

## Assignment 3

### CS414, Multimedia Systems (Instructor: Klara Nahrstedt)

Posted: March 7, 2008

Deadline: April 4, 2008

#### Disclaimer

Details in the following assignment are subject to change. Although no additional requirements will be added, it is possible that some current requirements might be relaxed. Clarifications and additional hints, which will also be posted on the course website, might also be added to this document.

#### Introduction

This is the third assignment for your Internet Television (IPTV) project where you will experiment with the following: (1) the concept of a proxy, and (2) streaming protocols for video and audio from two servers to two clients via the proxy (see Figure 1). You will need to consider different scenarios (see Figure 2). The goal will be to stream audio data and video data from two multimedia servers to two multimedia clients with streams being relayed through the proxy. You should assume that two different audio and video data streams are stored in separate files on each multimedia server (i.e., audio1/video1 movie is stored on server1 and audio2/video2 movie is stored on server 2). Each media type will be separately streamed in chunks to separate viewers (i.e., audio1/video1 will be streamed from SERVER1 via proxy to CLIENT1 and audio2/video2 will be streamed from SERVER2 via proxy to CLIENT2). You should reuse as much code as possible from Assignment 1 and 2 in terms of reading from the audio and video files, playing audio and video on the client sides, and streaming audio video streams from servers to proxy and from proxy to clients.

