# Machine Problem 2: Video/Audio Streaming

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Mar 13, 2014 Due: 5:00pm April 9, 2014

## 1. Introduction

Twenty years ago, the only method you could watch a movie was to either go to a theatre or buy/rent a videotape. Things have changed a lot since then. People now prefer watching everything online. YouTube becomes the place where we watch most video. Even the DVD movie rental companies, like NetFlix, are investing most of their money in expanding the online streaming business. What is the magic that makes all this revolution happen? Network Streaming!

In the first MP, you have already learned how to capture the raw images and raw audio data from external devices, and got some experience in video/audio playback. Therefore, we now focus on how to stream the captured video/audio contents from one machine to another machine through connected network, which is also known as the Video on Demand (VoD) service model. Through this MP, you will have a deep understanding of the concepts like bandwidth, QoS, and why they matter so much.

You can use the Linux workstations in the EWS lab rooms. You can use your own machine as well. You choose to use the Logitech Webcam you have borrowed or you can use your own video/audio I/O devices.

## 2. Problem Description

### 2.1 System Overview

For this MP, you have to create a server and a client component. They should run on separate machines. However, to implement the bonus functionality, your media server should support multiple clients running concurrently. The functionalities of server and client components are given below.

#### Server

The server always listens to a port for any incoming client request. On receiving request from a client, the server will stream a pre-recorded video to the client. The video was encoded in motion JPEG format (all I frames) and was captured at a fixed rate. We use 30fps for the video capture and 8000Hz for the audio capture. (Note: you have to capture at these fixed rates even though a client may request for lower rates of audio and video. You will need this so that multiple clients [bonus functionality] can request in different rates <= 30fps). Figure 1 describes the setup required for MP2.

# Machine Problem 2: Video/Audio Streaming

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Mar 13, 2014 Due: 5:00pm April 9, 2014

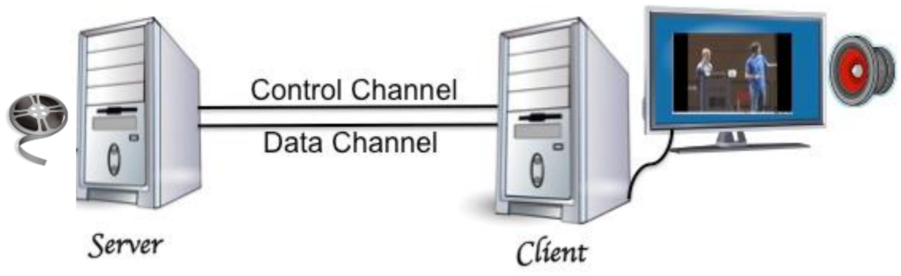


Figure 1. Server-Client communication architecture

### Client

A client is a separate machine with Graphical User Interface that can send a connection request to the server for video and/or audio streams of different rates. The client works in two modes: *active mode* and *passive mode*. In active mode, the client requests for both audio and video streams. The video rate may vary between 15fps to 25fps depending on the resource availability and the audio rate is fixed to 8000Hz. However, in passive mode, the client requests for only video streams (no audio) with 10 fps. The GUI should allow the user to change between modes. The client should display the video on the screen and send audio to the sound card. In active mode, the audio and video should be properly synchronized.

## 2.2 Functional Components

As a part of this MP, you will have experience in implementing different multimedia concepts such as Resource Admission, Resource Reservation, QoS Negotiation, QoS Enforcement and Real-time streaming and Session Control. Below we describe the required functional components for this MP.

### Client Resource Admission

Each client maintains a "resource.txt" file that stores the available application bandwidth ($AB_A$) of the local machine. Your client GUI should allow us to change this value. When a client is ready to send the streaming request to the server, it first performs the resource admission at the local machine. The admission process should consider defined media rate (by the user), the estimation of media frame size (**you can assume the resolution of required video is known by the client and consider average frame size to allow optimistic allocation**) and the available bandwidth at the client (can be obtained from resource.txt). Note that depending on the client mode and the required video, the average media frame size, frame rate and media type (video with audio or without audio) vary. A streaming request is sent to the server only if the admission process is successful. The admission process becomes successful if $B_A <= AB_A$, where $B_A$ is the required application bandwidth. If the admission process is not successful, show the rejection to the users. Remember that, in active mode, the admission process considers both the audio and video streams, where in the passive mode, the admission process considers only the video stream.

