**National Institute of Technology, Raipur**

**Department of Computer Science & Engineering**

****

**TCP over UDP**

**A Term Project on Network Programming**

**GitHub Project Link: https://github.com/NPPojectGroup/TCP-over-UDP**

**Submitted By:**

Roll no: 14115076 Name: Rishi Rai

Roll no: 14115070 Name: Pratyush Brahma

Roll no: 14115054 Name: Meghal Agrawal

Roll no: 14115078 Name: Saurabh Shrivas

Roll no: 14115084 Name: Shubham Agrawal

**Abstract**

The need of fast and reliable and reliable flow of information across the connected devices has always been a chief concern and prevalence of Real Time Application over internet requires a connection which is fast and reliable .The two major TCP/IP transport protocols fail to completely provide the desired feature for Real Time Applications to communicate. Thus to provide a suitable protocol that can provide flow of real time data across the internet efficiently the approach is to incorporate the functionality of both the protocols. Many protocols have been developed that try to provide TCP like functionalities to UDP in order to meet the needs in various networking scenarios .The main aim of this project is to highlight a general solution which can be used to achieve the desired functionality over internet for efficient communication between connected devices. This project aims to present a implementation of TCP-over-UDP for various applications.

The Github link of the project is:

<https://github.com/NPPojectGroup/TCP-over-UDP>

**List of Figures**

1. TCP Header
2. UDP Header
3. TCP State Diagram
4. UDP Packet Flow

**Table Of Contents**

**Introduction(Motivation/Problem/Project Outline4**

**Motivation to build TCP over UDP5**

**Chapter 1: TCP vs UDP6**

**Chapter 2: Methods to Implement TCP Over UDP8**

**Conclusion9**

**References9**

**Introduction (Motivation / Problem / Project Outline)**

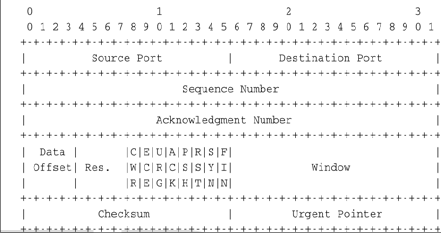
For communication among the different nodes of the network over Internet either of the two prevalent TCP/IP protocols are used which are Transmission control protocol (TCP) and User Datagram Protocol (UDP). But both of these protocols fails to provide the necessary requirements for the Real Time Applications to communicate, that is to provide a communication which is reliable as well as fast. Thus the main purpose of this project is to provide a solution to this problem; TCP-over-UDP is based on the better features of both TCP and UDP to provide a fast and reliable protocol which can be established in the TCP/IP environment.

The features and implementation of Transmission control protocol (TCP) and User Datagram Protocol (UDP) are as follows:

**Transmission Control Protocol (TCP):**

TCP provides a connection-oriented, reliable, and byte stream service. The term connection-oriented means the two applications using TCP must establish a TCP connection before they can exchange data. For achieving reliability, TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments. TCP transfers a continuous stream of bytes. TCP does this by grouping the bytes in TCP segments, which are passed to IP for transmission to the destination. TCP itself decides how to segment the data and it may forward the data at its own convenience. In addition to the properties above, TCP is also a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.

TCP header :



**Figure 1 TCP header**

**User Datagram Protocol (UDP) :**

UDP uses a simple [connectionless communication](https://en.wikipedia.org/wiki/Connectionless_communication) model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. UDP uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagram may arrive out of order, appear duplicate, or go missing without awareness of hosts. UDP assumes that the error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network transport level.

UDP Header:



**Figure 2 UDP Header**

**MOTIVATION TO BUILD TCP-OVER-UDP:**

From the explanation above it is explainable that both TCP and UDP are incapable of providing the communication overheads –fast and reliable simultaneously, thus it is not feasible for the Real Time Applications to be able to use any of the protocol mentioned above.

UDP being a connectionless unreliable service is fast, which is required for the real time communication but it being unreliable makes it unusable in Real Time Environment. On the other hand TCP provides Reliability and Flow Control but being slow and expensive to use it cannot be used in real time communication. But, Mixing the better features of both the Protocols is can provide a solution which is fast and reliable. This is the main aim of this project to highlight the need of such a solution.

As TCP instance can be made on top of UDP which provide exactly same functionality as TCP. It provides exactly the same congestion control, flow control, reliability, and extension mechanisms as offered by TCP. It is intended for use in scenarios where applications running on two hosts may not be able to establish a direct TCP connection but are able to exchange UDP packets. Thus TCP-over-UDP for Real Time Applications can be a suitable option for the developers to create real time network applications.

**Chapter 1: TCP VS UDP**

TCP and UDP can be compared using following points:

1. Connection Oriented vs Connection Less

First and foremost difference between them is TCP is a connection-oriented protocol, and UDP is connectionless protocol. This means a connection is established between client and server before they can send data. Connection establishment process is also known as TCP handshaking where control messages are interchanged between client and server. Attached image describe the process of TCP handshake, for example, which control messages are exchanged between client and server.  
  
