Computer Networks CS F303

Assignment II - Phase I

Design Document

Group Details

Group I

Jain Jai Sandeep 2017A7PS1585H

Prateek D Hiranandani 2017B4A70578H

Sahil Madhu Nair 2017B5A71317H

Jatin Arora 2018A7PS0551H

Rusabh Rakesh Parikh 2018A7PS1217H

Group II

Venkateshwar Dhari Singh 2018A7PS0246H

Dhruv Maheshwari 2018A7PS0170H

Raaed Syed Ahmed 2018A7PS0218H

Lokesh Mehra 2018A7PS0268H

Reliable UDP

Introduction

User Datagram Protocol (UDP) is a prominent transport layer protocol considered “unreliable” as it does not guarantee if the packets will be delivered. Even in cases where packets are delivered, the packets may be reordered or corrupted. Reliable UDP is an application layer middle-ware protocol to provide reliable in-order delivery of packets. Layered on the UDP/IP Protocols, RUDP is designed to transfer packets over UDP sockets in a reliable manner so that the complete file being sent is intact.

Assumptions

Application

- The user will provide the file's path to be sent from the Sender's computer and the file's path to store on the Receiver's computer. As the file path to store on the Receiver's computer will be sent in the handshake packet by Sender to Receiver, it should fit in one packet.
- RUDP assumes that the application will provide the file size that needs to be sent, and this size is sent to the Receiver during the handshake.
- The File Transfer application can chunk the file and provide each chunk as a payload to the RUDP middle layer, using the payload to form the packets.

Network

- The network is unreliable, and there is no guarantee of a successful transmission of data. It does not provide any congestion control or flow control.

Design

Reliable UDP supports the following features. The sender and receiver refer to either clients or servers sending and receiving a data segment on a connection in the following description. The client refers to the peer initiating the connection. On the other hand, the server refers to the peer that listens for a connection. A connection is an interface that serves a unique peer IP address/UDP port pair. Multiple connections are possible on a particular IP address/UDP port between a server and a client, and each connection will be with a unique peer IP address/UDP port pair.

1. Retransmission timer

The sender has a retransmission timer with a configurable timeout value. Each packet is assigned with its own logical timer as only a single packet will be transmitted on timeout. The timer initializes every time a packet is sent/retransmitted. If an acknowledgement for this packet is not received by the time the timer expires, then the packet is retransmitted.

2. Retransmission Counter

The sender maintains a counter for the number of times a segment has been retransmitted. The maximum value of this counter is configurable. If this counter exceeds the maximum, the connection will be considered broken.

3. Window Size

The Sender's window is equal to the Receiver's Window. The window size will be less than or equal to half the sequence number. This helps avoid packets from being recognized incorrectly. The Sender can transmit new packets as long as their sequence number is within the window.

Protocol

Language of implementation: Python3

Connection Establishment

The Sender will first send a handshake packet to the Receiver. This packet will contain the file's name and the path where it needs to be stored, along with the number of successful packet transmissions required. This will help the Receiver to know when it has ultimately received the file.

The Receiver will send an ACK with the sequence number equal to the number of successful packet transmissions required.

The Sender will close the connection when it receives the ACK for the last packet.

Implementation

The protocol will contain three classes -

- **Packet Class**

This class's constructor will take the payload or ACK as arguments to create a packet. The Reliable UDP packet will consist of a header and a payload.

The header will contain:

Sequence number

Each packet contains a sequence number. When a connection is first established, the sender will send the number of packets required to send the file as the sequence number to the receiver. On each transmission, the sequence number gets incremented.

Acknowledgement Number (ACK)

The acknowledgement number is used to indicate the sequence number of the packet the receiver has received to the sender.

Checksum

A checksum is always calculated on the Reliable UDP header to ensure integrity. The checksum of the payload is calculated by an algorithm that uses a 32-bit hash.

