CS297 Report

Online Video Chatting Tool

Sapna Blesson
sapna.blesson@yahoo.com

Advisor: Dr. Chris Pollett
Department of Computer Science
San Jose State University
Spring 2008

Table of Contents

Introduction

The demand for social networking sites is increasing day by day. A social networking site that allows you to video chat online is the primary inspiration for my project. The goal of my project is to build an online video chatting tool that enables users to join real-time streaming video chat rooms where users can share their video with multiple users. Users can send instant messages and share their live web cam data to other users in the chat room. My online video chatting tool will also have methods to add and delete new members into the access list. It will also support multi-user video conferencing.

Some of the existing video chat applications are stickam,YouCam, ichat, blogtv. My online video chat tool offers similar functionality like these social networking sites, and it is simple to execute and doesn't have to rely on any third party sites. Users can directly enter webcam chat rooms after signing up. Also this video chat application doesn't require any additional software installation on the client side.

The main technology used in this project is Flex. Flex is an Adobe product used for the development and deployment of cross platform, rich Internet applications based on Adobe Flash platform. Interactivity is achieved through the use of ActionScript. ActionScript is a scripting language based on ECMAScript. ActionScript supports Video and Camera classes to process and control the webcam. This project requires Adobe Flash Media Server which will work as a hub. Adobe Flash Media Server is real time media server and support client-server architecture. The server can send and receive data to and from the connected clients using the sever-side ActionScript code. Remote procedure calls are used to send request between clients and server.

This project report summarizes the research I conducted for the preparation of CS 298. During my initial phases, I learned about Adobe Flash Media Server and scripting using ActionScript which are the basic technology for my final project. For my first deliverable I installed and configured the Adobe Flash Media Server. I also created a sample application in order to make the server work. So this sample application streams pre-recorded mp3 file from server to the connected clients. Streaming a file from server to client is one of the foundation steps for my project. My second deliverable is the first step towards building an online video chatting tool that make a two way audio/video chat application that requires one publisher and multiple subscribers. The publisher flash client publishes the webcam data to the Flash Media Server. The Flash Media Server then streams the webcam data to the connected subscribers. My third deliverable focuses on bandwidth experiments of Flash Media Server which are the crucial feature of any video application. My fourth deliverable deals with load balancing issues of the Flash Media Server which is an important aspect of online video chatting tool. Finally I concluded the report with the description of the future work I will be doing as part of CS 298.

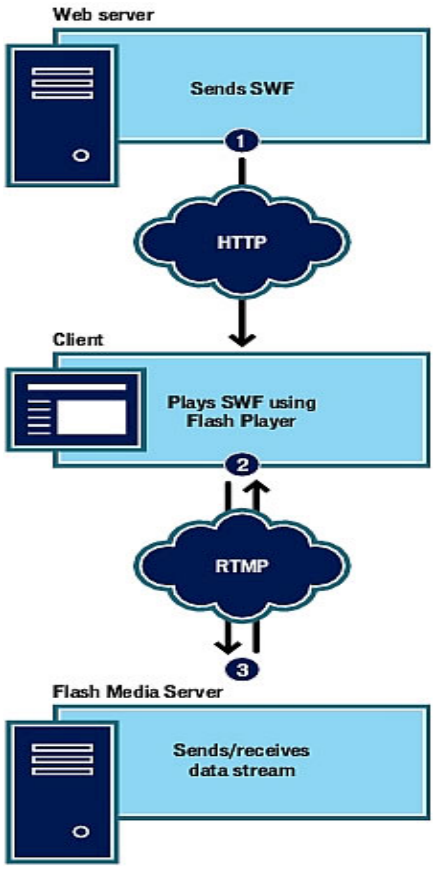
Deliverable 1

Streaming recorded mp3 file from Flash Media Server

My main focus of deliverable one was to learn about Adobe Flash Media Server and to build an application that shows how the server works. Adobe Flash Media Server works as the central server for my online video chatting tool. One of the main features of Adobe Flash Media Server is streaming audio/video over the internet. So in my deliverable one, I created a streaming application that streams recorded mp3 file from server to the connected clients. Streaming an application from server to connected clients is one of the key features of online video chatting tool. Deliverable one helped me to build the foundation for my project.

