DOI: 10.21917/ijct.2010.0008

CROSS LAYERED HYBRID TRANSPORT LAYER PROTOCOL APPROACH TO ENHANCE NETWORK UTILISATION FOR VIDEO TRAFFIC

Matilda.S[1], B.Palaniappan[2]

[1]*IFET College of Engineering, Villupuram, India*
Email: matildags@yahoo.com
[2]*Annamalai University, Chidambaram, India*
Email: bpau2002@yahoo.co.in

Abstract
Video data transfer is the major traffic in today's Internet. With the emerging need for anytime anywhere communication, applications transmitting video is gaining momentum. Real Time Protocol is the primary standard for transfer of video data, as; it requires timely delivery and can tolerate loss of packets. Streaming is the method used for delivering video content from the source server to the user. But this has many drawbacks: a) It sends only the amount of data equivalent to the streaming encoded rate to the client irrespective of the available bandwidth in the path. Hence the links are underutilized; b) It utilizes the link for the entire period of transfer and hence the link is not available to service other new clients. Thus as the number of clients increases, the network performance decreases. In this work, the advantages and disadvantages of the combination of different protocols in the application layer and transport layer are analyzed. The significant characteristics of each of these protocols are utilized and a combination of protocols for improving the network performance is arrived at, while retaining the QoS of video transmission.

Keywords:
Streaming, Real Time Protocol (RTP), Real time streaming protocol (RTSP), File Transfer Protocol (FTP), Catalyst FTP (CFTP)

1. INTRODUCTION

RTP is designed for real-time transfer of multimedia data and uses UDP at the transport layer for transmitting the packets. Another well known combination, FTP in the application layer with TCP in the transport layer is used to transmit real-time video. But the bandwidth required for transmission in both cases is high that alternatives are essential to save cost and time [1], [2], [3].

Example 1.1

Calculating the Streaming Bandwidth:

One hour of video encoded at 300 Kbps (encoded at 320 x 240 pixels window size) will be

$$\frac{3600 \text{ s x } 300000 \text{ bits/sec}}{8 \text{ x } 1024 \text{ x } 1024} = 128 \text{ Mb.}$$

If a unicast protocol is used for 1000 people to view this video, the Bandwidth required is

300 Kbps x 1000 = 300Mbps.

At this rate the entire spectrum will not be sufficient for today's Internet traffic. One way to work economically on bandwidth, without sacrificing the QoS of the transmitted video, is to use hybrid mechanisms for transmission and determine the combination of the protocols that can be used to achieve this.

2. EXISTING PROTOCOL COMBINATIONS

2.1 RTP OR RTCP WITH UDP

RTP (Real time transport protocol) and RTCP (Real time transport control protocol) was specifically designed for streaming media over networks [4].The transport layer protocol used is UDP which does not guarantee delivery of the media stream. If there is data loss the stream may suffer a "dropout". However this is the simplest protocols available and easy for use.

2.2 RTSP WITH TCP

RTSP (Real time streaming protocol) was also designed to stream media over networks, but it uses TCP in the transport layer. This ensures guarantee for delivery by using acknowledgements, retransmissions and timeouts. Hence it is more complex to implement.

The above two methods may work well if the server has to serve only one client. It utilizes an optimal bandwidth for serving the client. This bandwidth depends on the duration of the video to be streamed, minimum delivery stream required or requested by the client and the delay tolerance.

$$\text{Delivered stream rate} = \frac{CD_i . L_{\min}}{SD} + L_{\min} \qquad (1)$$

where CD_i – Delay tolerance by the Client

L_{min} – Badwidth of the weakest link in the path

SD – Streaming duration.

If the number of clients increases, they cannot be served as the weakest link in the path between the client and the server acts as a bottle neck. Clients occupy the entire links in the path for the entire streaming period and hence the links are not available for use by other clients. Hence fewer users are serviced and there is a chance that the higher bandwidth links are underutilized. The only option is to transmit the video upto a particular service point and then stream the data to the clients. The idea is to take the content as close as possible to the user using minimum bandwidth and network resources. The protocols used are FTP in the application layer and TCP in the transport layer [5]. As FTP is used the entire bandwidth along the path is utilized to the fullest and transmission time is reduced. But the complexity in

using TCP still remains as it does not support multicasting. The search for better options is continuous and the research and development in this area is gaining momentum.

