CS F303 - Computer Networks

DESIGN DOCUMENT

ON

ASSIGNMENT-2

By

GROUP A	GROUP B
Shivang Singh - 2018A7PS0115H	Nikhil Chandra - 2018A7PS0260H
Nishit Chouhan - 2018A7PS0446H	R.V. Srinik - 2018A7PS0266H
Deepak George - 2018A7PS0244H	Nitin Chandra - 2018A7PS0188H
Shubh Deep - 2018A7PS0162H	Thakur Shivank Singh - 2018A7PS0439H
	Tejas Rajput - 2018A7PS0253H

Abstract

This document discusses the design for a Middleware protocol that provides reliable features on top of the unreliable data channel - UDP. This middleware protocol exists between the application layer and the transport layer and ensures the reilable delivery of packets between client and server under various network conditions. This is done through the addition of various header fields to the application layer packet before passing the packet on to the transport layer. These header fields are discussed in further detail later on in the document. The packet is then intercepted by the middleware on the receiver end and appropriate actions are taken based on the values of these header fields to ensure reliability.

Assumptions about the Application and Network:

1. We have chosen the **Client Server** paradigm where computers do not communicate directly with each other (as is the case with P2P) , but clients establish a connection with an always-on host known as the server. In general, a client requests data from the server, and the server provides the required data after authorisation.
2. This protocol adheres to the **Selective Repeat paradigm** i.e., re-transmits only those packets for which ACK has not been received, or a NAK is received.
3. **Fixed sender/receiver window** (number of packets containing consecutive sequence numbers) at client and server sides of size N. A buffer to maintain in-order delivery is also considered to be of fixed size N.
4. Inorder delivery to the application is ensured as long as the buffer size is not exceeded by the out of order sequence numbers received, i.e. the difference (max received sequence number (-) min received sequence number) should always be lesser than the size of the buffer. If we get a packet that has a sequence number that cannot fit in the buffer, we simply ignore it.
5. Data that is sent to the middleware by the application is assumed to be divided and sent as chunks of 16 bits to be consistent with the header format (padded if necessary). This means that the **middleware continuously receives payload in the form of 16-bit chunks.**