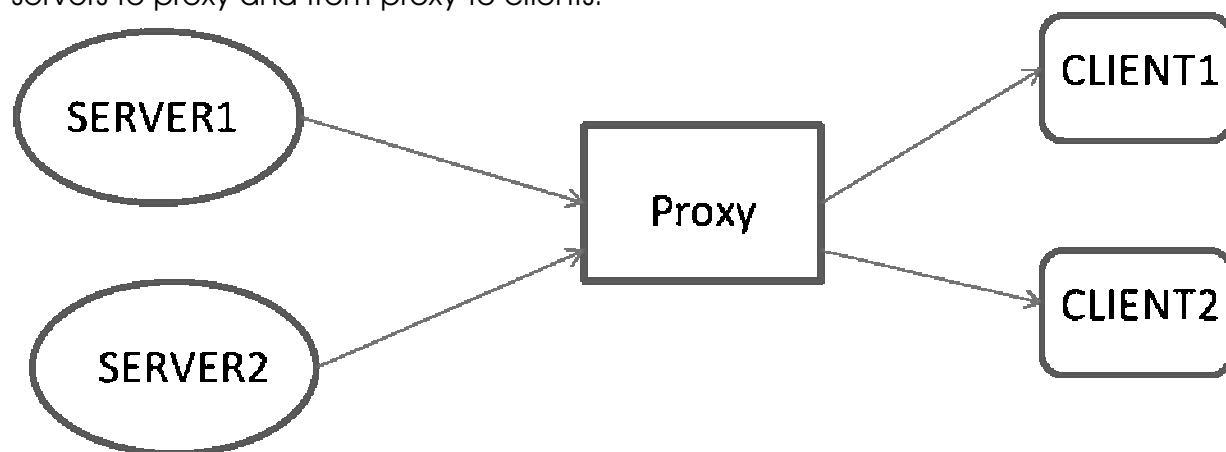


Figure 1: General Distributed Architecture for IPTV in MP3

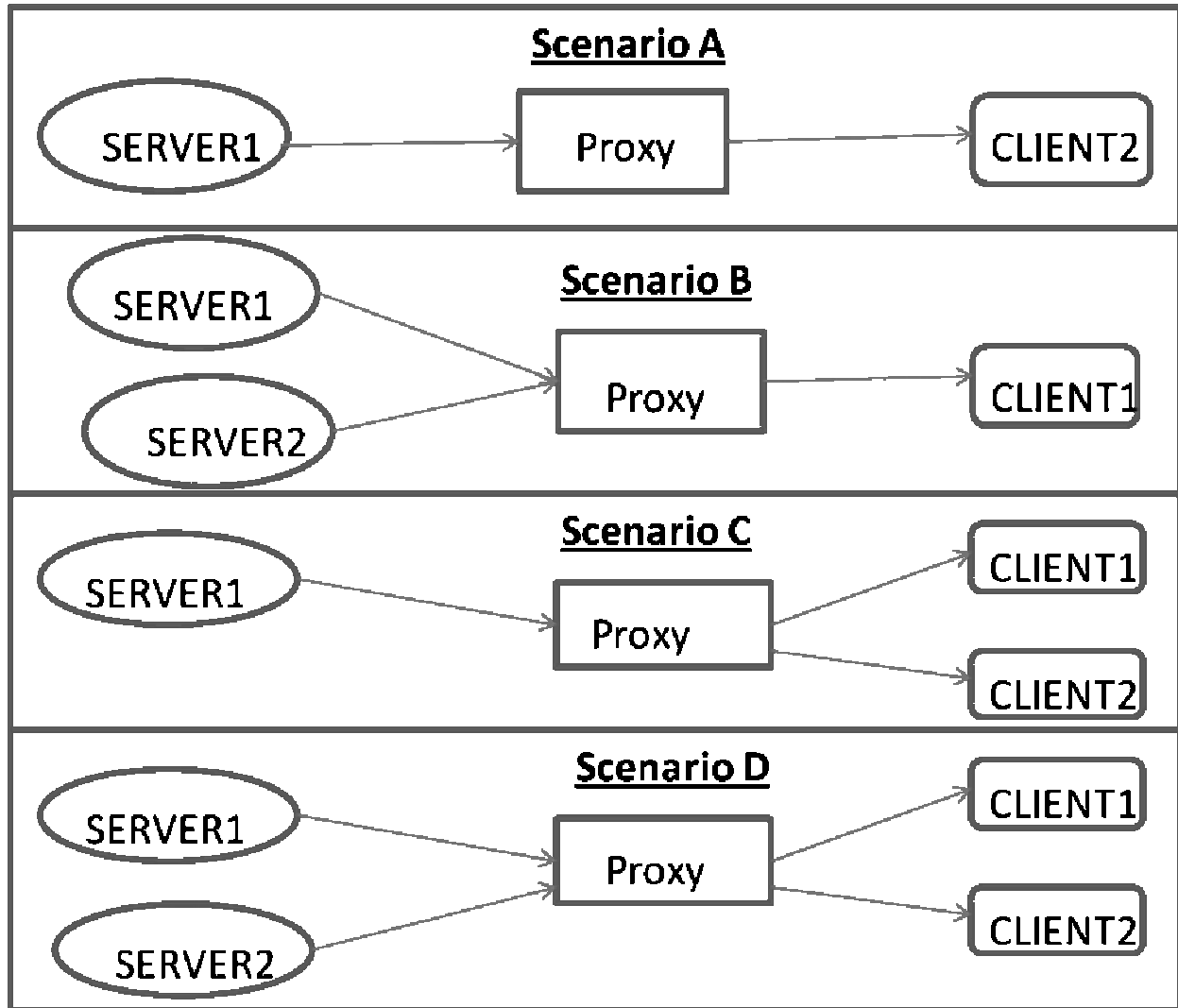


Figure 2: Different Scenarios that will be tested

## Assignment Description

You will send audio and video streams over separate connections from the server to the proxy and from the proxy to the client. The transmission of audio and video can be done over UDP/IP between servers and proxy and between proxy and clients. It is strongly recommended that you carefully consider the design of your server and client as a multi-threaded architecture with possibly the following threads on the server side: (1) read-audio (from file), (2) write-audio (to network), (3) read-video (from file), and (4) write-video (to network). The client side might have the following threads: (1) read-audio (from network), (2) play-audio (to speakers), (3) read-video (from network), and (4) play-video (to display). It is also strongly recommended that you carefully consider the design of your proxy as a multi-threaded architecture with possible threads such as: (1) read-video (from server1 network port), (2) read-video (from server 2 network port), (3) read-audio (from server1), (4) read-video (from server2), (5) write-video (to client1

network port), (6) write-video (to client2 network port), (7) write-audio (to client1), (8) write-audio(to client2).

