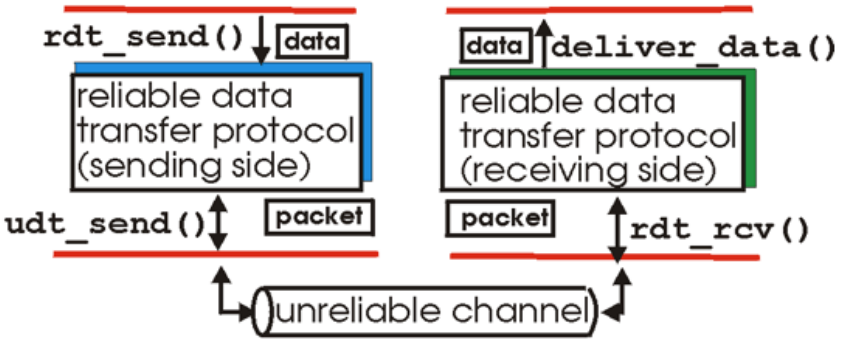
Programming Project 2 – Reliable Data Transmission (RDT)

Introduction

In this project, you will implement part of a simulation that provides reliable data transmission over an unreliable communication channel. Your goal is to design a transport-layer protocol that delivers all data correctly and in order, even when packets are lost, delayed, reordered, or corrupted.

Your design can draw on concepts like Go-Back-N or Selective Repeat, but it does not need to replicate any real-world protocol exactly. The main requirement is that your protocol uses a window-based approach, supports retransmissions, and successfully delivers all data in order despite an unreliable channel.



This simulation operates at a high level and does not use real TCP headers or bytes. Instead, it focuses on modeling the essential principles of reliable data transfer as described in your textbook.

Provided Code

The starter code includes the following:

- **unreliable** – simulates an unreliable communication channel.
- **segment** – represents data and acknowledgment packets, with checksum verification and iteration-based timestamps.
- **rdt_main.py** – runs your protocol and tests it with messages of different lengths.

Your main task is to complete the **RDTLayer** class. This single class must handle both sending (client) and receiving (server) roles. Each instance should determine its role based on the operation being performed, sending original data as a client or acknowledging received data as a server.

The RDTLayer relies on two key methods:

1. **processSend()** – handles outgoing segments
    - **Client:** sends data, waits for ACKs, and retransmits segments if lost or timed out
    - **Server:** sends ACKs and confirms receipt of data
2. **processReceiveAndSendRespond()** – handles incoming segments and generates responses
    - **Client:** receives ACKs, updates the send window, and marks segments as delivered
    - **Server:** receives data, validates checksums, sends ACKs, and delivers data in order to the application layer

The RDTLayer must maintain reliable communication even if the channel is highly unreliable. The **unreliable** can:

- Deliver packets out-of-order
- Delay packets by multiple iterations
- Drop packets
- Corrupt data packets (checksum failure)
- Combine any of these behaviors

Note: In this context, "segment" and "packet" are used interchangeably.

Key Features Your Implementation Must Include:

1. Run successfully with the provided **unreliable** and **segment** classes.
2. Deliver all data correctly and in order, even when all unreliable channel features are enabled.
3. Send multiple segments per iteration (pipelining) using the flow-control window.
4. Implement a flow-control window to limit the number of unacknowledged segments.
5. Use **Go-Back-N** or **Selective Repeat** for reliable delivery:
    - **Go-Back-N:** retransmit all unacknowledged segments if a segment is lost or corrupted.
    - **Selective Repeat:** retransmit only missing or corrupted segments, buffering out-of-order segments.
    - Either method is acceptable; correctness is measured by in-order delivery.
6. Include segment **timeouts**, based on iteration count, not actual time.
7. Follow the payload size and flow-control window size constants defined in **RDTLayer**.
8. Aim for efficient transmission with the fewest iterations and packets sent. Efficiency is achieved when multiple segments are transmitted per iteration and only lost or corrupted segments are retransmitted, minimizing total iterations and packets sent.

