times more than have ever been experienced, fitted to
 gether with a precision better than the finest watch, carrying all the equipment needed for propulsion, guidance, control, communications, food and survival, on an
 untried mission, to an unknown celestial body, and then return it safely to earth, re-entering the atmosphere at speeds of over 25,000 miles per hour, causing heat
 about half that of the temperature of the sun--almost as hot as it is here today--and do all this, and do it right, and do it first before this decade is out.

```

JFK - September 12, 196
-----
Time (iterations) = 368
Client-----
Sending segment: seq: 1240, ack: -1, data: 2

Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: -1, data:
Server-----
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1242, data:
Main-----
DataReceivedFromClient:

```

...We choose to go to the moon. We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, because th
 at goal will serve to organize and measure the best of our energies and skills, because that challenge is one that we are willing to accept, one we are unwilling to
 postpone, and one which we intend to win, and the others, too.

...we shall send to the moon, 240,000 miles away from the control station in Houston, a giant rocket more than 300 feet tall, the length of this football field, mad
 e of new metal alloys, some of which have not yet been invented, capable of standing heat and stresses several times more than have ever been experienced, fitted to
 gether with a precision better than the finest watch, carrying all the equipment needed for propulsion, guidance, control, communications, food and survival, on an
 untried mission, to an unknown celestial body, and then return it safely to earth, re-entering the atmosphere at speeds of over 25,000 miles per hour, causing heat
 about half that of the temperature of the sun--almost as hot as it is here today--and do all this, and do it right, and do it first before this decade is out.

```

JFK - September 12, 1962

$$$$$$$ ALL DATA RECEIVED $$$$$$$
countTotalDataPackets: 731
countSentPackets: 1424
countChecksumErrorPackets: 61
countOutOfOrderPackets: 44
countDelayedPackets: 164
countDroppedDataPackets: 95
countAckPackets: 325
countDroppedAckPackets: 27
# segment timeouts: 461
TOTAL ITERATIONS: 368
PS C:\Users\Shawn\Documents\Oregon State\2025\Fall 2025\Intro to Computer Networking\Project RDT>

```

Additional Comments/questions

I have the code transmitting correctly so that the complete, right message is always fully translated, no matter which issue is selected to be true. However, I think I may be sending some unnecessary packets which might be a problem for larger text sizes as my “Sent Packets” output is higher for the large file than the example screenshot provided.