CS5248: Systems Support for Continuous Media

Spring 2011

Project Assignment

Due: Report Draft – 4 Apr 2011 (11:59:59 pm)
Due: Code – 10 Apr 2011 (11:59:59 pm)
Due: Presentation & Final Report – 11 Apr 2011 (in class)

THIS IS AN INDIVIDUAL or TEAM-OF-2 HOMEWORK

REMEMBER TO CHECK FOR ANNOUNCEMENTS ON THE CLASS WEB SITE:
http://www.comp.nus.edu.sg/~cs5248.html

Homework Description

In this project, your task is to write a packetization routine that takes in an MP3 audio file and produces RTP packets from the file, following the standards and recommendations in RFC 2250 and RFC 5219.

Your implementation will be added to the Yima Personal Edition (PE v1.0) streaming media server code. You will be given the source code of Yima PE. You then need to implement your packetization routines within this framework. In addition to RFC 5219 (“A More Loss-Tolerant RTP Payload Format for MP3 Audio”) you will need to implement an earlier RFC 2250 (“RTP Payload Format for MPEG1/MPEG2 Video”) for comparison purposes. Your code should also be able to simulate packet losses so that the improved streaming quality with RFC 5219 can be measured and demonstrated.

You will be given the source code of Yima PE v1.0. It consists of several components. Instructions on how to install, compile and run Yima PE can be found in a PDF file here: http://www.comp.nus.edu.sg/~cs5248/l05/YimaPE_doc.pdf. More information will be given during the lecture on 7 February and can be found in the slides for that lecture. The Yima PE MP3 Player Client runs under Windows and we will give you a Microsoft Visual C/C++ 2005 project containing the source code.

Specifically, your implementation in this project consists of the following sub-tasks.

1) **Task 1: Creating Media Blocks for the Yima Server**
Your fundamental task is to produce media blocks that contain RTP packets such that the Yima server can load them and stream them to a client. You need to packetize audio data into the RTP format adhering to the standards of the RFC 5219 and the RFC 2550, respectively. Note that in order to partition the whole audio data file into the units suitable for packetizing, you will need to know the data structure of MPEG1/MPEG2 Audio Layer III. Yima uses a separate packetizing utility call ***yimasplit***. The code of this utility is in the directory *$HOME/YimaPE_v1.0/Splitter*.

2) **Task 2: Modifying the Network Module of Yima MP3 Player**
The second task is to modify Yima MP3 Player (client) so that it can successfully receive and demultiplex the RTP packets streamed from the Yima server. You do not need to worry about the decoding part – it is already implemented. Your goal is to modify the network module such that it performs the reverse process of what is achieved in Task 1, before feeding the received audio data into decoder.

3) **Task 3: Design Your Experiments**
The final task is to design your experiments to compare and measure the two packetization implementations under lossy network conditions. For this, first you need to simulate packet losses (i.e., using both a random and a Gilbert Model). Packet losses can be simulated on the server-side by selectively dropping outgoing packets (can be done on the client-side as well). In addition, you should prepare your test audio data. Moreover, you are expected to come up with reasonable criteria to quantify the performance of the two specifications. One possible criterion is the perceived audio quality, which can be measured with Perceptual Evaluation of Audio Quality (PEAQ) .

In addition to the implementation, you will need to write a research paper to describe your implementation, experimental setup, experimental results and your analysis.

Reference and Software Information

All the students talking this module are given an account on **cervino.ddns.comp.nus.edu.sg**. The machine is running Red Hat Enterprise Edition Linux RHEL 4.6 (http://www.redhat.com/rhel/). This is a 32-bit version of Linux. Note that Yima PE may not compile and run on 64-bit Linux. The necessary X11 libraries for Yima PE are already installed on this machine. You will need to use `ssh` to connect to the machine.

To analyze the RTSP protocol several network sniffing tools are useful as discussed in previous lectures.

The default RTSP port is set to 55,554 on both the server and the client. Ports above 1,024 can be used without root (i.e., administrator) privileges. Note that the port number needs to match between the client and the server. Note further that only **one** program can use a certain port number at a time. Therefore, the different teams in CS5248 have to select different port numbers – otherwise testing will become difficult.

