Multimedia Communications
@CS.NCTU

Homework 1: audio streaming over TCP/UDP

Outline

- Prerequisite
- Tasks
- Socket Programming
- Traffic Control
- Performance Evaluation
- Grading
- Summary

Prerequisite

- Quickly learn online how to write TCP or UDP socket programming
  - Server and client example with C sockets on Linux
  - Linux Howtos: C/C++ -> Sockets Tutorial
  - C Socket Programming for Linux: Example Code
- Install Linux systems (e.g., Ubuntu) and VM (optional) (e.g., VirtualBox, VMware workstation(free))
- Make sure you have an operable microphone and speaker

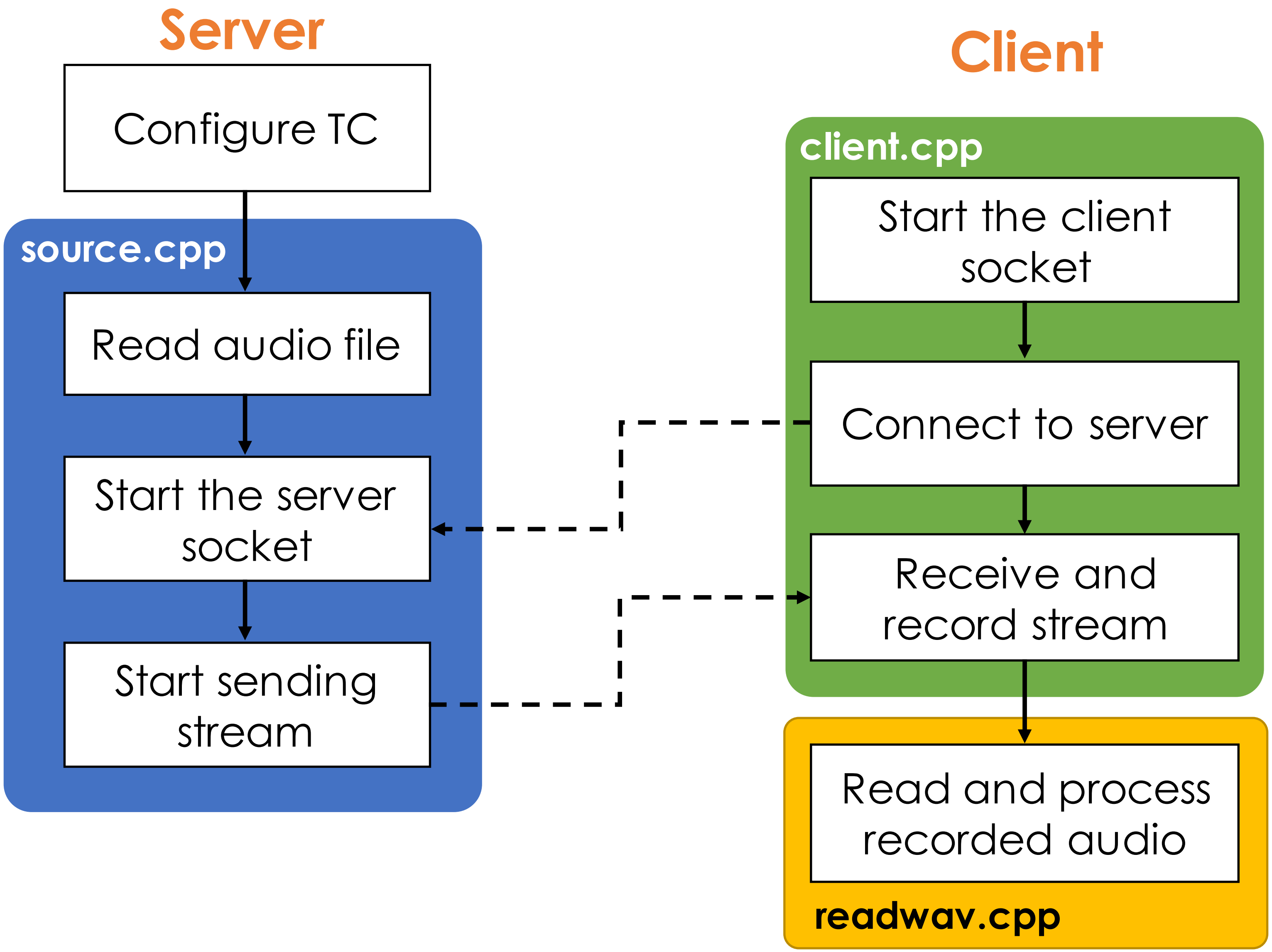
Outline

- Prerequisite
- **Tasks**
- Socket Programming
- Traffic Control
- Performance Evaluation
- Grading
- Summary

Task: Step by Step

- At server side, launch the shell script to configure the `tc` setting for controlling the network condition
- At server side, read the wave file
- At both side, use TCP/UDP to deliver audio signal
- At receiver side, call `aplay` to real-time play the received audio signal
- At receiver side, save and merge the received packets as the output file `ratexx_tcp/udp.wav`
  - For TCP, log the time-stamp of each received packet
  - For UDP, drop the out-of-order packets and pad ‘0’ bits for the lost packets
- Finally, evaluate the quality of the received signal
  - For TCP, calculate the per=byte delay
