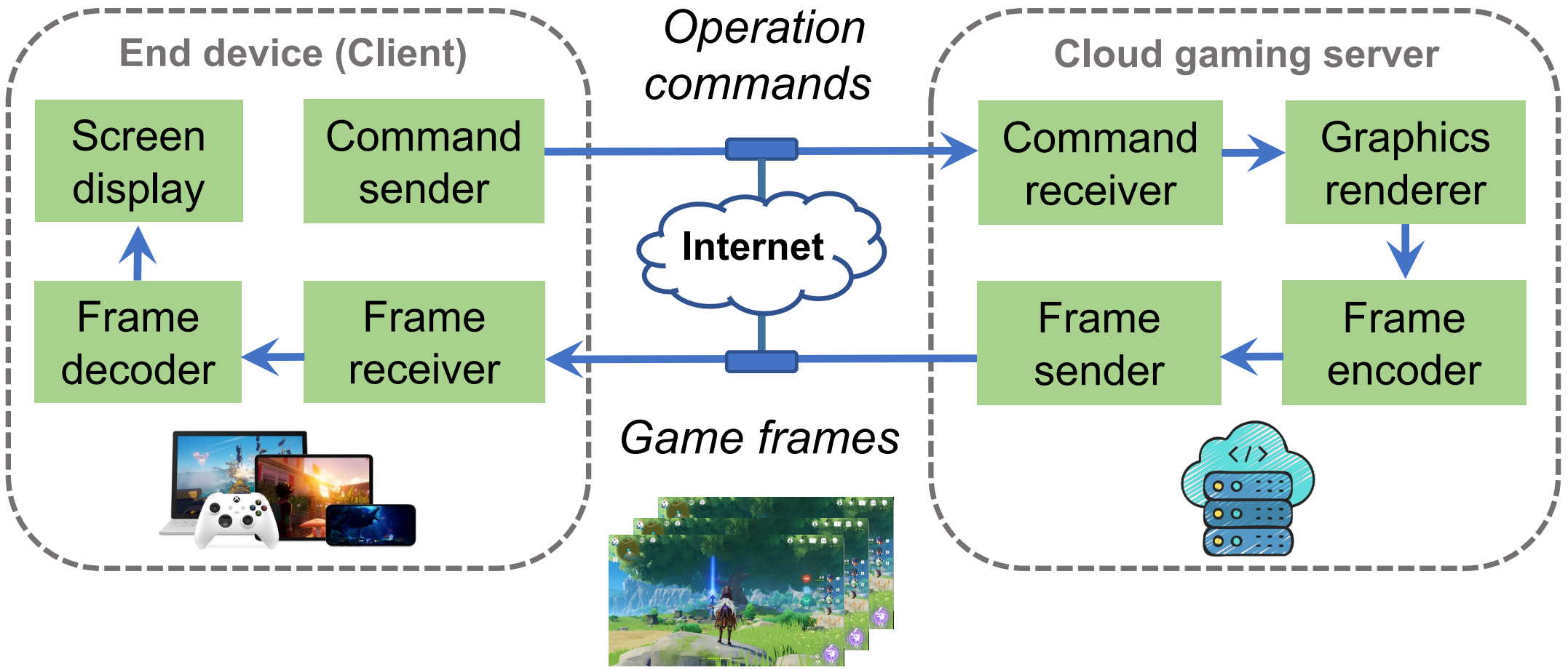
Pudica: Toward Near-Zero Queuing Delay in Congestion Control for Cloud Gaming

Shibo Wang, *Shusen Yang, Xiao Kong, Chenglei Wu, Longwei Jiang, Chenren Xu, Cong Zhao, Xuesong Yang, Jianjun Xiao, Xin Liu, Changxi Zheng, Jing Wang, Honghao Liu*