Data Flow

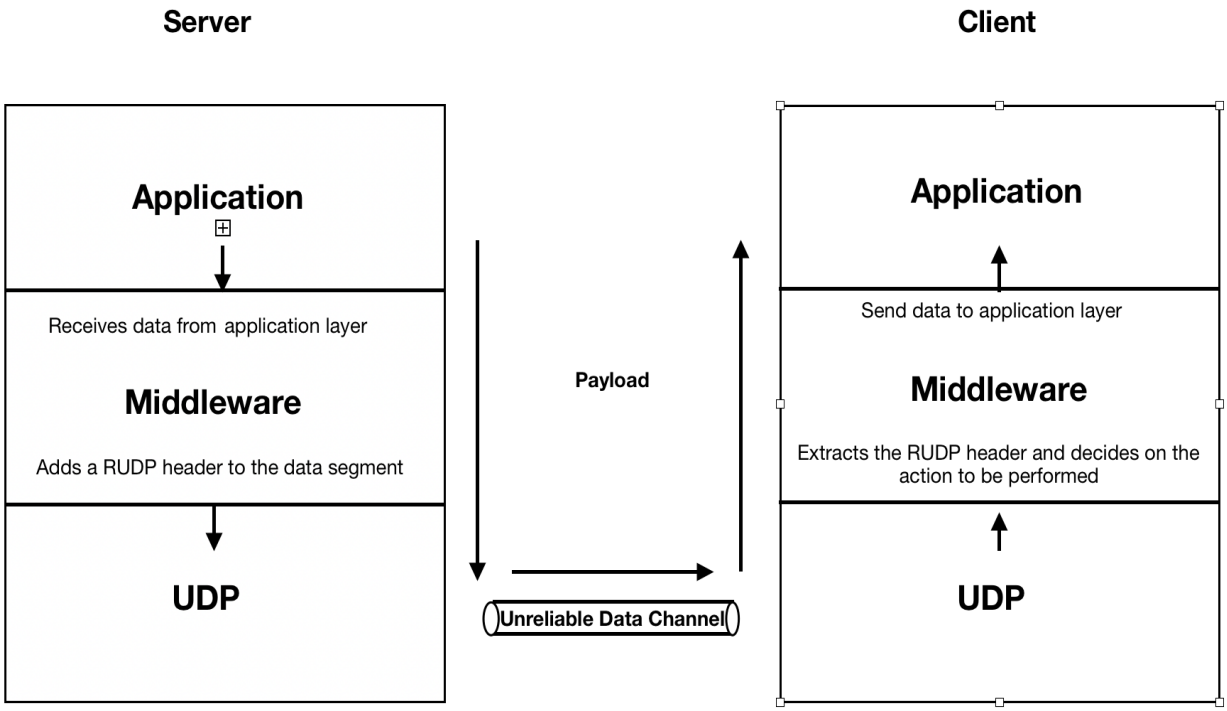

Figure 1

General RUDP Header Format for Data Transmissions :

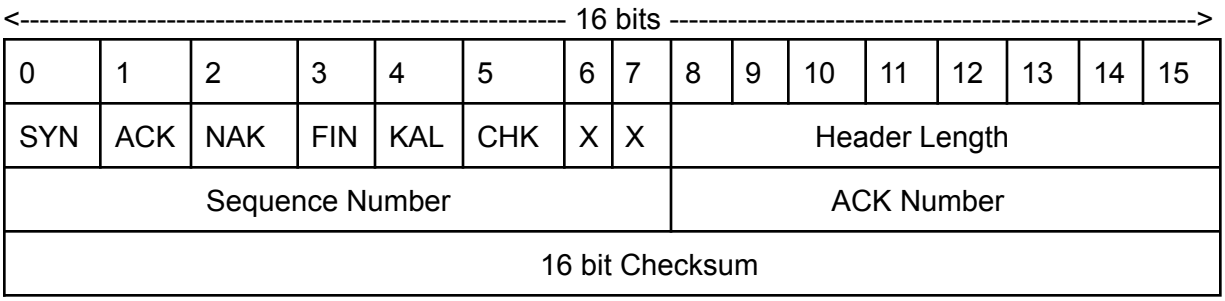

Figure 2

Control Bits:

1. **SYN -** Indicates whether the segment is a SYN segment or not.
2. **ACK -** Indicates whether the segment is an ACK segment or not.
3. **NAK -** Indicates Negative Acknowledgement (ACK for corrupted data)
4. **FIN -** Initiates connection close on sender's end (won't send more new data).
5. **KAL -** Indicates whether the segment is a KAL segment or not.
6. **CHK -** Indicates whether checksum is for the header only (0) or header and data (1).

Header Length :

This field indicates the number of bytes in the packet that represent the RUDP header i.e. the offset in the packet from where the actual user data begins. It has a fixed value (equal to the length of the packet) for the segments that do not contain any data (SYN, ACK, KAL, FIN).

Sequence Number :

This field contains an 8-bit number that indicates the order of the data being sent so as to ensure in-order delivery of data. It is given a random value while establishing the connection using a SYN Segment. It is then incremented by the sender each time we send Data or a KAL Segment.

ACK Number :

This field contains an 8-bit number that contains the sequence number of the packet that is being acknowledged (might be a NAK)

Checksum :

It is a 16-bit number computed using different 16 bit fields in the header (CHK=0) or the header and the body (CHK=1). It is used to detect Data corruption that might have occured while using the UDP channel.

Specific RUDP Segments – SYN, ACK/NAK, KAL, FIN

SYN Segment :

A SYN segment is used to initiate establishment of connection between the Client and Server. It is different from the general RUDP header since it specifies additional parameters required for the connection. It can also be combined with an ACK to form a SYNACK Segment i.e. it acknowledges the SYN of the sender and itself initiates the connection using SYN.