3. ALTERNATIVES IN THE MARKET

By default FTP uses TCP for transfer of video files. The advantage of this combination is that, the entire link is utilized resulting in minimum bandwidth wastage (characteristics of FTP). Reliability increases as TCP guarantees that no packet is lost. The disadvantage of this combination is that, TCP responds to latency by adjusting the amount of unacknowledged data that can be on the link before waiting for acknowledgement. But TCP has limits on the window size. Degradations due to slow-start, window size, and acknowledgment frequency serve as a bottle-neck. Hence when the bandwidth delay product exceeds a certain threshold, the result is a lot of waiting or "dead air". Hence the ulterior motive of link utilization fails.

Catalyst has come out with a solution in the form of Catalyst FTP called CFTP [6]. This protocol is used in the application layer and UDP is used in the transport layer. UDP is not reliable as it just sends packet to fill up the pipe. With no TCP window throttling the transmission and wasting time for acknowledgements, UDP can be used to utilize the link to the fullest. But packets might be lost due to the unreliable UDP, resulting in poor video quality during playback.

To compensate for this shortfall CFTP incorporates retransmission and congestion control in the application layer. Functions of CFTP include sending new frames, taking care of lost frames and retransmission. In addition it incorporates congestion control algorithm and hence does not overwhelm the network. All the transfer parameters are under the control of the end user. Conditions such as delay tolerance, latency and packet loss can be dictated by the client [6]. The net result is that the link is utilized to the fullest and guaranteed delivery of all frames.

The main drawback in using CFTP lies in the fact that, it performs the duties of the lower layers in the application layer. Application layer retransmissions must be sent over the entire path, thus incurring higher latency and wasting bandwidth. Link layer retransmissions are more efficient than application layer retransmissions and congestion control. The application layer gets overloaded by performing the duties of the lower layers and hence the concept of OSI model fails as all the functions are pushed to the upper layer. The complexity of implementing congestion control if passed onto the application layer is a burden which many would prefer to avoid.

4. HYBRID TRANSPORT LAYER PROTOCOL APPROACH

4.1. BASICS OF VIDEO ENCODING

Any video is encoded from its raw form, so that it is compressed and can be easily transmitted over the network. The commonly used encoding formats are MPEG-4 and H.264. MPEG-4 is able to crunch massive video files into pieces, which are small enough to send over mobile networks. The intention behind H.264 is to provide good video quality at substantially lower bit rates than previous standards. It is the format of convergence in the digital video industry regardless of the video playback platform [7].

Both these encodings have a common feature – The entire video content is divided into frames. MPEG 4 uses three types of frames I frame (Intra frame), P frame (Predicative frame) and B frame (Bidirectional frame).H.264 uses only I frame and P frame. Inside a 9000 transmitter, frames of video are captured from the camera and sent to the internal encoder to be compressed. Each frame of video is then compressed: as an I-frame or as a P-frame or as a B frame.

An I-frame is a video frame that is encoded without reference to any other frame of video. A video stream or recording will always start with an I-frame and will contain regular I-frames throughout the stream. These regular I-frames, also called intra frames, key frames or access points, are crucial for the random access of recorded files, such as with rewind and seek operations during playback. The regularity of these I-frames is known as the I-frame interval. [7] [8].

However, I-frames are compressed as any still image is compressed (using the DCT, quantization, run-length encoding etc.) and hence are large. There are two or more (typically 3 to 6) I frame, each second in a group of frames. Complex frames are encoded as I frames.

P-frames are motion-compensated frames, as the encoder makes use of the difference between the current frame being processed and a previous frame of video (forward prediction), ensuring that information that does not change, e.g. a static background, is not repeatedly transmitted. MJPEG, H.264 not only looks for differences but searches for motion that has occurred in the video [7] [8].

A B-frame, or bi-predictive inter frame, is a frame that makes references to both an earlier reference frame and a future frame (forward and backward prediction) [7] [8].