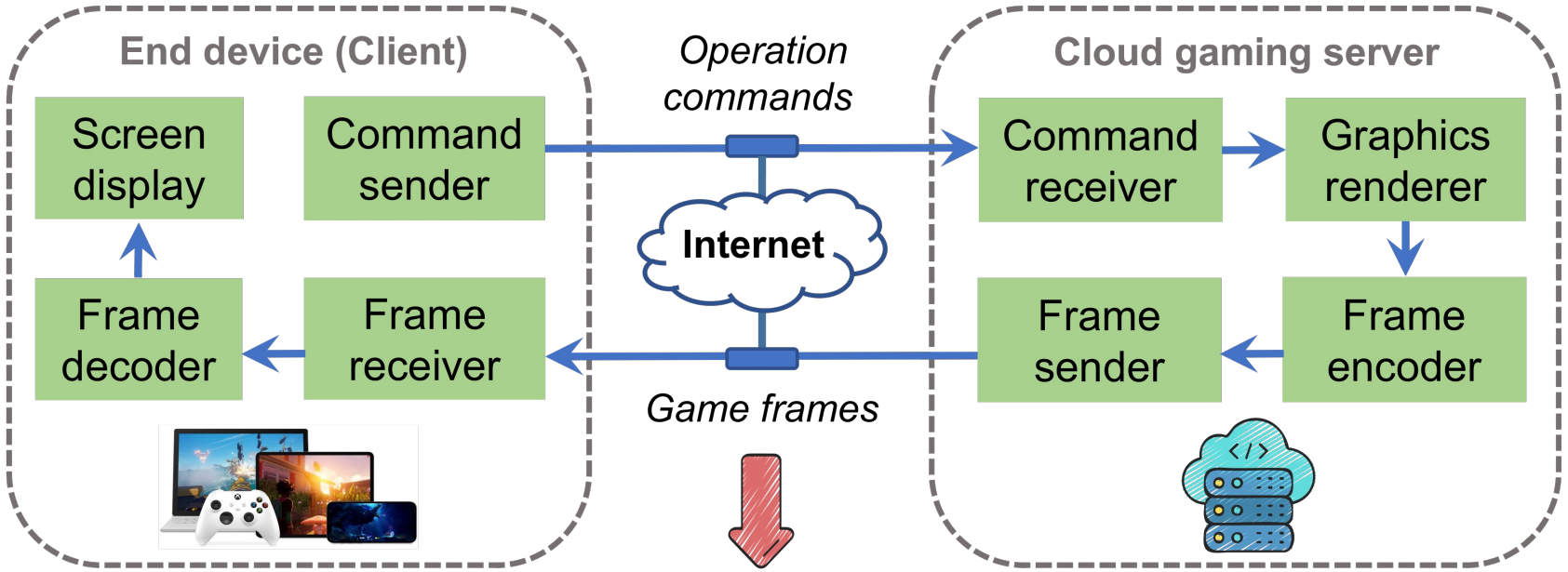
Xi'an Jiaotong University, Tencent Inc., Peking University,
Bonree, Tencent America, Columbia University

Cloud gaming has already gained world-wide popularity while still under rapid growth