The client, which is the initiator of TCP connection, sends SYN message to the server, which is listening on a TCP port. The server receives and sends an SYN-ACK message, which is received by client again and responded using ACK. Once the server receives this ACK message,  TCP connection is established and ready for data transmission.  
  
On the other hand, UDP is a connectionless protocol, and point to point connection is not established before sending messages. That's the reason, UDP is more suitable for multicast distribution of the message, one to many distributions of data in single transmission.

1. Reliability

TCP provides the delivery guarantee, which means a message sent using TCP protocol is guaranteed to be delivered to the client. If a message is lost in transits then its recovered using resending, which is handled by TCP protocol itself. On the other hand, UDP is unreliable, it doesn't provide any delivery guarantee. A datagram package may be lost in transits. That's why UDP is not suitable for programs which require guaranteed delivery.

1. Ordering

Datagram packets may arrive in any order. That's why TCP is suitable for application which needs delivery in a sequenced manner, though there is UDP based protocol as well which provides ordering and reliability by using sequence number and redelivery

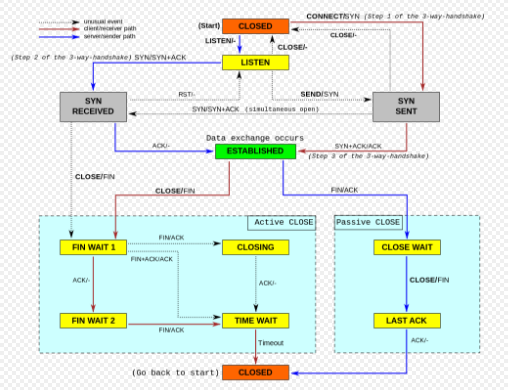
1. Speed

TCP is slow and UDP is fast. Since TCP does has to create a connection, ensure guaranteed and ordered delivery, it does a lot more than UDP. This cost TCP in terms of speed, that's why UDP is more suitable where speed is a concern.

1. Flow Control

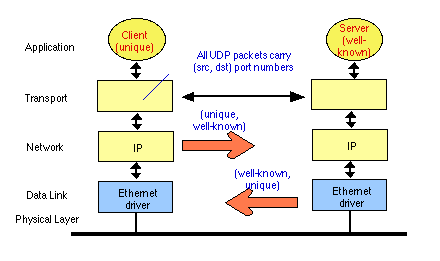
TCP does Flow Control. TCP requires three packets to set up a socket connection before any user data can be sent. TCP handles reliability and congestion control. On the other hand, UDP does not have an option for flow control.

TCP State Diagram:



**Figure 3 TCP state diagram**

UDP packet flow:



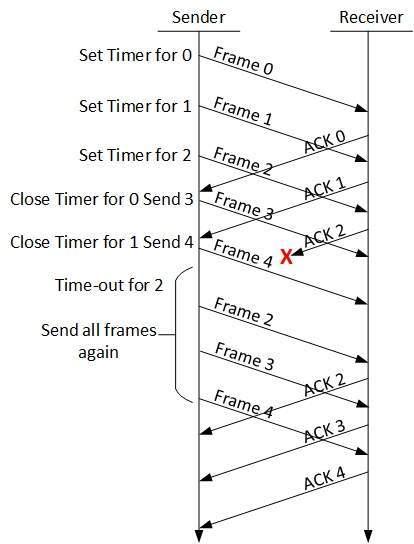
**Figure 4 UDP packet flow**

**Chapter 2: Methods used to implement the TCP over UDP**

As error and flow control are the important task in TCP. We implemented Go-back-N as described in [1] protocol for error and flow control. The protocol is briefly described as bellow.

**Go-back-N :**

The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame’s sequence number. When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not received any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

****

**Figure 5: Go-Back-N ARQ**

We have also used the predefined CRC32 class in java to implement checksum bytes, to check if the sent data frames were corrupted or not.

We used two threads one for sending the frames and one for receiving the acknowledgement bytes. We use our window size to be 10 and a packet size to be of 1024 bytes. We use base as our variable as the starting number of the frame in a window. We update the value of the next packet to be sent if the sent packet gets corrupted or if timeout Occurs. And we update the base value whenever a successful acknowledgement is received for the packet which was not corrupted.

Initially we use one thread to send out our packet one by one until 10 packets (window Size) are sent out, and then the tread is suspended for some time. The receiver thread then starts to receive the acknowledgement for the packets and updates the Base and the variable that holds the value of the index of the packet that is needed to be sent next.

**Conclusion**

It is obvious that TCP can be a more reliable option for communication than a UDP connection. Implementing a TCP-Over-UDP can be useful in different scenarios where error and flow control are needed. Such a combination can be used as base for Real Time Application developers to redesign the protocol in the way that suits their application. Such a protocol would have great implementation in different media applications which require audio and video transmission.

**References**

<http://www.acm.org/publications/authors/reference-formatting>

<https://www.youtube.com/watch?v=PYhNHX-MNXs&feature=youtu.be>

[1] Kurose, James F.; Keith W. Ross. Computer Networking: A Top-Down Approach.