Cpr E 489: Computer Networking and Data Communications
Lab Experiment #3 – UDP Socket Programming
(Total Points: 100)

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

- Review sockets programming materials.
- 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. After the lab, write up a lab report. Be sure to:

1) Attend the lab. **(5 points)**
2) Summarize what you learned in a few paragraphs. **(25 points)**
3) Include your answers to the **two exercises** stated in the Exercise section below**. (20 points in total, with 10 points each)**
4) Include your **well-commented** code and **demo** your code to the TA. **(50 points)**
    - See the detailed point breakdown in the "Problem Description" section.

Demonstration Policy

This lab will require you to **demo** your code to the TA. Review the lab manual carefully to ensure you have completed all demonstration requirements. Labs may be demonstrated during:

- **this lab section**
- **the TA office hours**
- **the first hour of the next lab** (everyone will have a chance to demo **once**)

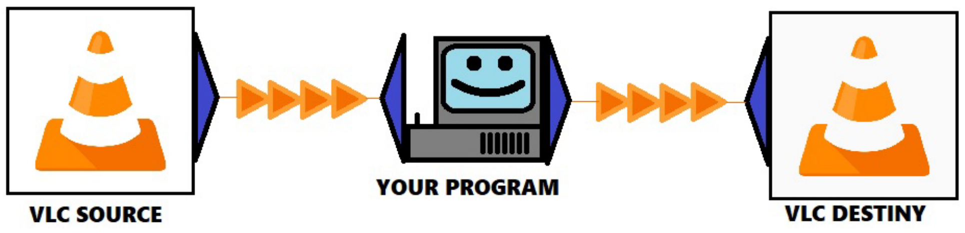
Lab reports are due on the day **before** the next lab.

Note: If a lab is assigned as a partnered lab, the demonstration **must** be performed while all team members are present. Exceptions to this "team rule" may be made for extenuating circumstances and with prior notice.

Problem Description

In this lab experiment you must 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. **(10 points)**
- 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. **(10 points)**
- Packets will be forwarded to the machine and port specified by the destination IP and destination port parameters, respectively. **(20 points)**
- The program will decide to forward each packet received based on the specified loss rate. *(Hint: you may implement a loss rate by comparing a random number with the loss rate.)* **(10 points)**



Procedure

- Write the program in C under Linux.
  - 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.
- Run your program by typing the full path to your compiled executable with its command-line arguments:
  ```
  ./file <SOURCE IP> <SOURCE PORT> <DESTINATION IP> <DESTINATION PORT>
  <LOSS RATE>
  ```
- `<SOURCE IP>` and `<SOURCE PORT>` are where the VLC streams out the video data. How to set up VLC to stream out data is given in Step 3 of “Testing Your Program” section.
- `<DESTINATION IP>` and `<DESTINATION PORT>` are where you want to dispatch the video, which is discussed in Step 4 of the “Testing Your Program” section.
- `<LOSS RATE>` is the number of packets out of 1000 that are dropped by your program.
- All communication is done using UDP.
- The source port and destination port parameters should use separate UDP port numbers chosen between 1024 and 65535.

Testing Your Program

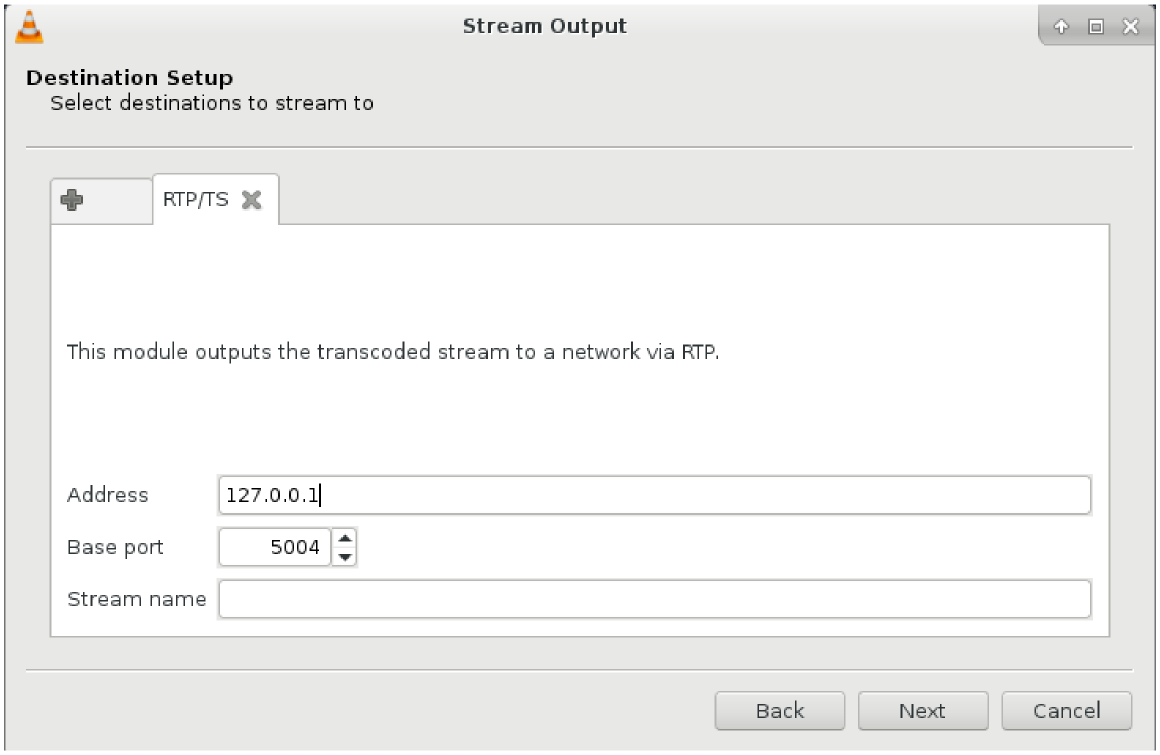
To test your program, you will be using a media player called “**VLC media player.**” VLC media player allows for playing multimedia files of various formats, while providing the ability to stream the file being played to a remote machine specified by an IP address and a port number. Additionally, VLC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

