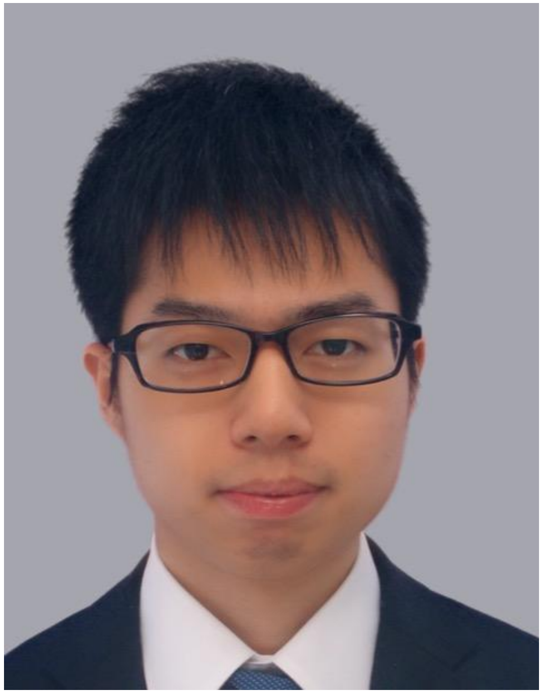
On the FullMesh Path Selection of Multipath TCP Video Streaming

Yosuke Komatsu, Dirceu Cavendish
Daiki Nobayashi, Takeshi Ikenaga

Kyushu Institute of Technology, Japan
komatsu.yousuke620@mail.kyutech.jp
{cavendish@net.ecs, nova@ecs, ike@ecs }.kyutech.ac.jp

About Me

- Yosuke Komatsu
  - First-year master’s student
  - Kyushu Institute of Technology, Japan
  - komatsu.yousuke620@mail.kyutech.jp

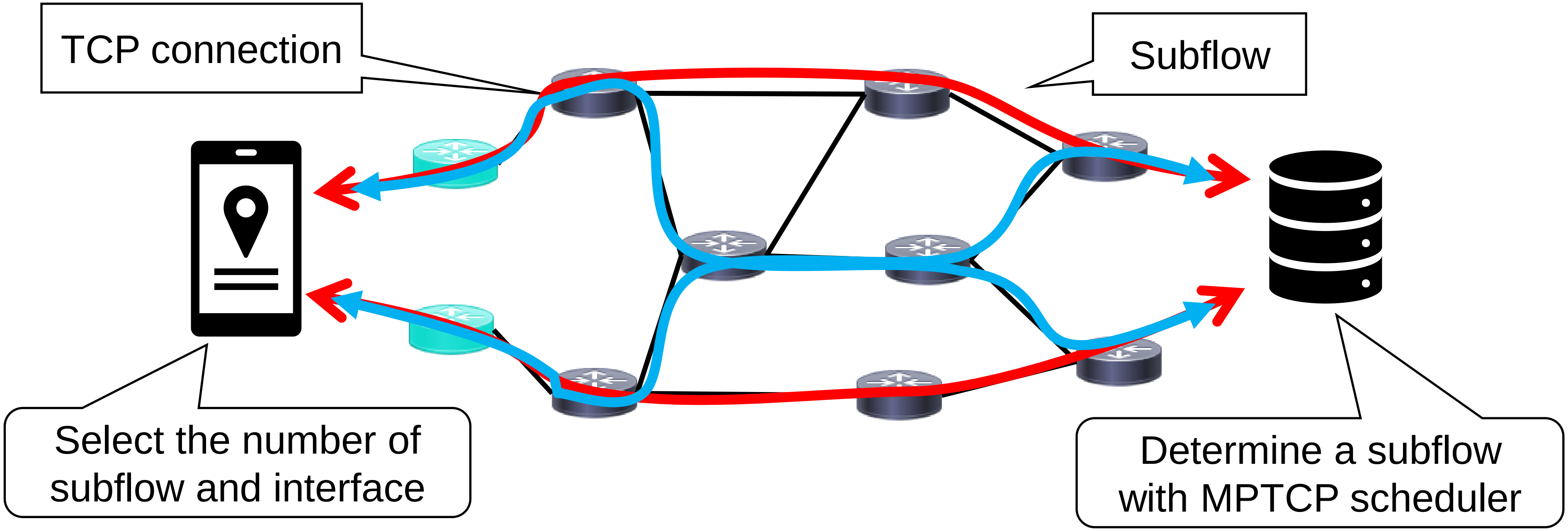


- Field of Study
  - MPTCP
  - Transport Protocol

Video Streaming

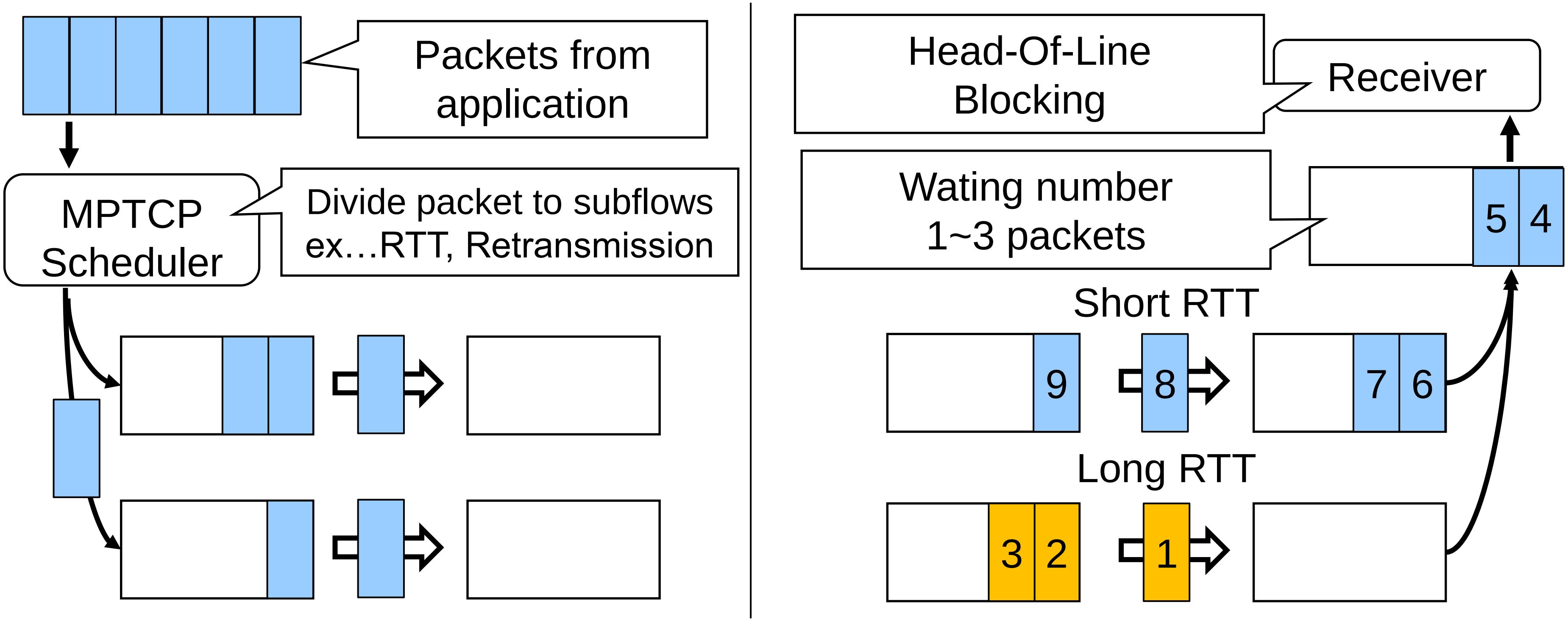
- Examples of video streaming platforms
  - Youtube, Netflix, Amazon Prime Video …etc.
- TCP connection
  - Use a single interface
  - New connections need to be made when switching interfaces
- MPTCP connection
  - Multiple interfaces can be used simultaneously
  - Increased communication stability by eliminating the need to switch interfaces
  - Enables more bandwidth than TCP
  - ➢ Effective because many devices now have multiple interfaces, wireless and wired

About MPTCP


TCP connection
Subflow
Select the number of subflow and interface
Determine a subflow with MPTCP scheduler

- MPTCP control multiple TCP connection
- Added connections are called “Subflow” by MPTCP
- The Receiver (Sender) can change the number of interfaces and subflows
- The Sender decide a subflow to send a packet following MPTCP scheduler

