Iowa State University
Department of Electrical and Computer Engineering
Cpr E 489: Computer Networking and Data Communications
Lab Experiment #5
UDP Sockets Programming

Objective

To use UDP sockets to write a program to implement network impairments (i.e., loss and delay), and study this effect on a video stream.

Pre-Lab

Write the code to create a UDP socket and bind it to the IP address of the machine your program is running on, and to a given port number.

Lab Expectations

Work through the lab and let the TA know if you have any questions. **Demonstrate your program to the TA after you have completed it.** After the lab, write up a lab report with your partner. Be sure to

- summarize what you learned in a few paragraphs
- include your answers to the two exercises
- include your **well-commented code**
- specify the effort levels of each group member (totaling to 100%)

Your lab report and demonstration are due at the beginning of the next lab.

Problem Description

In this lab experiment you are required to implement a UDP program that will forward UDP packets from a given source to a given destination, while introducing packet loss.

- The program will run on a Linux machine and will accept five parameters: source IP, source port, destination IP, destination port, and loss rate.
- The program will listen on the port number specified by the source port parameter and will accept packets with the source address specified by the source IP parameter.
- The program will decide to forward the received packets based on the specified loss rate. *(Hint: you may implement a loss rate by comparing a random number with the loss rate.)* Packets will be forwarded to the machine and port specified by the destination IP and destination port parameters, respectively.

Procedure

- Write the program in C under Linux.
- All communication is done using the UDP protocol.
- The source port and destination port parameters should use separate UDP port numbers chosen between 1024 and 65535 (see the **Notes** below for hints on how to choose the port numbers).

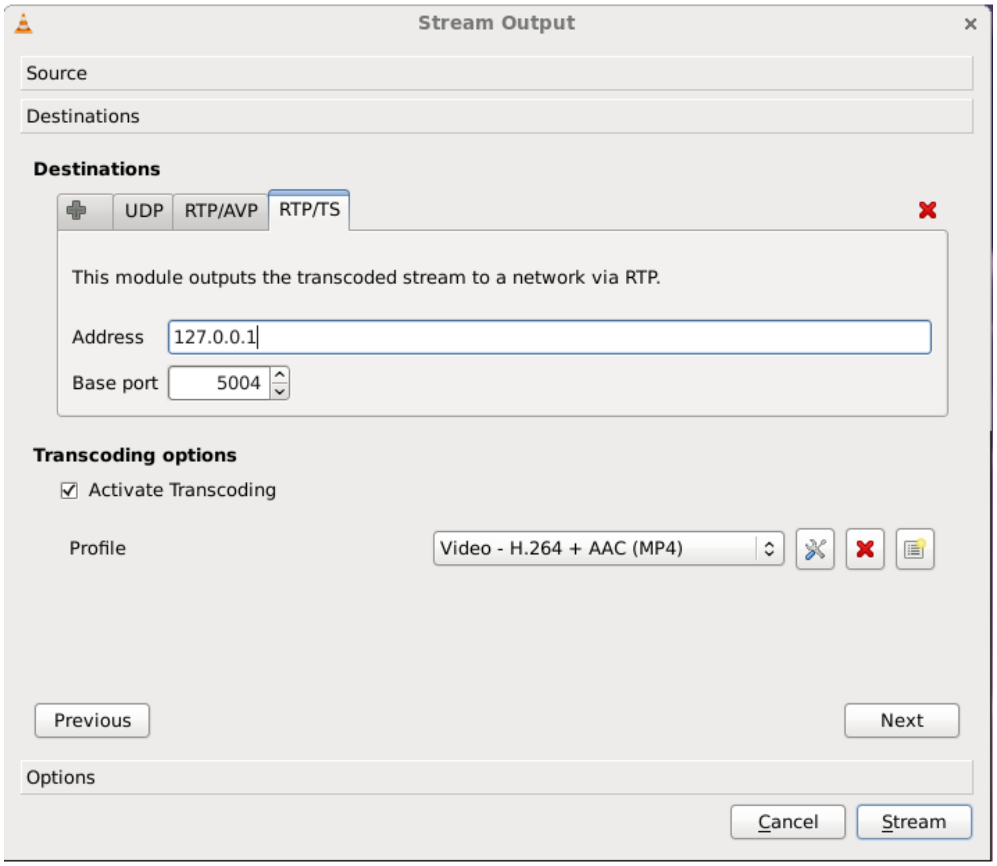
Testing Your Program

In order to test your program, you will be using a media player called “VLC media player.” VLC allows for playing multimedia files of various formats, while providing the ability to stream the file being played to a remote machine specified by an IP address and a port number. Additionally, VLC can display media available through a network stream.