  - For UDP, calculate the PSNR

Task: Diagram



Outline

- Prerequisite
- Tasks
- **Traffic Control**
- Socket Programming
- Performance Evaluation
- Tasks

Traffic Control

- User-space utility program used to configure the Linux kernel packet scheduler
- Shape traffic by
  - **Shaping**: limited the transmission rate (egress only)
  - **Scheduling**: scheduling the packet to different "class" (egress only)
  - **Policing**: deal with reception traffic
  - **Dropping**: If traffic exceeding a set bandwidth, drop the packet (both ingress and egress)

Traffic Control: How to

- **qdisc**
  - Short for “queueing discipline”
  - It is elementary to understanding traffic control
- **class**
  - Qdisc contains classes
  - A qdisc may prioritize certain kinds of traffic by dequeening from certain classes
- **filter**
  - A filter is used by a classful qdisc to determine in which class a packet will be enqueued.

Traffic Control: How to

- Use SHAPING to deal with transmission of traffic

Hint:

1. Using lo(localhost) network interface for this homework
2. Create one qdisc on lo interface
3. Create a class rule for it
4. Create filter

Reference：

http://lartc.org/lartc.html
https://puremonkey2010.blogspot.tw/2015/01/linux-tc-traffic-control.html
https://www.cyberciti.biz/faq/linux-traffic-shaping-using-tc-to-control-http-traffic/