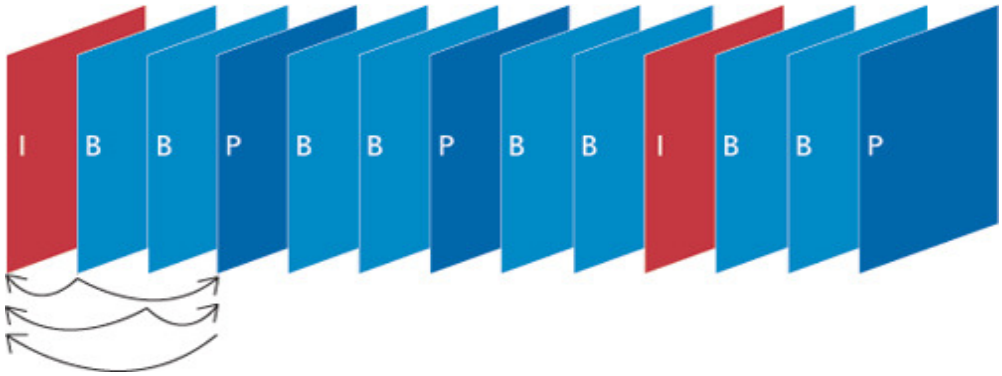

Fig -1 – Arrangement of video frames in a GOP

An I frame plus the following B and P frames before the next I frame define a GOP (Group of Pictures). The size of the GOP can be 8, 12 or 16 depending on the video and display formats. By decreasing the frequency of I-frames (longer GOP), the bit rate can be reduced.

4.2. TCP - UDP HYBRID APPROACH

The size of the I frame is large when compared to P or B frame. Loss of an I frame can distort the quality of video, as the following B or P frames cannot be decoded, while loss of P or B frame will have negligible effect. Hence to retain quality I frames are to arrive intact at the receiver. To enable this TCP is used to transmit the I frames while the P and B frames are transmitted using UDP. FTP is used in the application layer. Thus both the transport layer protocols are used simultaneously to serve the video application [9] [10] .

4.3. ALGORITHMS FOR CHECKING FRAMES

I and P/B frame can be segregated in two ways. One method is based on the size of the frame and the other is based on the GOP. In both cases it is sufficient if the I frame is identified. The remaining frames could be any other type with any type of prediction. Thus these methods are suitable for all types of encoding.

4.3.1. Differentiation based on size of frame:

I frame is encoded like any still picture and hence there is no actual compression in this case. P and B frames are encoded with reference to an I frame and hence are 50% and 25% of the size of the I frame. The first frame of any video is an I frame. The size of this first frame is to be determined. With this as a base number the size of the remaining frames can be judged. If the size is approximately equal to the base number then that frame is to be sent to TCP for transmission. If the size is significantly less than the base number UDP is used for transmitting the frame.

Algorithm- 1

When a frame arrives at the transport layer
/ Determine the size of the first frame in bytes */*
Sizeof_Iframe = size of first frame
/ Compare the size of all the frames with 75% of the size of I frame */*
If size of frame< 0.75 (Sizeof_Iframe)
transmit using udp
Else
Transmit using tcp

End

4.3.2. Differentiation based on GOP:

The frames usually follow the same pattern in a GOP. For example if the size of the GOP is 8 then the normal pattern is I B P B P B P B, I B P B P B P B (Fig -1). The size of the first I frame could be determined. Using this size, occurrence of the next I frame can be determined. The difference between the positions of the two frames, gives the size of the GOP. If this value is 8 then every 8^{th} frame could be sent to TCP and the remaining frames could be sent to UDP. The disadvantage here is that if any intermediate complex frame is encoded as an I frame, then it is sent using UDP, while the first method covers this problem.