Cloud Gaming System

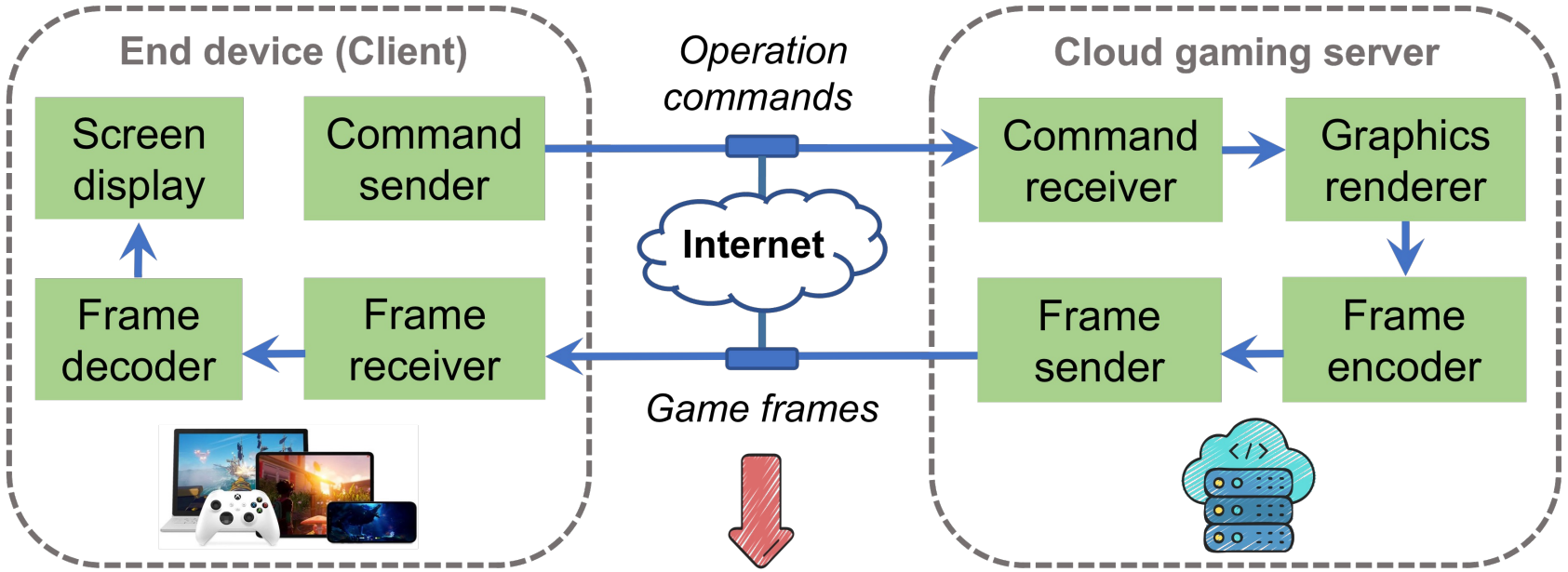


Cloud Gaming System



Consistently demand a low transmission delay for frames

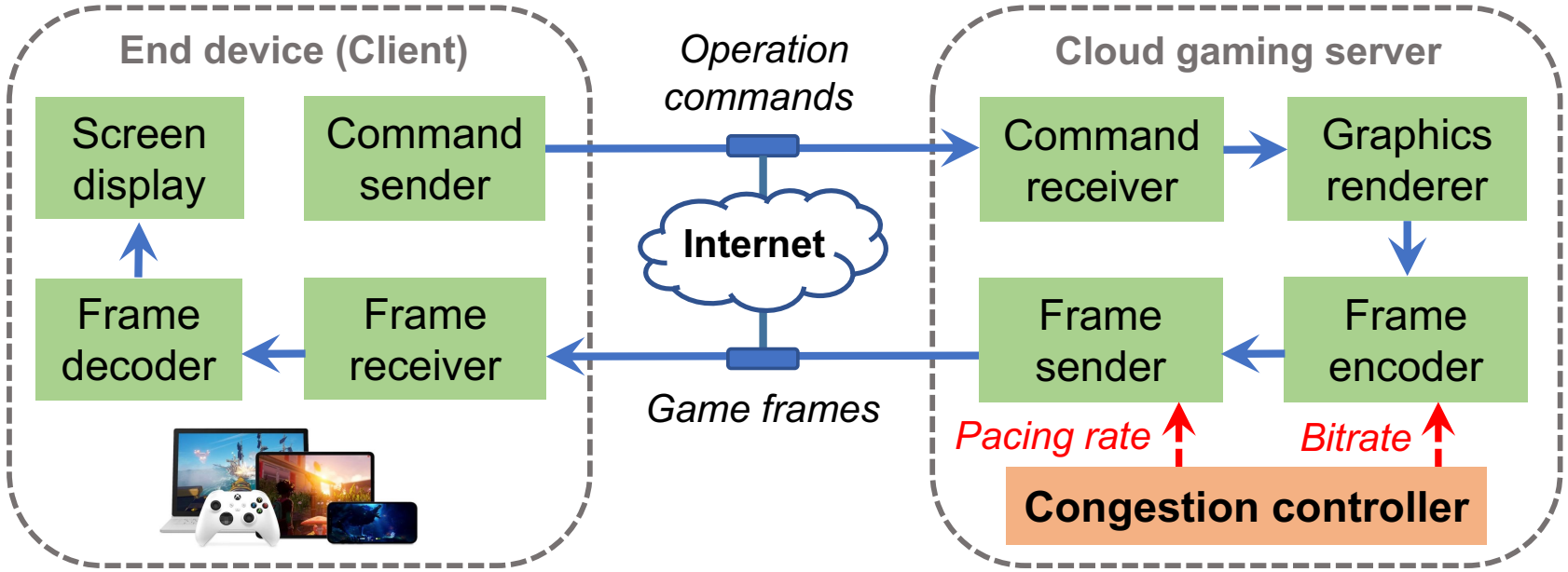
Cloud Gaming System



Consistently demand a low transmission delay for frames

A cloud gaming system needs a carefully designed congestion control (CC) algorithm to manage frame delay.

Congestion Control (CC) for Cloud Gaming

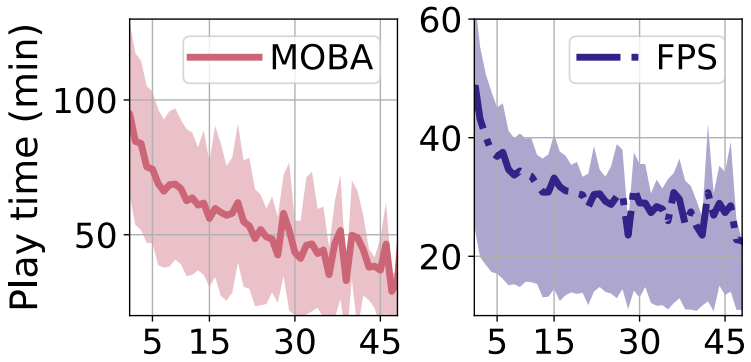


- Cloud gaming CC operates on both **application** and **transport** layers, which controls two factors on the fly, namely the ***frame bitrate*** and ***packet sending pace***.