The communication protocol between the server, the proxy, and the client will have three phases: (1) establishment/negotiation phase, (2) transmission phase, and (3) tear down phase.

### 1. *Establishment/Negotiation Phase*

**Proxy-Client:** During this phase, you will implement a receiver-driven (i.e., client-driven) control negotiation protocol over the TCP/IP connection between the proxy and the clients to exchange needed negotiation Quality of Service (QoS) parameters and other parameters for setup of end-to-end connections between the proxy and clients. The considered negotiation QoS parameters are desired video frame rate and any other parameter (e.g., file name, time duration) as you see appropriate. All these parameters are sent over the control TCP/IP negotiation connection. The proxy will need to compare the available video frame rate (received from the server) with the desired video frame rate (received from the client).

The negotiation control protocol could have the following steps:

- a. Setup a control TCP/IP connection between proxy and clients.
- b. Each client specifies the following to the proxy: movie/file name, video frame rate, and time duration. Each proxy then matches the available QoS with the client's requested QoS.
- c. Proxy checks resources and setup QoS parameters with resources if bandwidth admission control is positive, otherwise renegotiate requested QoS.
- d. Send feedback to each client over TCP/IP to acknowledge the session initiation.

**Server-Proxy:** During this phase, you will implement a control negotiation protocol over the TCP/IP connection between the server to the proxy to exchange needed negotiation Quality of Service (QoS) parameters and other parameters for setup of end-to-end connections between the server and the proxy. This negotiation occurs in response to a request from the client to the proxy. The considered negotiation QoS parameters are recorded video frame rate and any other parameters (e.g., file name, time duration) as you see appropriate. All these parameters are sent over the control TCP/IP negotiation connection.

The server, proxy, and client will need appropriate **rate control** during the transmission to avoid congestion in the network, buffer overflow at the receiver side, and other undesirable effects.

The negotiation control protocol could have the following steps:

- a. Setup a control TCP/IP connection between the servers and the proxy.
- b. Proxy specifies the movie/file name and time duration, to each server and each server returns recorded QoS (e.g., frame rates) of the specified movie.
- c. Proxy needs to check resources and setup QoS parameters at the proxy if enough resources are available (i.e., **proxy must do bandwidth admission control at the application layer to see if it can accept one, two, or more servers**).
- d. Send feedback to each server over TCP/IP to acknowledge the session initiation.

## 2. *Transmission Phase*

During the transmission phase, you may use the following transmission protocols for audio/video data (no synchronization is required at this point between audio and video) between servers and proxy and between proxy and clients.

- a. Open new transmission UDP/IP sockets for audio and video from the servers to the proxy.
- b. Open new transmission UDP/IP sockets for audio and video from the proxy to clients.
- c. Setup the rate control on the servers, proxy, and clients sides.
- d. Start streaming audio and video from servers to proxy.
  - i. Start your streaming protocol by pre-fetching 5 seconds of audio/video data at the proxy and 1 second of audio/video data at the client. After the pre-fetching period, start to play the data on the client side.
  - ii. **At the proxy, you must cache 5 seconds of audio/video for each stream combination (i.e., cache audio1/video1 and audio2/video2) using circular buffers. Unlike client buffering in MP2, a circular buffer is required at the proxy for MP3. In Scenario C, assume that the two clients will start their sessions within 5 seconds of each other (i.e., same buffer should be used for both sessions).**
  - iii. After the pre-fetching step, stream your audio and video chunks according to the rate control algorithm on a per M-JPEG video frame or per audio sample basis from each server via proxy to each client.

## 3. *Tear Down Phase*

Close the connections of the control protocols and the streaming protocols.

**Logging and Monitoring:** In addition to control and data streaming protocols, this machine problem has the additional requirement to implement logging and monitoring at the servers, proxies, and clients. Logging will be critical to the demonstration of the components of this machine problem. Log format suggestions will be given during the

MP3 help session and posted on the course website in the help session slides. The following types of events should be written to a local log file when they occur.

