Multimedia Communications
@CS.NCTU

Homework 2: adaptive-rate video streaming

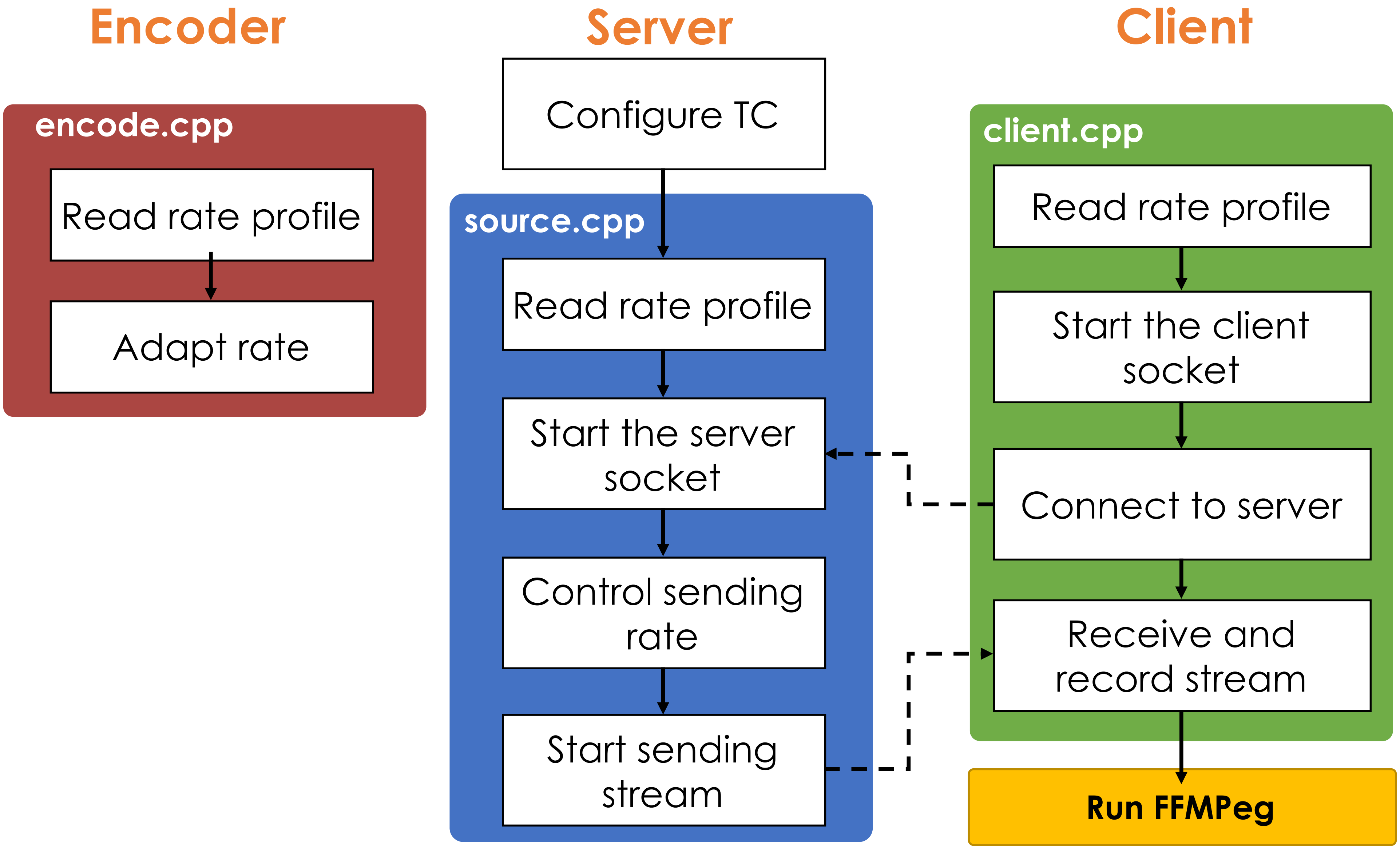
Outline

- **Tasks**
- H.264 Encoding
- Video Streaming
- Shell script and Makefile
- Performance Evaluation
- Submission and Grading

Task: Step by Step

- Video encoder
  - Read the profile of adaptive video rate
  - Modify the H.64 example code to configure the rates accordingly
- Traffic shaper
  - User tc to control the bandwidth of the s-d link
- Streaming server
  - Build a UDP socket
  - Send the compressed video file to the destination
  - Perform rate control (send K bits every 50ms)
- Streaming client
  - Build a UDP socket
  - Received UDP packets and pad ‘0’ for lost packets
  - Log the time-stamps of the first and last packet
- PSNR estimation
  - Use FFMpeg to calculate the average PSNR

Task: Diagram



Outline

- Tasks
- **H.264 Encoding**
- Video Streaming
- Shell script and Makefile
- Performance Evaluation
- Submission and Grading

Per frame encoding

- The encoding flow is frame by frame
- Adaptive video rate (example)
  - Initial with: 2048 kbps
  - Reconfig to 1024 kbps at 20s (Assume fps is 24.)
  - Reconfig to 512 kbps at 40s
  - Reconfig to 128 kbps at 60s

- Note: the unit of bandwidth is bytes/sec in tc(8), while the unit of video rate in h264 bits/sec

Rate Configuration Profile

videorate.txt

```
0        2048
20        1024
40       512
60        128
```

tcbw.txt

```
0            128
```

```
0       256
20      128
40      64
60      16
```

Outline

- Tasks
- H.264 Encoding
- **Video Streaming**
- Shell script and Makefile
- Performance Evaluation
- Submission and Grading