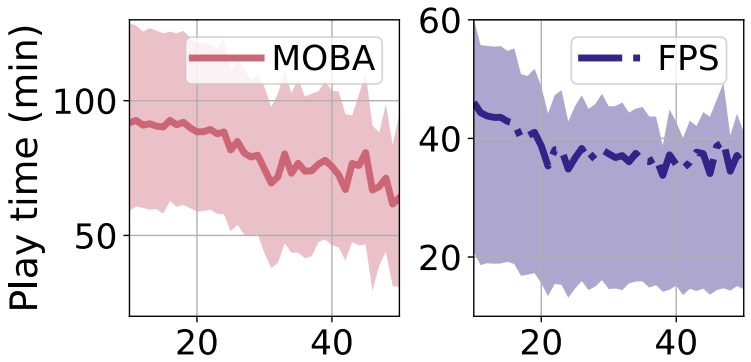
What Cloud Gaming Players Cares?

What Cloud Gaming Players Cares?

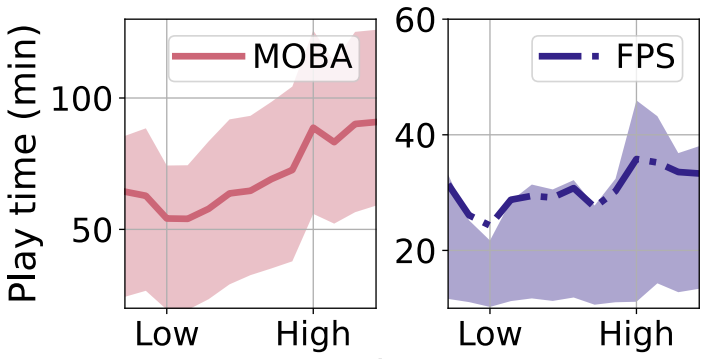
- **We conducted case studies on two widely-played games[1] to examine the metrics that drive user engagement in cloud gaming:**
  - *As the **stall rate** (frame ratio of >100 ms) increases, the play time rapidly decreases.*
  - ***Frame delay** and **bitrate** also impact the play time, while not as significant as stall rate.*



***Stall rate** (unit: $\frac{1}{10000}$)*



***Avg. frame delay** (ms)*



Avg. frame bitrate

[1] *A multiplayer online battle arena (MOBA) game and a first-person shooting (FPS) game.*

Our Goal of Congestion Control

- **Ultra-low** (or **nearly zero**) **queuing delay** at the bottleneck to minimize the frame delay and stall rate.

Our Goal of Congestion Control

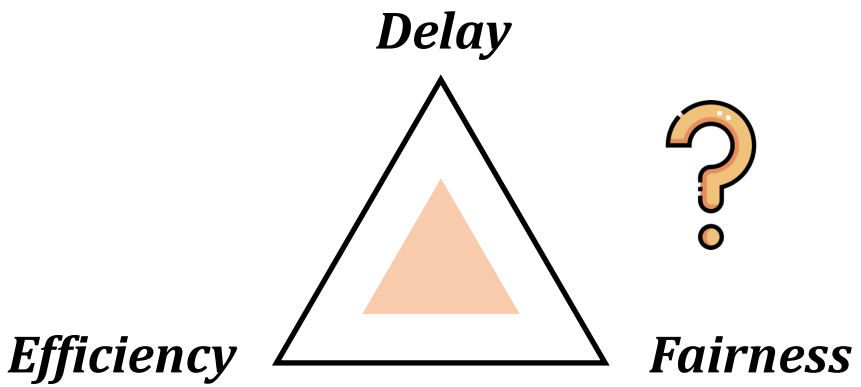
- **Ultra-low** (or **nearly zero**) **queuing delay** at the bottleneck to minimize the frame delay and stall rate.
- **Efficient link utilization** to support high bitrate.

Our Goal of Congestion Control

- **Ultra-low** (or **nearly zero**) **queuing delay** at the bottleneck to minimize the frame delay and stall rate.
- **Efficient link utilization** to support high bitrate.
- Decent fairness among homogeneous flows.