## Resource Negotiation

After the admission is successful at the client, it sends a connection request to the server to specify the requested media (client can request in audio and video in active mode or only video in passive mode) data, the resolution of the required video (to compute the frame size) and the required rates. Depending on these values, the server should perform resource admission as follows.

## Server Resource Admission

The server also maintains a "resource.txt" file that stores the available application bandwidth of the local machine. The admission process should consider client defined rates, the estimation of media (audio and/ or video) frame size, the capture rate of the requested video and the available bandwidth at the server (can be obtained from resource.txt). The request is accepted only if the admission process at the server is successful. If the admission process is not successful, the request is rejected and the rejection is informed to the client.

## Resource Reservation

Both server and client should maintain a resource table storing states describing the allocated resources for each client session. For multiple clients (optional feature) this table is important since, you cannot allocate resources that already been allocated to the other clients. If $B_A$ is the allocated bandwidth to other clients and $AB_A$ is the available application bandwidth (given in resource.txt), then when a new client request comes, the available application bandwidth to support the client request is $AB_A$- $B_A$. **Remember, you do not need to consider transport packet size, RTP header size or transport packet payload in your application level bandwidth computation.**

## QoS Enforcement

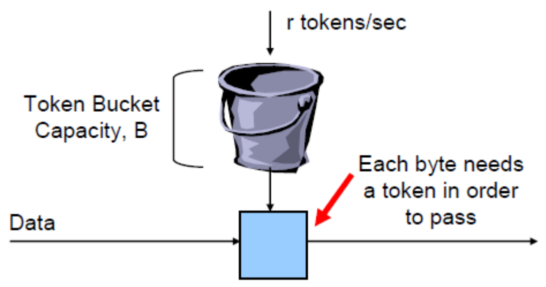
Once the request is accepted at the server, it uses a rate control mechanism to enforce the negotiated rate in the data communication (or streaming). You can implement token bucket, leaky bucket, gstreamer specific rate control or any simple queuing mechanism to control rates. Leaky bucket scheme is introduced in the class (March 10th) so we only briefly introduce token bucket here. You can skip this part if you are already familiar with the topic (CS438).

# Machine Problem 2: Video/Audio Streaming

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Mar 13, 2014 Due: 5:00pm April 9, 2014



The idea of token bucket is depicted in the picture on the left (borrowed from slides of Prof. Matthew Caesar and Prof. Robin Kravets). 'r' tokens are issued to the regulator per second and it has a bucket that can hold at most 'B' tokens. For one byte of data to pass through, it costs one token in the bucket. Thus, we can adjust parameters 'r' and 'B' to shape our traffic. As you can imagine, three situations can happen in this scheme. 1) As a data packet comes, there are enough tokens (# of tokens >= size of packet). 2) There are not enough token, thus the packet is discarded. 3) No packet comes before the bucket is full of tokens, thus extra tokens are discarded. You should handle all three situations if you choose to implement the scheme.

## Data Plane Communication

The audio and video data between the server and the client should consider RTP (or RTP like) and UDP. The RTP header should include media (audio and video) capture timestamp (the time when the data is captured from the webcam). You can use "rtpbin" of gstreamer for data communication or you can build your own RTP-like protocol over UDP.

## Audio Video Synchronization

If a client receives both audio and video data from the server (in active mode), the playback should synchronize the media data. To, synchronize, you need the capture timestamp from the RTP header of the video and audio frames.