Traffic Control: Tested Configuration

- Configuration 1:
  - Bandwidth: htb rate 256kbps ceil 300kbps

- Configuration 2:
  - Bandwidth: htb rate 196kbps ceil 200kbps

- Configuration 3:
  - Bandwidth: htb rate 128kbps ceil 150kbps

Outline

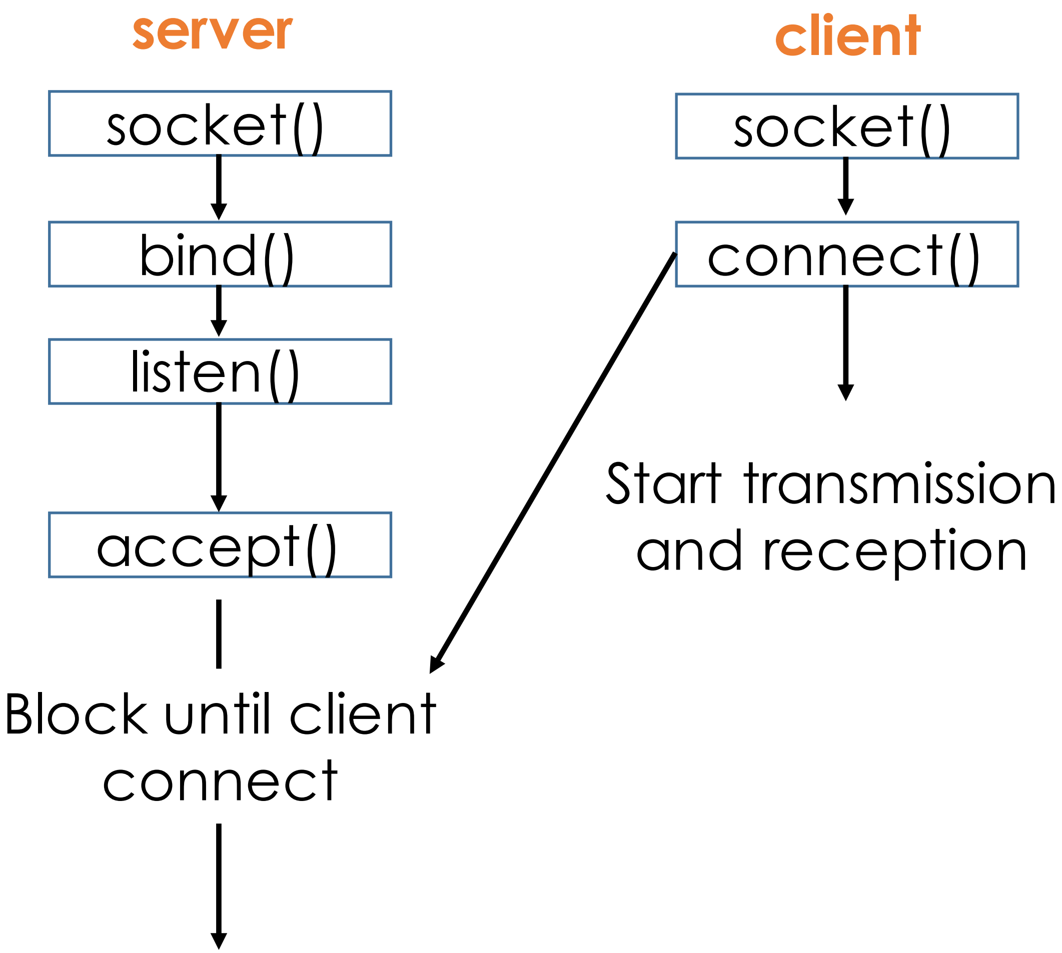
- Prerequisite
- Tasks
- Traffic Control
- **Socket Programming**
- Performance Evaluation
- Submission and Grading

Socket Programming

- Write both TCP and UDP socket to test audio streaming performance with different transportation protocols
  - Read the example codes
  - Complete the part with *the* ***"TODO"*** *tag*
  - Parameters:
    - IP, Port: 127.0.0.1, 1024~65535
    - Buffer size: 1024

Reference :
http://fanli7.net/a/caozuoxitong/Unix/20120625/175942.html

TCP Socket



server

socket()

bind()

listen()

accept()

