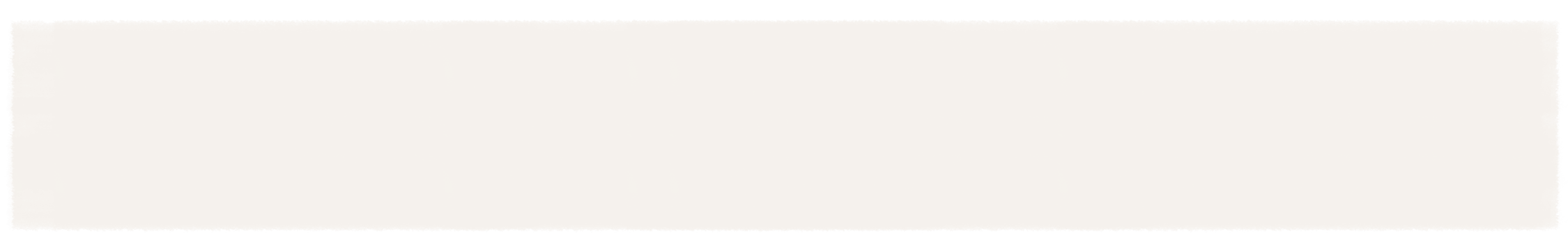
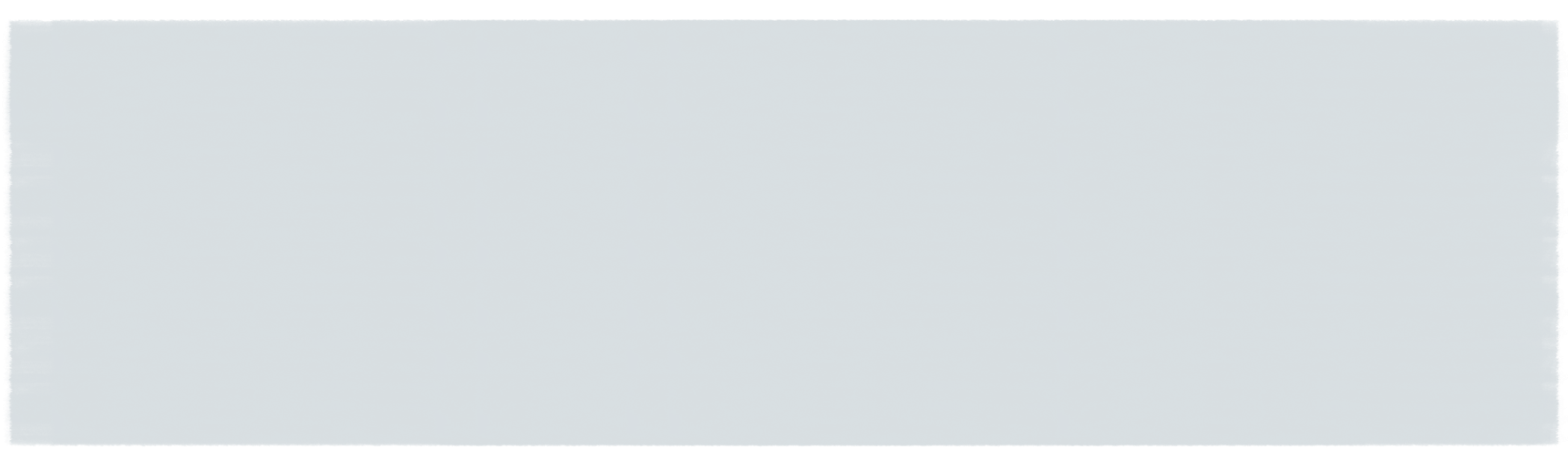
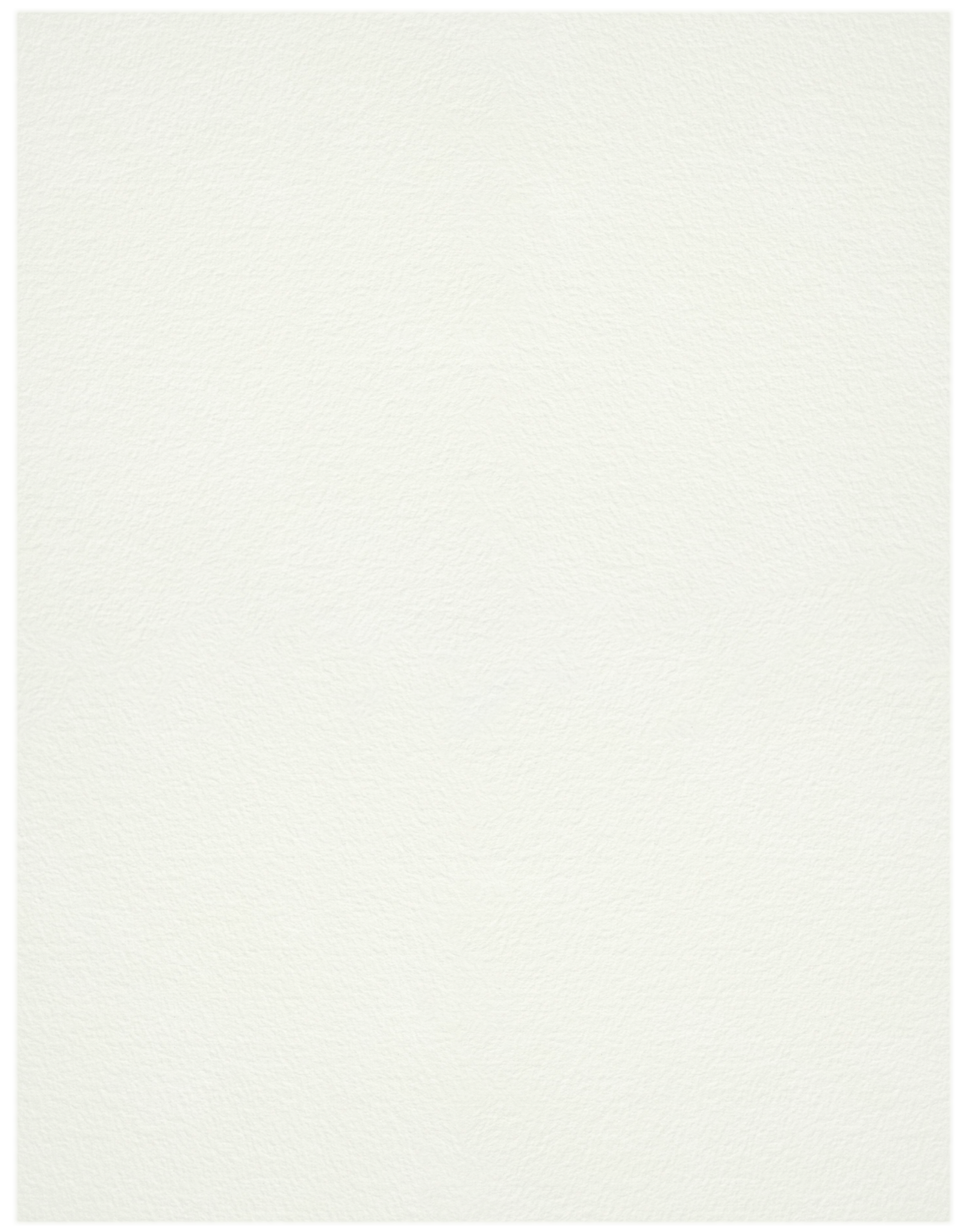
# A Comparison of Streaming Media and Real-Time Media Applications *With reference to their use of network resources*