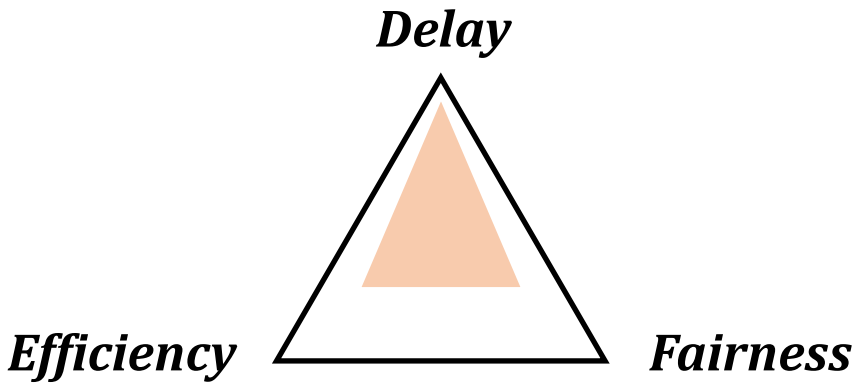
Our Goal of Congestion Control

- **Ultra-low** (or **nearly zero**) **queuing delay** at the bottleneck to minimize the frame delay and stall rate.
- **Efficient link utilization** to support high bitrate.
- Decent fairness among homogeneous flows.



Our Goal of Congestion Control

- **Ultra-low** (or **nearly zero**) **queuing delay** at the bottleneck to minimize the frame delay and stall rate.
- **Efficient link utilization** to support high bitrate.
- Decent fairness among homogeneous flows.



Frame delay has a higher priority than efficiency or fairness.

Why Existing Congestion Control Methods Fail?

- **Existing solutions fall short in achieving consistent low frame delay.**

Algorithm	Avg. frame delay	95%ile frame delay	Stall rate (>100 *ms*)
Copa [*NSDI'18*]	**39.8 *ms***	**114 *ms***	**3.2%**
Salsify[1] [*NSDI'18*]	**66.7 *ms***	**186 *ms***	**6.3%**
SQP	**109.4 *ms***	**287 *ms***	**3.6%**
Pudica	**23.2 *ms***	**38 *ms***	**0.7%**

[1]*Salsify refers to its frame size control part solely.*

Why Existing Congestion Control Methods Fail?

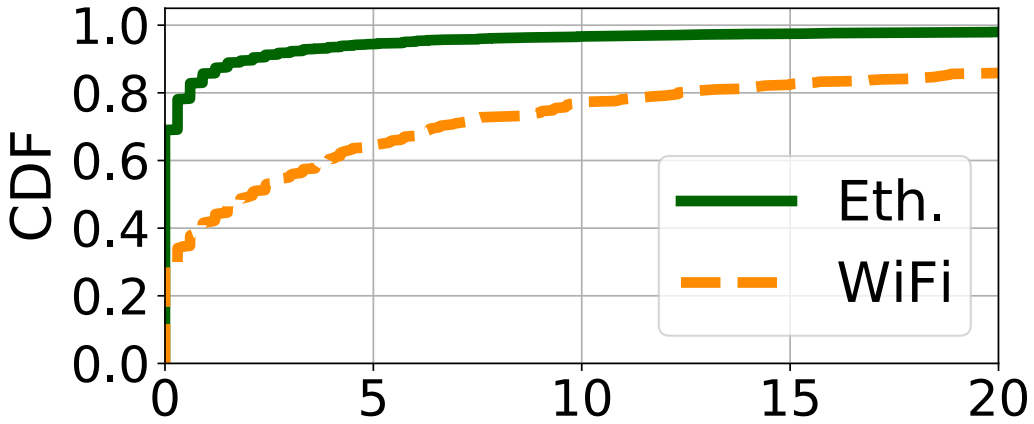
- **Periodical self-induced queue buildups for effective network probing.**

Algorithm	Indicator
Copa	RTTstanding
Salsify	Packet inter-arrival time
GCC	Delay gradient
SQP	Frame transport bandwidth

When the queue is nearly empty, these indicators fail to provide precise signals.

Why Existing Congestion Control Methods Fail?