Block until client connect

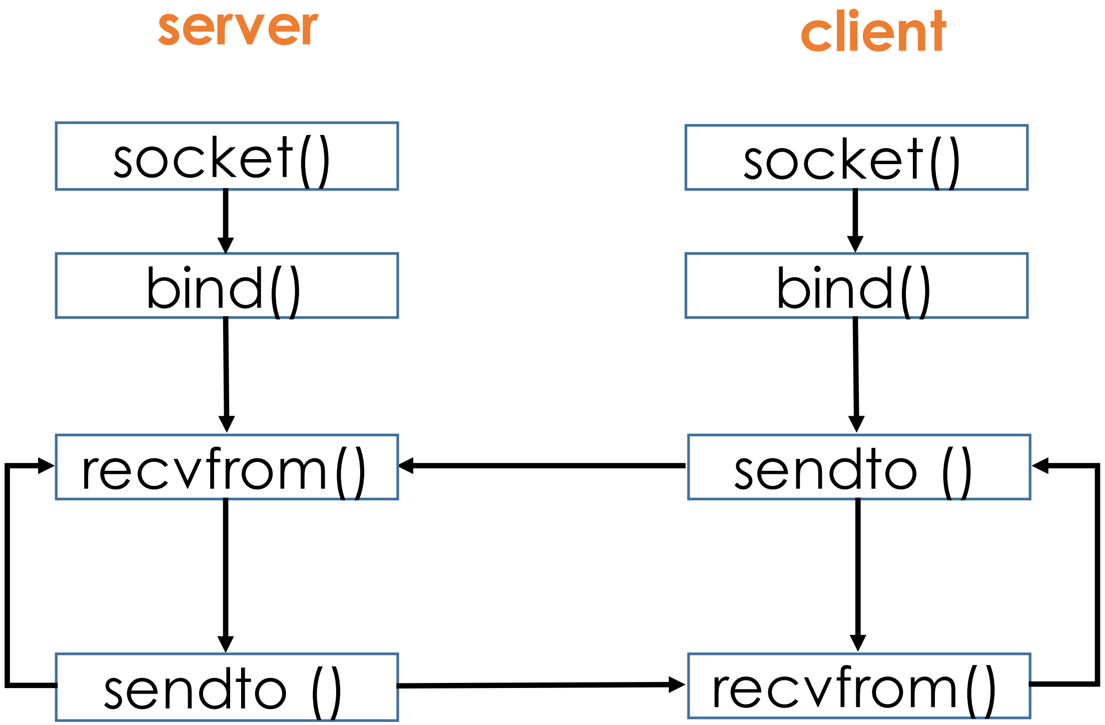
client

socket()

connect()

Start transmission and reception

UDP Socket (connectionless)



UDP Socket (connectionless)

- Why UDP dropped the packet
  1. The sending rate exceeds the configured bandwidth of traffic control
  2. The transmission buffer or reception buffer is full when you push packets
     → Enlarge the socket buffer or slow down the transmission rate (see example)

Outline

- Prerequisite
- Tasks
- Traffic Control
- Socket Programming
- **Performance Evaluation**
- Submission and Grading

How to Read Audio File

- Use libsndfile library

```
% sudo apt-get install libsndfile1-dev
```

- Compile the example file

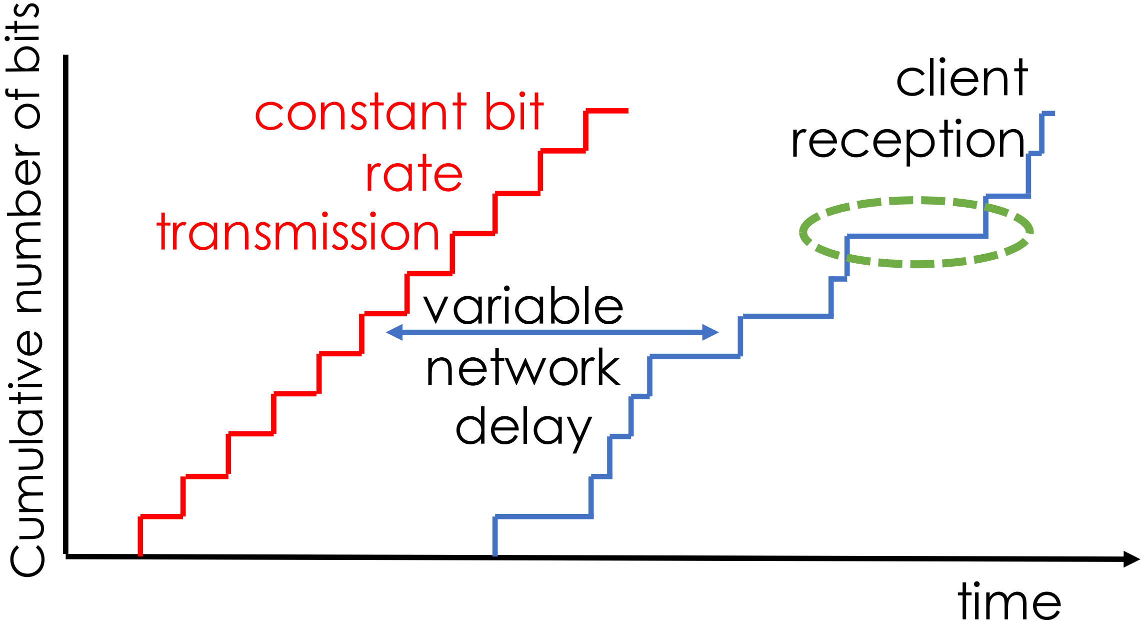
```
% g++ -w -o readwav readwav.c -lsndfile
```