The RTSP port number in the **server** can be changed in the following file:
YimaPE_v1.0/Server/RTSPS/rtsps.h

```
#define RTSPSPORT       55554          // well-known RTSPS TCP port; RTSPS
```

The RTSP port number in the **client** can be changed in the following file: protocol.h

```
#define RTSP_DEFAULT_PORT          55554
```

Each team has to choose a port in the range 50,010-50,014 or 50,020-50,024. Also note that most UDP ports are blocked on the NUS network. Use the VPN to connect on the client side.

Additional information and links:

- Yima PE documentation: get it from the course web site (http://www.comp.nus.edu.sg/~cs5248/l05/YimaPE_doc.pdf)
NOTE: The documentation also describes a Linux player client, which you don't need.
- Yima PE server source code tarball (YimaPE_v1.0.tgz): get it from the IVLE workbin; You uncompress the file with the command 'tar zxvf YimaPE_v1.0.tgz'.
- Yima PE MP3 player Visual Studio 2005 ZIP file (YimaMP3Player_VS2005.zip): get it from the IVLE workbin
- Slides: get them from the course web site (http://www.comp.nus.edu.sg/~cs5248/l05/05-project.ppt)
- Network Sniffing Tools: MS Network Monitor (http://blogs.technet.com/netmon/), WireShark (http://www.wireshark.org/download.html)

If you have any questions, email either the TA or the professor!

Submission Guidelines

1. Materials which you need to submit

(1) Your modified Yima PE source code. Create a tarball of all your **sources** (please exclude sample videos, object files, etc.).

(2) A detailed README.txt file, that should include the following information:
a) Your name(s), matriculation number(s), and your username(s) on the host cervino.
b) A brief description of each of the modified/new files and how it works.

Please create a tarball that collects all your modified files into a package with your matriculation number. Then compress it (with gzip) before you submit the file. **Make sure that your code compiles without ANY errors!**

(3) A project report. Your write-up should be similar to a standard research paper (see details below). Please submit a PDF file.

(4) Submit to your report and compressed package into the course module's IVLE workbin to the folder called 'Student Submission' by 23:59 on 04/04/10 (report draft), 23:59 on 10/04/10 (source code), and 23:59 on 11/04/10 (final report).

(5) Late submission policy for report.
- late within 24 hours: 30% reduction in marks;
- late within 3 days: 50% reduction in marks;
- late within a week: 70% reduction in marks;
- after one week: zero mark.

Note that the project demonstrations will be held on 11 April and cannot be done later.

Grading Policy (35 full marks)

1. **Project Demonstration (15 marks)**. You will present your project in class on 11 April 2011. You will have approximately 20 minutes to demo your project, explain how it works and show how your RFC 5219 compliant MP3 packetization improves over the standard RFC 2250 packetization. You should be able to demonstrate the following:
   - Correct excution of RFC 5219. **(4 marks)**
   - Correct excution of RFC 2250. **(3 marks)**
   - Switch between RFC 5219 and 2250-style packetization by GUI operation or configuration. **(1 marks)**
   - Simulate (i.e., ”turn on”) packet losses in the network. The amount of losses and the type of loss model should be selectable, for example 3% random packet loss. **(3 marks)**
   - Your presentation: Explain your experimental setup, show your experimental results and justify them. **(4 marks)**

   For the demo, make sure you are ready and everything works. We will not have time to debug any issues.

2. **Project Source Code (5 marks)**. I will have a look at your code. Criteria are for example, how well it is documented, how well it is understandable, how robust it is (i.e., can it process all relevant input files).

3. **Project Report (15 marks)**. Your write-up should be similar to a standard survey paper. Please include (at least) the following sections:
   a) Abstract
   b) Introduction and motivation for your project work (your specific design and implementation choices)
   c) Description of your implementation, i.e., details on what you did, any difficulties encountered, any special features that you implemented, etc.
   d) Results, comparison and discussion, i.e., what should we learn from this (and maybe possible improvements)
   e) Conclusions
   f) References

   Most of the marks will be given for sections b), c) and d).

Additional Notes

No plagiarism is tolerated. While you can talk to other teams and you are encouraged to used the IVLE Forum, you cannot copy source code and/or text in your report. If you are not sure whether something is permissible, either talk to the instructor or consult the university policy at http://www.scholars.nus.edu.sg/uspinfo/academiccode.html.