Streaming Server

- Convert the mp4 video to the yuv (raw) video

```
$ ffmpeg -i sample.mp4 -f rawvideo -vcodec rawvideo -pix_fmt yuv420p sample.yuv
```

- Shape the bandwidth using TC
  - For each video file, the server should try different bandwidth profiles
- Send a packet every 50 msec
  - The size of a packet should be determined according to the average video rate
  - For example, if the video rate is 128kbps, the packet size per 50ms should be

$$128 * 10^6 * 0.05 \text{ (bits)}$$

FFMpeg commands: See the readme file **`README.md`**

Streaming Client

- Track lost packets
  - For each lost packet, insert ‘0’ bits to the received bit-stream
- Save the received bits as a video file **`sample.h264`**
- Log the time-stamps of the first and last received packet
- Log the sequence numbers of the lost packets
- User FFMpeg to create a container

```
$ ffmpeg -i source.h264 -c:v copy -f mp4 myOutputFile.mp4
```

- Use FFMpeg to calculate the PSNR of the received video file

```
$ ffmpeg -i input_video.mp4 -i reference_video.mp4 -filter_complex "psnr" "output_video.mp4"
```

Outline

- Tasks
- H.264 Encoding
- Video Streaming
- **Shell script and Makefile**
- Performance Evaluation
- Submission and Grading

Shell Script

streaming.sh

```
# First encode and output sample h264
# Launch server in background
# Launch control.sh in background
# Launch client
```

control.sh

```
# Write a tc flow match the profile
# ..
# ..
```

Run ./streaming.sh

Makefile

- CXX = g++
- INCLUDE_DIR = ./include
- SRC_DIR = ./src
- OBJ_DIR = ./obj
- CFLAGS = -std=c++11 -g -O2 -Wunused-result

- PROG = x264_encode server client

- all: $(PROG)

- %: $(SRC_DIR)/%.cpp
- 	$(CXX) -o $@ $(CFLAGS) $< -lx264

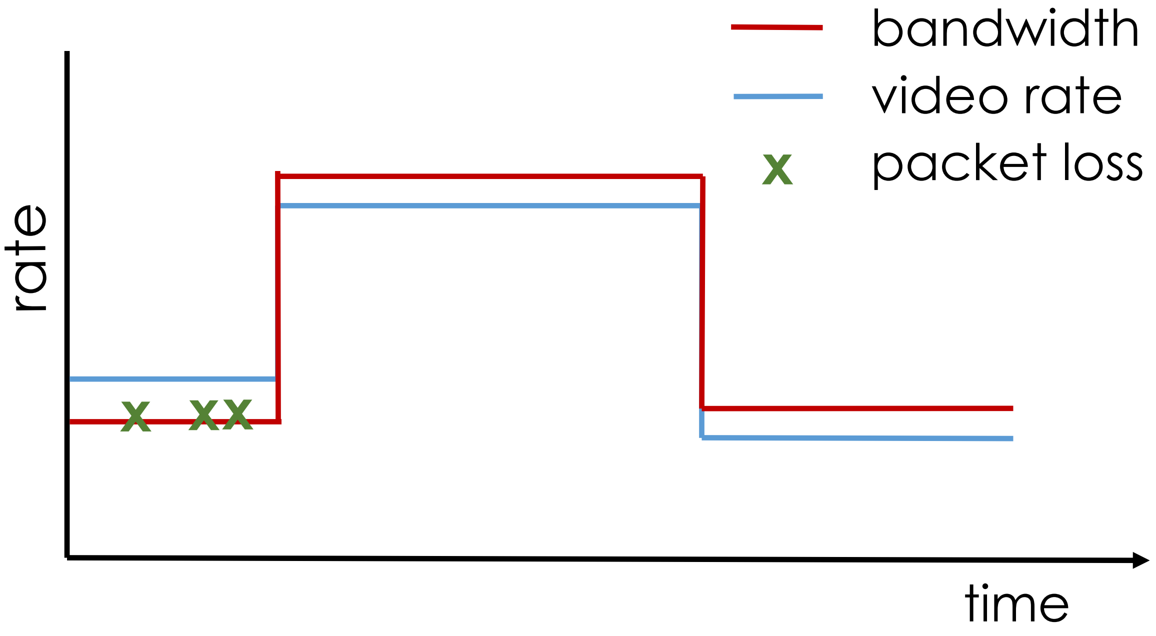
- clean:
- 	rm -rf $(PROG)
- # It will work fine as you place your code in src directory.
- # Please submit with it!

Outline

- Tasks
- H.264 Encoding
- Video Streaming
- Shell script and Makefile
- **Performance Evaluation**
- Submission and Grading

Output

- PSNR
- Playout duration: $\text{time}_{\text{last}} - \text{time}_{\text{first}}$
- figure



Outline

- Tasks
- H.264 Encoding
- Video Streaming
- Shell script and Makefile
- Performance Evaluation
- **Submission and Grading**

Submission and Due

- Submit the following files as a compressed file `hw2_yourID.zip` to mmcom.nctu@gmail.com by May. 31 23:59
  - Makefile
  - Shell scripts (`streaming.sh` and `control.sh`) running all your code (may need to add `sleep` if necessary)
    - `./streaming.sh[videorate.txt]`
    - `./control.sh [tcbw.txt]`
  - Source and output files
    - `x264_encode.cpp, server.cpp, client.cpp`
  - 1-2 page report (`report.pdf`) including your results/figures and a short discussion

Grading

- Shell script: 10%
- encoder: 25%
- Streaming server: 25%
- Streaming client: 25%
- Report: 15%