## Session Control

You should implement session control functionalities at the client such as START session, STOP session, PAUSE streaming, RESUME streaming, REWIND video, FAST_FORWARD video, and SWITCH streaming modes. STOP session button should teardown the ongoing session, while PAUSE session button only stop the streaming but maintains the control connection. Therefore, the RESUME functionality does not require further negotiation of resource availability. However, START session button will start from the initial connection setup (Resource Admission, Negotiation and Resource Reservation). **Session control command can be given at any time during the streaming.**

## Session Adaptation

# Machine Problem 2: Video/Audio Streaming

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Mar 13, 2014 Due: 5:00pm April 9, 2014

Also, users can change the "resource.txt" (using GUI) at the client and the updated resource should be reflected in the current communication if necessary. For example, if we lower the machine bandwidth at the client machine, there should be again resource admission, and QoS negotiation both at the server and client.

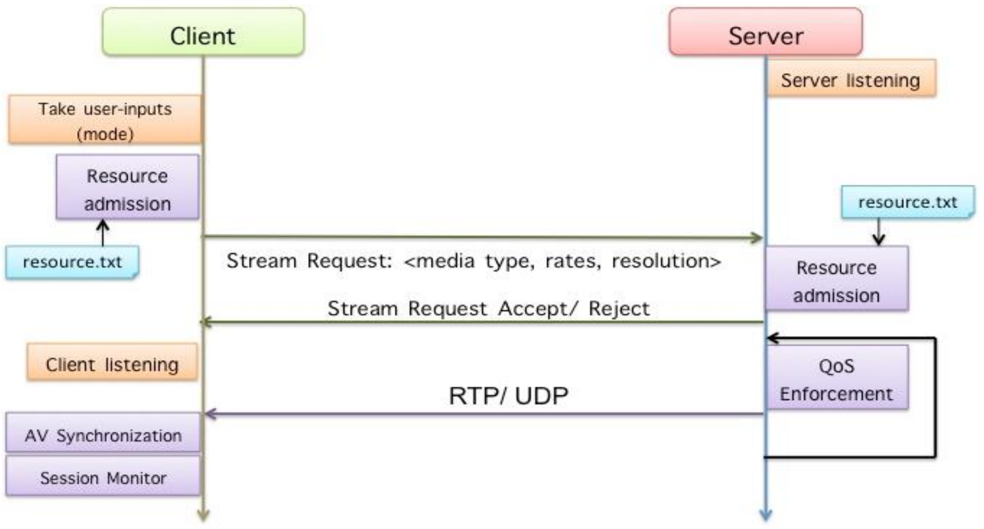


Figure 2. The information flow chart and the placement of functional components

## Session Monitoring

You should monitor the session at the client. The session monitoring should include: 1) Incoming bandwidth, 2) Synchronization skew, 3) Media Jitter (both for video and audio if any) 4) Failure rate.

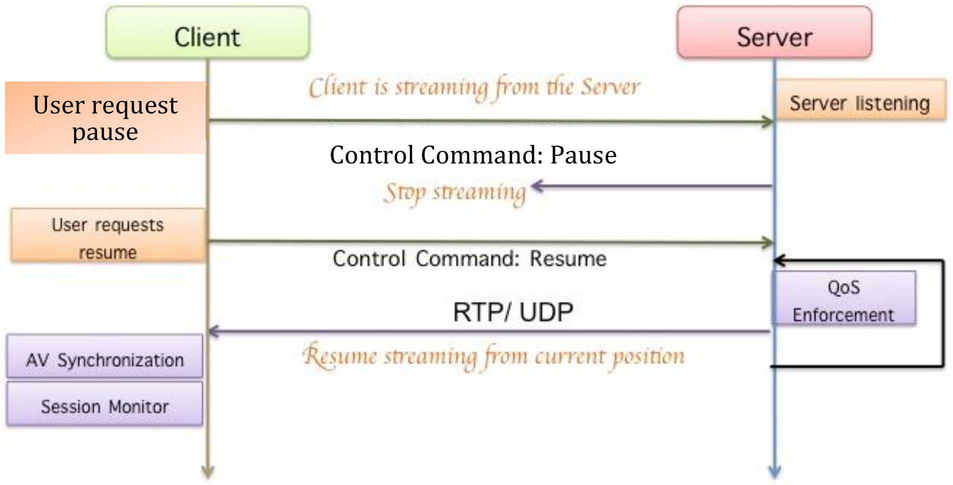


Figure 3. The information flow chart and the placement of session control

Figure 2 shows the placement of different functional components during the connection setup and streaming phase. Figure 3 shows the flow of session control during the streaming with pause and resume. The pause command from the users should stop the streaming and the resume command should re-start the streaming. The resume continues the playback from the frame when 'pause' was hit.