1. *Server Log*: This log should contain a summary of all the control messages sent and received. All control messages should contain the message type, sender IP address and port, receiver IP address and port, timestamp, and any other relevant information (e.g., QoS parameters).
2. *Proxy Log*: This log should contain a summary of all the control messages sent and received similar to the server log. This log should also record a summary of any data packets received out-of-order from the server or not received at all. The log should record an event summary whenever the proxy's buffer runs out of data or overflows. These events should have timestamps.
3. *Client Log*: This log should summarize all of the control messages and events similar to the proxy.

## Runtime Execution Commands:

1. Each Multimedia Server (sender side):

```
iptv SERVER port
```

After executing this command, the server will listen for control negotiation messages from the client to initiate an audio or video streaming session. *Note: you will need to start two servers on different machines during your demo.*

2. Multimedia Proxy

```
iptv PROXY port serveraddr1 sport1 serveraddr2 sport2
```

e.g., `iptv PROXY 9999 csil-core10 9998 csil-core11 9997`

The proxy will connect to 'serveraddr1' on 'sport1' and 'serveraddr2' on 'sport2'. After connecting to the servers, the proxy will listen to its local 'port' for connections from clients. *Note: the proxy will act as a client to the servers and a server to the clients.*

3. Multimedia Client (receiver side):

```
iptv CLIENT proxyaddr pport filename rate duration (filename2 duration2)
```

e.g., `iptv CLIENT csil-core10 9505 a.mjpg 25 10 b.au 10`

After executing this command, the client will start the control negotiation phase by connecting to the proxy at the specified address 'proxyaddr' and port 'pport'. The client will receive the named file(s) at the specified rate(s) for the specified duration(s). The received stream should be played out on the players that you implemented in Assignment 1 (or re-implemented for Assignment 2).

## Comments on Transport Streaming Protocol

The transport streaming protocol for both media must satisfy the following characteristics:

- Streaming Mode - this means that when you get a frame from the disk, the server or proxy will immediately send it over the network to the receiver (i.e., proxy or client, respectively).
- Buffer Management at the Proxy and Client - you will implement a circular buffer at the proxy, which will allow you to balance for network jitter. For the client only, you can re-use your buffering mechanism from MP2. The circular buffer management must be implemented in the read-from-network task. At the proxy, the buffer cache should be shared between the read-from-network and write-to-network tasks as proxy relays video and audio from server to client. **For Scenario C, you can assume that the clients will be started within five seconds of each other (i.e., same buffer should be used to serve both clients).** You will implement the maxbuf scheme, which means that you will pre-fetch several frames at the beginning of the transmission phase (1 second at the client and 5 seconds at the proxy). If your video frame rate is 10 fps, then you will always buffer a maximum of 10 frames at the client and 50 frames at the proxy.
- Adaptive Algorithm for Video Protocol - you will implement a frame-dropping and feedback-based algorithm at the client side. It means that if your receiving buffer for video is empty (starvation happens at the client side) and the next video frame does not arrive on time, then once it arrives, you drop it and get the next frame (sometimes frames arrive in a train manner and to pick up on timing, you drop the late first frame in the train sequence and display the next frames). Be aware that if you drop a frame in the networking task, then the display task needs to be skipped as well. If the dropping occurs five times or more in a consecutive manner, then it means that the network is too congested and you should decrease the frame rate of video, i.e., you need to create feedback at the receiver (client) side and send it to the proxy side to decrease the frame rate of the video stream at the proxy side. **Keep in mind that the server sides do not decrease the rate at which they are sending. The decrease of the rate should only happen at the proxy side (i.e., rate adaptation will only occur between the client and proxy).** Also, keep in mind that if you renegotiate the frame rate, you need to change the rate control at both the client and proxy. If you receive more frames at the client side than you can handle (i.e., overflow happens at the client side), it means that the client is slower than the proxy and network is not congested, then you should send feedback to the proxy and ask for lower frame rate. Again, don't forget to adjust the rate-control at both sides. *Note: we will only work with adaptation that will decrease the video frame rate.* If you decrease the frame rate, you should stay at that decreased rate.