Adobe Flash Media Server has client-server architecture. The client code is written using client-side ActionScript and server side code is written using server-side ActionScript. Client code is compiled into SWF file.

The server and the client communicate over a persistent connection using Real-Time Messaging Protocol (RTMP). RTMP is a reliable TCP/IP protocol for streaming and data services. In a typical scenario, a web server delivers the client SWF file over HTTP. Once the client starts executing, it creates a socket connection to Flash Media Server over RTMP. The connection allows data to stream between client and server in real time.



The first step for streaming recorded mp3 file is to establish a connection between the flash client and the server using RTMP protocol. The client side ActionScript code has NetConnection and NetStream class that enables the client to establish connection with server. The server side code has Stream class which creates the stream and plays the pre-recorded mp3 file stored in the server's application folder.

```
//Client side ActionScript code

public var nc:NetConnection = new NetConnection(); //creates NetConnection object

public var subscribe_ns:NetStream;

nc.connect("rtmp://localhost/playmp3");          //connects to server

nc.call("play_mp3", null);                       //calls server-side function play_mp3

subscribe_ns = new NetStream(nc);                //creates NetStream object

subscribe_ns.bufferTime =1;
```

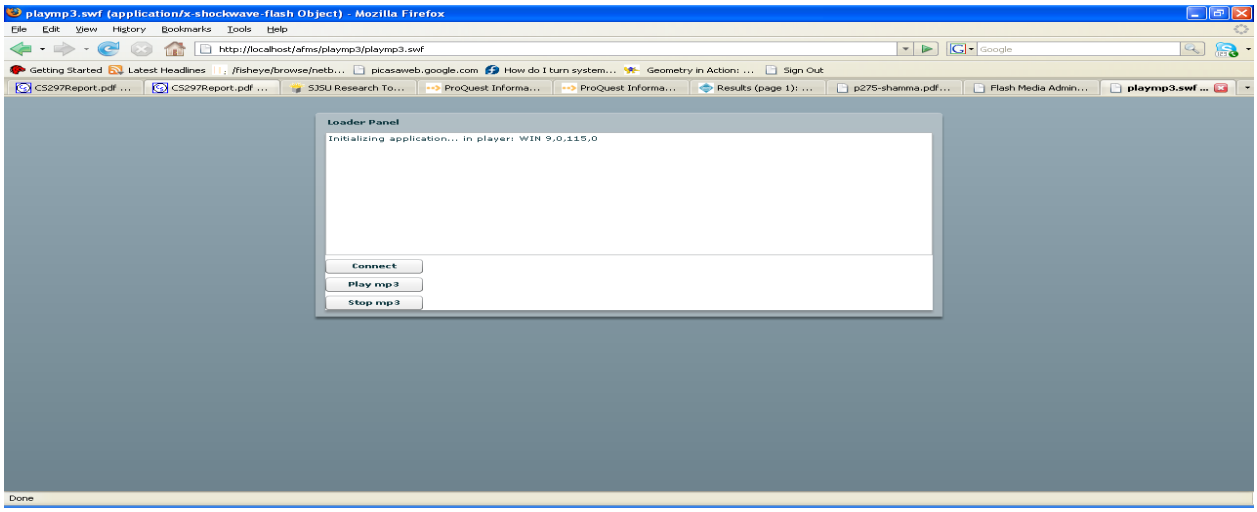
subscribe_ns.play("myAVStream"); //plays the stream