Remember that, any changes in the resource.txt should perform the re-admission control and QoS negotiation. **The streaming in active mode should be done with the highest possible rate (between 15 to 25fps) considering the client bandwidth defined in resource.txt**.

## 3. Required Features

**Video Streaming (10 points):** The client should be able to display the video from the server depending on the specified mode. Playback in active mode requires video with 15 to 25 fps. Playback in passive mode requires video with 10 fps. **For test videos stored at the server, you need to prepare at least two videos. One with 640x480 resolution and one with 320x240 resolution. You may start to record multi-view videos (two videos concurrently shot from different angles at the same object) since these will also serve as test videos for your next MP.**

**Audio Streaming (10 points):** The client should be able to play the audio from the server. Note that, in passive mode, there is no audio.

**A/V Synchronization (15 points):** In active mode of playback, the audio and video captured from the server should be properly synchronized. You will be judged depending on your synchronization perfection.

**Resource Admission and Negotiation (10 points):** You should implement a simple resource admission and negotiation protocol mentioned above. The server and client bandwidths are given in resource.txt file at each machine.

**QoS Enforcement (Rate control) (15 points):** You should create a proper rate control mechanism so that the network always gets the packet on the negotiated rate.

**Session Control (10 points):** The client GUI should implement the START/STOP/PAUSE/RESUME/REWIND/FAST_FORWARD functionality. User can also change the mode of the client at any time (SWITCH mode).

**Session Adaptation (10 points):** We should be able to update resource.txt at the client machine. Depending on the modification, the rate of the current session should be re-admitted and re-negotiated.
**Session Monitoring (10 points):** Display the monitoring output on the GUI in the text form.
**Report Writing (10 points):** Write a clean report describing your approach, algorithm and assumptions.

## 4. Bonus Features

**Supporting Multiple Clients (20 points):** We will provide 20 bonus point if your code supports multiple concurrent clients watching videos in different modes.

## 5. Submission

Pack all source codes and documents into a zip file or tar ball and submit them through Compass. Do not submit your solutions through email unless there are technical problems with the Compass system. The submission deadline is April 9th at 5:00 pm. You can get up to two days bonus for each MP, but please remember you can use only 3 bonus days throughout the semester. Further late submissions are not accepted and will get 0 point.

### 5.1 Source Code

You should only submit your source codes (.java, .c or .cpp files), Makefile, and any open source libraries (or jar files) you use in your solution into compass. Do not include any pre-compiled obj files (.o), binary execution, or pre-recorded media files in the directory.

### 5.2 Documentation

Your documentation should include two parts: user manual and development manual. The user manual should include all instructions on how to compile and install your source code, and how to run your program and test all features. If you have designed any GUI, it is better to attach some screen shots to explain. The development manual should have the implementation details of all features. The implementation details include program flow, key data structures; media file formats, important algorithms, and so on.

## 6. Evaluation

The evaluation will be done by face-to-face interview with each group. You will run your program. We may ask you to show the source code implementing different functionalities. Your solution is evaluated based on how many features you have implemented and demonstrated. In order to get full score (100 points) of this MP, you need to implement all 100-point required features. 20-point optional features will be used to offset your lower marks at MPs, homework's or exams. Evaluation will happen on Friday April 11th at lab SC218 at 5:00 pm. A sign-up sheet will be provided (by email or using compass) to schedule your demonstration slot.

**[Please Attend MP2 Lecture on Friday March 21th, 2014. TA will discuss various design alternatives and gstreamer functionalities that you can use for implementing this MP.]**