**Dipen Delvadiya**

A20346457

04/27/2016

**ABSTRACT**

This following report will consist of an analysis of different Wireshark traces of real-time media and streaming media application taken under two different testing environments. The purpose of this report is to observe the different packets used to stream these applications. By doing so, it will allow us to understand the traffic and protocols in the network when these different application are being used.

Contents

[A Comparison of Streaming Media and Real-Time Media Applications *With reference to their use of network resources* 1](#_Toc449643876)

[Introduction 4](#_Toc449643877)

[Test Environment 6](#_Toc449643878)

[First Test – The Baseline of the Test Environment 7](#_Toc449643879)

[Second Test – YouTube Video 8](#_Toc449643880)

[Third Test – Blackboard Video 10](#_Toc449643881)

[Fifth Test – Google+ Hangout 14](#_Toc449643882)

[Observations, Analysis and Conclusions 16](#_Toc449643883)

# Introduction

Nowadays, most of the business, professional or personal work are relying more on the networking industry. Everyone knows how to use the Internet, but really few people know how it works. Also sometimes administrators have had to face a loss of the performance of the network managed. People usually assume that it is because of slow speed of the network. However administrators know that cases are like those are not easy to solve due to lack of sources available to find out the reasons behind that. At some point, everyone face some problems while streaming any media on their personal devices. This could be the traffic in the network or any third party attack that tries to put the server out-of-service. As an IT person, you are supposed to find out the point of source causing any problem in your network. In doing so, I used Wireshark tool to check my network traffic and flow of the packets. Wireshark is an open-source protocol analyzer that runs on any platforms. Wireshark provides a range of filters that allow us to perform certain tasks to observe the flow of the network.

The main goal of this paper is to gain an understanding of how these applications communicate with each other using Wireshark. It captures the packets of the network. Packets are the units of data that are routed between source and a destination on the Internet to communicate with each other. When your computer connects to a network, it relies on network interface card to send and receive these packets. Multimedia defines multiple forms of information such as audio, video, graphics, and images. Today with advancing technology in these medias, multimedia has become really popular feature on the Internet. Today, there are two types of multimedia that are being used the most on the Internet: real-time media and streaming media applications. In this report, I took two different traces of two streaming media applications, YouTube and Blackboard. One trace was taken while streaming a video and second one was taken while watching a video. To know the difference between real-time media and streaming-time media applications, I took two different traces of two real-time media applications: Skype and Google hangout. In this case, I took one trace while making a connection, and I took second one when connection was made and it was steady. Before I jump to protocols, let me explain real-time media and streaming media applications.

**Real-time media application:**

Real-time media application is an application program that runs within a time frame that users sense it immediately. The latency of these applications must be always less than a defined value measured in seconds. The examples of this type of media are Skype, Google hangout, Face Time, and Magic jack. In this context, the term "real-time" is synonymous with "live". The media is interactive in this type of applications. The end-users send the outputs of their webcams to each other using these applications. Both users can send and receive data to each other.

**Streaming media application:**

Streaming media is video and audio content sent in compressed form on the Internet. Users do not have to wait to get streamed since they can pause, forward or go back. The media itself is not interactive. Examples of these media applications are YouTube, and blackboard. The only difference is that end-user cannot send any data to the server. The end-user request to stream a data, and once request is accessed; it’s only a sever that communicates with the end-user.

The main goal of this paper is to study the impact of these different applications on the network and computer. I want to find out the protocols that are used to stream these applications. Protocols are the special set of rules that specify the interactions between the communicating devices. Examples of the protocols are UDP, ARP, HTTP, TCP, IP, and SFTP. Also this report aims to explain the characteristics of the IP datagrams that carry the transport and application layers.