MPTCP Scheduler and Head of Line Blocking



- MPTCP scheduler selects a subflow with several methods
  - RTT, Retransmission etc.
- Head of Line (HOL) blocking can occur by MPTCP scheduler
- Preventing HOL blocking leads to high performance in applications

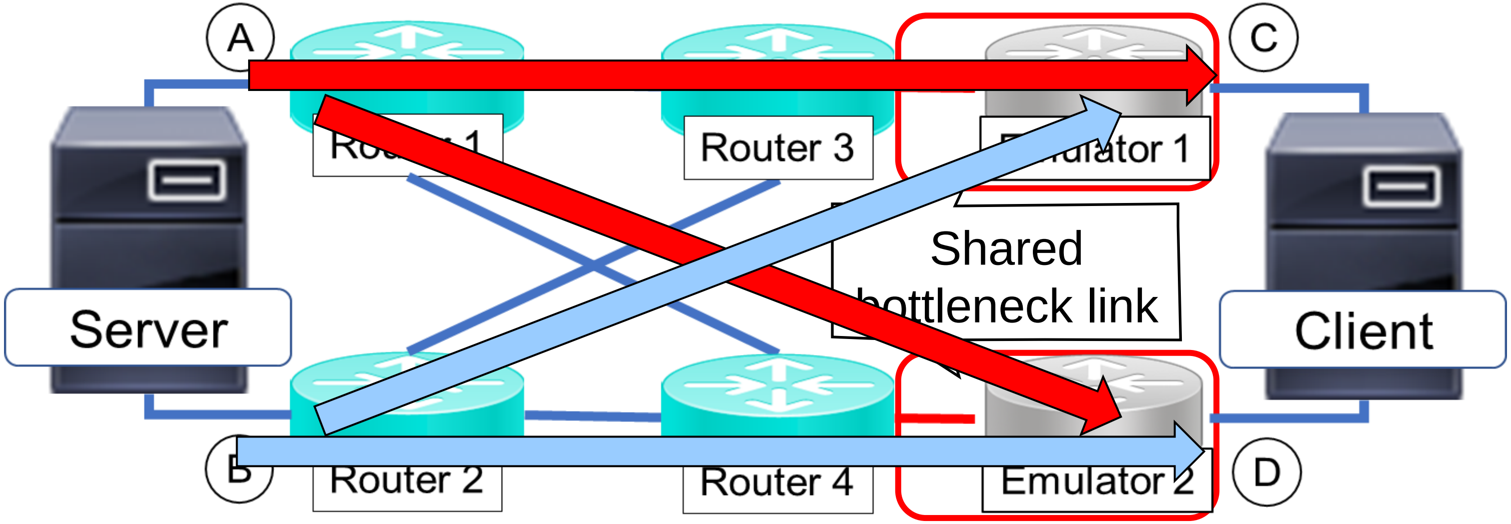
Video Streaming by MPTCP

- Video streaming needs stable throughput
  - Low throughput, latency variation and HOL blocking cause interruptions to video streaming

- MPTCP is required adaption to any topology, connection path, path quality

- We conduct MPTCP video streaming with and without a shared bottleneck and propose an efficient scheduler

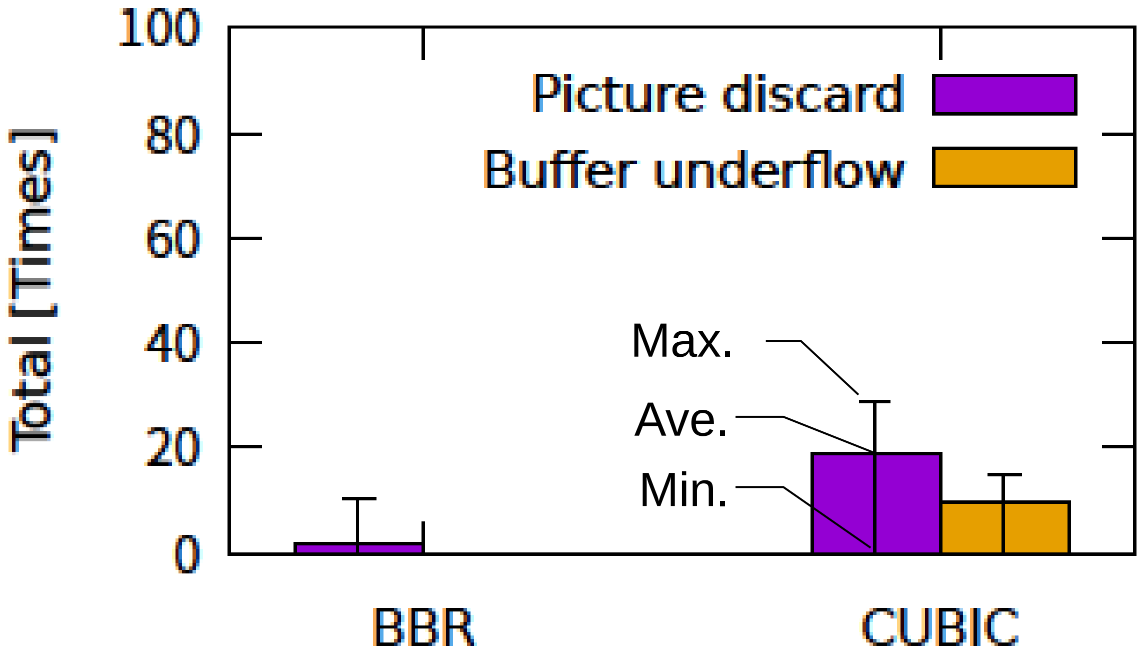
Fullmesh Test Experiment with Default Scheduler



- Testbed experiment
  - Establish fullmesh routes on all interfaces used by each other (A-C, A-D, B-C, B-D) → Emulators are shared bottleneck links
- Emulator setting
  - BW: 3Mb/s, Packet loss rate: 0.1%, RTT: 120ms
- Video values
  - Bitrate: 5.24Mb/s, Playout time: 6min
- Congestion control
  - CUBIC, BBR

➢ Evaluate video quality (Picture discard, Buffer undefflow) results in five experiments