- **Slow adaptation to abrupt decreases in available bandwidth.**

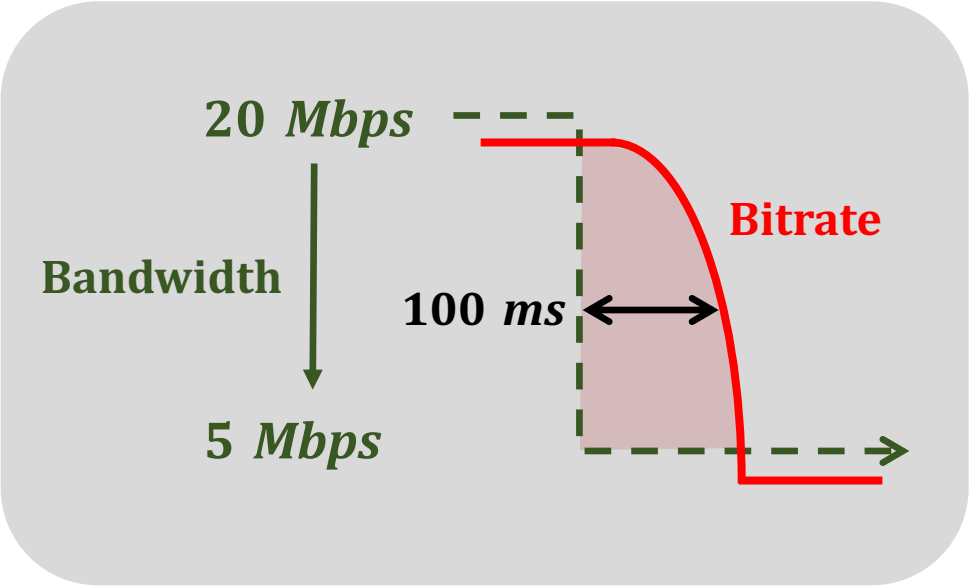


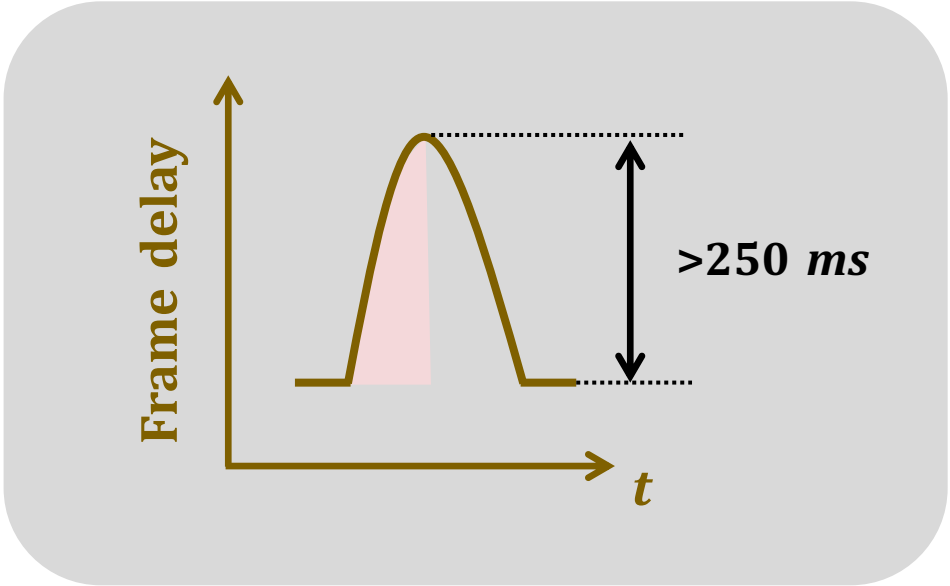
Num. of bandwidth reductions (≥ 50%) per minute

Internet users frequently encounter significant reductions in available bandwidth.

Why Existing Congestion Control Methods Fail?

- **Slow adaptation to abrupt decreases in available bandwidth.**





A delayed response of only 100 *ms*

A frame delay spike exceeding 250 *ms*

Pudica Design: Requirement and Overview

- **Basic requirements for achieving near-zero queuing:**

- Convergence to efficiency (i.e., high link utilization) and fairness **without resorting to overshoot-based network probing**.

- **Prompt reaction** to abrupt decreases in available bandwidth.

Pudica Design: Requirement and Overview

- **Basic requirements for achieving near-zero queuing:**

- Convergence to efficiency (i.e., high link utilization) and fairness **without resorting to overshoot-based network probing**.

- **Prompt reaction** to abrupt decreases in available bandwidth.

- **Solutions in Pudica:**

- Probing the **bandwidth utilization ratio (BUR)** while avoiding frame-level overshooting.

- Bitrate adjustment based on both ***smoothed BUR estimations*** and *more responsive* ***short-term BUR signals***.

Pudica Design: Network Probing and Estimation

- **What to probe?**

- Pudica probes the ***bandwidth utilization ratio (BUR)*** rather than the bandwidth per se.
  - BUR is the ratio of current bandwidth usage to the link capacity.