//Server-side code

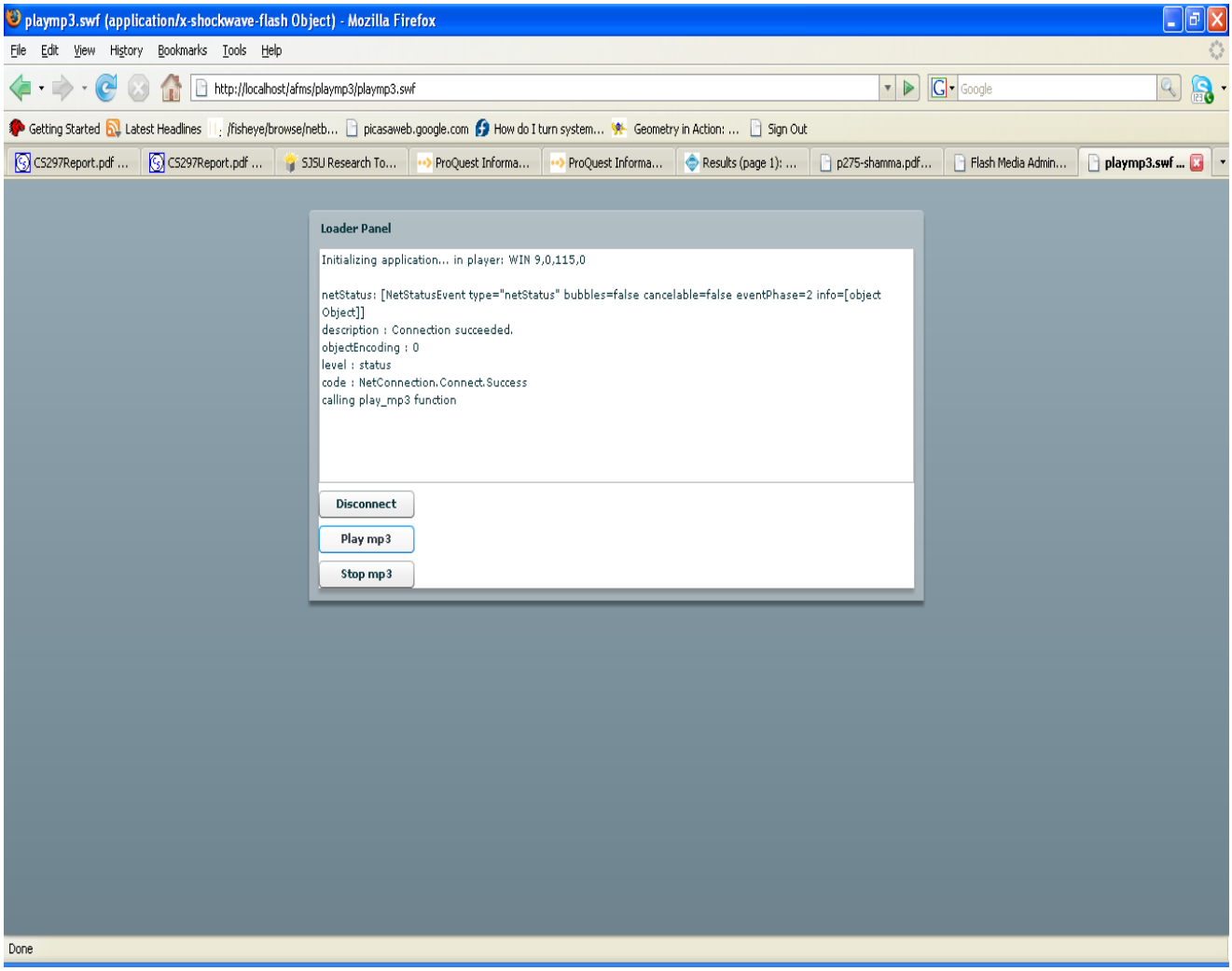
application.my_stream = Stream.get("myAVStream"); //creates stream

application.my_stream.play("mp3:music",0,-1); //plays the pre-recorded mp3

Front end of deliverable1



Click the connect button. When it establishes successful connection with server, sucess status message will be displayed in the panel and after that click the play mp3 button. Server then starts streaming the mp3 file.



After finishing deliverable 1, I understood the basics of client-server communication of Flash Media Server.

