**Assignment 2: RELIABLE UDP**

**Introduction:**

The User Datagram Protocol (UDP) is a transport layer protocol, providing minimal services. It is a connectionless protocol without any 3 way handshaking. Being an unreliable data transfer service, it does not offer any flow control or congestion control. Thus to make the User Datagram Protocol more reliable, we are designing the Reliable UDP which will add more some additional services including the flow control and congestion control.

Below are the different components of the reliable UDP:

**Header Design:**

**Sender:**

Sequence number: Each packet that is sent from the sender to the receiver is associated with a sequence number. The sequence number also keeps track of in order packets. In real time scenario, the sequence number is usually randomly selected by the server. So here, I have started the sequence number with 2700.

**Receiver:**

File name: this header field contains the name of the file that is being requested by the client over the reliable UDP. The file directory location is hardcoded in the server code. The location at which the file is located is: u/vensamba/bin

Acknowledgement number: Each packet that is sent by the sender needs to be acknowledged by the client, this way one can confirm that the packets are successfully received at the client side. This can also be used to check in order packet delivery.

Sequence number: Same way as the sender has a sequence number, the client also has a sequence number. Haven’t used it anywhere in the code though.

**Advertised window:** The advertised window size is provided separately in the command line interface. This is the maximum number of packets that the sender can advertise at a time. So if we choose 10 as the advertised window, then only a maximum of 10 packets can be advertised. But when we introduce congestion control into the picture, it will be the minimum of the advertised and the congestion window.

**Sliding Window:**

I have implemented GO-BACK-N transmission methodology. The file is divided into chunks and the receiver sends packets based on the advertised window with its associated sequence numbers. When the correct packets are received, the packets are cumulatively acknowledged.

So every time a correct packet is received, I store 1 in the array to indicate that the receiver has received the correct packet. If the receiver receives an out of order packet, then I store a value 0 in the array. This is done so that the correct sequence number of out of order packet can be calculated and sent over to the sender in order for re-transmission.

**Duplicate acknowledgment:**

When we receive the packet on the receiver side, the sequence number is compared with the expected sequence number. If the expected sequence number doesn’t match, then the receiver sends a duplicate acknowledgment back to the sender.

On the sender side, when the sender gets this duplicate acknowledgment, it checks if this duplicate acknowledgment is less than its current sequence number. If yes, then it increments it duplicate acknowledgment count by 1. If this happens 3 times, then the sender retransmits that packet. This is how duplicate acknowledgments work.

**Cumulative acknowledgments:**

Here I have used cumulative acknowledgements in my code. So every time a successful packet in order is received, it is displayed on the screen directly. In my code I have set the payload value to 300, so the acknowledgement number is incremented by 300 to indicate 300 bytes were received successfully. Similarly the final cumulative acknowledgement is sent over to the receiver. For example: If all the 10 packets are received successfully in order, then the cumulative acknowledgement will be incremented by 3000 and sent over to the sender side. This acknowledgement will be used by the sender to get a confirmation that the packets were sent successfully, else retransmission will be needed based on 3 duplicate acknowledgements.

**Round Trip Time and Timeout Interval:**

The sample round trip time is the time taken for a packet to transmit from sender to receiver and for the acknowledgement to traverse back to the sender side. Below are the 3 formulas that I have used to calculate the time out interval.

Estimated\_RTT = (1 - α) Estimated RTT + (α) Sample\_RTT

Dev\_RTT = (1 - β). Dev\_RTT + β. |Sample\_RTT – Estimated\_RTT|

Timeout Interval = Estimated\_RTT + 4. Dev\_RTT

Where:

α – 0.125

β – 0.25

The timeout is calculated based on the above formulas. So every time the timer times out, the packet has to be re transmitted.

**Instructions to run the file:**

**Sender side: ReliableUDPServer 12345**

Parameter Details: server file, port number.

**Client side: ReliableUDPClient silo 12345 a.txt 10**

Parameter Details: client file, server name, port number, file to be read, advertised window size.

(The file location is hardcoded: Please change the location at the time of execution. Line 72 in the server file.)

Regards,

Venkat Sambandhan

0003544188