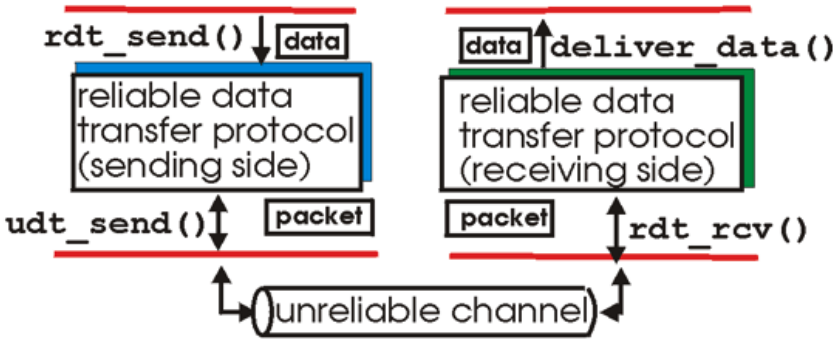
Programming Project 2 – Reliable Data Transmission (RDT)

Introduction

In this project you will code part of a simulation of reliable data transmission over an unreliable underlying system. You will complete the RDTLayer class that allows the transfer of string data through an unreliable channel in a simulated send-receive environment.



The provided code includes the **UnreliableChannel** class, the **Segment** class (for data and acknowledgment packets), and the main function. Your task is to finish the implementation of the **RDTLayer** class. When you review the rdt_main.py file, you'll notice that the **RDTLayer** class is used for both the client and server. Sending data happens in the processSend() method, and receiving data happens in the processReceiveAndSendRespond() method. Each method must determine if the instance is a client or server and act accordingly, meaning the **RDTLayer** will handle sending and receiving data and acknowledgment segments.

Your code should be able to simulate many of the features discussed in your textbook for RDT and TCP. Note that we will not be using actual TCP headers or bytes. This is more high-level than that, but the principles are the same.

RDTLayer

The RDTLayer class needs to provide reliable transport no matter how bad the unreliable channel is at delivering packets. And it is bad! The UnreliableChannel class may do any of the following:

- Deliver packets out-of-order (data or ack packets)
- Delay packets by several iterations (data or ack packets)
- Drop packets (data or ack packets)
- Experience errors in transmission (failed checksum - data packets only)
- Various combinations of the above

***The words 'segment' and 'packet' are frequently used interchangeably with TCP communications.*

The RDTLayer class must handle all of these and deliver the data correctly and as efficiently as possible. In this project, you must finish the implementation of the RDTLayer class. To help build your RDTLayer implementation, the UnreliableChannel class takes flags that can turn on/off the different types of unreliability. It is recommended that you use those flags, starting with completely reliable channels, get that working, and go from there, enabling each of the unreliable features in turn.

Note that the Segment class already has methods for calculating and checking the checksum. It also has methods for saving a 'timestamp' (iteration count). We will be using iteration count instead of time, because the simulation executes much too quickly to use actual time.

Your final RDTLayer implementation must:

- Run successfully with the original provided UnreliableChannel and Segment classes
- Deliver all of the data with no errors
- Succeed even with all of the unreliable features enabled
- Send multiple segments at a time (pipeline) (This project program is operating in discrete time slices/iterations rather than continuous time. If you are sending the maximum amount of data you can at each iteration (based on the state of the available flow control window) instead of strictly a single segment at each iteration, you are likely meeting the pipeline requirement.)
- Utilize a flow-control 'window'
- Utilize Go-Back-N or selective retransmit
- Include segment 'timeouts' (use iteration count, not actual time)
- Abide by the payload size constant provided in the RDTLayer class
- Abide by the flow-control window size constant provided in the RDTLayer class
- Efficiently run with the fewest packets sent/received. Who can transmit all of the data with the fewest iterations?

Example screenshots:

Output with smaller texts:

```
-----------------------------------------------------------------------------------------------------------
Time (iterations) = 5
Client----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1, data:
Sending segment:  seq: 13, ack: -1, data: rkin
Sending segment:  seq: 17, ack: -1, data: g rd
Sending segment:  seq: 21, ack: -1, data: t pr
Server----------------------------------------
Length of Receive Unacked Packets List: 9
Sending ack:  seq: -1, ack: 5, data:
Main----------------------------------------
DataReceivedFromClient: We h
-----------------------------------------------------------------------------------------------------------
Time (iterations) = 6
Client----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1, data:
Sending segment:  seq: 25, ack: -1, data: otoc
Sending segment:  seq: 29, ack: -1, data: ol!
Sending segment:  seq: 33, ack: -1, data:
Server----------------------------------------
Length of Receive Unacked Packets List: 11
Sending ack:  seq: -1, ack: 5, data:
Main----------------------------------------
DataReceivedFromClient: We h
-----------------------------------------------------------------------------------------------------------
Time (iterations) = 7
Client----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1, data:
Sending segment:  seq: 5, ack: -1, data: ave
Sending segment:  seq: 9, ack: -1, data: a wo
Sending segment:  seq: 37, ack: -1, data:
Server----------------------------------------
Length of Receive Unacked Packets List: 2
Sending ack:  seq: -1, ack: 41, data:
Main----------------------------------------
DataReceivedFromClient: We have a working rdt protocol!
$$$$$$$$ ALL DATA RECEIVED $$$$$$$$
countTotalDataPackets: 20
countSentPackets: 25
countChecksumErrorPackets: 4
countOutOfOrderPackets: 2
countDelayedPackets: 1
countDroppedDataPackets: 2
countAckPackets: 7
countDroppedAckPackets: 0
# segment timeouts: 11
TOTAL ITERATIONS: 7
```

Output with larger texts:

What to turn in

1. Include a copy of all your code.
2. Include the answers and screenshots for the following questions:
    - Where did the bulk of logic occur? (3 pts)

The bulk of the logic occurred in processSend() and processReceiveAndSendRespond(). They were responsible for managing the sending and handling of segments of data chunks and acknowledgements.

