INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

COMPUTER COMMUNICATION LABORATORY
A REPORT ON

"Simulation of Go-back-N ARQ protocol using Sockets"

November 15, 2020

Y SAI SANJEET
(16EC35025)

DEPT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

Contents

OBJECTIVE

The aim is to simulate the go-back-N ARQ flow control protocol between processes using socket programming.

DISCUSSION

- The socket server and client are realized using the same socket library in the second experiment.
- The propagation delays across the channel are modeled by introducing a delay between receiving a data packet and sending an acknowledgment.
- Loss of data packets due to errors in the channel is modeled by sending a NACK instead of an ACK with a small probability, which is the probability of error of the channel.
- A large probability of error results in a higher number of NACKs being sent, which in turn reduces the data rate.
- The receiver, which is the server, has to be run first followed by the sender, which is the client. Else the client would terminate as the server is not reachable. The code continues to run until the sender transmits all the data.
- End of data transmission is indicated by sending a newline character `\n`.
- Stop-and-wait protocol is a special case of go-back-N where the value of N is 1.

CODE EXPLANATION

This section gives a brief overview of the code present in `send.cpp` and `recv.cpp`.

Sender:

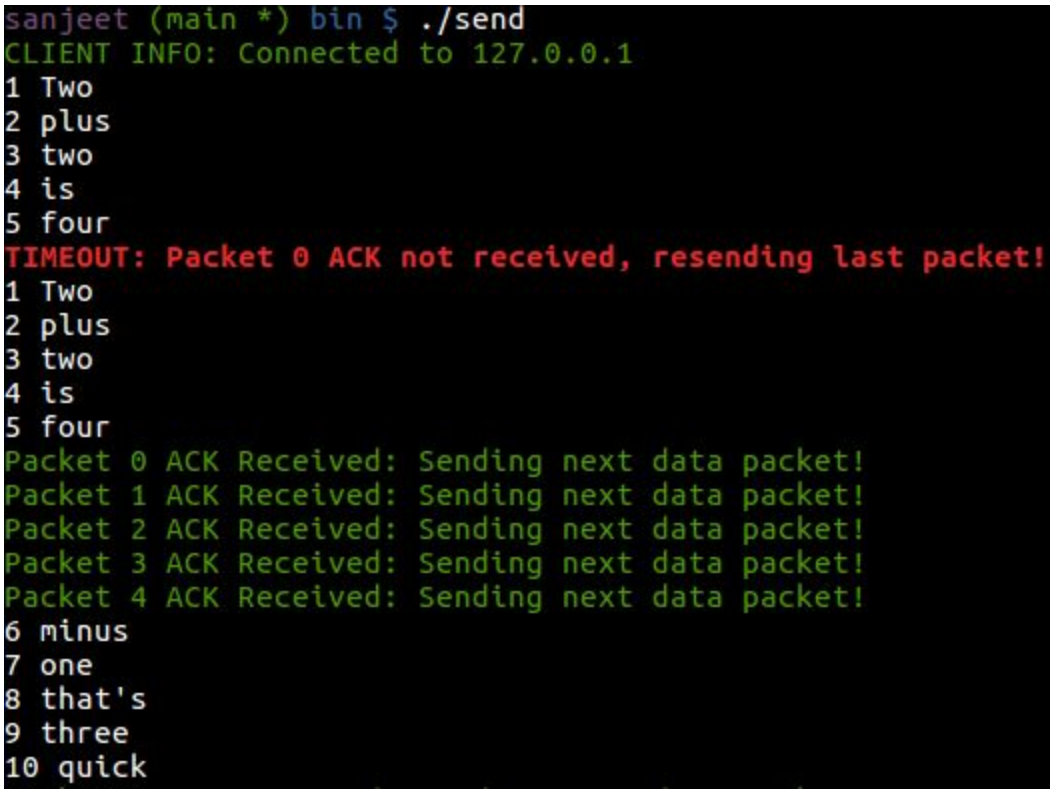
- The `sender` has a vector containing the data to be sent, whose last element is `\n`.
- An instantiation of the `SocketClient` object is used to send the data.
- The sender sends N data packets and waits for N `ACKs`.
- The `sentDataIndex` and `recvDataIndex` variables keep track of what data is to be sent and how many `ACKs` have been received.
- The `recvDataIndex` is incremented if an `ACK` is received, and the `sentDataIndex` is assigned the value of `recvDataIndex` if a `NACK` is received.

Receiver:

- An instantiation of the `SocketServer` object is used to receive the data.
- The receiver receives N data packets and generates a random number to decide whether to send an `ACK` or a `NACK` and adds them to a queue.
- It stores the received data in a vector if an `ACK` is to be sent.
- It sends all the queued acknowledgments once N data packets are received.
- The receiver stops receiving if it successfully receives a `\n`.

RESULTS

The sender-side:



The receiver-side:

CONCLUSION

- The go-back-N ARQ flow control protocol is implemented.
- The Socket library written in the second experiment is used to create the socket server and client.

REFERENCES

https://en.wikipedia.org/wiki/Go-Back-N_ARQ