



RELIABLE DATA TRANSFER PROTOCOL

Multi-client-Server Application.

Zeyad Ayman El-Zanaty
Raffy Bekhit

3320
3248

Program Report

1. Structures.py

 **Packet:** A packet class is created with:


- Length of the packet in Bytes
- Sequence number
- Checksum value
- Data, which is limited to a size of 500 bytes per packet.

To send a packet through a socket, it's packed into data bytes using the structures module in python. When data bytes are received in client-side, they're unpacked into a new packet.

 **Ack:** An ack class is created with:

- Sequence number
- Checksum value

An ack is packed and sent using the method in packet class, also the same process is done when receiving an ack.

 **Checksum:** Checksum value is calculated using an implemented method with the same logic as discussed in the lectures.

2. Client.py

 **Client:** A client class is created with:

- Server IP address
- Server port
- Client port
- Client socket.

The first method a client invokes is `send_request()`, which connects the client socket to the initial server port, and tries to send a packet with the requested file name and waits for an ack, if not received it will re-send the packet until an ack is received from the server.

Clients then wait to receive the following:

- New port number that it will connect to the server and download the required file through.
- The algorithm used by the server to receive using the appropriate method.
- Requested file length in packets.

Finally, the `recv()` method is called which takes the algorithm used by the server as a parameter and calls the appropriate method to receive the requested file.

Receiving Methods:

- Go back n: Client receives packets continuously until reaching the end of the file using the previously received file length, if the packet received is in order, not corrupted nor lost, an ack is sent to the server and data is added to file. If not, it discards the packet, and re-receives the packet again.
- Stop and wait: Packets are received, if a packet is received in order and no corruption occurred, it's written to the file and an ack is sent to the server so it can send the next packet.
- Selective repeat: Packets are received. Once a packet at the beginning of the window is received the window is shifted by changing the base sequence number. Every time a packet is received the window is rearranged by sequence number starting from the base sequence number. If the base packet is received it is delivered and written to file.

3. Server.py

 Server: A server is created with:

- IP address (localhost)
- Initial port number
- Random seed value
- PLP Value
- Window Size

First, when the server runs, it constantly waits for a new request from clients, if a packet is received, and is not corrupted, it sends an ack to the corresponding client to ack that the request is received successfully. Then it searches for a free port number and sends it to the client to send the rest of the data on, finally it forks a new process to handle the request (send required file and any required data) while the parent process accepts more requests from multiple clients, and the whole process is repeated for any request received.

Sending Methods:

- Go back n: First, `get_packets_from_file()` is called, which converts the required file into a list of packets. A subset of the list is created (`current_pkts`) with the size of the window of server, then the `n` packets are sent or lost according to the PLP value, server then waits for `N` acks, for each ack received, it increments `current_pkts` range, if an ack isn't received it goes back and sends starting from the sequence number of the lost ack till the window size. The receiver handles the out of order packets.
- Stop and wait: Servers sends a packet with a sequence number equal to 0, and then waits for its corresponding ack to be received, if received sends the other packet with a sequence number equal to 1 and repeats the whole process over again.

- Selective repeat: Server sends the window size to the client. For each packet sent a new thread is created with a timer waiting for the acknowledgement to be received for the packet. When the acknowledgements are received and some are timed out then the window is shifted by number of acknowledged packets at its beginning. The Unacked packets only are resent.

4. Packet Loss and Corruption:

- ✚ Packet loss is implemented in server and corruption is implemented in client, a list of random sequence number is generated with the probability and seed value given in the server.in file, if the sequence number of the packet about to be sent is in said list, it's discarded and not sent in case of packet loss, and corrupted by changing its checksum value so when compared with the packets' data it doesn't send the ack.

5. Work Flow

✚ Server

- Server starts, waits for a download request.
- Request arrives, checks authenticity, sends an ack.
- Sends a new port number to the requesting client.
- Forks a new process to handle the request.
- Connects the client to the new port number.
- Sends algorithm used (Stop-and-wait, Go-back-N, Selective-Repeat) to client.
- Sends required file length in packets to client.
- Simulates packet loss.
- Send the requested file packets using one of the 3 algorithms.
- Repeat.

✚ Client

- Client starts, sends a download request to server.
- Waits for the request ack to arrive, resends request if it times out.
- Receives new port number from server.
- Connects the socket to the new port number.
- Receives the algorithm used by the server.
- Receives the requested file length in packets.
- Simulates packet corruption.
- Receives the requested file using the algorithm specified by the server.
- Exit

6. Running

Program can be run through terminal with options, or using an IDE with default values hard coded into the program and in server.in/client.in.

Server

- Terminal Command: `python3 server.py -[algorithm]`
 - Algorithms:
 - -saw: Stop and wait.
 - -gbn: Go back n.
 - -sr: Selective Repeat
- Default algorithm is Stop and wait (Hard Coded) which will be used if program is run using an IDE.

Client

- Terminal Command: `python3 client.py [requested file name]`
- Default file name is the file name specified in client.in
- Multiple clients can be run at the same time.
- All file types are supported.

Testing

- Testing was on a 3.7 bytes file which resulted in roughly 7k packets.
- Window size in go-back-n and selective-repeat was equal to 5

PLP	Stop and Wait	Go Back N	Selective Repeat
1%	376.3 sec	690.4 sec	4.45 sec
5%	1,435.98 sec	2,520.65 sec	4.50 sec
10%	2,983.21 sec	4,965.68 sec	4.55 sec

Conclusion

- As seen, Selective repeat is the fastest [multi-threading is used], Go back N is the slowest by far, Stop and wait comes second.