- Execute the binary file

```
% ./readwav wav_filename
```

For TCP: Delay

- Save the received bits as `ratexx_tcp.wav`
- Log the time-stamp of each received packet
- Let t_1 be the time-stamp of the first packet
- Define `delay = ti - t1` ($t_i - t_1$) and save the delay values as `ratexx_tcp_delay.csv`
- Plot the following figure (blue curve only) and mark any packet experiencing a noticeable lag



For UDP: PSNR

- Drop the out-of-order packets (check the sequence number)
- Pad the lost packets as the same number of value ‘0’
  - For example, if you receive p1, p2, p4, p3, p6, then save p1, p2, p3, (000…0), (000...0), p6, where the number of 0 padded will be the number of bits per packet
  - You may need special process if the lost packets are the last ones
- Merge the packets and save as **`ratexx_udp.wav`**
- Use **`readwav.cpp`** to read audio samples and modify it to calculate the PSNR

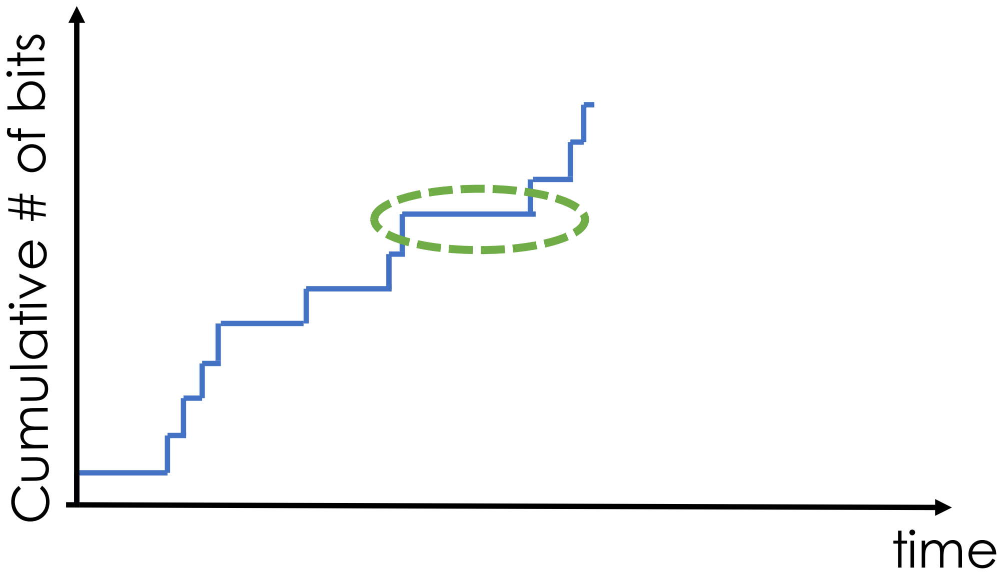
$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}[r(i) - s(i)]^2$$