Two most useful protocols in multimedia are UDP and TCP. Both of these protocols used for sending bits of data over the Internet. TCP stands for Transmission Control Protocol and UDP stands for User Datagram Protocol. Both of these protocols are widely used in multimedia applications to communicate. The difference between these two protocols is how they work. TCP guarantees that the recipient will receive in given order; while UDP will not wait to make sure that recipient received the packet. For example, you are watching live stream program on your computer, and live streams are often broadcast using UDP. If you lose your connection for few seconds, it will skip the bits you missed. It is interesting to see how different media applications use different protocols and packets in order to get the desired result. I have traced so many frames in trace that carry some information within it. Each frame has to go through the steps of IP datagram, and that makes interesting to see how each layer carries one of these streaming applications in different characteristic ways. Traces I have taken for this report will help us identify these different ways.

This report is organized to give an explanation of the processes that take place during any conversation. Next, we will be talking about the impacts of both of these media applications on the network. I hope this report will help the readers to gain some knowledge and get a better understanding of real-time and streaming media applications.

# Test Environment

I used my personal computer MAC OS X 10.10.5 mac Book pro to conduct all the traces. My CPU’s maximum RAM is 8GB and storage is 128GB. Most of my networks were located at my apartment, but there was one time when I took one trace at my friend’s apartment. I took blackboard streaming media trace on different network. Hence only for this trace, IP address of my computer and Gateway router is different. I took all other traces under same network. For Google+ hangout, YouTube, Skype, and Baseline trace, my computer has an address has an IP address 10.0.0.14 and gateway router has an address 10.0.0.1. For blackboard trace, my computer and gateway router have IP address 192.168.0.26 and 192.168.0.1 respectively. For all the networks, Comcast was a service provider. I tested my upload and download speed using [www.speedtest.net](http://www.speedtest.net). My download and upload speed at my apartment were 64.46 Mbps and 11.87 Mbps. For my other network, my download and upload speed were 15.58 Mbps and 10.85 Mbps.

I was getting Internet through LAN hosted by Comcast.

# First Test – The Baseline of the Test Environment

I took a trace of my network for 31 seconds with all applications turned off on my computer. I added a filter, which is able to remove all other conversations except my computer’s conversation with others. My computer had total 34 conversations during the test. It shows the traffic in my Wi-Fi. That also means that there are some applications running that making the computer to communicate with the networks. Some of the protocols I collected are TCP, SMB, MDNS, and NBNS. My computer was connected with 3 layer 2 addresses of conversation. Layer 2 basically means the data link, which is responsible for moving data across the physical links in the network. I found 3 addresses my computer was connected with on layer 3. Layer 3 is responsible for forwarding packets through the routers. Layer 3 has two different types of addresses: IPv4 and IPv6. My computer found three addresses IPv4 and none with address IPv6. Most of the protocols I found are TCP in most of the frames. I captured total 9329 bytes in 31.65 seconds of trace. In layer 2, total 3498 bytes were used. From those, 1850 bytes went from my computer to others, and 1688 bytes went from others to mine. I captured 3498 bytes in layer 3. From those, 1924 bytes went from my computer to others, and 1574 bytes went from other computers to mine. In my trace, I found the average number of captured bytes per captured packet is 100.311, and average number of displayed bytes per displayed packet is 106. I had all application turned off, so my computer exchanged most of the bytes and packets with my gateway router. Still it surprised me how my computer still had conversation outside my network with an IP address 224.0.0.251 with MDNS protocol. I found that it was an ITunes, which I forgot to log off. Hence you can find out if there are any applications running in the background. Figure 1 displays the statistics of the conversations my computer had using different protocols. For all of the conversations, since I was not exchanging any data, all frames had flag 0x0000. In this trace, I had 6 HTTP request packets too. I couldn’t fine out all the applications that were running, but I could find only ITunes. Average captured bits rate is 2358 bits/second, and average displayed bits rate is 17000 bits/second. Before I ran the test my CPU usage was around 1.80% and memory usage was 2.62 GB. After I ran the test CPU usage dropped down to 1.40% and memory usage got increased to 2.72 GB

Figure 1

# Second Test – YouTube Video

Connecting Trace:

The second test was to watch the video in YouTube and capture the packets and traffic through Wire shark that was running on the network while watching the video. The resource that was used is browser, flash player and the Wire shark running at the back to capture the data. My computer had an IP address 10.0.0.14. I took two traces for YouTube video. First one was taken when I was trying to connect with a server, and as soon as connection was made and video started I stopped the trace. This trace was taken for 24.59 seconds. To find out the conversations my computer had, I had to filter computer’s IP address. In this trace, average captured bits/sec were 1416000 bits/second. There were two HTTP messages in the frame numbers 787 and 807. In server key exchange frame number 807, there were two secure sockets layer. There were four frames carrying application data, and those frames are 727, 728, 827, and 922. Each frame has heavy load around 775 bytes as well. All these frames have secure socket layer 5, so it is not possible to see what was sent. In frame 10, three-way handshake took place, since my server was requesting foe video to get streamed. After the video has started, there are mostly DNS protocols in the trace. In this trace, my computer exchanged most of the bytes with an IP address 162.250.144.215. After searching for that IP address, I found that it is belonged to Incero LLC. My computer was requesting to get data from that server that is why it had exchanged most of the bytes with it. My computer had 2 Ethernet conversations, 12 IP conversations, 13 TCP conversations, and 19 UDP conversations.

**Figure 2 – connecting YouTube**

Steady State trace:

After the video was started, I took this trace for around 12 seconds to understand the conversation while the video is being streamed. In this trace, my computer has exchanged 2781 packets and 25632338 bytes. To make it secure connection, QUIC protocol has been used to make the connection. QUIC stands for Quick UDP Internet connection. The different protocols used to stream this YouTube video are BROWSER, NBNS, DNS, ICMP, TCP, and TLSv1.2. Frame numbers 1856, 2565, and 2566, which had BROWSER protocol, those conversations are getting backup list response. It is really interesting to see, how application layer protocol is displaying the hostname and my computer information as well. Also it has three other protocols named SMB, SMB Mailslot Protocol, and Microsoft Windows Browser Protocol. In this trace, my computer had 3 Ethernet, 5 IP, 1 TCP, and 11 UDP conversations. In this trace, UDP was used as application layer protocol. As I said before, this protocol is mostly used to broadcast. My computer had only one TCP conversation with socket 23.52.44.193:443, which are owned by Akamai Technologies. I don’t know why I had conversation with outsider party, but I had one conversation with it that exchanged 61000 bytes with each other. I might have had any aid while paying video or something else. The amount of average bytes per second was 200000 bytes/s. I also had DNS protocol while streaming a video. After checking layer 5 of DNS protocol frame, I found it was ITunes, which was communicating with Apple server. In number 2679 frame, server couldn’t reach the destination. So it gave me error messages using ICMP protocol, which is one of the main protocols of the Internet protocol suite that is used to send any error messages.

**Figure 3- steady YouTube**

Figure-2 and Figure-3 proves that UDP is mostly used to broadcast any videos on streaming media.

# Third Test – Blackboard Video

Connecting trace:

The third test was performed through Blackboard’s “IIT Online” videos. What we did here was we used Wireshark to capture data while we were streaming an online lecture video through media player. After about 30 seconds, the capture was stopped and analyzed. Then second trace was taken while playing this video, and it was taken for about 10 seconds. For this test, I took a trace with other network. For that network, my computer had an IP address 192.168.0.26, and my gateway router had an IP address 192.168.0.1. I had 24 QUIC messages on frame numbers 19, 70,73, 75,117, 118, 126,127, 129,130, 131,217, 218,219,220,221,233, 389, 402,409, 455,464,466, and 468. A flag during conversation was 0x0d. I found first QUIC conversation with a destination address of 2607:f8b0:4002:c09::93. However I could not find anything about this destination address. Figure 4 displays the statistics of conversation with my computer.