Algorithm -2

When the first frame arrives at the transport layer

/ Determine the size of the first frame in bytes */*
Sizeof_Iframe = size of first frame
Transmit using tcp
/ Determine the size of the GOP */*
gop=1
For Count = 1,
/ Compare the size of all the frames with 75% of the size of I frame */*
If size of frame> 0.75 (Sizeof_Iframe) transmit using tcp
End for loop
Else
Count = Count+1
gop = gop+1
Transmit frame using udp
Continue for loop
For Count = Count

If mod(Count/gop)=0, transmit frame using tcp
Else
Transmit frame using udp
Count = Count + 1
End

All these algorithms are with the assumption that the bandwidth and bit rate does not change during transmission.

5. SIMULATION USING OMNET++ AND INET FRAMEWORK

5.1 NETWORK DESIGN

A network with a server and five regions is assumed for simulation. The number of clients in each region is generalized and is varied for each run of the simulation [11] [12] [13]. The time at which the client requests the video file from the server and the delay tolerance and minimum speed required at the client end for continuous playback is given in the table.

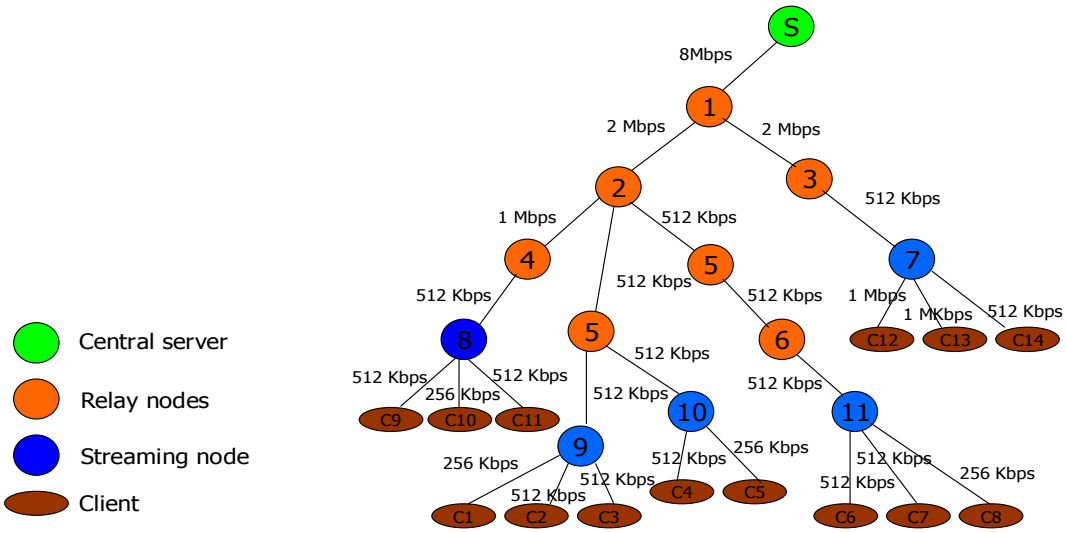

Fig -2 Sample Network used for Simulation

Table -1 Client Parameters

Clients	Time at which request is made in Minutes	Client's requirement	
		Delay tolerance in secs	Rate required for playback in Kbps
C1	0	10	256
C2	5	90	512
C4	15	30	256
C11	15	30	1024
C14	15	60	1024
C3	100	30	512
C13	110	30	512

5.2 DESIGN OF VIDEO FILE

A GOP of size 12 was chosen for the video. The frames were designed in a ".mpg.gdf" file and served as video input. Sample coding which defines the size of the frames is as shown [11].

25	[frames/second]	Frame Rate
0.08	[seconds]	Initial Delay
148976	[bits]	I-Frame
42216	[bits]	B-Frame
59312	[bits]	B-Frame
32316	[bits]	P-Frame
43849	[bits]	B-Frame
59245	[bits]	B-Frame
59356	[bits]	P-Frame
29576	[bits]	B-Frame
31994	[bits]	B-Frame
56660	[bits]	P-Frame
29944	[bits]	B-Frame
29999	[bits]	B-Frame
120944	[bits]	I-Frame

The same pattern is repeated so that the length of the video lasts for 120 minutes.