## Delivery

Each group delivers:

- Source code of your iptv program in the your group directory. The source code evaluation will be based on how well your code is documented. If you use some code that you found on the Internet (you must understand the code you found and include in your code, not just blindly copy the code !!!) or in a local system directories, then document it. It is very important that you give credit to people who developed the previous code. Your own code should include the following information at the beginning of each file (Note: your code will be used next Fall by another class so it is important that you give yourself credit.)
- Each major source file should include
  - File Name: Name of the File
  - Description: Short description what the file includes (general description, what kind of functions are embedded in the file).
  - Version: version of your code. You start with version 0 and as you improve the code, at some point you increase the version.
  - Programmer's Name: your name(s) who developed the code
  - Company/University Name: you put the name of the course, department and university you implemented the code for;
  - Date
- Each function in your source code file should have a header with information:
  - Function Name: Name of the Function
  - Description: Short description what the function does.
  - Arguments: Specification of each input argument parameter entering the function and its description.
  - Results: Specification of returning parameters exiting the function and their description.
  - Comments: some special system issues connected with this function
- Group representative(s) come at the scheduled time (we will have a sign-up sheet) between 5 and 7pm on Friday, April 4th and shows a demo of the required programs in 0216 Siebel Center.

## Evaluation of the Assignment (100 Points)

- **NEGOTIATION PROTOCOL (20 points):** Show that you can negotiate different video frame rates with different client QoS specifications and different server QoS capabilities between the two servers and the proxy and between the proxy and two clients. This part will also evaluate the proxy design, especially the admission control and capabilities to negotiate two different video frame rates for two clients via proxy and one server – demonstration 15 points, correct answers of questions to this part 5 points
- **AUDIO/VIDEO STREAMING WITHOUT SYNCHRONIZATION (60 points):**
  - (10 Points – Scenario A) Start one server, one proxy, and one client. Stream audio1/video1 from the server via proxy to the client with different

video frame rates and durations. Demonstrate also adaptive behavior of the protocol - demonstration 5 points, correct answers to questions about this part 5 points

- (10 Points – Scenario B) Start two servers, one proxy, and one client. Start audio1/video1 movie on SERVER1, start audio2/video2 on SERVER2, and stream them via proxy to one CLIENT1. – demonstration 5 points and answers 5 points
- (15 Points – Scenario C) Start one server, one proxy, and two clients. Start audio1/video1 movie on SERVER1 and stream this movie via proxy to clients CLIENT1 and CLIENT2. Allow that CLIENT1 plays the video with different rate than CLIENT2. Demonstration 5 points, and questions 5 points.
- (15 Points – Scenario D) Start two servers, one proxy, and two clients. Start audio1/video1 movie on SERVER1, and stream to CLIENT1 via proxy, start audio2/video2 and steam to CLIENT2 via proxy.
- (10 Points) Implement monitoring and logging tool to log the specified messages and events. Demonstration of the logging tool – 10 Points and answers to questions 5 points.
- **DOCUMENTATION (20 points):** Each group should include a short PDF file (2-4 pages) that gives a brief overview of the components from your program followed by a description detailing how these software components implement the protocols and functionality specified for this assignment. Email this document to the TA [wconner@uiuc.edu](mailto:wconner@uiuc.edu), and also store it in your group directory. In particular, your documentation should describe your design and implementation of the following:
  - Client, proxy, and server threads
  - Negotiation protocol – message sequence, QoS parameters considered, algorithm used by server to determine its available resources, **your control protocols for server-proxy and proxy-client should be well-documented**
  - Streaming protocol – audio/video data transmission from server to proxy and from proxy to client at the appropriate rate
  - Session initiation and termination – brief discussion of how streaming is started and stopped after negotiation
  - Rate adaptation – prevention of starvation and overflow at the client
  - Buffer management – algorithms and data structures used for buffering and playback at the proxy and client, **particular emphasis will be placed on the design/implementation of your circular buffer at the proxy and a description of how the same proxy cache can serve two clients in Scenario C**

The demonstration of the whole assignment for one group should take 30 minutes.