MM’21 Grand Challenge (GC)

ShengMing Tang

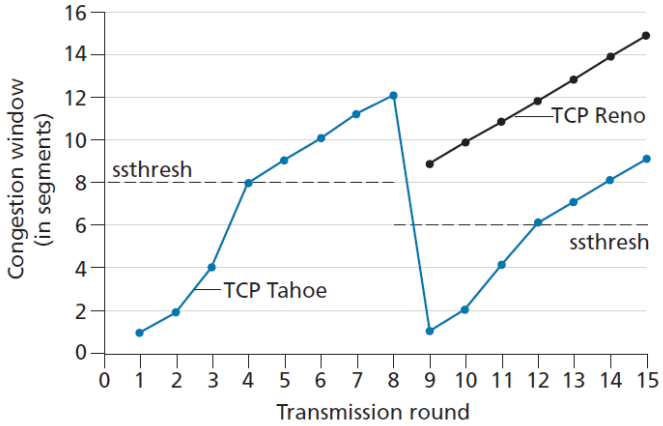
Introduction

- [GC web]
- Real Time Communication (RTC) application are popular
  - Online conferencing
  - Control (commands, reponse ASAP)
  - Video (large b/w consumption, stalling is tolerable)
  - Audio (annoying if sounds are not smooth)
- Delay senstitive multimedia challenges
  - Meet deadline requirements

- Can we send everything out once a block is available at highest rate?

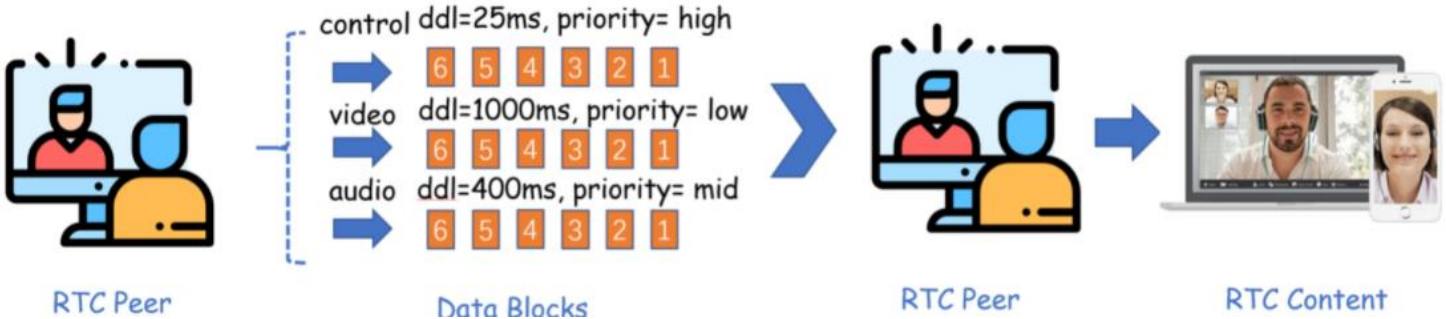
Introduction

- Transmission
  - Application data are in unit of blocks with different priority
    - Control (high, priority-0)
    - Video (low, priority-1)
    - Audio (middle, priority-2)
  - Blocks are divided into packets to fit into lower network layers
- Congestion Control
  - Send at unlimited speed will congest the link along the way
  - Congested link will introduce packet losses
    - Miss a deadline