Video Quality of Default scheduler

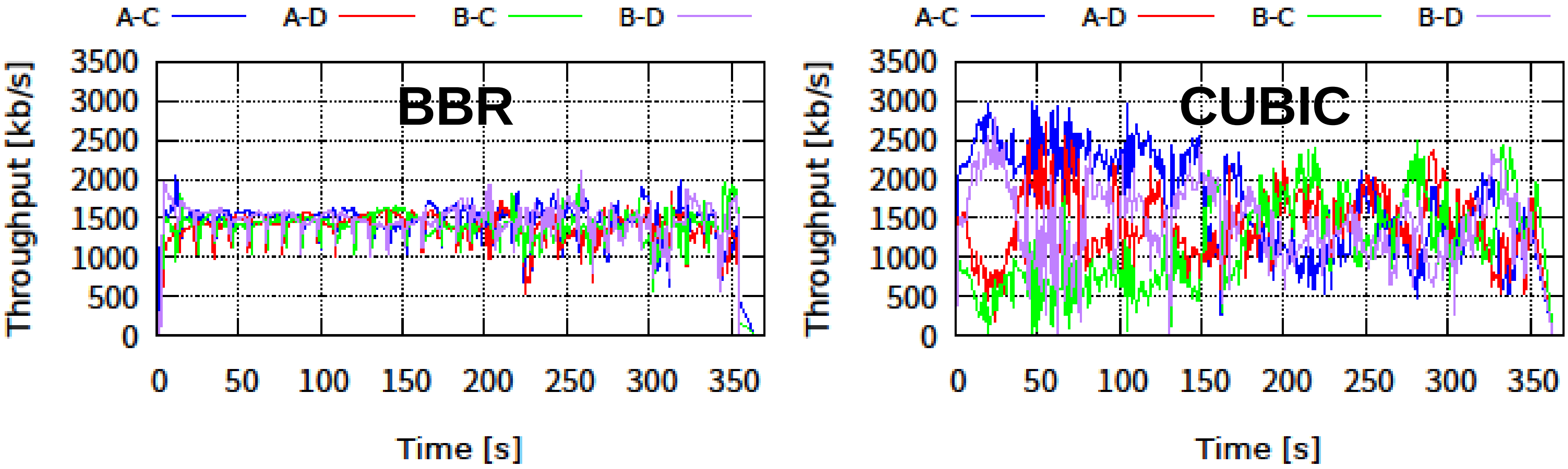


BW: 3Mb/s, Packet loss rate 0.1%, RTT 120ms

- BBR had good video quality
- CUBIC caused degradation of video quality

➢ Congestion control has a big effect on video quality

Client’s Downlink Throughput

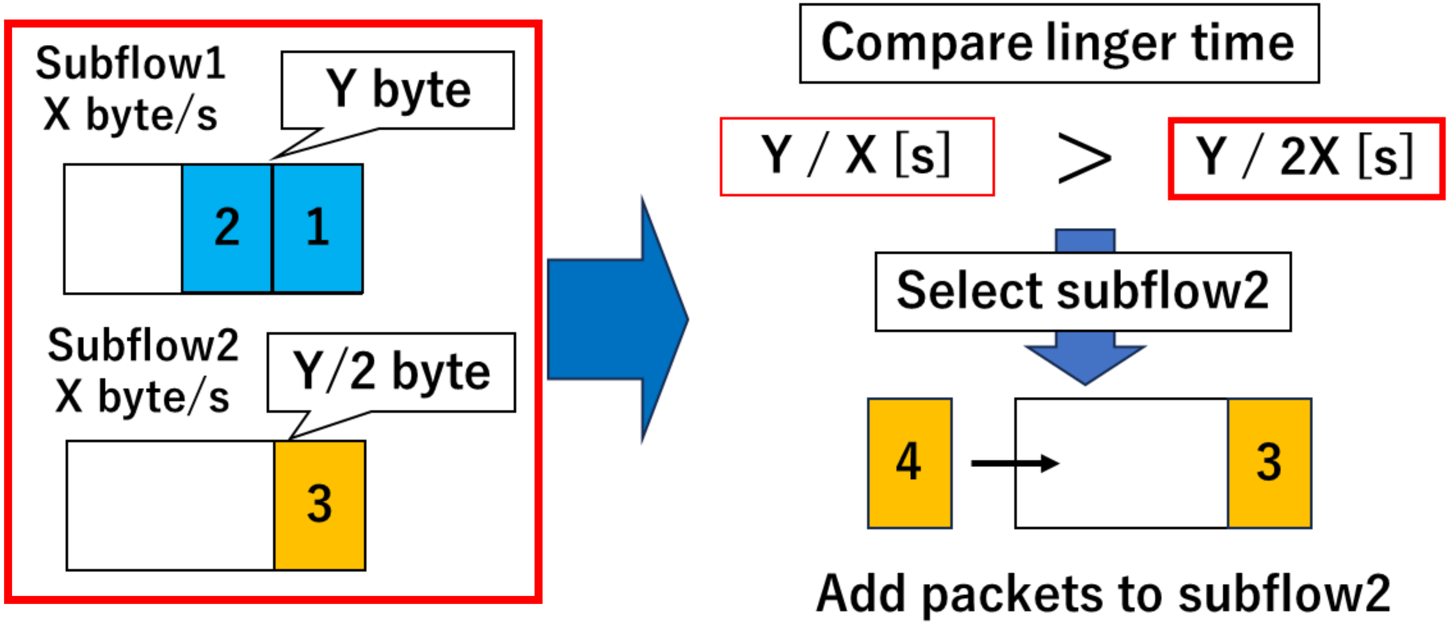


- In BBR, all subflows use the bandwidth fairly, but in CUBIC they are competing for it

- MPTCP in fullmesh requires consideration of shared bottleneck links

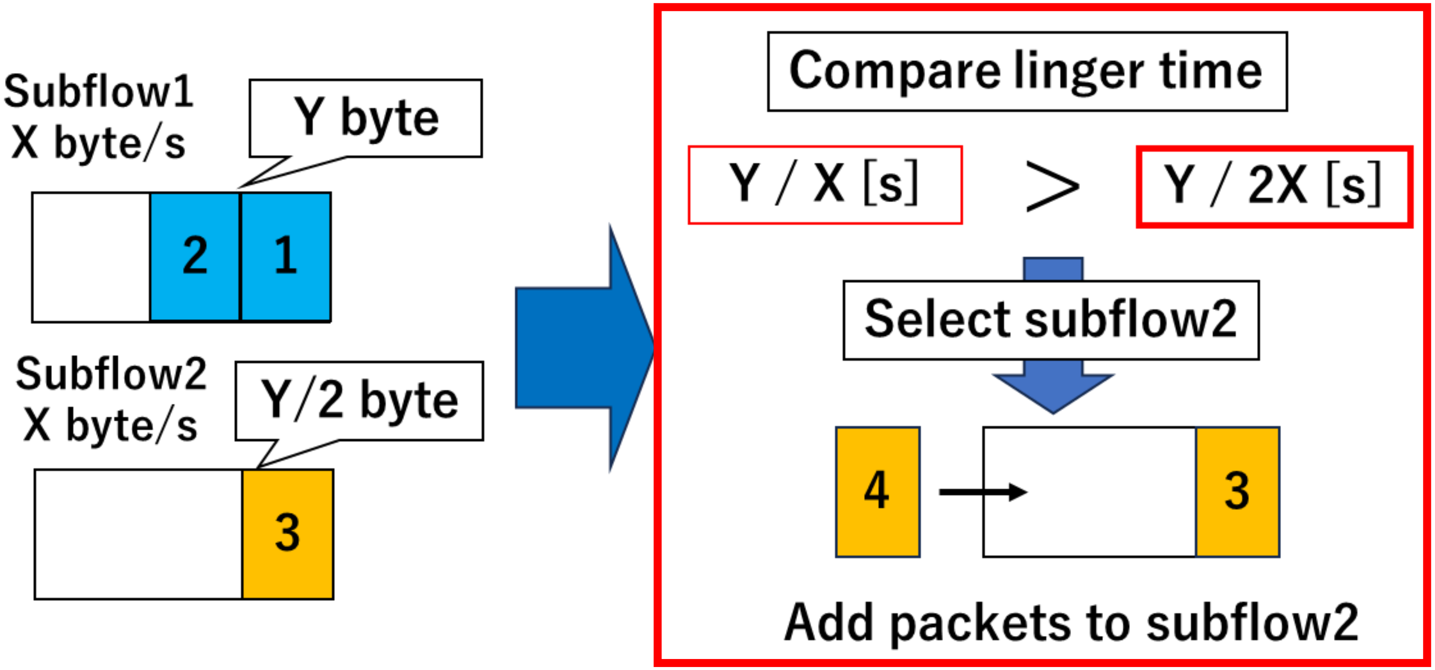
About Default Scheduler #1

- Select the subflow with the shortest transfer time
- Transfer time is calculated from the total packet size in the send buffer and the pacing rate
  - Pacing rate: Packet transmission rate



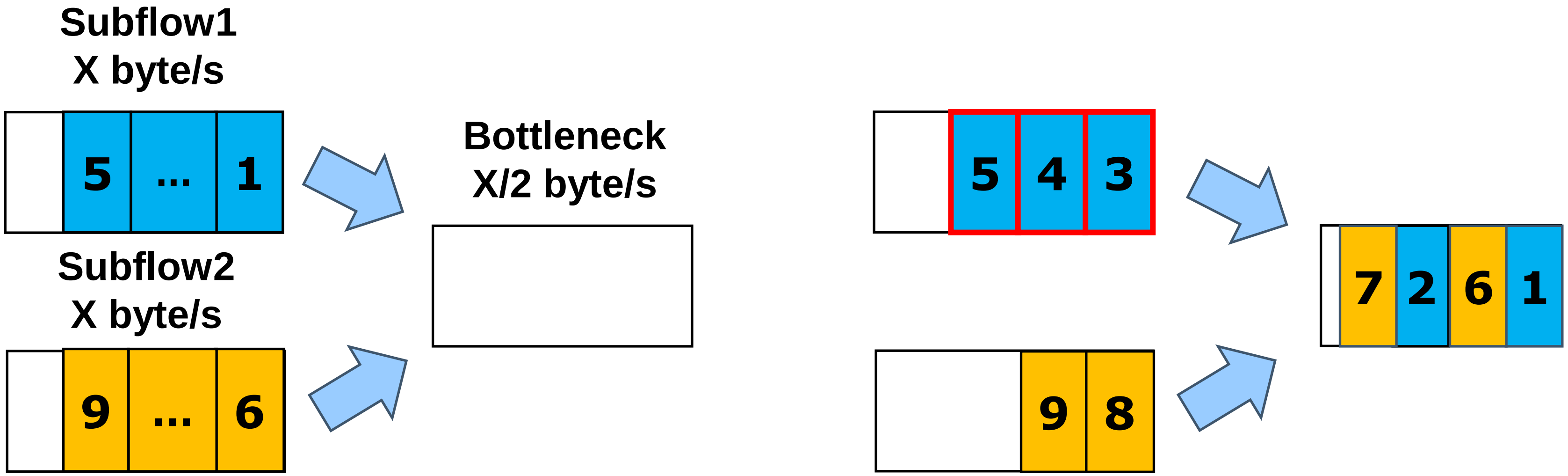
About Default Scheduler #2

- Transfer times for Subflow1 and Subflow2 are “Y/X [s]” and “Y/2X [s]”
- Add new packets to Subflow2's send buffer because Subflow2 can send all packets the fastest