Pudica Design: Network Probing and Estimation

- **What to probe?**

- Pudica probes the ***bandwidth utilization ratio (BUR)*** rather than the bandwidth per se.
  - BUR is the ratio of current bandwidth usage to the link capacity.
  - BUR provides an indicator of the precise level of link utilization.

Pudica Design: Network Probing and Estimation

- **What to probe?**

- Pudica probes the ***bandwidth utilization ratio (BUR)*** rather than the bandwidth per se.
  - BUR is the ratio of current bandwidth usage to the link capacity.
  - BUR provides an indicator of the precise level of link utilization.
  - BUR has been leveraged by ECN-based CC methods for achieving high link utilization and low bottleneck queuing.

Pudica Design: Network Probing and Estimation

- **How to probe?**

Pudica Design: Network Probing and Estimation

- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** for each frame period.

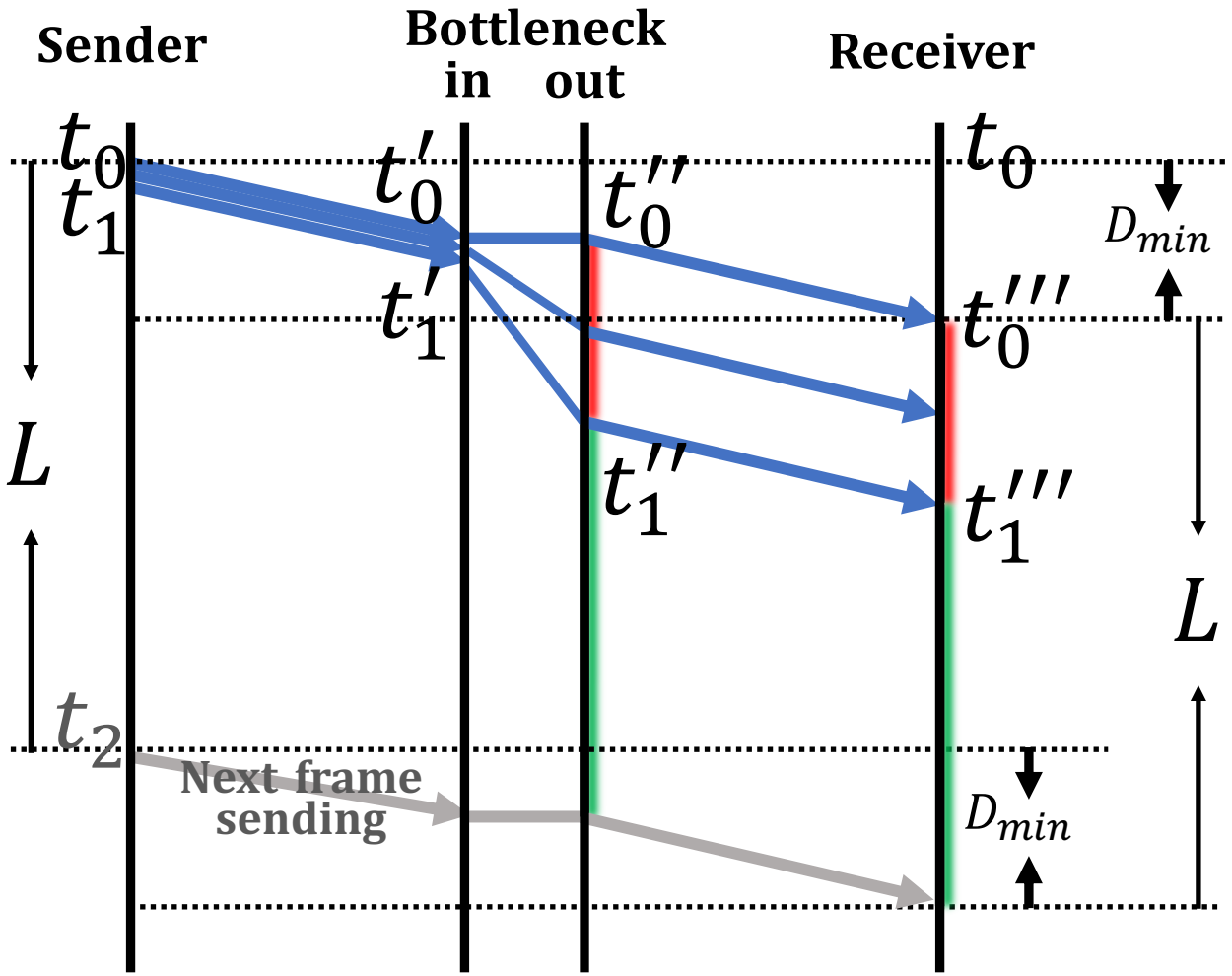
Pudica Design: Network Probing and Estimation

- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (= red shadow in right fig) for each frame period.

