TCP/IP Internetworking - Project Report

BUTP SERVER AND CLIENT

Krishna Varaynya Chivukula- krch13
Haider Rehman- hara12

Following techniques of TCP were used for the UDP based protocol-

1. Error Control-

Error control is necessary in order to have a reliable transport protocol. It enables the protocol to identify and handle the errors in the packets like the out of order arrival, missing or duplicated packets. In the design of BUTP, this is achieved by sending a sequence number and checksum of the data along with the data packet.

Implementation -

Sequence number-

- All packets have a sequence number, which helps identify the missing or dis-ordered packets.
- To verify this, a wrong sequence number after every 7 packets is sent randomly, which is detected by server at the receiving end and sends back an acknowledgement as a sequence error. In response, client drops the current packet and sends the correct sequence packet.

Checksum-

- All packets have checksum of the sent data, which is compared to the checksum of received packets at the server end. Any deviation identified resembles corrupted data during transmission. To verify this, data of every 4^{th} packet is changed manually. After observing the packet drop, it is resent.

2. Flow Control-

Flow Control is the process of managing the rate of data transmission. TCP provides flow control mechanism using acknowledgements of TCP sequence numbers. In BUTP, this is achieved by controlling the data accepted by server. Server requests the acceptable window size, which client transmits.

Implementation-

A timer is used for the client to wait for server response before terminating. After being acknowledged by server, next packet is sent. ACK contains the response of

accepting the data and the windows size acceptable for the next packet to avoid overflow. For this, the window size is randomly changed.

3.Congestion Control-

Congestion control is the mechanism concerning the traffic and avoiding overload of the network.

In BUTP, Congestion is handled by limiting the window size to which it is acceptable to server without causing any data loss.

Implementation-

Every 4th packet is manually dropped at receiving end. After noticing that the packet is not received successfully, the packet is retransmitted.

Algorithms-

BUTPClient-

- Create a datagram socket.
- Get the file contents using file chooser.
- Creating First chunk to be sent to the server with file and packet info.
- If server accepts, proceed to next chunk, if not timeout after 2000ms.
- While, receiving content from file, get the string of data packet of size 1000bytes.
- Create checksum for data packet.
- Initialize the sequence number.
- Checking for Random congestion control on the sequence number, if true add garbage text to the data packet.
- Checking random wrong sequence number, if true, adding "1" to current sequence number.
- Sending created chunk to server.
- Waiting for server response.
- Splitting the response of server with a delimiter- "_".
- If the length of split Server response is greater than 1, it means server responds with congestion error.
- Window size will be reduced to random number less than 1000 bytes. Else, window size is reset to 1024 and packet size to 1000 bytes.
- If the server response is NAK, the packet is dropped, a message is displayed and the packet is resent.
- Else, sequence number at pointer position in the file is incremented with the packets size and the size of file is reduced by the packet size.

BUTPServer-

- Creating a new Datagram socket.
- Display message of server running on a defined port.
- Data packet received from client is split with a delimiter – “”.
- The data packet from client contains Opcode at position 0, data at position 1, checksum identifier at position 2 and sequence number at position 3.
- If the Opcode = write, the client is trying to sent file info as packet 1.
- Else checksum is generated for the data and compared with a checksum identifier received from client. If it is not a match, response is set to NAK with checksum error. Else random congestion is compared with the sequence number sent from the client.
- If match server responds with NAK congestion error, it sets the new window size for next chunk. Else if the packet number does not match the sequence number, the server sends response with NAK and gives unexpected sequence number. Else, the server responds with ACK and creates new file with received content and increment packet umber.