Deliverable 2

Streaming the output of webcam via Flash Media Server

The first step towards building an online video chatting tool is to make a two way audio/video chat application that requires one publisher and multiple subscribers. The publisher flash client publishes the webcam data to the Flash Media Server. The Flash Media Server has a server-side script which receives the published data and streams back to the subscribers. The subscribers will play the stream. Deliverable 2 focuses on

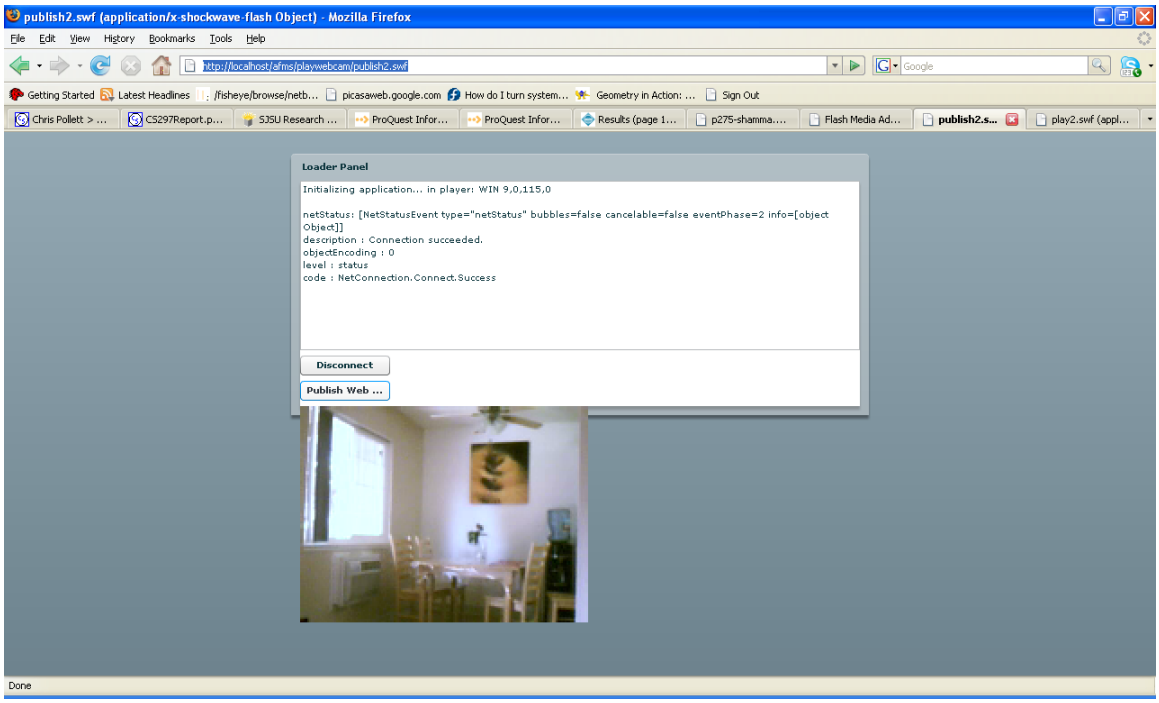
streaming the output of webcam via Flash Media Server. The client side ActionScript code has camera and video class that allows to process and control the webcam data.

```
//client-side ActionScript code for publishing webcam data

public var nc:NetConnection = new NetConnection();
public var ns:NetStream;
public var video:Video;
public var camera:Camera;

 nc.connect("rtmp://localhost/playpublish1");                    //connect to server
 ns = new NetStream(nc);
var mic:Microphone = Microphone.getMicrophone();
 video = new Video(camera.width * 2, camera.height * 2);
 video.attachCamera(camera);
ns.attachCamera(camera);                                  //Give video a ride on the stream

ns.attachAudio (mic);                                    //Give audio a ride on the stream

 myViewer.addChild(video);                                    // Show video in publisher
 ns.publish("livestream", "live");                           //publish webcam data

//client-side ActionScript code for subscribing to the webcam data stream

public var nc:NetConnection = new NetConnection();
public var ns:NetStream;
public var video:Video;
public var camera:Camera;

nc.connect("rtmp://localhost/playpublish1");                  //connect to server
ns = new NetStream(nc);
video = new Video(320,240);
video.attachNetStream(ns);
myViewer.addChild(video);
ns.play("livestream", "live");

//server-side ActionScript code for streaming the webcam data

application.myStream = Stream.get("livestream");
if (application.myStream){

   application.myStream.play("livestream", -1);
}
```

The server first gets the published webcam stream from the flash client using Stream.get(). After that the server broadcasts the live webcam stream to the requested clients using Stream.play().