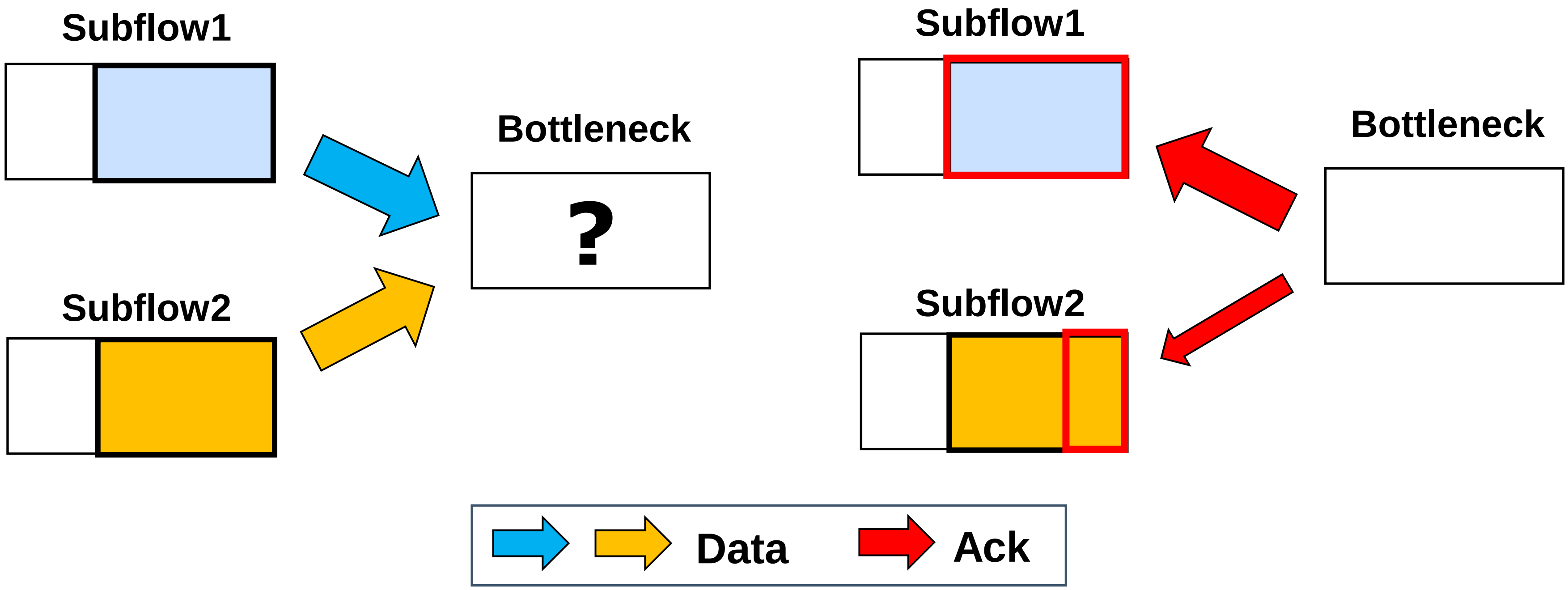
Problems with Fullmesh by Default Scheduler

- Buffering delay increases on the bottleneck link when each subflow over-transmits packets to the bottleneck link
- Delayed transmission of packets that should have arrived first
- If the scheduler continues to add new packets, they will also be affected



Approach against a Shared Bottleneck

- Limit the send buffer size for subflows
  - Unknown if shared bottleneck link exists immediately after communication starts
  - However, it is possible to assume that the packet size of the packets sent with an ACK is the appropriate bandwidth for the subflow



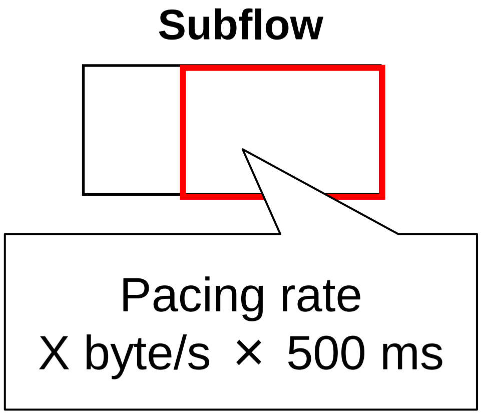
Proposed Method

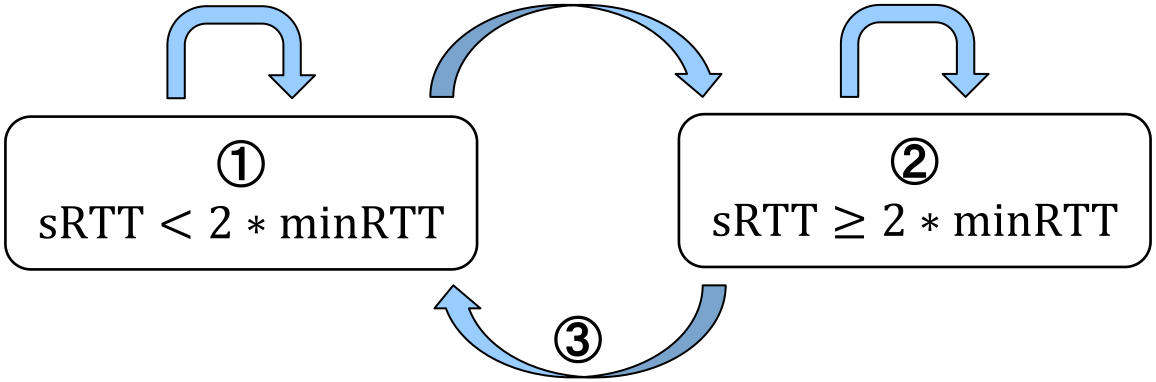
- Bottleneck aware scheduler
  - Added send buffer size limit to Default scheduler

- Benefits
  - Subflow selection by transfer time can avoid head-of-line blocking
  - Prevent over-transmission to shared bottleneck links that occurs in fullmesh connections

Bottleneck aware Scheduler #1

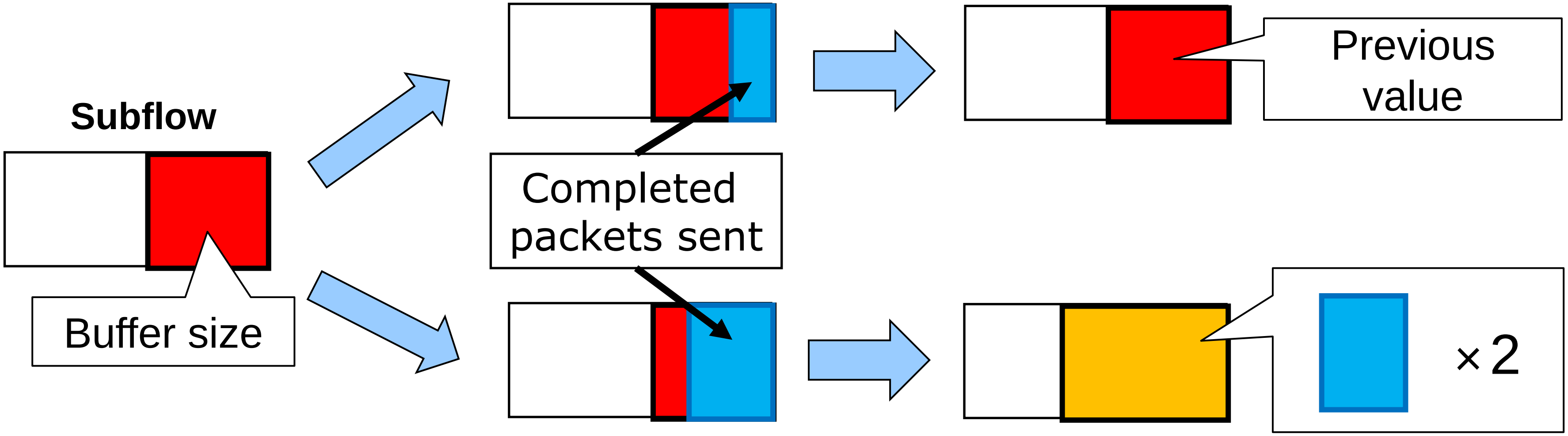
- The initial value is 500 ms of the pacing rate
- Each subflow updates its value from its own minimum RTT (minRTT) and smooth RTT (sRTT)
  - $\mathrm{sRTT} < 2 * \mathrm{minRTT}$ ...①
  - $\mathrm{sRTT} \geq 2 * \mathrm{minRTT}$ ...②
  - ② → ① ...③