5.3 RUNNING TCP AND UDP ON THE SAME MACHINE

As I frames are essential while decoding and cannot be lost, they are transmitted using TCP, while the less important B and P frames are transmitted using UDP. Hence both TCP and UDP are to be configured in the transport layer. The new files are in the "inet-framework-inet d37c1fb\src\transport\TCPUDP" folder of INET framework and are imported into the workspace directory.

5.3.1 Starting the Server:

Starting a server on TCP port 54999 and UDP port 54888 [9].

```
Server server = newServer();
server.start();
server.bind(54999, 54888);
```

The `start()` method starts a thread to handle incoming connections, reading/writing to the socket, and notifying listeners. This adds a listener to handle receiving objects:

```
server.addListener(newListener() {
  public void received (Connection
connection, Object object) {
    if (object instanceof SomeRequest) {
      SomeRequest request =
(SomeRequest)object;
      System.out.println(request.txt);

      SomeResponse response = new
SomeResponse();
      response.txt = "Success!";
      connection.sendTCP(response);
    }
  }
});
```

Listener class has other notification methods that can be overridden. Typically a listener has a series of `instance of` checks to decide what to do with the object received. In this example, it prints out a string and sends a response over TCP [9].

The SomeRequest and SomeResponse classes are defined as follows:

```
public class SomeRequest {
  public String text;
}
public class SomeResponse {
  public String text;
}
```

5.3.2 Connecting a client:

To connect to a server running on TCP port 54999 and UDP port 54888:

```
Client client = new Client();
client.start();
client.connect(500, "172.16.25.7", 54999,
54888);
SomeRequest request = new
SomeRequest();
request.txt = "Request for realtime video!";
client.sendTCP(request);
```

The `start` method starts a thread to handle the outgoing connection, reading/writing to the socket, and notifying listeners. The thread must be started before `connect` is called, else the outgoing connection will fail. In this example, the `connect` method blocks for a maximum of 500 milliseconds. If it times out or connecting otherwise fails, an exception is thrown. After the connection is made, it sends a "SomeRequest" object to the server over TCP [9].

This code adds a listener to print out the response:

```
client.addListener(new Listener() {
  public void received (Connection
connection, Object object) {
    if (object instanceof SomeResponse) {
      SomeResponse response =
(SomeResponse)object;
      System.out.println(response.txt);
    }
  }
});
```

For each run of the simulation the number of clients served was increased [9].

6. RESULTS

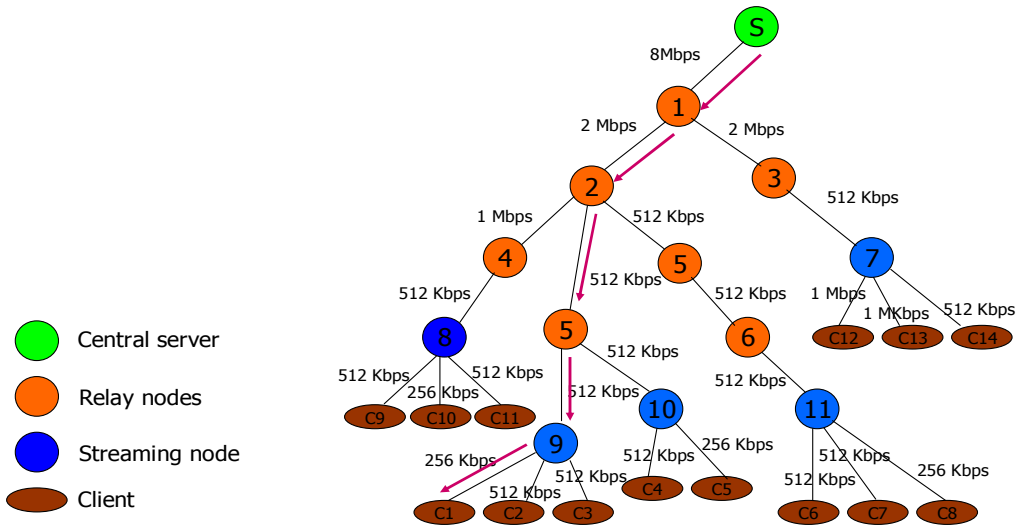

Fig.3 Analyzing the route to the Client

Client C1 is the first client to request the video file from the Server. If the entire video is to be streamed from the server, the shortest path for transmission is (S-1-2-4-10-C1) and the streaming duration is 120 minutes. Client requirements are 256Kbps speed and 10 secs delay tolerance [3].