1. Identify three machines on which you will run the source VLC, the destination VLC, and your program. Alternatively, if there are not enough machines available in the lab, you can test everything on your own machine using 127.0.0.1 as the IP address for both the source and the destination IP addresses.

2. To demonstrate and test your program, you will need to forward packets between a source VLC player and a destination VLC player. In other words, your single program will act as a server (to accept UDP packets from the source) and a client (to forward UDP packets to the destination).

3. Start the source VLC player (Applications -> Sound & Video -> VLC media player). From the "Media" menu, choose "Streaming". A dialog box will appear, and click "Add" within the "File" tab to add the multimedia file you want to display, and then click "Stream" in the same tab. A new dialog will open. Verify that the "Source:" field has the file you want to display, and then click "Next". Check the box that says "Display Locally", and from the dropdown menu, configure "RTP/TS" (RTP/MPEG Transport Stream) or "UDP". For each one, select it from the dropdown menu, and click "Add". In the address field, enter the IP address where the program you developed resides and a port number (remember this port number because this is the source port parameter passed to your program). Click "Stream".



This module outputs the transcoded stream to a network via RTP.

Address 127.0.0.1

Base port 5004

☑ Activate Transcoding

Profile Video - H.264 + AAC (MP4)

Figure 1 – The source VLC player Stream output settings dialog box

4. Start a new terminal and enter the command:
```
$ vlc -vvv rtp://@IP_ADDRESS:PORT
```
or
```
$ vlc -vvv udp://@IP_ADDRESS:PORT
```

depending on whether you started streaming as RTP or UDP, where IP_ADDRESS is the destination address passed to your program and PORT is the corresponding port number.

5. Start your program with the following parameters: the source IP is the IP address of the source VLC player, the source port is the port number entered in Step 3 above, the destination IP is the IP address of the destination VLC player, the destination port is the port number entered in Step 4, and the loss rate is any value between 0% and 100%.

6. Test your program with various loss rates and demonstrate your program to the TA.

7. Submit the source code of your program with your lab report to the TA for grading.

Exercises

1) What was the effect of increasing the loss rate on the video quality?
2) To counter the effect of network loss, someone suggested using TCP instead of UDP for multimedia communication. Do you agree? Why? Why not? *(Hint: Answer the question in terms of buffer space, jitter, and retransmission time.)*

Notes

- In order to avoid contentions and confusions, use a server port number that is equal to the rightmost five digits in your student number. For example, if your Student ID number is 123-45-6789, then the port number should be 56789. If this number is less than 1023, greater than 65535, or equal to any reserved port number (see the `/etc/services` file) you may use any random port number.
- If there is any missing information, you may make any reasonable assumptions, but clearly state these assumptions in your solution.
- Please refer to the **"How to"** below for general guidelines on how to write and run your sockets programs.
- If you are having issues launching VLC, use the following command:
  ```
  MALLOC_CHECK_=1 vlc
  ```

How to Write and Run Your Networking Programs

All programs will be written in C, under the Linux environment.

You should follow these steps:

- Edit your program using any of the editors available under UNIX (e.g., pico, vi, emacs, etc.)
- Compile your program using `gcc`.
- For programs using sockets, use the following command to compile:
  ```
  gcc –o file file.c
  ```
  where `file.c` is your code, and file is the required executable file. Note that you can link to other libraries as needed, such as the math library using `–lm`. Also, please note that under other versions of UNIX besides Linux (e.g., Solaris), you need the "`–lsocket`" and "`–lnsl`" options for socket programs. (This shouldn't be an issue with the computers in Coover 2048.)
- You may run your program by typing the full path to your compiled executable:
  ```
  ./file
  ```