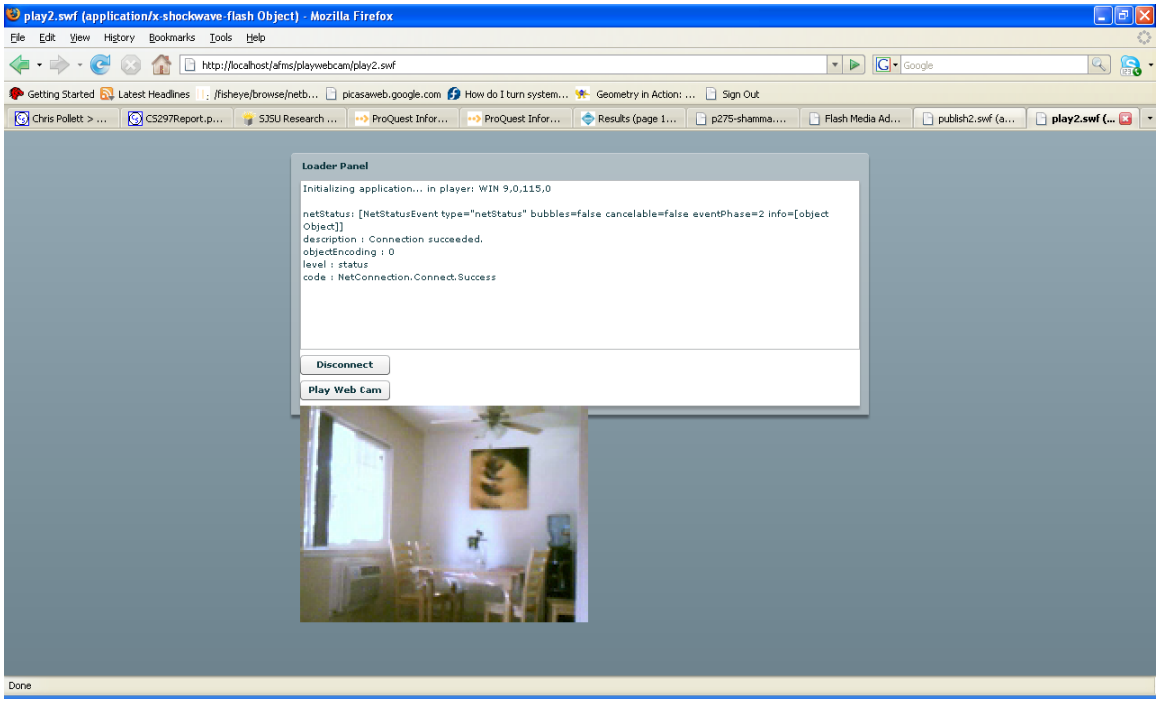
Front end of deliverable 2

Publisher publishes webcam stream from client to server



Click connect button. When connection succeeds, click publish webcam button. Flash client publishes the webcam data to the server.

Subscribers play the webcam stream via Flash Media Server



Clicks connect button and when connection succeeds, click play webcam button. Flash Media Server streams the webcam data from server to the connected clients.

Deliverable 3

Bandwidth experiments of Flash Media Server

The third deliverable focuses on the bandwidth experiments of Flash Media Server which are the crucial feature of any video applications. I conducted the experiments using Adobe Flash Media Development Server free edition. Its license limit allows only up to 10 simultaneous connections.

I experimented with bandwidth of Flash Media Server 3 using deliverable 2 (video chat application). In this example experiment, there is one publisher which publishes the web cam data to the server and the server streams this webcam data to nine subscribers. At a time the server can support 10 simultaneous users.

I experimented with 10 simultaneous connections in local system. I got a graph like this.