**Figure 4 – connecting Blackboard**

Steady state:

I took a trace for 20 seconds after the video was started. My computer had TCP and TLSv1 conversations only. I am surprised to see how I had to UDP conversation with server. Then after reading some articles, I found that UDP is the best for live streaming events, but UDP can’t buffer lost bits. Hence, it is not appropriate for replaying video-on-demand. I am pretty sure everyone has observed, when you watching any lecture online on blackboard and you try to skip more than it was loaded, it restarts from the beginning. I had only one TCP conversation with an IP address 216.47.143.62. This ISP is owned by Illinois Institute of Technology. In this test, I found 1 Ethernet, 1 IP, 1 TCP, and 0 UDP conversation. 4 client packets and 3 server packets were used to take place TCP conversation. At frame 16, client Hello message was sent, then it sent encrypted handshake message back and conversation started. After that, it started to get large application data starting from frame 21. Largest application data received was 1244 bytes. In this test, 193.7 average bytes/packet were sent from the server and 217.5 average bytes/packet were sent from my computer to a server. This shows that the streaming media sends you really large packets of application data encapsulated in TCP and TLS, and your computer sends really small packets to acknowledge recipient that you have receive the data. That is the reason why TCP was used to make the conversation.

**Figure 5 – Steady Blackboard**

If you compare, figure 4 and 5, you can see the difference between usage of protocols as need. To connect to blackboard, it used most UDP bytes, then in purpose of making secure connection, it used TCP bytes to exchange large number of application data.

**Fourth Test – Skype Call**

Connecting trace:

The forth test was performed by using an application called Skype under same circumstances as first three. Skype provides a two-way communication through video, audio and text in real-time. When using Skype, users don’t realize how much background things are going on and are not aware of it because they are too busy with the video call or an audio call. This will allow us to analyze the background data. The connecting trace was taken for 20 seconds. There was no three-way handshake in this conversation since it is real-time media application. The protocols used in this conversation were TLSv1, TCP, and UDP. UDP protocol is the most useful protocol for live conversation. Hence all of the data packets were sent through UDP messages. Highest data that was sent throughout this test was 444 bytes. Total 242 packets were exchanged throughout the test. In this test, I found 1 Ethernet, 18 IP, 10 TCP, and 12 UDP conversations. Total 33213 bytes were exchanged during the test.

**Figure 6 – connecting Skype**

Steady state:

After my fiend answered my call, I took this trace for like 10 seconds. The protocols used in this conversation were TCP, TLSv1, and UDP. In connecting trace, my computer had multiple UDP conversation outside the network, but it had only one conversation outside the network in steady state. That outside IP address is 172.56.12.213. This ISP is owned by T-Mobile USA. I think my friend used his data when we had conversation on Skype. Since UDP messages were carrying all the data, my computer sent 404000 and received 336000 bytes from a server. All those bytes were sent and received from that outside address. I couldn’t understand why it sent some application data using TLSv1 message since it was using UDP for exchanging the most bytes! I also had 7 TCP conversations during the test. My computer exchanged most bytes with TCP socket 104.208.31.113:443. When I looked up online, I found that Microsoft Corporation owns that ISP. Interesting thing is that I did not send any byte to that socket; I only received bytes from that socket. Highest bytes sent throughout the conversation were 519 bytes. During the test, my computer had one Ethernet, 8 IP, 7 TCP, and 1 UDP conversation.

**Figure 7 – Steady Skype**

From figure 6 and Figure 7, you can see how UDP used as application layer protocol was carrying huge amount packets, and network layer protocols carrying this application layer protocol also had huge amount of packets.

# Fifth Test – Google+ Hangout

Google Hangouts is fairly new real-time application and is very similar to Skype. It can also do two-way communication via text, audio or even video. With Google Hangout though, you are able to chat with up 10 users at once compare to Skype. I took two traces under same circumstances as first four.