Minimum Link speed along the path is

Min (8Mbps, 2 Mbps, 512Kbps, 512 Kbps, 256 Kbps) = 256 Kbps

Delivery stream rate at client1 (using formula given in (1)) is (10*256/120*60) + 256 = 259.55 Kbps. Hence playback is smooth. After 5 minutes Client 2 sends a request, but this request cannot be serviced as the two 512Kbps links serves as a bottle-neck. At this time clients C3, C4, C5, C6.C7 and C8 cannot be serviced, while one client among C9, C10, C11 and one among C12, C13 and C14 can be serviced. Hence at 5 minutes only 3 out of 14 clients can be serviced if streaming is done from the server [3]. Using hybrid approach 12 out of 14 clients can be served for the same setup. Simulation was done upto a maximum of 30 clients and the results in the graphical form is as shown in Fig.4.

The undulations in the initial stage are due to the manner in which the clients were chosen. The first two clients were from the same region ie. they were served by the same streaming node. The next two clients were from the next region and so on. This was done intentionally to observe the pattern in which the values would change.

Based on ITU standard measurement, delay and jitter of 150 msecs is acceptable and 150 -400 is acceptable provided the client is aware of the transmission time. Analysis shows that the network performance is acceptable for video traffic and is capable of supporting more number of clients than streaming.