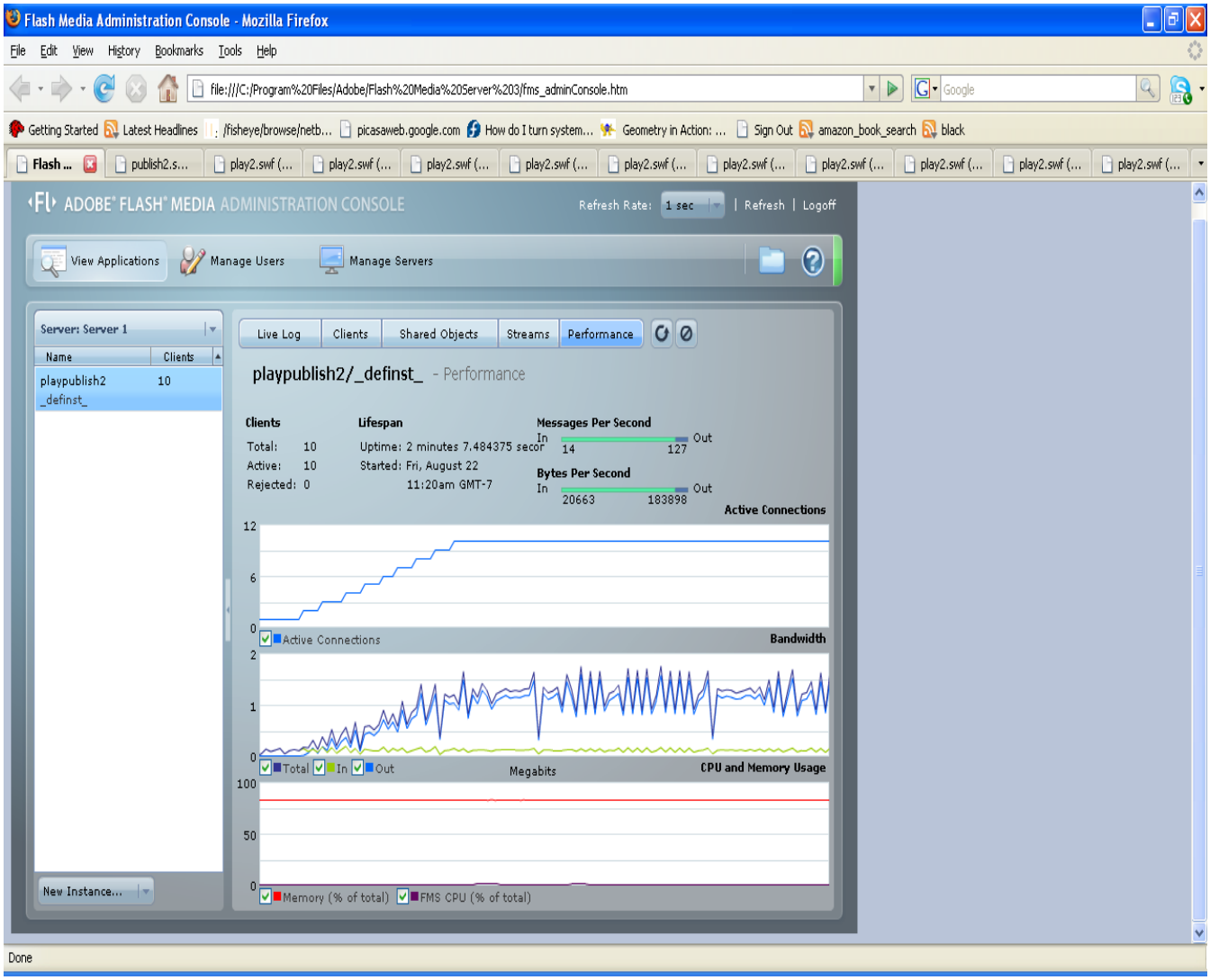


Figure: Bandwidth of 10 simultaneous connections in local system.

The first graph shows the total active connections which is 10 in this experiment. The second graph shows the bandwidth output. Green line is the input bandwidth and blue and dark blue lines show the output bandwidth and total bandwidth. The graph clearly shows that as the number of connections increases, bandwidth also increases. The spikes in the graph show that traffic is spontaneous rather than a smooth flow. The server sends data to the clients between intervals.

I experimented with 10 simultaneous connections on the same server using two machines connected through local area network (LAN).