①
$\mathrm{sRTT} < 2 * \mathrm{minRTT}$
②
$\mathrm{sRTT} \geq 2 * \mathrm{minRTT}$
③

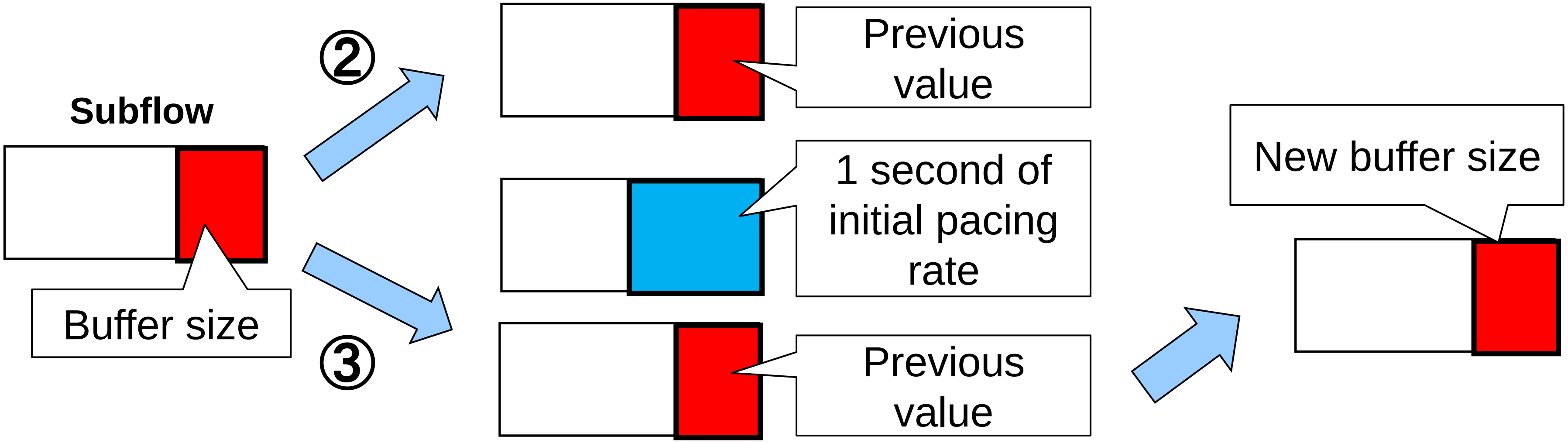
Bottleneck aware Scheduler #2

- $sRTT < 2 * minRTT$ …①
  - Select the larger value of the following
    - Previously used value
    - Twice total packet size of completed transmissions from the previous subflow selection phase to the current selection phase

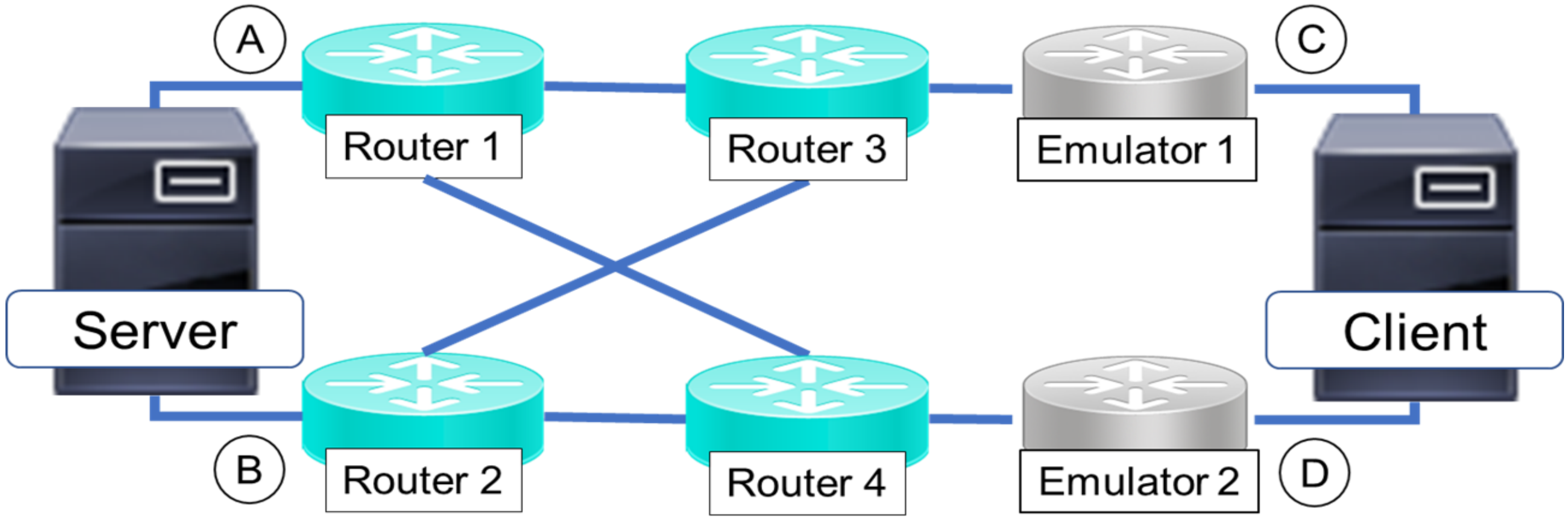


Bottleneck aware Scheduler #3

- $\mathrm{sRTT} \geq 2 * \mathrm{minRTT}$ …②
  - Maintain previous value
- ② → $\mathrm{sRTT} < 2 * \mathrm{minRTT}$ …③
  - Select the smallest value of the following
    - 1 second of initial pacing rate
    - Previous value



Experimental Environment (Fullmesh)

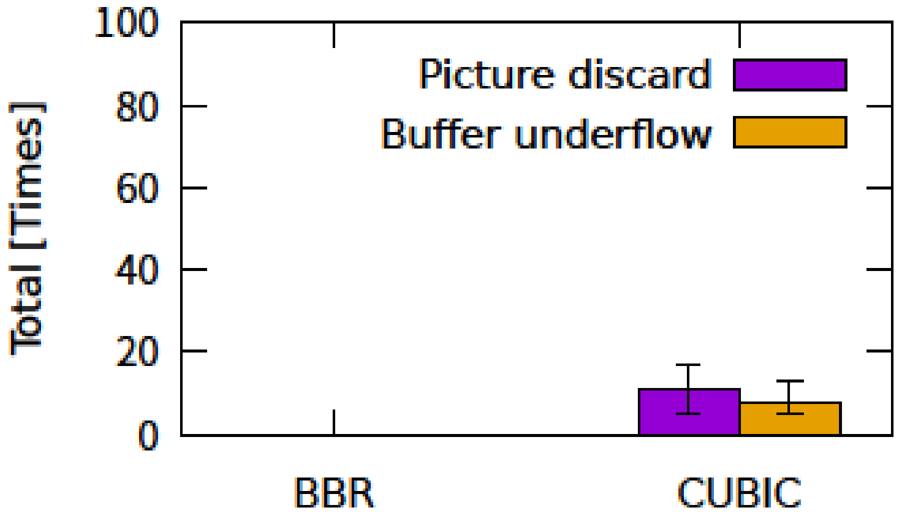


- Testbed experiment
- Video values
  - Bitrate: 5.24Mb/s
  - Playout time: 6min
- Emulator setting
  - BW limitation: 3Mb/s
  - Packet loss rate: 0.1%
  - RTT: 60ms, 120ms
- Congestion control
  - CUBIC
  - BBR
- MPTCP scheduler
  - Default
  - Bottleneck aware (proposed scheduler)