<---------- 16 bits ---------->

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	ACK	X	X	X	X	X	X	Header Length = 12							
Sequence Number								ACK Number							
Sender Window / Buffer Size (value of N as per assumption)															
KAL Segment Timeout Value															
Retransmission Timeout Value															
16 bit Checksum															

X - Don’t Care Condition

Figure 3

KAL Segment Timeout Value:

Indicates the maximum duration after the latest data segment that was sent for which the receiver keeps the connection alive.

Retransmission Timeout Value :

Indicates the maximum duration for which the sender waits for the acknowledgement of the packet being sent before it retransmits it.

ACK/NAK Segment :

An ACK segment is used to acknowledge the current segment (SYN/Data/KAL) that was received. It uses the general RUDP header format with ACK bit = 1 and the ACK number specifies the sequence number of the packet received. On receiving a packet, we check if the data is corrupted or not using the checksum value, if found corrupted, we send a NAK segment (same as ACK segment, but with NAK bit=1) otherwise we send an ACK segment.

KAL Segment :

A KAL segment is used to know if the connection is active or not. It uses the general RUDP header format with KAL bit = 1. If the sender hasn’t received a response that it has been expecting for a while, it would send the KAL segment before the KAL segment timeout value

expires. Upon receiving a KAL segment, the receiver must immediately acknowledge it to notify the sender that the connection is still alive. This segment never contains user data.

FIN Segment :

A FIN segment is used to initiate closing of a connection between the client and the server. It uses the general RUDP header format with FIN bit = 1. Once the client sends a FIN segment, it can no longer send data but can still receive data. The server acknowledges this, and sends a FIN segment once it sends all the pending packets back to the client. Now, the client sends an ACK segment to the server acknowledging the FIN segment of the server, resulting in the complete closure of the connection.

Strategy to Handle Various Network Conditions:

Delay

In the event that high latency conditions are prevalent in the network, if a packet was not transmitted within the time, there is a retransmission time-out for that particular packet, and that packet alone is retransmitted by the sender.

Packet Corruption

On receiving a packet, the receiver calculates the checksum and compares it with the checksum in the packet. If the packet is corrupted, the checksums will not match and a NAK segment is sent (ACK segment with NAK bit set).

Packet Reordering

According to the Selective Repeat Paradigm, the receiver acknowledges all correctly received packets and these are buffered as needed for eventual in-order delivery to the upper layer. The out-of-order ACK-ed packets are buffered and eventually when missing packets are received, we deliver the buffered in-order packets to the application layer and advance the window to the next not-yet-received packet. Hence the buffer of size N, handles and maintains the reordering of the packets.

Packet Duplication

If the received packet's sequence number is marked as received in the receiver window/buffer, then we discard the packet(not passed to the application) and send an acknowledgement of the duplicate packet to the sender. This ensures that the sender is aware of the fact that the packet was successfully received, and does not continue to attempt multiple duplicate transmissions.

Packet loss

If the receiver is expecting a certain packet, and this packet is lost during transmission, the sender will wait for the acknowledgement of the packet it had transmitted for the duration given

in the retransmission timeout value field. Once this timeout period has elapsed, the packet will be re-transmitted, and the sender will await acknowledgement of the sent packet.

Modifications made to the Design Document submitted in Part 1:-

Sr No.	Original document	Modifications made based on implementation
1	Sender and Receiver window were assumed to be of the same size as the buffer (for in-order delivery).	The Buffer size is taken to be of smaller size (than sender/receiver window) since we are transferring only the correct inorder packet (only 1 packet) to the application.
3	FIN control bit was used.	FIN control bit was not used in the implementation.
4	CHK control bit was intended to be used to identify if the checksum was only for the header or for both header and data.	CHK control bit was used to identify if the packet received is a data / SYN segment or an ACK/NAK/KAL segment.
5	The packet structure for various types of segments were assumed to be different.	The packet structure remains the same for all different segment types and redundant header fields are dropped from the packet structure.
6	The connection establishment parameters are passed through the SYN segment to the server.	The connection establishment parameters are assumed to be known to both the client and the server before the file transfer.