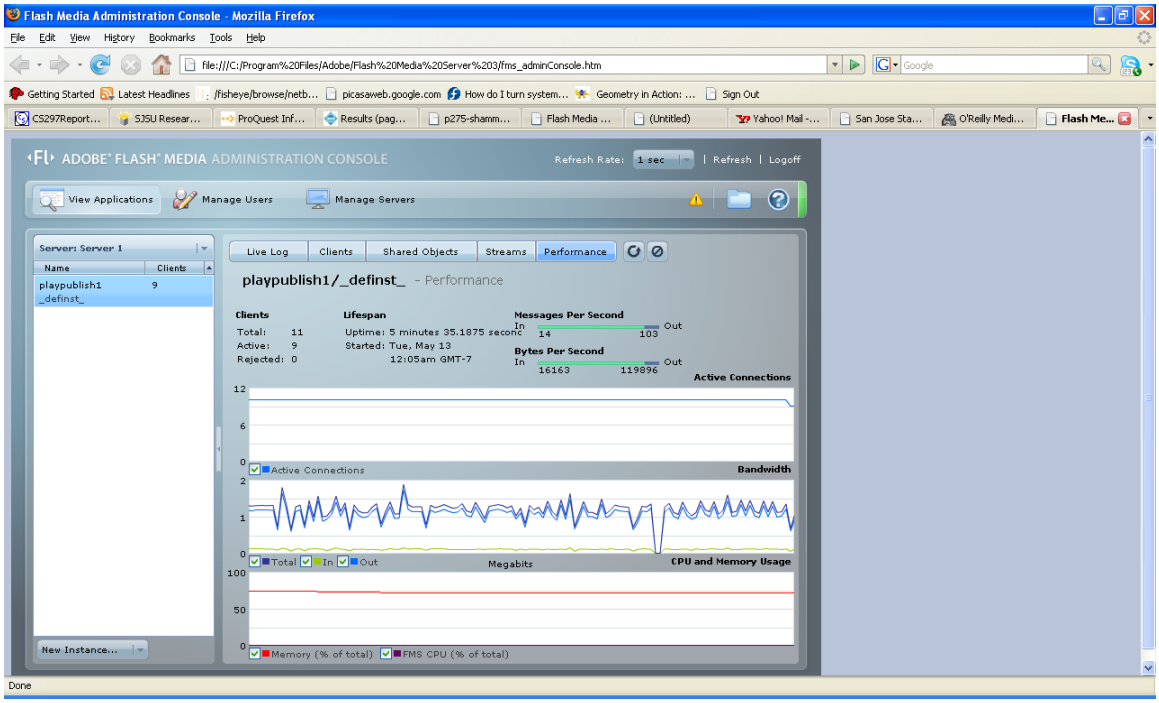


Figure: Bandwidth of 10 simultaneous connections through local area network (LAN).

The above graph shows that connection through LAN causes more congestion. Also the bytes of data per second are less compared to the connection in local system. So the data loss is more in this case. The picture quality also degraded.

From these experiments, I concluded that LAN connection can cause more congestion and for that reason the data transfer is less compared to connection within the local system. Also the picture quality of LAN connection is poor compared to the connection within the local system.

Deliverable 4

Load Balancing

Balancing the load among multiple servers is a very important aspect of online video chatting tool. Deliverable 4 focuses on balancing the load among multiple Flash Media Servers.

Load balancing can be achieved through different ways. A typical method is implemented using a caching server which sits between the clients and the servers. All the clients connect to caching servers and caching server will forward the request to the application servers. When the caching server services the request, it will make a copy of the data and store it locally. Next time when the client connects to the caching server, the caching server can provide the data without requesting the application servers.There can

be multiple application servers behind the load balancer and load balancer can send the request to the application server in a round robin fashion. There is lots of load balancing software available in the market. One example of load balancing software is squid.

But there is a potential problem with load balancing soft wares. Those are best fit for stateless application services. A typical web server is a stateless application service. Each request to the web server is independent and doesn't matter which machine services its request each time. So load balancing is very easily achieved through round robin and all the details of the multiple servers servicing the requests can be masked from the client.