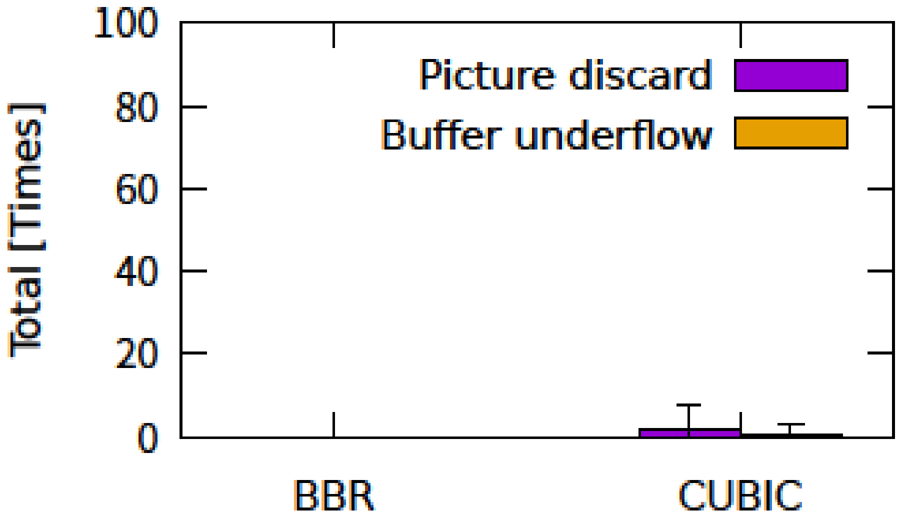
Experimental Scenario and Evaluation Index

- Experimental scenarios
  - Fullmesh
    - 1… BW: 3Mb/s, Loss rate: 0.1%, **RTT: 60ms**
    - 2… BW: 3Mb/s, Loss rate: 0.1%, **RTT: 120ms**
- Evaluation index
  - Video quality
    - Picture discard
    - Buffer underflow
  - Network quality
    - Number of Out-oF-Order（OFO）
- Each scenario was conducted 5 times

Compared Results (Fullmesh-1)#1



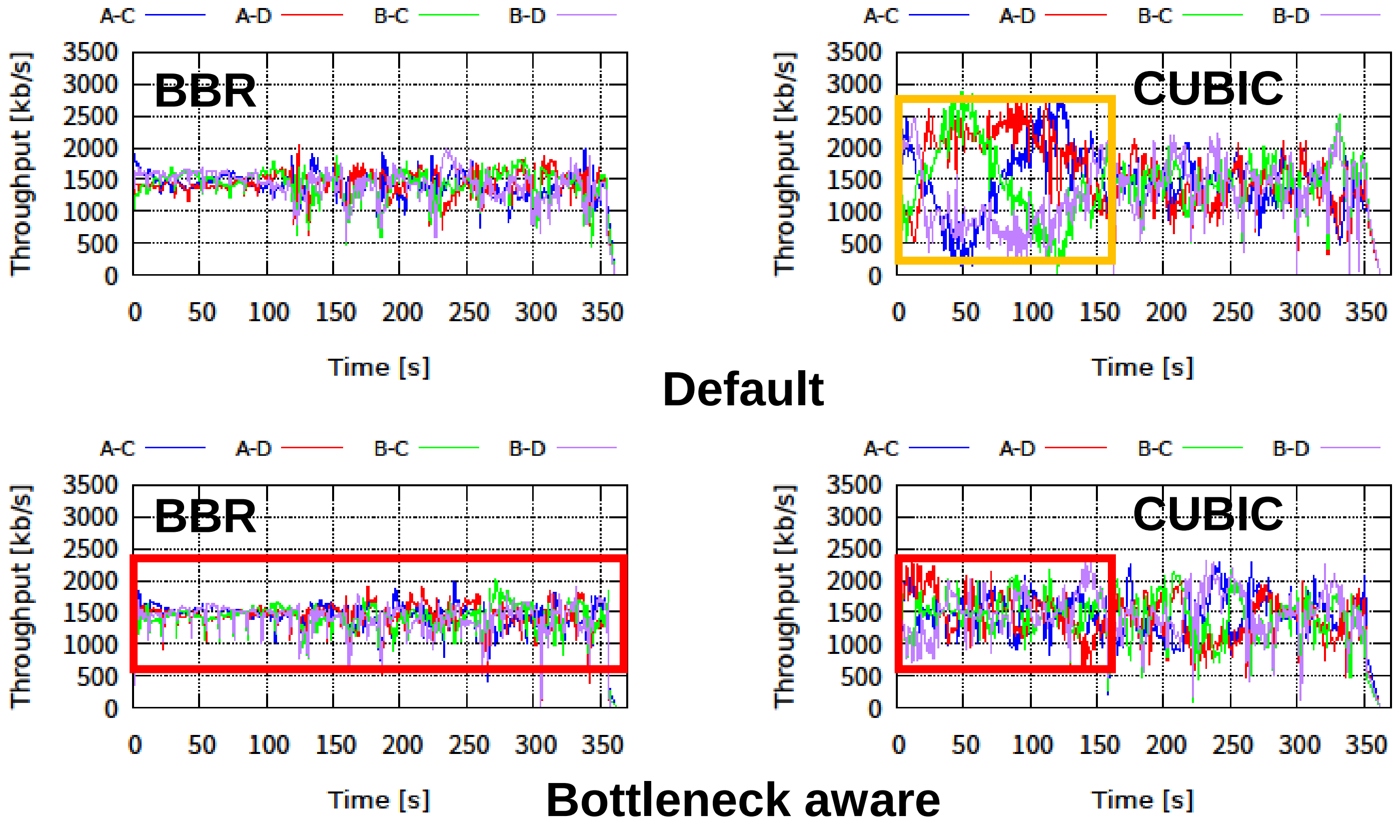
Default



Bottleneck aware

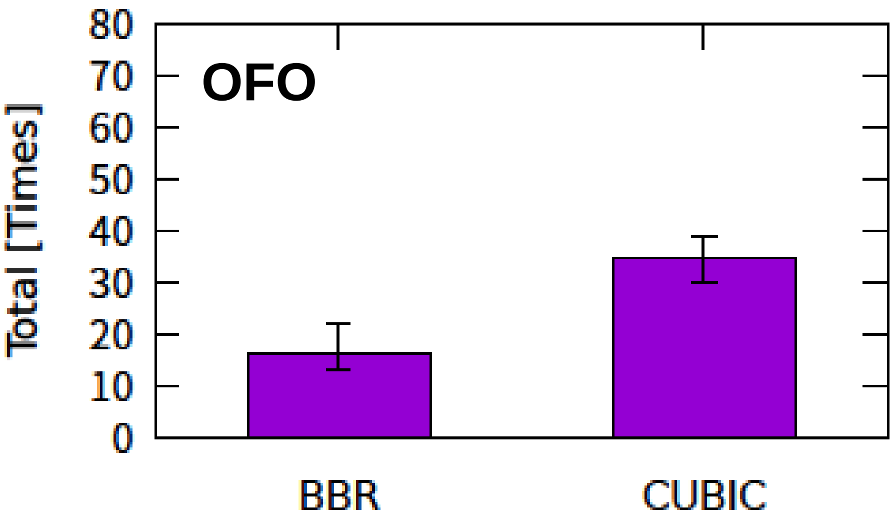
- ◆Prevents video quality degradation when using CUBIC
- ◆The number of retransmissions and the impact of OFO remain almost the same

Compared Results (Fullmesh-1)#2

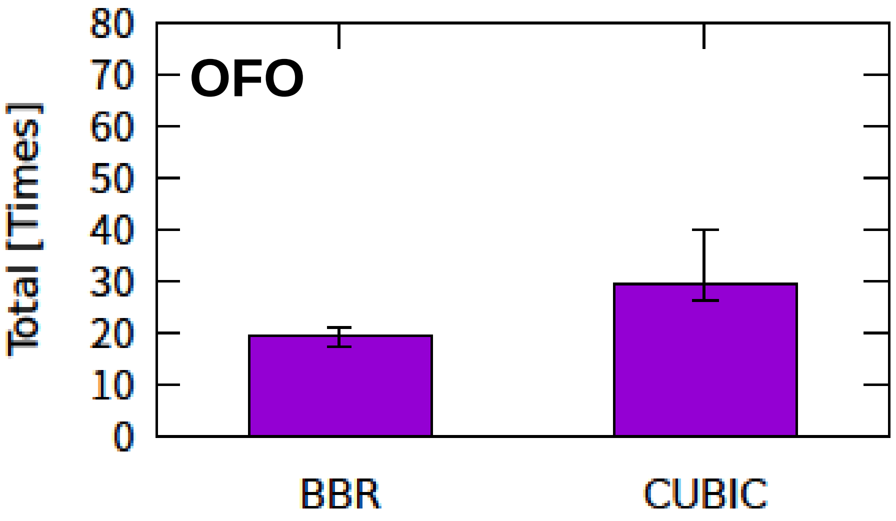


- BBR results similar to Default scheduler
- CUBIC reduces throughput fluctuations
- Prevents over-transmission on bottleneck links, resulting in better video quality