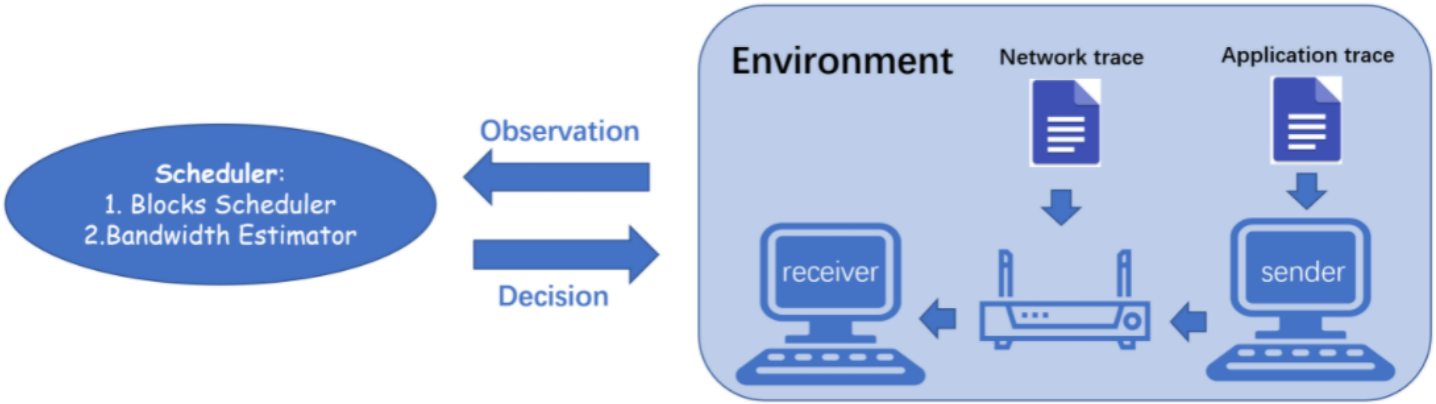
Introduction (cont'd)

- Objective
  - Serve highest possible QoE, which is composed of
    - Priority
    - Meetting the deadline or not
  - Design congestion control (CC) algorithms
    - Which block to send
    - Congestion window control (if your alg. is congestion window based)
    - At what speed



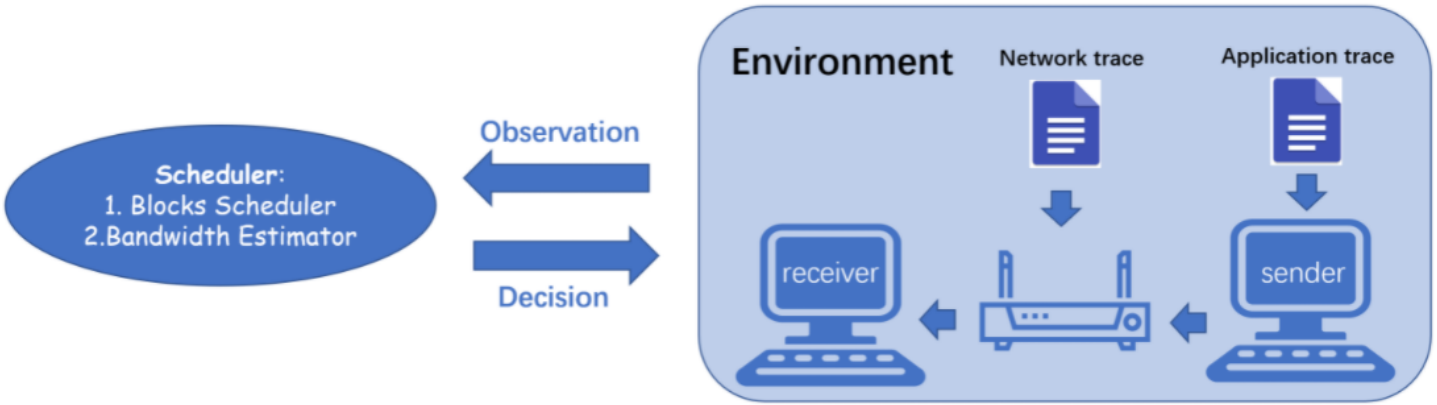
Simulation Platform

- Platform
- Network trace
  - Network condition record which can be replayed
- Application trace
  - Block record which also could be replayed
- Use the two traces to replay different scenarios



Task description

- Block scheduler
  - select which application block to send
- Bandwidth estimator
  - bandwidth estimation (not necessary if you don’t need this)
  - congestion controll triggered on packet ACK/loss/sent



Function Explained

- def select_block(self, cur_time: float, block_queue: list) -> int:
  - return block index to be sent relative to the block_queue
- def on_packet_sent(self, cur_time: float) -> dict:
  - called on every packet is sent
  - state update
  - return {“cwnd”: int, “send_rate”: float}
- def cc_trigger(self, cur_time: float , event_info: dict) -> dict:
  - called on packet ack-ed or droped
  - state update (return in the same form as on_packet_sent(...))
- Examples are here.
  - TCP Reno

Dataset

- Open dataset (on the same Github)

Application Trace

We provide application traces w
priority of the application trace
means the highest priority in thi
trace's name "block-priority-0-d

For each file of application trace

Time (s)	Block Size (B)
0.0	514
0.06	305
...	...

