**Reliable UDP Protocol**

**ABSTRACT:**

“Reliable data transfer using UDP (User Datagram Protocol – RFC 768) for file transfer” is a client-server application. The reliability over UDP is achieved using **Stop n Wait (SW**) or **Selective Repeat (SR)** protocol. UDP, unlike TCP (Transmission Control Protocol), is an unreliable, connectionless data transfer protocol existing in the transport layer of TCP/IP architecture. The primary purpose of this project is to download a big file from a server residing in different IP location

In this method I shall be using a modified version of Selective Repeat protocol.

**MESSAGE SENDING:**

In this protocol the user can send as messages at any rate without keeping in mind about the loss or traffic condition the messages to be sent would be queued in for sending purpose. Before sending the messages, the messages would be concatenated together with the sequence number for the latest message corresponding to a user.

A map would store the latest sequence number of a message that has been sent for each user. Every time a message has to be sent to a user, the latest sequence number for the user in the mapping is updated. The message with the sequence number (together called packet) is then encoded with a static key using CRC and the encoded message is then sent to server for delivery This way the user does not have to wait on previous message's delivery.

Now once the message has been sent to server, there can be several scenarios:

**CASE 1: PACKET LOST**

For each message being sent, there is timer that waits for a specific amount of time. Once the time expires, the code checks for the acknowledgement received for the particular packet. This is done with the help of a mapping, that stores the username to the list of sequence number for the acknowledgment received from that particular user.

If any acknowledgement number in the list matches the sequence number of the message sent, the job is done and the processor moves on to next task, otherwise the message is re-scheduled for delivery. Re-scheduling the unsent packet helps in balancing the load/traffic rather than continuing to resend that particular packet unless successful.

**CASE 2: ACKNOWLEDGMENT LOST**

If the ACK is lost, the sender of the message packet would assume that the message has been lost on the expiry of the timer. So it would resend the packet and there would be duplicity of the packets in the receiver’s end. To solve this, I have stored the received packets in a list corresponding to the sender's id using a mapping.

On arrival of duplicate packets, I the code would check through the list of message sequence numbers. If the message is present store the ACK is sent to the sender with proper sequence number and owner’s id but not stored, otherwise in addition to sending messages the message along with sequence number is stored in the list

**CASE 3: ACKNOWLEDGMENT DELAYED**

If the ACK is delayed, again the sender of the message packet would assume that the message has been lost on the expiry of the timer. So it would resend the packet and there would be duplicity of the packets in the receiver’s end.

This problem is solved the same way as in when ACK is lost. In addition to this problem, if some other message is send and it gets lost on the way (critical point), but the delayed ACK reaches now, the sender would assume that the currently sent message has be accepted on the other side and so it would move on to next message.

In order to handle this, the sender would also store a mapping from user id to list of ACK numbers that are already received. Now if the delay AK arrives at some critical point, the client would check if it has already received the ACK with same sequence number. If it has, then no need to resend the packet, otherwise resend the corresponding packet.

**ERROR IN PACKET**

The error check in packet is done through CRC method which can detect almost all kinds of errors in most cases. As for this reason its it very frequently used error check mechanism. A static key is used for the CRC process. If error is present, no ACK is sent while the packet is discarded and the message is eventually resent on expiry of time on the sender's side, otherwise an ACK is send and the message is stored for processing

**UNORDERED PACKETS**

For each user, a mapping is stored that indicated that last sequence number of the message that was printed. This number (n) implies all the messages with sequence number from 1 to n has been sequential and thus has already been printed. The rest of the message list for that particular user is sorted and sequential messages starting from the last sequence number of the message that was printed is looped and printed. For each message printed, an ACK is send for confirmation of message acceptance. Unless messages with continuous range of sequences received, no messages are printed nor any ACK is sent out.