Compared Results (Fullmesh-1)#3



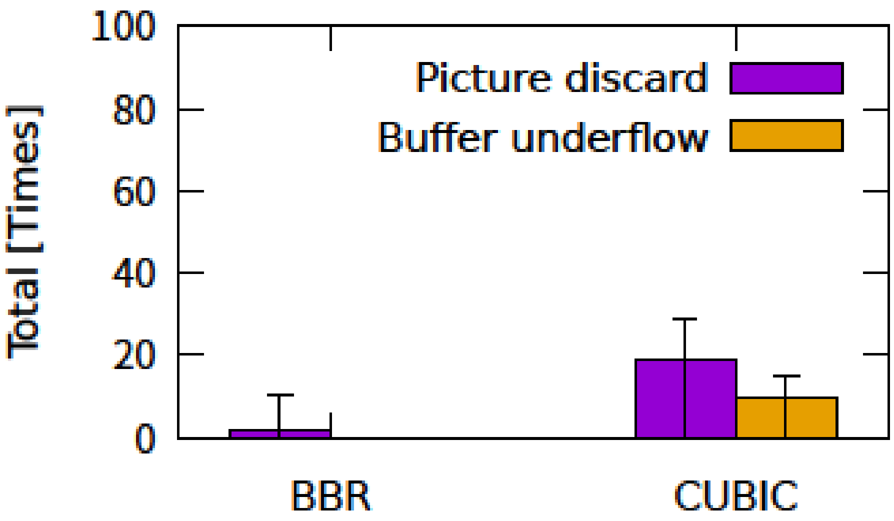
Default



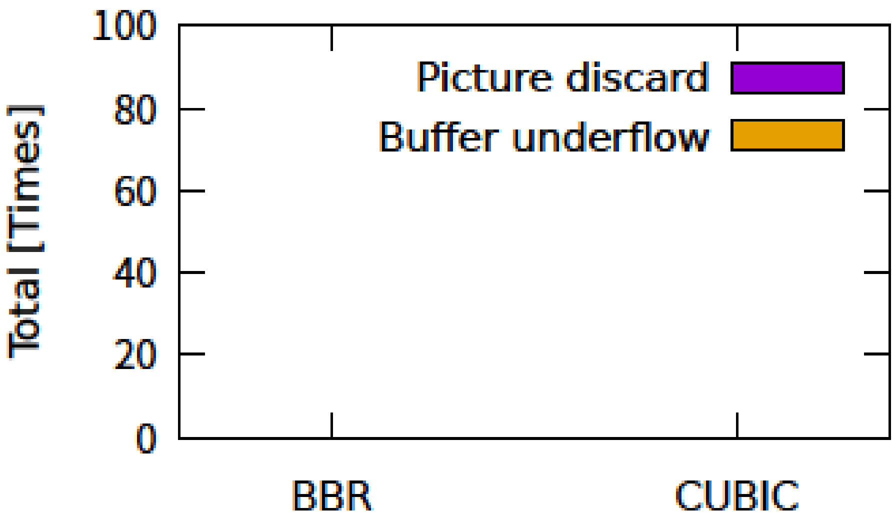
Bottleneck aware

- The proposed method maintains the same network quality as the Default scheduler

Compared Results (Fullmesh-2)#1



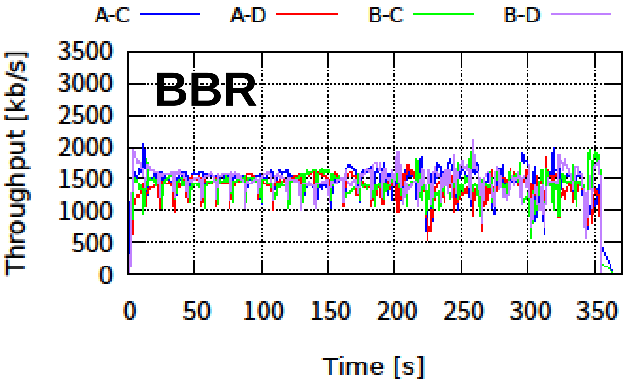
Default

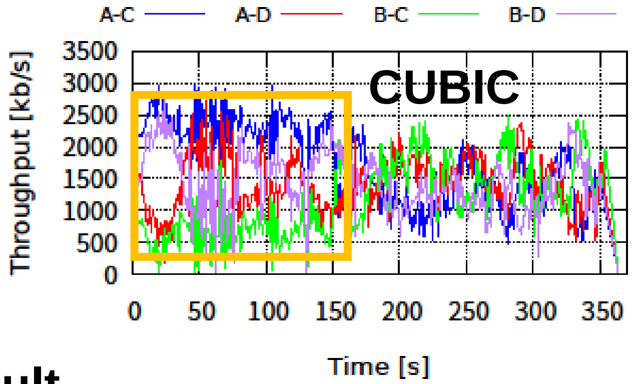


Bottleneck aware

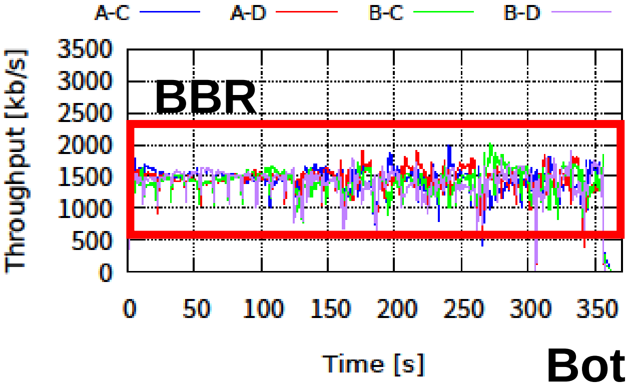
- The proposed method has good video quality regardless of congestion control
- Prevented degradation of video quality even with long delays