Network Trace

For each network trace file, the first column is the timestamp. The second colu
loss rate of the link. The last column is the fixed propagation delay in seconds

Time (s)	Bandwidth (MB)	Loss Rate	Propagation Delay (s)
0	19.38592070201254	0	0.001
1	24.832955411664393	0	0.001
...	...	...	...

The QoE Model

Evaluation Metrics

After the submission of "Scheduler", the simulator will go through each pair of video traces and network traces to simulate the transmission using the algorithm implemented in "Scheduler". Each simulating will calculate a QoE which representing the score of a point. The QoE model about deadline requirements can be expressed as:

$$QoE = \sum_{n=1}^{N} \phi(P_n) \times meet_n - \mu \sum_{n=1}^{N} \psi(P_n) \times (1 - meet_n)$$

where $meet_n$ is a binary-state value and indicates whether the $block_n$ arrives before the deadline (i.e., $meet_n = 1$ means $block_n$ met deadline, $meet_n = 0$ means $block_n$ missed deadline). P_n denotes the priority of $block_n$. $\phi(P_n)$ expresses the QoE improvement if $block_n$ arrives before its deadline. The block with high priority, which is significant to experience of user, corresponds to high $\phi(P_n)$. $\psi(P_n)$ penalizes the deadline missing event of $block_n$.

In this challenge, we consider a simplified choice of $\phi(P_n)$, $\psi(P_n)$ and μ :

$$\phi(P_n) = P_n, \qquad \psi(P_n) = P_n, \qquad \mu = 1$$

Then the QoE model can be expressed as:

$$QoE = \sum_{n=1}^{N} P_n \times meet_n - \sum_{n=1}^{N} P_n \times (1 - meet_n)$$

Pn for meeting the deadline
-Pn otherwise

Additional Resources

- Grand Challenge paper
  - Paper that won the first award
  - Simply reproducing the algorithm will result in zero score
- Basics
  - TCP Reno (congestion control algorithm)
- Simulation
  - Emulator class explained
    - Instantiation details

File Structure and Self Test

- Clone or fork this https://github.com/ShengMingTang/Meet-Deadline-Requirements-Challenge
- Put everything you need under CS3570/s{StudentID}
  - Read the document string in ‘TA_eval.py’ carefully
  - ‘solution.py’, ‘requirements.txt’, and so on
- We will change to your directory when evaluating your code
  - Use relative path when reading/writing files
- Run ‘TA_testcaseGen.py’
  - A small batch of test cases will be generated in ‘hidden/’
- Run ‘TA_eval.py’
  - Some output will logged and qoe.csv will be generated to bookkeep the details

Submission Details (for code)

- solution.py
  - Your code must start from this template (in CS3570/s000000000)
  - The class must be named ‘MySolution’
- sideModule.py
  - Optional, if you have any code that is separated
- requirements.txt
  - $ pip freeze > requirements.txt
  - To separate your environment from others’
- others
  - Any of additional files
  - Model files

Evaluations

- Test cases will be released after your submission
- We will use exactly the same code as ‘TA_eval.py’ to evaluate your performance
- Make sure that your structure is compatible with our requirements
- Live demo is required if your code throws any exception
  - Scores will be heavily deducted if this comes from implementation not setting errors
  - Make sure that you have passed the self test
  - Zero score if the team does not show up for makeup demo
- Actual run time is limited to 20 sec / test case (maybe extended)
  - Each test case is of length 20 seconds
- Benchmark Scores (summation of QoE overall testcases)
  - top ⅓: 10 points
  - next ⅓: 7 points
  - last ⅓: 4 points