1. Start the source VLC player by typing `vlc -q &` on the command line. From the **“Media”** menu, choose **“Stream.”** 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 **"Next"** twice to confirm and then click **"Stream."**

2. Identify three machines on which you will run the **source VLC**, the **destination VLC**, and **your program**. You may test everything on your own machine using 127.0.0.1 as the IP address for both the source and the destination IP addresses.

3. 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).

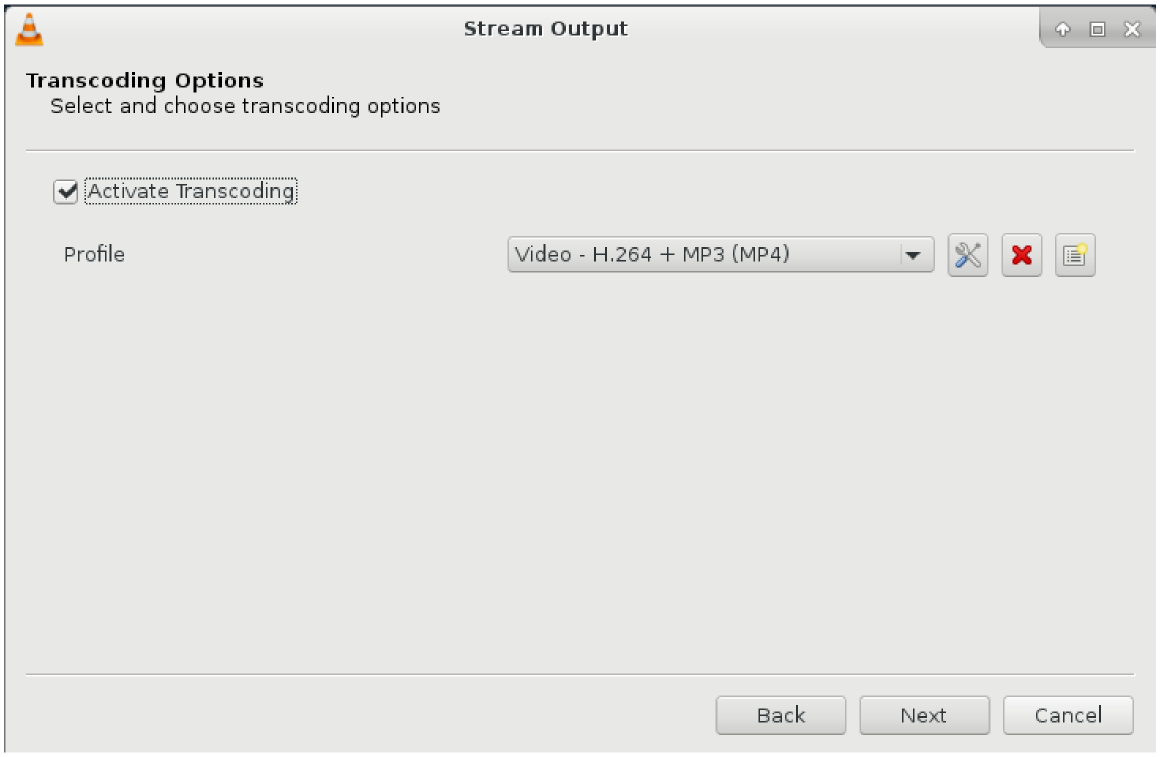


Destination Setup
Select destinations to stream to

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

Address: 127.0.0.1
Base port: 5004
Stream name:

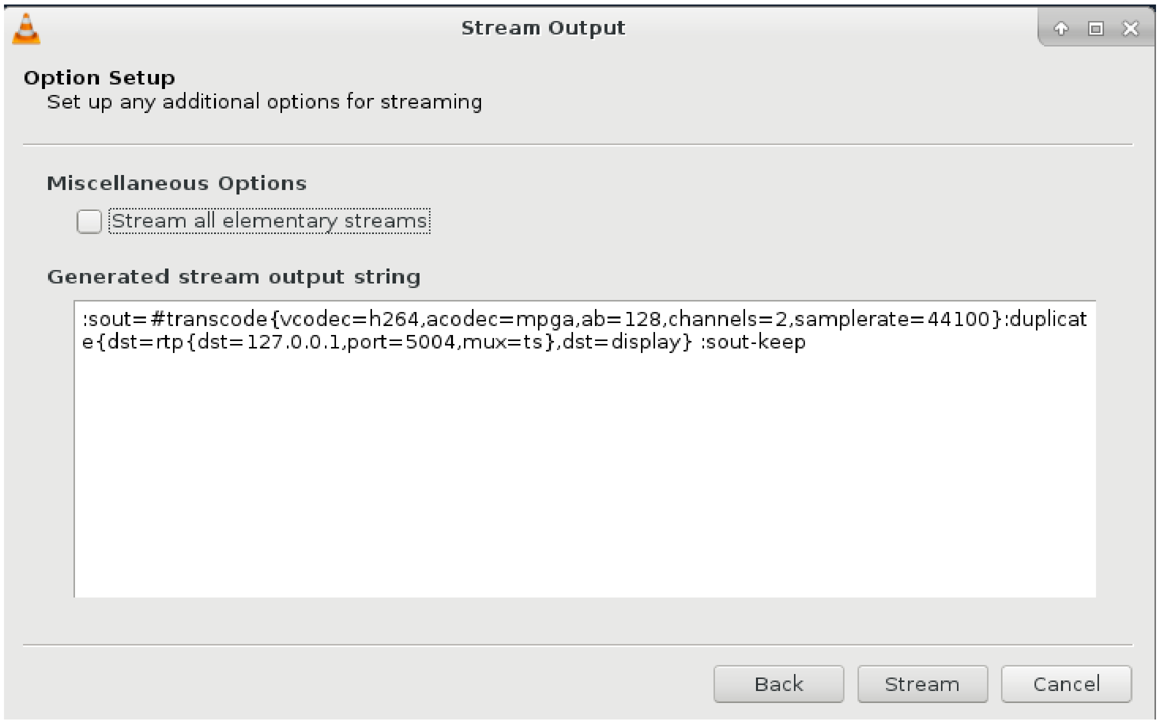
Figure 1 – The source VLC player Stream output destination setup dialog box. In this example, when you set Address to 127.0.0.1 and Port to 5004, VLC streams out to the IP and Port, which you would provide as `<SOURCE IP>` and `<SOURCE PORT>` parameters to the above executable C code.



☑ Activate Transcoding

Profile: Video - H.264 + MP3 (MP4)

Figure 2 – The source VLC player Stream output transcoding options dialog box.



☐ Stream all elementary streams

Generated stream output string

```
:sout=#transcode{vcodec=h264,acodec=mpga,ab=128,channels=2,samplerate=44100}:duplicat
e{dst=rtp{dst=127.0.0.1,port=5004,mux=ts},dst=display} :sout-keep
```

Figure 3 – The source VLC player Stream option setup dialog box.

4. To watch the video data sent by the above program at the destination, you would start a new terminal on the destination computer 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 to listen on.** The IP_ADDRESS and PORT would typically be destination computer's IP and a random port number, which were also given as **`<DESTINATION IP>`** and **`<DESTINATION PORT>`** parameters to the above executable C code.
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 from 0 to 1000.
6. Loss rate of 0 should produce lossless video at the destination, loss rate of 1 should allow 999 out of every 1000 packets to arrive at the destination, loss rate of 2 allows 998 out of every 1000, etc.
7. Test your program with various loss rates.
8. Submit the source code of your program with your lab report to the TA for grading.

In summary, the sequence of the programs or commands is as follows:

- First, you can set up the VLC media source as described in step 3 of the Test procedure. You can consider this as the source computer that streams the video. Whenever you stream this video, it sends video data to the program specified or executed in the above first step. The program relays video data based on the loss rate and forwards the video data to the destination IP and port. The forwarded video data could be displayed on the destination computer by the above second step VLC command.
- Second, you would run the destination VLC command **`$ vlc -vvv udp://@IP_ADDRESS:PORT`.** This should be run on the destination computer.
- Third, you would run the program as **`$ ./file <SOURCE IP> <SOURCE PORT> <DESTINATION IP> <DESTINATION PORT> <LOSS RATE>`.**

Exercises

1) What was the effect of increasing the loss rate on the video quality? **(10 points)**
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.)* **(10 points)**

Notes

- If there is any missing information, you may make any reasonable assumptions, but clearly state these assumptions in your solution.
- Please refer to the "**Procedure**" section above 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
```