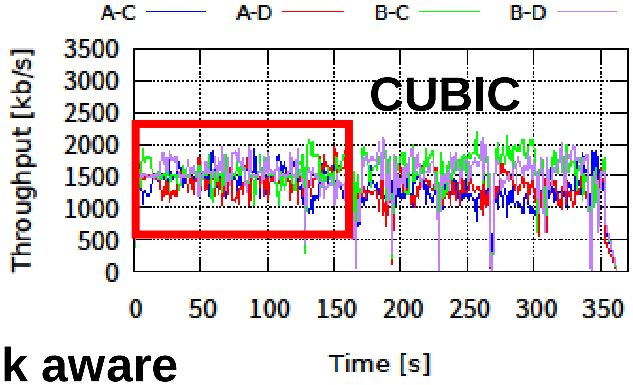
Compared Results (Fullmesh-2)#2





Default

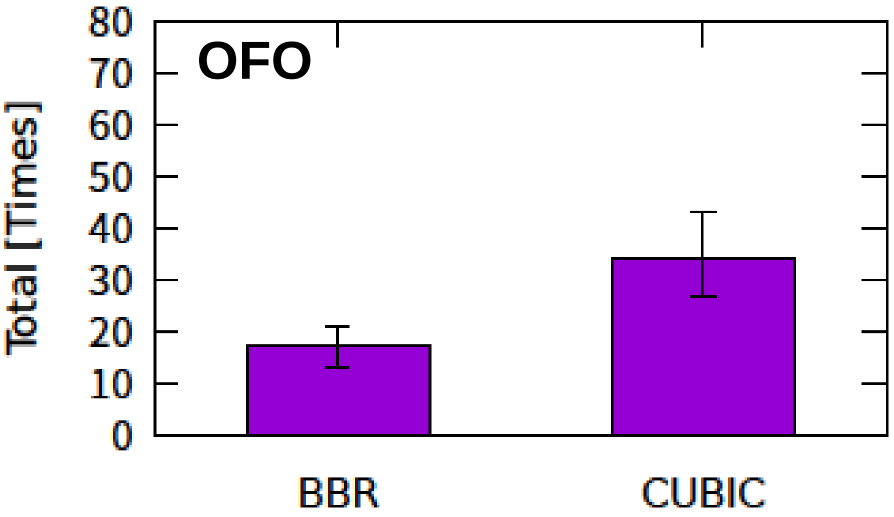




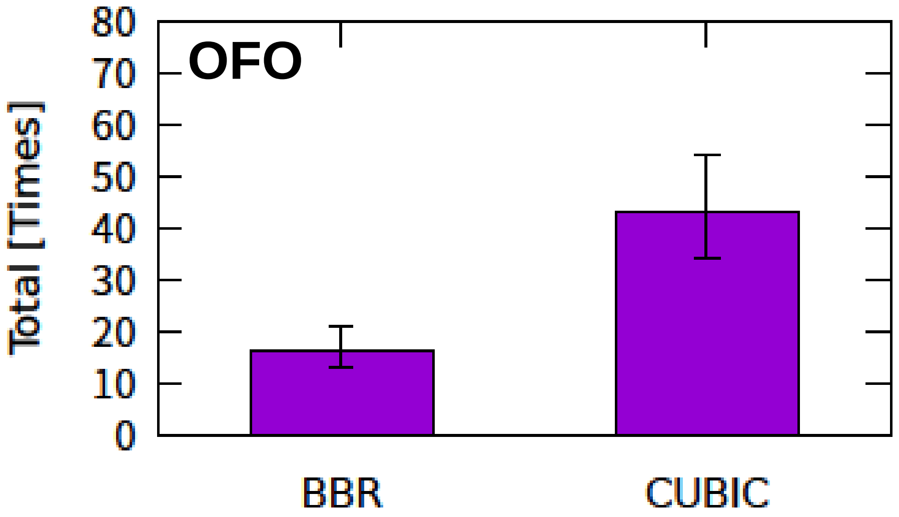
Bottleneck aware

- BBR results similar to Default scheduler
- CUBIC suppresses throughput fluctuations
- Buffer size limitation avoids over-transmission even with long delays

Compared Results (Fullmesh-2)#3



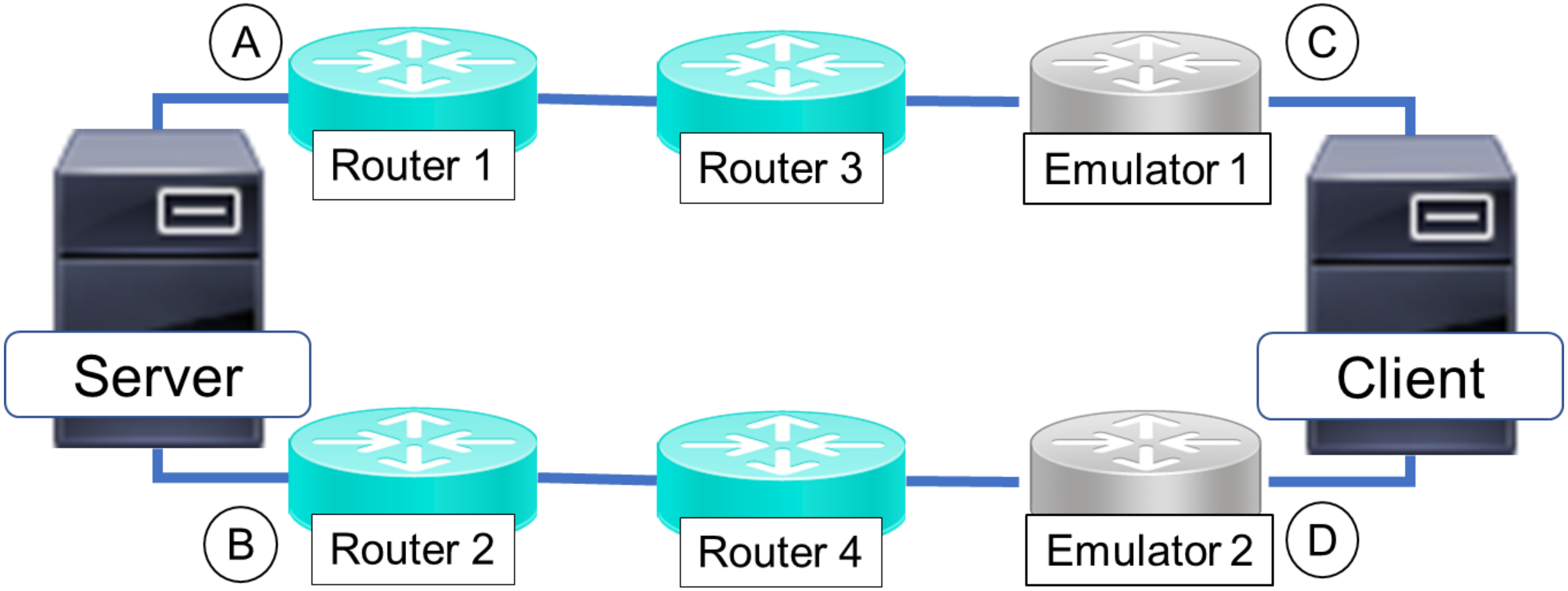
Default



Bottleneck aware

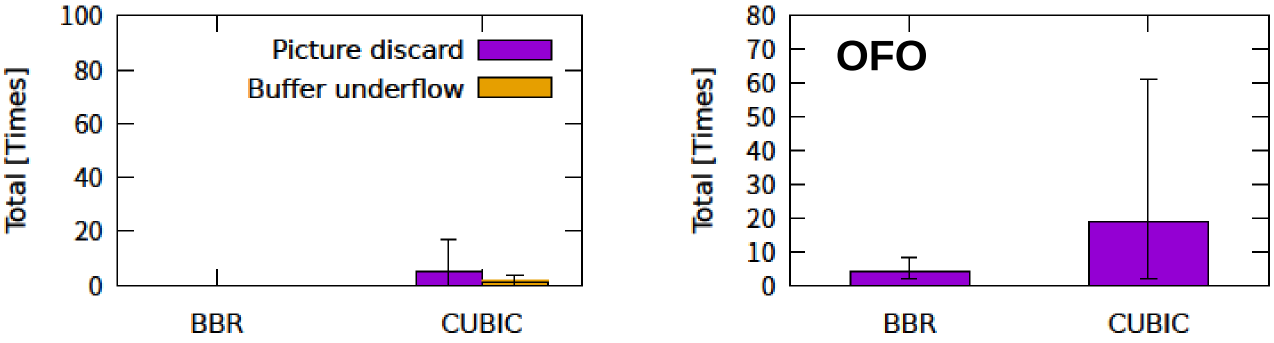
- The proposed method maintains the same network quality as the Default scheduler

Experimental Environment (Parallel)

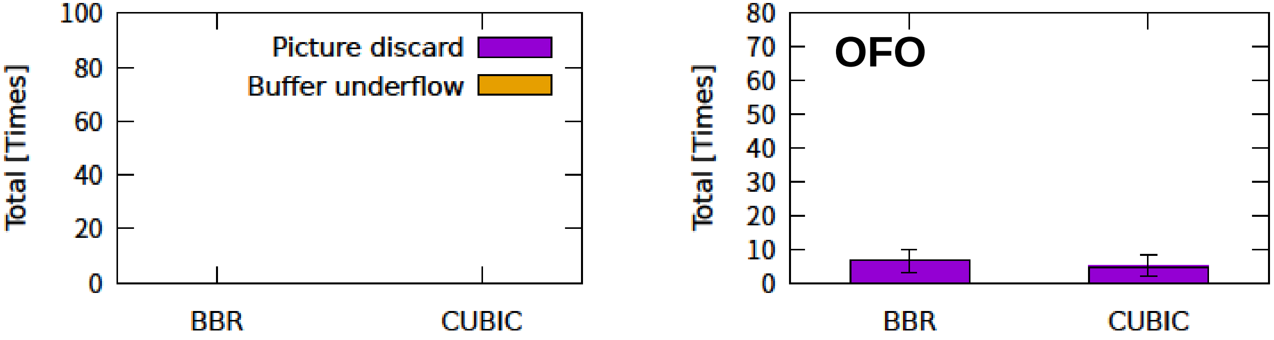


- Testbed experiment
  - Can only connect with A-C and B-D
- All videos, etc. used are the same as in the Fullmesh environment
- Experimental scenario
  - BW: 3Mb/s, Loss rate: 0.1%, **RTT: 60ms**
- Scenario was conducted 5 times

Compared Results (Parallel)



Default



Bottleneck aware

- Video quality is good in both schedulers
- Same level of network quality impact
- The proposed method performs better than the Default Scheduler with and without shared bottleneck links

Conclusion and Future Work

- MPTCP video streaming over shared bottleneck link
- Default scheduler degrades video quality when shared bottleneck links exist and subflows compete for bandwidth when using CUBIC
- The proposed method avoids over-transmitting to the shared bottleneck link and achieves good video quality by limiting the transmission buffer size
- Select subflows according to environment with or without shared bottleneck links
- Future work includes confirming the stability of the proposed method and further improving it through experiments in real environments.