Assignment 1

Computer Networks

Word Count: 1097

Stop and Wait Protocol

Sender - Sending a Packet

The Sender computes a checksum for the packet by converting the application data into bytes, performing a character by character summation of the data and adding this to a sequence number (initially 0) and an acknowledgement number of 0. The packet is created and sent consisting of a sequence number, calculated checksum and the data. The timer for the packet then starts. The sender alternates between sequence numbers of 0 and 1 by checking the sequence number of the packet just sent and flips this ready for the next packet. The sender also stores the acknowledgement number it is expecting back from the receiver by checking the sequence number of the packet just sent and assigning it to be equal to this sequence number. When a packet is sent the sender stores this into a lastSentPacket variable used for retransmission if needed.

Sender - Receiving Acknowledgements & Handling Corruption and Packet Loss

The sender computes the packet's checksum, however given acknowledgement packets contain no data it checks if data.isEmpty() and if so then it adds 100 to the checksum, resulting in checksums of 100 for ACK 0 packets and 101 for ACK 1 Packets solving the problem of getting a 0 checksum. If the computed checksum matches the packet's checksum and the acknowledgement number is expected then the timer of the packet is stopped, however if the sender receives a corrupted acknowledgement or never receives an acknowledgement due to packet loss then the timer for the packet it just sent expires and the lastSentPacket is retransmitted and the timer is restarted.

Receiver - Receiving Packets

When a packet is received it computes the checksum and if this is equal to the packets checksum and the sequence number of the packet is what the receiver expected (initially 0) then it extracts the packets data and delivers it to the application layer.

Receiver - Sending Acknowledgements

The if statement is used to check the expected sequence number of the received packet, then a checksum is calculated for an acknowledgement 0 packet or an acknowledgement 1 packet depending on this sequence number, finally an acknowledgement packet is created and sent consisting of a sequence number of 0, the appropriate acknowledgement number and empty data. The receiver then flips the expected sequence number to match the next packet that is expected. The receiver also stores the last sent acknowledgement in a lastSentACK variable used to retransmit lost acknowledgement packets if needed.

Receiver - Handling Packet Corruption, ACK Loss and Packet Loss

If the packet is corrupted and it's the first packet received, the receiver does nothing and the timer for the packet expires because it's checksum won't match the expected checksum and there is no record of a previously sent acknowledgement.

If the packet is corrupted but it's not the first packet received then it resends the lastSentACK. The if statement checks for the expected sequence number as well as the checksum because the acknowledgement sent could get lost, resulting in the sender retransmitting the same packet again. The receiver handles this because if the sequence number is not expected then it does not redeliver the data to the application layer, it just resends the last sent acknowledgement. If the receiver never obtains a packet due to packet loss then it does nothing and the sender retransmits the packet again due to a timeout.

Go Back N Protocol

The buffer is represented as an ArrayList with variables repressing the front (base) and end of the window (endofWindow). Initially the base and nextSeqNum are set to 0 and endofWindow is 8.

Sender - Sending a Packet

The sender checks whether the window is full by checking if the nextSeqNum is less than the base + windowSize. If this is false then the sender can't send any packets because the nextSeqNum slot is outside the usable window so it refuses data from the application layer. If nextSeqNum is in the usable window then it creates and sends a packet consisting of the nextSeqNum, calculated checksum and the application layer data. It then uses the ArrayList 'buffer' to put this packet in the next sequence number slot. It then checks if this is the first packet sent (base is equal to nextSeqNum?) and this is initially true, therefore the timer is started for this packet as it is the oldest currently unacknowledged packet. The nextSeqNum is then increased ready for the next packet to send.

Sender - Receiving Acknowledgements & Handling Corruption/ACK Loss

The sender checks if the acknowledgement is corrupted, if so then it does nothing, otherwise it stores it at the base of the window. The sender handles cumulative acknowledgements by increasing the base by the packets acknowledgement number + 1, effectively sliding the window as acknowledgements are received. When the receiver sends an acknowledgement for a packet with sequence number n this indicates all packets with a sequence number up to and including n have been received, therefore the window could be moved more than once. Finally the endOfWindow is also increased because this will complete the window sliding action. The sender finally checks whether all packets sent are acknowledged (base is equal to nextSeqNum?) if so then it stops the timer because the application has no more data to send, if not then it restarts the timer for the next packet waiting to be sent, effectively every time the window slides the timer is restarted.

Sender - Timeout

If the timer for the oldest sent unacknowledged packet expires due to packet loss, the sender resends all packets that have previously been sent but not yet acknowledged. This functionality has been implemented by initialising a for loop iterating through the buffer in the range starting from the base up to the nextSeqNum, obtaining the packet at position i and sending the i-th packet to the receiver.

Receiver - Receiving Packets & Sending Acknowledgements

The receiver checks for corruption and whether the packet's sequence number matches the expected sequence number. If so then it delivers the data to the application layer and creates an acknowledgement consisting of a checksum and the acknowledgement number (equal to the sequence number). It then sends this to the sender and also stores this in the lastSentACK variable which is used for retransmission if needed. The expected sequence number is increased so the receiver is ready for the next expected packet.

Receiver - Handling Corrupted/ Out of Order/ Lost Packets

If the received packet's sequence number doesn't match the expected sequence number (an out of order packet) then it discards this packet (ignoring the if statement) or if it is corrupted it resends the lastSentACK (most recent acknowledgement for the packet with the highest sequence number). If it's corrupted and it's the first packet sent then it does nothing because there is no last sent acknowledgement available. Finally if the packet sent by the sender is lost then the receiver does nothing as the timer eventually expires at the sender side.