Machine Problem 3 (Option 2): Multi-Party Video Conferencing

CS414 Spring 2011: Multimedia Systems
Instructor: Klara Nahrstedt

Posted: Apr 4, 2011
Due: 11:59pm Apr 29, 2011

Introduction

In this final MP, you will use all the components built in previous MPs to build a three-party video conferencing system. You will be able to claim that you have some experience with the "large-scale" system after you finish this MP. Trust me! It will be more complicated than you think. You are required to work on the Linux platform for this machine problem. You can use the Linux workstations in the EWS lab room SC216 and SC220. You need all three Logitech Webcams your group have borrowed because each party needs one. You also need your own headphone for the audio playback.

Unfortunately, you will not be eligible for the final competition if you take this option. Do you want to switch to the mobile project now?

Problem Description:

Implement ONE Linux program named as *videoconf*. Run *videoconf* on three difference machines. Each *videoconf* should be able to send the captured video/audio data to two other peers, and play the audio/video streams from two other peers. Here are the detailed feature requirements:

Required Features (80 points)

1. *Fast Connect (20 points)*: each *videoconf* should connect to two other peers as fast as possible after the program starts. Each *videoconf* should print how long it takes to finally connect to other two peers.

2. *A/V Streaming (10 points)*: after all three peers are connected to each other, each *videoconf* should send its own audio/video streams to two other peers and receive the audio/video streams from two other peers. The recommended video resolution is 320×240 and the frame rate should be no less than 10 frames per second.

3. *Audio Mix (10 points)*: each *videoconf* should receive two audio streams from two different peers. Mix the audio streams and play them together.

4. *Video Window (10 points)*: each *videoconf* should open three video windows: one for its own captured video stream and two for the remote video streams from two other peers. Add the title to each video window to mark.

5. *A/V Sync (10 points)*: all audio/video played by your program should be synchronized.
6. *Bandwidth (10 points)*: *videoconf* should calculate the incoming and outgoing network bandwidth and print the stats every second.
7. *Robustness (10 points)*: your programs are expected to run without crash for up to one hour.

Optional Features (70 points)

8. *GUI (5 points)*: design any graphic user interface for *videoconf*.
9. *Name Server(5 points)*: set up a name server for the video conferencing peers to register and query IP addresses based on user names.
10. *Remote Control (10 points)*: each *videoconf* can remotely pan/tilt the webcam controlled by other peers.
11. *More Party (10 points)*: can you do four-party?
12. *Proxy (15 points)*: instead of sending audio/video streams to other peers directly, send them to a proxy that will forward for you. Compare the difference of bandwidth usage before and after using the proxy.
13. *NAT Penetration(15 points)*: find a solution to penetrate NAT. You can use existing protocols, tools, or set up your own server to help.
14. *Your Feature (10 points)*: design a feature of *videoconf* by yourself and get 10 extra points. The feature must be related to the video conferencing. Whether your feature deserves 10 points will be determined based on its novelty and workload. For example, you can do file sharing or short message during video conferencing.

Examples

Here is an example of what you are expected to implement and some explanations about the features.

For the required features, you should run *videoconf* on three different machines. You are strongly recommended to use the config file to input all parameters and the name server to help set up initial connections. Once the three-way connections are established, each *videoconf* is expected open three titled video window to play the local captured video stream and two other remote video streams. An alternative is to open only one video window and display either the local video or the remote video in the window. However, in this case, you have to provide the user interface to switch to different video streams. The remote audio streams should be mixed, which means you can hear two remote peers talking at the same time. Your local audio stream should not be mixed and played. Each *videoconf* should print the bandwidth stats during the whole video conferencing.

```
[1s] Outgoing Bandwidth: *** bps
[1s] Incoming Bandwidth: *** bps
[2s] Outgoing Bandwidth: *** bps
[2s] Incoming Bandwidth: *** bps
......
```

For the optional features, feature 9 has the same requirements as the same feature in MP2. You need to write a third program *name_server*, which should start before any *videoconf*. For feature 10, your *videoconf* can remotely pan/tilt the webcam controlled by other two peers*desktop*. For feature 11, run four *videoconf* on four different machines. Ask for borrowing one more Logitech Webcam or you can choose to use a video file to replace webcam as the video source. For feature 12, you should add a network proxy that helps you forward packet to other servers. In this case, you don't have to send two copies of your audio/video streams to two other peers, but send one copy to the proxy. The proxy forwards all streams to the destination. For feature 13, you need to run at least one *videoconf* on the external server which is in a subnet using the private IP address (e.g., 192.168.*.*). If you have problem finding such a server but still want to implement this feature, contact TA for help.

Submission and Evaluation

The submission and evaluation of this final MP is different from previous two MPs. Although you are not eligible for the competition, you are also expected to present your project face-to-face to the instructor and TA on the afternoon of Apr 29. More details about the time schedule of the final presentation and the evaluation criteria will be released one week before the due date.

All groups are required to pack your source code and documentation files in one zip file and submit through Compass by midnight Apr 29th. You lose up to 20% points if you don't submit your code or don't write documentations.