Spitzer Assignment 3: RDT

Release 1.0

Landon Spitzer

Mar 23, 2025

CONTENTS:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CHAPTER

ONE

COMPUTERNETWORKSASSIGNMENT3

1.1 client module

class client.**Client**(*listen_port*, *server_port*, *server_ip*, *query*, *filename*)

Bases: `object`

A client that can either send a file to a server or query a file from the server.

query_file()

Queries the server for a file and writes the received data to a local file.

Sends a filename to the server, receives the file data from the server, and writes the received data to a local file with the same name as the queried file. Ensures that an ACK is received from the server after sending the filename, and sends ACKs as the data is received from the Server

run()

Execute the client's main functionality.

send_file()

Sends a file to the server.

Reads the file specified by the class variable filename into a byte string, sends the filename to the server, and then sends the file data to the server. Ensures that an ACK is received from the server after sending the filename.

1.2 intermediate module

class intermediate.**Intermediate**(*listen_port*, *sender_ip*, *sender_port*, *receiver_ip*, *receiver_port*, *loss_prob*, *reorder_prob*, *corrupt_prob*)

Bases: `object`

An intermediate node that simulates network impairments such as packet loss, reordering, and corruption between a sender and a receiver.

corrupt_packet(*packet*)

Corrupts a packet by randomly flipping bits in it.

Parameters

packet – The packet to corrupt.

Returns

The corrupted packet.

`handlePackets()`

Handles incoming packets, distinguishing between data and ACK packets.

`handle_ack_packet(`*packet*`)`

Handles incoming ACK packets, simulating packet loss and corruption as specified.

Parameters
`packet` – The incoming ACK packet.

`handle_data_packet(`*packet*`)`

Handles incoming data packets, simulating packet loss, corruption, and reordering as specified.

Parameters
`packet` – The incoming data packet.

`start()`

Starts the intermediate node, continuously handling packets until terminated.

`intermediate.main()`

Parses command-line arguments and starts the intermediate node.

1.3 receiver module

`class receiver.Receiver(`*listen_port*, *receiver_ip*`)`

Bases: `object`

Go-Back-N Receiver.

This class listens for incoming packets, verifies their integrity using checksums, acknowledges received packets, and handles packet loss and reordering.

`reassemble_data()`

Reassemble received packets into a complete data sequence for files.

Returns
The complete data reconstructed from received packets.

`return_filename()`

Retrieve the filename from the received data.

Returns
The filename extracted from the first received packet.

`start_receiving()`

Listen for incoming packets, validate checksums, and send acknowledgments.

This method continuously listens for packets, processes them in sequence, and acknowledges correctly received packets. It also handles out-of-order packets and retransmission scenarios.

`receiver.calculate_checksum(`*data*`)`

Compute the checksum of the given data

Parameters
`data` – data to compute the checksum

Return checksum

`receiver.main()`

Parse command-line arguments and start the receiver.

1.4 sender module

`class sender.Sender`(*receiver_ip*, *receiver_port*, *listening_port*, *data*)

Bases: `object`

`handle_acks()`

Handles acknowledgment reception and manages the sliding window.

`retransmit_next_packet()`

Retransmits the next unacknowledged packet in the window.

`send_data()`

Handles the transmission of data packets and manages acknowledgments.

`send_packet`(*seq_num*, *data*)

Sends a single packet with a sequence number and checksum.

Parameters

- **`seq_num`** – Sequence number of the packet.
- **`data`** – Payload data to be sent.

`update_window`(*ack_seq_num*)

Shifts the sliding window based on received acknowledgments.

Parameters

`ack_seq_num` – Sequence number of the received acknowledgment.

`sender.`**`calculate_checksum`**(*data*)

Compute the checksum of the given data

Parameters

`data` – data to compute the checksum

Return checksum

`sender.`**`main()`**

Parses command-line arguments and initializes the sender.

1.5 server module

`class server.Server`(*server_port*, *server_ip*)

Bases: `object`

A server that can receive files from a client or respond to file queries.

`run()`

Receive a file from a client or respond to a file query.

If the server is receiving a file, it writes the received data to a local file. If the server is responding to a file query, it sends the file data to the client.

PYTHON MODULE INDEX

INDEX