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

Times are detected in the processSend() function. After each iteration, it checks if an ack has been received for the sent segments, if no acks are received after 5 interactions, a timeout is triggered. If timeouts had been resolved, the program increments the countSegmentTimeouts variable and keeps resending the window until the acks are received.

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

Packet dropping was handled by detecting when no ack packets were received for sent data. When a timeout occurs, the program resends the unacked segments from the current window.

- How was the retransmission policy implemented (5 pts)? Please provide a screenshot of the output to show the policy clearly.

The retransmission policy is implemented by monitoring timeouts for sent packets. If no ack is received for a segment after a point, the program finds the current unacked packets and resends those segments. The window slides forward as acks are received until all the data is transmitted successfully.
Screenshot:

```
Timeout reached. Resending window: [0, 4]
Sending segment: seq: 0, ack: -1, data: The
Sending segment: seq: 4, ack: -1, data: quic
```

3. In rdt_main.py, you will see two different length texts to send through the RDTLayer. Start with the smaller one, then try the larger one.
4. Take screenshots of your running code for both the larger and smaller texts and include it in your submission doc.

```
----------------------------------------------------------------------------------------------------------------
Time (iterations) = 5
Client-------------------------------------------
Received ACK for sequence number 32
Received ACK for sequence number 32
Received ACK for sequence number 32
Sending segment: seq: 28, ack: -1, data: ver
Sending segment: seq: 32, ack: -1, data: the
Server-------------------------------------------
Received data segment with seqnum 28 and data: ver
Received data segment with seqnum 32 and data: the
Main---------------------------------------------
DataReceivedFromClient: The quick brown fox jumped over the
Press enter to continue...
----------------------------------------------------------------------------------------------------------------
Time (iterations) = 6
Client-------------------------------------------
Received ACK for sequence number 40
Received ACK for sequence number 40
Received ACK for sequence number 40
Sending segment: seq: 36, ack: -1, data: lazy
Sending segment: seq: 40, ack: -1, data:  dog
Server-------------------------------------------
Received data segment with seqnum 36 and data: lazy
Received data segment with seqnum 40 and data:  dog
Main---------------------------------------------
DataReceivedFromClient: The quick brown fox jumped over the lazy dog
$$$$$$$$ ALL DATA RECEIVED $$$$$$$$
countTotalDataPackets: 17
countSentPackets: 34
countChecksumErrorPackets: 0
countOutOfOrderPackets: 0
countDelayedPackets: 0
countDroppedDataPackets: 0
countAckPackets: 17
countDroppedAckPackets: 0
# segment timeouts: 0
TOTAL ITERATIONS: 6
```

```
--------------------------------------------------------------------------------------------------------------
Time (iterations) = 119
Client-----------------------------------------
Received ACK for sequence number 944
Received ACK for sequence number 944
Received ACK for sequence number 944
Sending segment: seq: 940, ack: -1, data: d th
Sending segment: seq: 944, ack: -1, data: en r
Server-----------------------------------------
Received data segment with seqnum 940 and data: d th
Received data segment with seqnum 944 and data: en r
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 a
re easy, but because they are hard, because that 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 w
e 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 fee
t tall, the length of this football field, made 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 together with a precision be
tter 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 r
Press enter to continue...
--------------------------------------------------------------------------------------------------------------
Time (iterations) = 120
Client-----------------------------------------
Received ACK for sequence number 952
Received ACK for sequence number 952
Received ACK for sequence number 952
Sending segment: seq: 948, ack: -1, data: etur
Sending segment: seq: 952, ack: -1, data: n it
Server-----------------------------------------
Received data segment with seqnum 948 and data: etur
Received data segment with seqnum 952 and data: n it
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 a
re easy, but because they are hard, because that 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 w
e 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 fee
t tall, the length of this football field, made 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 together with a precision be
tter 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
Press enter to continue...
```

```
-------------------------------------------------------------------------------------------------------------
Time (iterations) = 156
Client-----------------------------------------
Received ACK for sequence number 1240
Received ACK for sequence number 1240
Received ACK for sequence number 1240
Sending segment: seq: 1236, ack: -1, data:  196
Sending segment: seq: 1240, ack: -1, data: 2

Server-----------------------------------------
Received data segment with seqnum 1236 and data:  196
Received data segment with seqnum 1240 and data: 2

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 a
re easy, but because they are hard, because that 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 w
e 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 fee
t tall, the length of this football field, made 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 together with a precision be
tter 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--almos
t 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: 467
countSentPackets: 934
countChecksumErrorPackets: 0
countOutOfOrderPackets: 0
countDelayedPackets: 0
countDroppedDataPackets: 0
countAckPackets: 467
countDroppedAckPackets: 0
# segment timeouts: 0
TOTAL ITERATIONS: 156
```

5. Add any comments/questions to your submission doc.