Note: Sending only one segment per iteration is automatically a failure. Full marks require effective use of pipelining.

Development Tips

- Start with **all unreliable channel flags set to False**. Once basic functionality works, enable flags incrementally:
    1. Delay
    2. Out-of-order delivery
    3. Packet drops
    4. Data corruption
- Use the **checksum** and **timestamp** methods in the Segment class.
- Keep track of **unacknowledged segments** for retransmission.
- Use **cumulative acknowledgments** to confirm receipt of multiple segments.
- The simulation is **iteration-based**, so timeouts are counted by iterations rather than real time.

Running the Simulation

In rdt_main.py:

1. Two text messages are provided: a short message for debugging and a long message for testing under load.
2. Observe the simulation output to verify:
    - Segments sent and retransmitted
    - Acknowledgments sent
    - Effects of packet loss, delay, corruption, and reordering
3. Use **step mode** (press Enter each iteration) for debugging.

Example Screenshots *(Exact matching with these outputs is not required)*:

Output with Shorter 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:

```
Time (iterations) = 202
Client-----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1, data:
Sending segment:  seq: 1241, ack: -1, data: 2

Length of Sent Unacked Packets List:  5
Sending segment:  seq: 1245, ack: -1, data:
Length of Sent Unacked Packets List:  6
Sending segment:  seq: 1249, ack: -1, data:
Length of Sent Unacked Packets List:  7
Server-----------------------------------------
Length of Receive Unacked Packets List: 6
Sending ack:  seq: -1, ack: 1225, 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 that goal will serve to organize and measure the best of our energies and s
kills, 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, 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 better than the finest watch, carrying all the equipment needed for propulsion, guidance, control, communications, food an
d 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 a
s hot as it is here today--and do all this, and do it right, and do it first before this decade is out.

JFK - S
------------------------------------------------------------------------------------------------------------------------
Time (iterations) = 203
Client-----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1, data:
Sending segment:  seq: 1225, ack: -1, data: epte
Sending segment:  seq: 1253, ack: -1, data:
Length of Sent Unacked Packets List:  8
Sending segment:  seq: 1257, ack: -1, data:
Length of Sent Unacked Packets List:  9
Server-----------------------------------------
Length of Receive Unacked Packets List: 0
Sending ack:  seq: -1, ack: 1257, 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 that goal will serve to organize and measure the best of our energies and s
kills, 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, 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 better than the finest watch, carrying all the equipment needed for propulsion, guidance, control, communications, food an
d 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 a
s 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: 549
countSentPackets: 737
countChecksumErrorPackets: 65
countOutOfOrderPackets: 19
countDelayedPackets: 90
countDroppedDataPackets: 58
countAckPackets: 173
countDroppedAckPackets: 15
# segment timeouts: 294
TOTAL ITERATIONS: 203
```

Submission Requirements

1. Submit **all code**, including your completed RDTLayer implementation.
2. Include an **introduction, screenshots and answers** for each of the following questions:
    - Where did the bulk of your logic occur?
    - How were timeouts resolved? What happened when a timeout occurred?
    - How was packet dropping handled? (include output screenshots)
3. How was your retransmission policy implemented? (include code and output screenshots showing retransmissions clearly)
4. Test with **smaller and larger messages**, and include **2 to 4 additional screenshots** of final results beyond those for the questions above. More screenshots can help demonstrate your protocol clearly.
5. Add any comments, challenges, or questions you encountered during your implementation.

Output Guidelines

While the simulation's output may vary due to channel randomness, your output should clearly show:

- Segments being sent and retransmitted
- ACKs being sent
- Timeouts occurring
- Segments arriving out-of-order and/or being buffered
- The final reassembled message at the server matches the original message exactly

This output will demonstrate that your protocol handles all types of unreliability correctly and efficiently.