Flash Media Server can intelligently direct traffic to a multiple server cluster using server-side scripting. This option is used for multi-way communication applications that require connections to be routed to a specific server. In the video chat application of the Flash Media Player there is one publisher and several subscribers. Publisher publishes the live web cam data to Flash Media Server which then streams the web cam data to connected subscribers. Both the publisher and the subscriber need to connect to the same Adobe Flash Media Server instance.

There will be a master Flash Media Server. Master server stores the details of all the other servers. Publisher client first connect to master server. In the server-side code it first accepts the client connection. It then checks whether the server capacity is exceeded. For example the Flash Media Development Server free edition supports only 10 simultaneous connections at a time. So if the total number of clients in the application exceeds 10, server then creates a new NetConnection object and then connects to the next server. Then the stream name is attached to a NetStream object and published live. This way Flash Media Server balances the load among multiple servers.

```
//server-side code of master server
var nc;
var ns;
application.onConnect = function(client)
{
   application.acceptConnection(client) ;
}


application.onPublish = function(client,livestream)
{

   if (application.clients.length>10)
   {
        trace("current server fully loaded");
        nc = new NetConnection();
        nc.connect( "rtmp://192.168.0.6/NextSever" );  //connect to next server

        nc.onStatus=function (info){
```

```
        if (info.code==”NetConnection.Connect.Success”)  //connection succeeded
        {
         ns = new NetStream(nc);
        trace(livestream.name+ " is up.");
         ns.setBufferTime(2);
        ns.attach(livestream);
        ns.publish( livestream.name, "live" );   //master publishes the stream to next server
         }}
     }

    else                          //master server capacity is not exceeded, so it plays the stream
      {
       application.myStream = Stream.get(livestream);
        if (application.myStream){
        application.myStream.play(livestream, -1);
             }
      }
}
```

Master server maintains a list of all Flash Media Servers. Master Server uses NetConnection object to connect to other server. All other servers first check whether their load is full. If their capacity is exceeded, they reject connection from master server. Otherwise they play the stream that is send from master server to the connected clients. If the NetConnection fails, the master server picks next server in the list and try to connect the next server and the so on. If capacity of master server is not exceeded, it plays the stream to the connected client.

Future Work

For my CS 298 I will extend the deliverables I worked on my CS 297 course. I will create a video conferencing application, where users can share their video with multiple users. Users will have the ability to invite new members to view their video. All users will have to authenticate to join chat rooms. I will create a user interface which looks more similar to real world video chat applications. The system will also have the capability to share its work across multiple servers.

The deliverables I worked in this semester helped me to build the foundation for implementing my final project.

References

[2008] Learning Flash Media Server 3.William B. Sanders. O'Reilly. 2008

[2008] Flex Tutorial from Adobe. "http://www.adobe.com/devnet/flex/quickstart/coding_with_mxml_and_actionscript/"

[2008] Adobe Flash Media Server. "http://www.adobe.com/devnet/flashmediaserver/"

[2008] Adobe Flash Media Server.”http://livedocs.adobe.com/fms/”

[2008] Adobe Flash Media Server.” http://fmsguru.com/”

[2008] Adobe Flash Media Server. ”http://fczone.com/”

[2008] Flash video player sources.http://www.flashstreamworks.com

[2008] Adobe Flash Media Server. “http://en.wikipedia.org/wiki/Adobe_Flash_Media_Server/”

[2008] Flash Media Server.http://www.sandlight.com/

[2008] Dynamic Stream Switching with Flash Media Server 3. “http://www.adobe.com/devnet/flashmediaserver/articles/dynamic_stream_switching_print.html/”

[2007] Programming Flex 2.0. Joey Lott, Chafic Kazoun.O'Reilly. 2007

[2007] Calculating your bandwidth and software needs of flash media server 2. fms_whitepaper_bandwidth.pdf. Chris Hock. 2007

[2006] Adobe Flex 2: Training from the Source. Jeff Tapper, James Talbot. Pearson Education. 2006

[2004] Developing Rich Clients with Macro Media Flex. Steve Webster, Alistair McLeod. Peachpit Press. 2004

[2003] ActionScript Cookbook.Joey Lott.O'Reilly. 2003