$$PSNR_{dB} = 10 * \log_{10}\left(\frac{MAX_r^2}{MSE}\right)$$

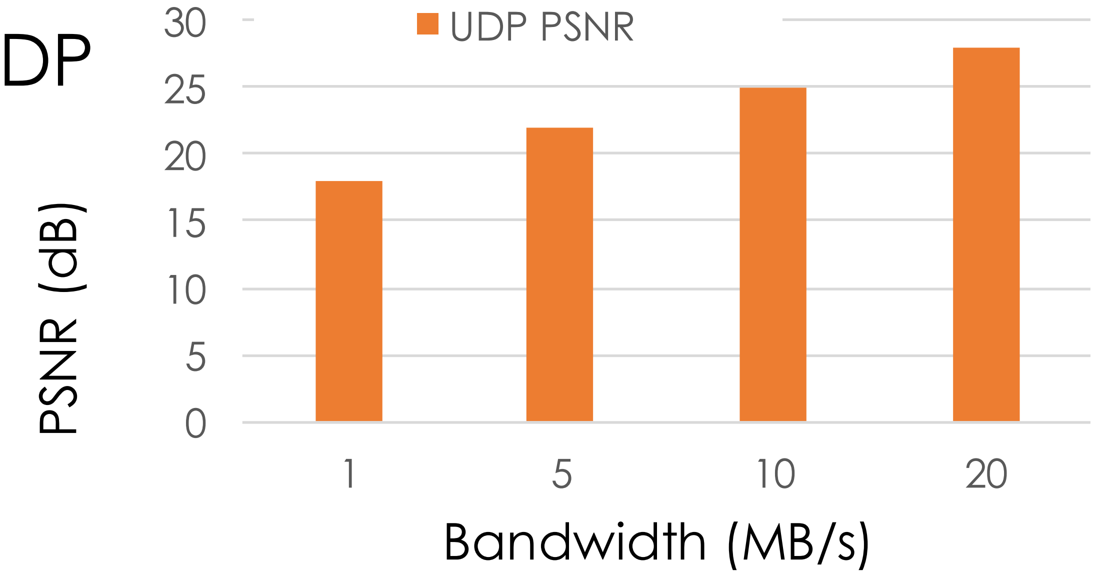
r(i): recorded samples
s(i): original samples
MAX$_r$: the maximum amplitude of the recorded audio

Performance Evaluation: Figure

- TCP



- UDP



Outline

- Prerequisite
- Tasks
- Traffic Control
- Socket Programming
- Performance Evaluation
- **Submission and Grading**

Task: Example Shell Scripts

server.sh

```
# configure tc to rate=$ARGV2
% sleep 1
# if $ARGV1=tcp
# ./tcp_server
# elseif $ARGV1=udp
# ./udp_server
```

client.sh

```
# if $ARGV1=tcp
# ./tcp_client | buffer | aplay
# elseif $ARGV1=udp
# ./udp_client | buffer | aplay
# ./readwav wav_filename
```

Run server.sh before client.sh

Submission and Due

- Submit the following files as a compressed file **`hw1_yourID.zip`** to mmcom.nctu@gmail.com by Mar. 31 23:59
  - Shell scripts (**`source.sh`** and **`client.sh`**) running all your code (may need to add **`sleep`** if necessary)
    - **`./source.sh [tcp/udp] [rate]`**
    - **`./client.sh [tcp/udp]`**
  - Source and output files
    - **`tcp_source.cpp, tcp_client.cpp`**
    - **`udp_source.cpp, udp_client.cpp`**
    - **`readwav.cpp`**
    - **`ratexx_tcp.wav, ratexx_udp.wav`**
    - **`ratexx_tcp.csv`**
  - 1-2 page report (**`report.pdf`**) including your figures and a short discussion

Grading

- Shell script: 10%
- TCP socket: 25%
- UDP socket: 25%
- Audio processing: 25%
- Report: 15%

Outline

- Prerequisite
- Tasks
- Traffic Control
- Socket Programming
- **SOX Audio Signal Processing Library**
- PSNR Calculation
- Tasks

SOX

- Install the library

```
% sudo apt-get install sox
```

- How to real-time play?

```
% ./udp_client | buffer | play -t wav -
```

- Record the sound
  - Use fork and exec in the client (already given in the example code)

Reference : http://sox.sourceforge.net/sox.html