

## **Assignment 2**

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

**Posted: February 08, 2008**

**Last Updated: February 15, 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 second assignment for your Internet Television (IPTV) project where you will experiment with two streaming protocols: (1) one streaming protocol for audio and (2) one streaming protocol for video. The goal will be to stream audio data and video data from the multimedia server to a client, which will play the audio and video separately. You should assume that the audio and video data are stored in separate files on the multimedia server. Each media type will be separately streamed in chunks to a viewing user. You should reuse as much code as possible from Assignment 1 in terms of reading from the audio and video files and playing audio and video on the client sides. However, you are also free to re-implement portions of your audio and video players from Assignment 1 if it will help your overall IPTV design in this course.

#### **Assignment Description**

You will send audio and video over separate connections from the server to the client directly. The transmission of audio and video can be done over UDP/IP. Streaming audio over TCP will also be allowed as long as you justify your design decision. However, streaming video over TCP is not allowed (i.e., UDP must be used for video). 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).

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

## 1. Negotiation Phase

During this phase, you will implement a control negotiation protocol over the TCP/IP connection from the client to the server to exchange needed negotiation Quality of Service (QoS) parameters and other parameters for setup of end-to-end connections between the client and the server. The considered negotiation QoS parameters are requested video frame rate and any other parameters as you see appropriate. All these parameters are sent over the control TCP/IP negotiation connection.

The server and the 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 should have the following steps:

- a. Setup a control TCP/IP connection between the client and the server.
- b. Client specifies desired QoS (e.g., frame rates) and other parameters to the server.
- c. Check resources and setup QoS parameters at the server if enough resources are available.
- d. Server sends resource availability information in terms of QoS parameters (e.g., available frame rate) over the TCP/IP connection to the client.
- e. Check resources and setup all parameters at the client to prepare for the reception of audio/video data if resources available.
- f. Send feedback to the 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):

- a. Open new transmission UDP/IP sockets for audio and video from the server to the client.
- b. Setup the rate control on the sender and the receiver side.
- c. Start streaming audio and video from server to client.
  - i. Start your streaming protocol by pre-fetching 0.5 to 1 second of audio and video data. After the pre-fetching period, start to play the data on the client side.
  - ii. 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.

### 3. Tear Down Phase

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

### 4. Audio retransmissions (optional extra credit)

Implement a negative acknowledgement retransmission mechanism for audio for the case when audio streaming is done over UDP/IP. Since the human ear is very sensitive, audio should be handled carefully. The client should check if any audio packet/sequence number is lost. If a packet loss is detected, the client sends feedback to the server identifying which packet is missing. The server should retransmit any audio packet for which it receives a negative acknowledgement. *Note: you don't have to implement a retransmission scheme for video, i.e., if a video frame is lost, we will live with it.*

### 5. Fast forward and rewind (optional extra credit)

Implement fast forward and rewind for video streaming from the server to the client similar to MP1, but over the network.

## Runtime Execution Commands:

#### 1. 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.

#### 2. Multimedia Client (receiver side):

```
iptv CLIENT address port filename rate duration (filename2 rate2 duration2)
```

e.g., `iptv CLIENT csil-core10 9505 video.mjpg 25 10`

After executing this command, the client will start the control negotiation phase by connecting to the server at the specified address and port. 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, you will immediately send it over the network to the receiver.
- Buffer Management at the Receiver side - you will implement a circular buffer (or other appropriate data structure) at the receiver side, which will allow you to balance for network jitter. The circular buffer management should be implemented in the read-from-network task. The buffer should be shared between the read-from-network and playback tasks. You will implement the maxbuf scheme, which means that you will pre-fetch several frames at the beginning of the transmission phase (up to 0.5-1 second). I would suggest that you consider a maximum buffer size corresponding to 500 ms. This means that if your video frame rate is 10 fps, then you will always buffer a maximum of 5 frames. However, other buffer sizes may be used as long as you can justify your design decision.
- Adaptive Algorithm for Video Protocol - you will implement a frame-dropping and feedback-based algorithm. 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 sender (server) side to decrease the frame rate of the video stream at the sender side. Keep in mind that if you renegotiate the frame rate, you need to change the rate control at both, client and server. If you receive more frames at the receiver side (client) than you can handle (overflow happens at the client side), it means that the client is slower than the server and network is not congested, then you should send feedback to the server (sender) 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, February 22nd and shows a demo of the required programs in 0216 Siebel Center.

## Evaluation of the Assignment (100 Points)

- **NEGOTIATION PROTOCOL (10 points):** Show that you can negotiate different QoS parameters, such as the video frame rate, with different client QoS specifications and different server QoS capabilities – demonstration 5 points, correct answers of questions to this part 5 points
- **VIDEO STREAMING ONLY (25 points):** Show iptv streaming protocol for video only from sender to the receiver – demonstration 20 points, correct answers of questions to this part 5 points
- **AUDIO STREAMING ONLY (25 points):** Show iptv streaming protocol for audio only from sender to the receiver – demonstration 20 points, correct answers of questions to this part 5 points
- **AUDIO/VIDEO STREAMING WITHOUT SYNCHRONIZATION (20 points):** Run iptv with both audio and video (does not need to be synchronized but both streams should run from the same program) – demonstration 15 points, correct answers to questions about this part 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) by 11:59pm on February 22nd, and also store it in your group directory. Please remember to include your group number and the names of all group members in your document. In particular, your documentation should describe your design and implementation of the following:

- Client and server threads
- Negotiation protocol – message sequence, QoS parameters considered, algorithm used by server to determine its available resources
- Streaming protocol – audio/video data transmission from server 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 (e.g., circular buffer) used for buffering media data at the client for playback
- **AUDIO RETRANSMISSION (OPTIONAL, 10 bonus points):** Implement a mechanism that causes the audio server to drop audio packets with a specified probability. Show audio streaming from such a server both with and without negative acknowledgement retransmissions from the client for comparison – demonstration 5 points, correct answers of questions to this part 5 points
- **FAST FORWARD (OPTIONAL, 10 bonus points):** Similar to MP1, implement fast forward for video streaming from the server to the client – demonstration 5 points, correct answers of questions to this part 5 points
- **REWIND (OPTIONAL, 10 bonus points):** Similar to MP1, implement rewind for video streaming from the server to the client – demonstration 5 points, correct answers of questions to this part 5 points

The demonstration of the whole assignment for one group should take no more than 20 minutes.