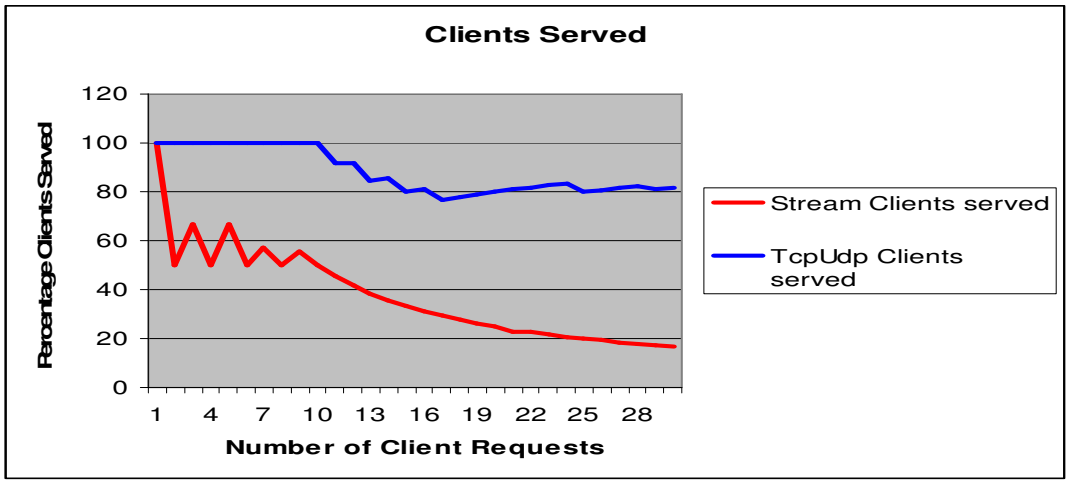

Fig -4 Comparison of Clients Served

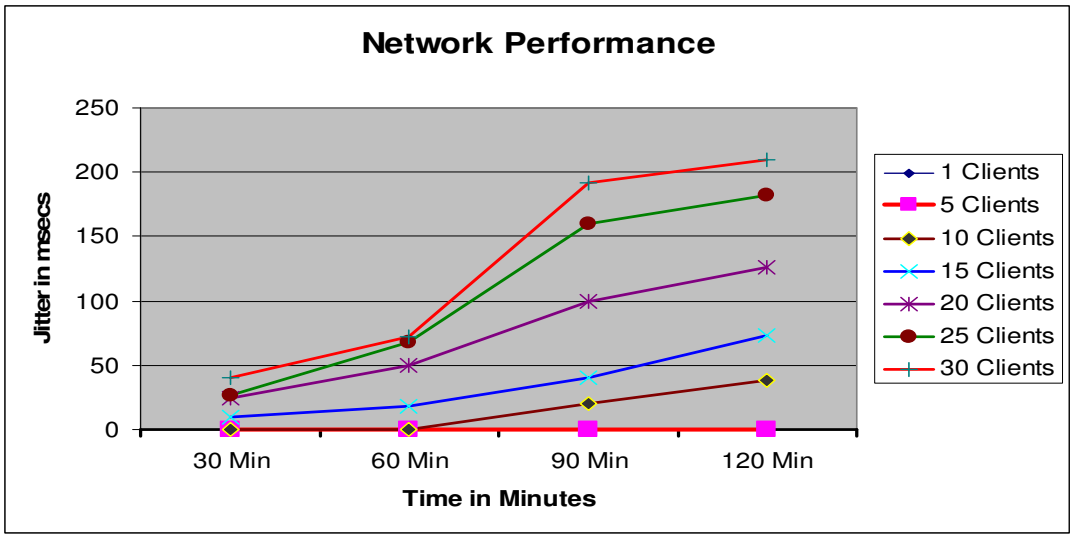

Fig -5 Network Performance based on Jitter

6. CONCLUSION AND FUTURE WORK

In this paper the network performance for transmitting video using cross layered hybrid transport layer protocol approach has been investigated. It is shown that the network capacity is utilized to the fullest and the number of clients served increases dramatically, when compared to normal streaming methods. Omnet++ was chosen for creating the simulation environment, as extending the network in wireless and mobile environment could be done easily.

Future work includes increase of video QoS based on the network conditions and the size of the network. This would require employing a server that could change the video packetization scheme based on the network conditions. Mean Opinion Score (MOS) and bandwidth analysis will considered in future work.

REFERENCES

[1] Insight Research Corporation. (2006). *Streaming Media, IP TV, and Broadband Transport:Telecommunications Carriers and Entertainment Services 2008-2013*. Insight Research Corporation, Boonton, NJ. (http://72.167.184.78/reports/IPTV08.asp).

[2] Yevgeni Koucheryavy, Dmitri Moltchanov "*Performance evaluation of live video streaming service in 802.11b WLAN environment under different load conditions*", Jarmo Harju Institute of Communication Engineering, Tampere University of Technology, Finland,Pp-1-12.

[3] Annanda Th. Rath, (2006) *A Hybrid Streaming Mechanism for Delay- Tolerant Multimedia Applications* , M.Tech Dissertation, IIT Bombay.

[4] Cranley, N. Debnath, T. Davis, M. (2007) *The Effects of Contention among stations on Video Streaming Applications over Wireless Local Area Networks – an experimental approach*. ITT Conference, Blanchardstown, Ireland.

[5] FileCatalyst, March 1, 2009, Unlimi-Tech Software Inc. <http://www.filecatalyst.com>.

[6] http://www.reelseo.com/encoding-formats-mpeg4-vs-h264/ "*H.264 versus MPEG-4 – Video Encoding Formats Compared"* by Mark R Robertson.

[7] Understanding H.264 Video, http://www.indigovision.com.

[8] Project kryonet created by nathan.s@gmail.com -TCP and UDP client/server library for Java http://code.google.com/p/kryonet.

[9] Santhana Krishnamachari, Mihaela van der Schaar, Sunghyun Choi, Xiaofeng Xu, "*Video Streaming over Wireless LANs: A Cross-layer Approach"*, Packet Video 2003 École polytechnique de l'université de Nantes April 28-29, Nantes, France.

[10] András Varga, *"The Omnet++ Discrete Event Simulation System"* Department of Telecommunications Budapest University of Technology and Economics Pázmány Péter sétány 1/d. 1117 Budapest, Hungary, <https://labo4g.enstb.fr/twiki/pub/Simulator/SimulatorReferences/esm2001-meth48.pdf>.

[11] OMNeT++ Community. 2009. <http://www.omnetpp.org>.

[12] INET Framework, 2009, <http://www.inet.omnetpp.org>.