Connecting Trace:

I started the trace as soon as I opened the browser and stopped the trace after 10 seconds of conversation. Total duration of this trace lasted for 37 seconds. The total packages were 11957 and the average was 755.09 bytes per packet. The total bytes exchanged during the test were 9028690 bytes. The protocols used for the conversation were BROWSER, DNS, and STUN. I had 2 Ethernet, 3 IP, 0 TCP, and 4 UDP conversations. My computer had two conversations outside the network with the IP addresses 75.75.75.75 and 74.125.142.127. 75.75.75.75 is owned by Comcast cable, and it is obvious to know that Google owns the other ISP since I was using Google+ Hangout. I exchanged most bytes with Google’s ISP. My server received 4964 bytes, and my server sent 5236 bytes. I was surprised to see no TCP conversation, because I expected to send smaller packets through TCP when I was connecting to Google+ Hangout. My computer had most of the STUN conversation with server. STUN is a client-server protocol. A STUN client sends a request to discover its public Internet Protocol and Ports, and STUN server then return a response. In my trace, I had all Binding request and Response. Binding request is usually sent over UDP.

**Figure 8 – Connecting Google+ Hangout**

Steady State:

After connection was made, I took this trace for 20 seconds. My computer had TCP and STUN conversation. Actually I did not expect TCP after the connection was made, but I found three frames 15164, 15227, and 15228 that had TCP conversation. Out of these three TCP messages, two TCP messages had data in it. Third TCP, which is acknowledging the connection, doesn’t have any data in it. Same as connecting trace, my computer exchanged the most bytes with UDP socket 74.125.142.127:19305. My device received 2920 bytes from that UDP socket and sent 3234 bytes with the server. I also had one TCP conversation with Apple socket 17.110.226.165:5223. During test, my computer had 1 Ethernet, 2 IP, 1 TCP, and 1 UDP conversation. The total bytes captured throughout the test were 14541803. It shows how huge amount of bytes get exchanged when having conversation with real-time media applications.

**Figure 9 – steady Google+ Hangout**

# Observations, Analysis and Conclusions

After going through all the traces, at least we get to know the difference how real-time media and streaming media applications work. When we were using streaming media applications, we seem to receive a lot of data. We were barely sending anything. We only sent data for requesting in order to get an access of all the information. After three-way handshake is done, we don’t really send anything; we only get data from a server. For example, we did not send any data to YouTube or Blackboard after the connection was made. While in the real-time media applications, we send and also receive huge amount of bytes. Google Hangout and Skype were all over the place. They were receiving and sending data at the same time. Real-time media applications always have UDP conversations, because UDP is the best choice for live streaming. On other hand, you can use either one of TCP or UDP for streaming media applications. It totally depends on the need and usage of that application. So if any important information were going to get transmitted then TCP would be a better choice that guarantees that recipient will receive the data. There are other protocols like STUN get used in real-time media applications to make your information secure and safe.

**References:**

"Blackboard | Reimagine Education | Education Technology & Services." Blackboard | Reimagine Education | Education Technology & Services. N.p., n.d. Web. 01 May 2015. <http://www.blackboard.com/>.

"Download." Wireshark · Go Deep. N.p., n.d. Web. 01 May 2015. <http://www.wireshark.org/>.

Gonzalez, Barb. "The Difference Between Streaming and Downloading Media." About Tech. N.p., n.d. Web. 28 Apr. 2015.

<http://hometheater.about.com/od/internethometheater/a/The-Difference-Between- Streaming-And-Downloading-Media.htm>.

"Google+ Hangouts." – Google Hangouts. N.p., n.d. Web. 01 May 2015. <http://www.google.com/hangouts/>.

"Skype Keeps the World Talking. Call, Message and Share Whatever You Want for Free." Skype. N.p., n.d. Web. 01 May 2015. <http://www.skype.com/>.

YouTube. YouTube, n.d. Web. 01 May 2015. <http://www.youtube.com/>.