- **Sender Class**

The constructor of this class will take the socket's IP address and port number as arguments to create a UDP socket for the Sender to send the file.

1. When the data is obtained from the application layer, the Sender allots the next immediate available sequence number for the packet. If the sequence number is not within the Sender's window, it asks the upper layer to wait until the window moves forward or buffers the packet; or else if it lies within the window, the data is packetized and sent.
2. To take into account timeouts, timers are used to protect against lost packets. Each packet has its own logical timer, as only a single packet will be transmitted on timeout.
3. If an ACK is received, the Sender marks the packet as received, given it is in the window. If the packet's sequence number is identical to sent base, the window base is moved forward till the next unacknowledged packet with the smallest sequence number. If the window moves and there are packets which have not been transmitted with the sequence numbers that now are in the range of the window, they will be transmitted.

This class will have 3 functions:

1. *execute(path_of_file, dest_ip_addr, dest_port)*

   This function will run the Sender's side of the protocol.

2. *send(payload, seq_num, dest_ip_addr, dest_port)*

This function will send a packet with the payload and sequence number passed as arguments to the socket bound at (dest_ip_addr, dest_port)

3. *receive()*

   This function will receive any packets (ACKs) bound to the Sender's socket.

- **Receiver Class**

The constructor of this class will take the socket's IP address and port number as arguments to create a UDP socket for the Receiver to receive the file.

1. Packet with sequence number in the range [rcv_base, rcv_base +N-1] is correctly received. In this scenario, the received packet falls in the range of the receiver's window following which a selective ACK packet will be returned to the sender. The packet gets buffered if it was not previously received. If this packet has a sequence number which is equal to the base of the receive window(rcv_base) then the packet and any previously buffered and consecutively numbered (starting with rcv_base) packets are delivered to the upper application layer. Next, the receive window is then pushed forward by the number of packets delivered to the upper layer.
2. Packet with sequence number in the range [rcv_base-N, rcv_base -1] is correctly received. In this case an ACK must generated even though its a packet that the receiver has previously acknowledged.
3. Otherwise the packet is ignored.

This class will have 3 functions:

1. *execute(path_of_file, src_ip_addr, src_port)*

   This function will run the Receiver's side of the protocol.

2. *send(payload, seq_num, src_ip_addr, src_port)*

This function will send a packet with the payload (ACK) and sequence number passed as arguments to the socket bound at (src_ip_addr, src_port)

3. *receive()*

This function will receive any packets bound to the Receiver's socket

Handling of different network conditions

	Packet	ACK
Loss	If a packet is lost, the ACK will not be received by the Sender. Hence, the Retransmission timer will expire, and the packet will be sent again.	If an ACK is lost, the Retransmission timer will expire, and the packet will be sent again. On receiving the packet with the sequence number that it had already ACKed, the receiver will resend the ACK.
Corruption	The receiver will check if the packet is corrupted by calculating the checksum. If the packet is corrupted, it will ignore the packet. The Retransmission timer will expire, and the packet will be sent again.	The sender will check if the ACK is corrupted by calculating the checksum. If the packet is corrupted, it will ignore the ACK. The Retransmission timer will expire, and the packet will be sent again. On receiving the packet with the sequence number that it had already ACKed, the receiver will resend the ACK.
Delay	If a packet/ACK is delayed, it means that the sender does not receive any ACK for that packet within the specified timeout. Hence, the Retransmission timer will expire, and the packet will be retransmitted. If the receiver receives the packet again, it will retransmit the ACK. If the sender receives the ACK for the packet sent two times, it will ignore the second ACK.	
Reordering	As each packet/ACK has unique sequence numbers, the receiver/sender knows each packet's position in the file. If packets/ACKs are reordered in the network, the receiver/sender will look at the sequence number and act according to the Receiver's/Sender's protocol mentioned above.	

References

Kurose. J. F., & Ross. K. W., (2012), *Computer networking: A top down approach*, Boston, MA: Addison-Wesley.