① **Assume w/o cross traffic:**

$$\widehat{BUR} = \frac{t_1'' - t_0'}{L} = \frac{t_1''' - t_0 - D_{min}}{L}$$



Pudica Design: Network Probing and Estimation

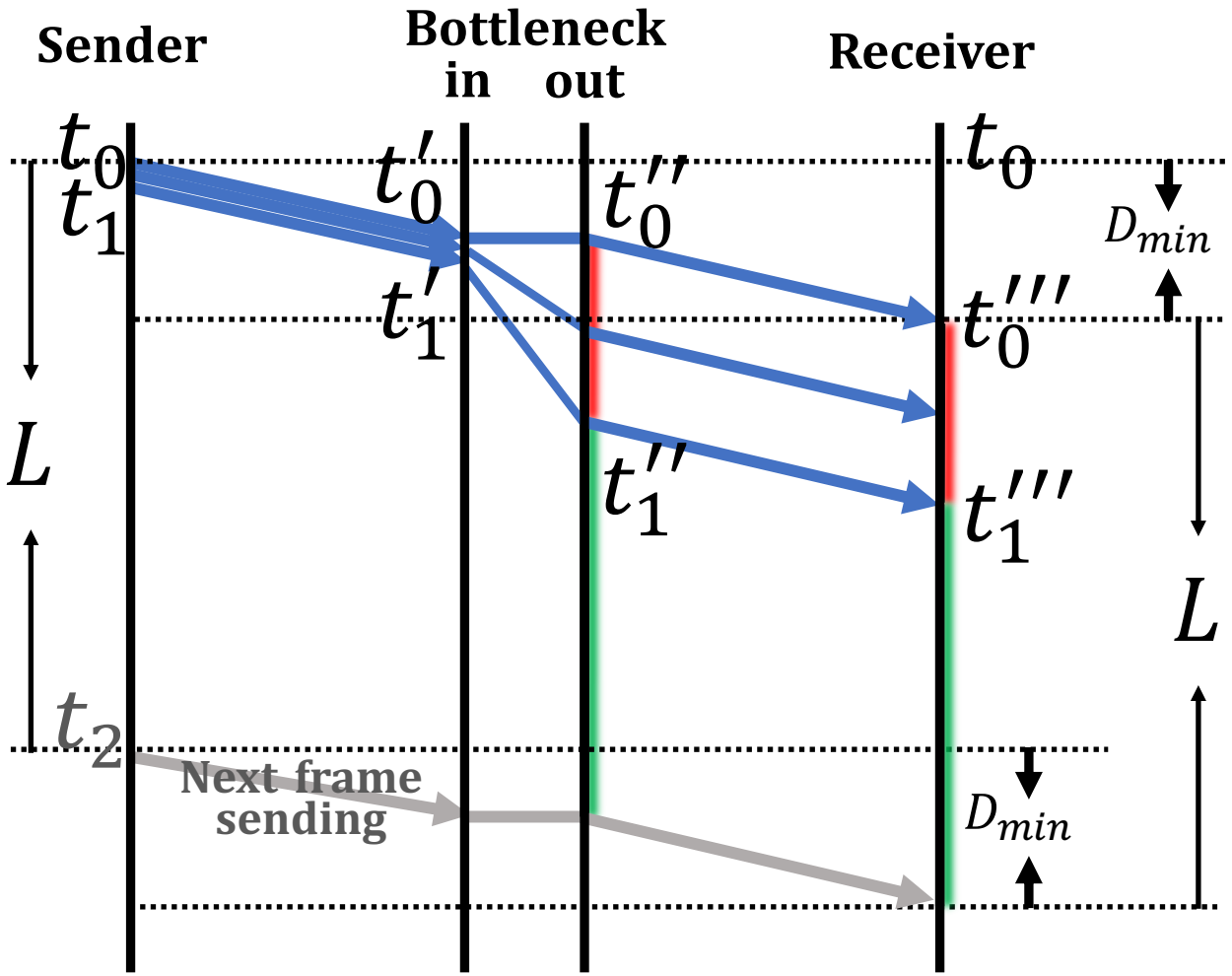
- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (= red shadow in right fig) for each frame period.

① **Assume w/o cross traffic:**

$$\widehat{BUR} = \frac{t_1'' - t_0'}{L} = \frac{t_1''' - t_0 - D_{min}}{L}$$

Frame-level queuing delay



Pudica Design: Network Probing and Estimation

- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (≥ red shadow in right fig) for each frame period.

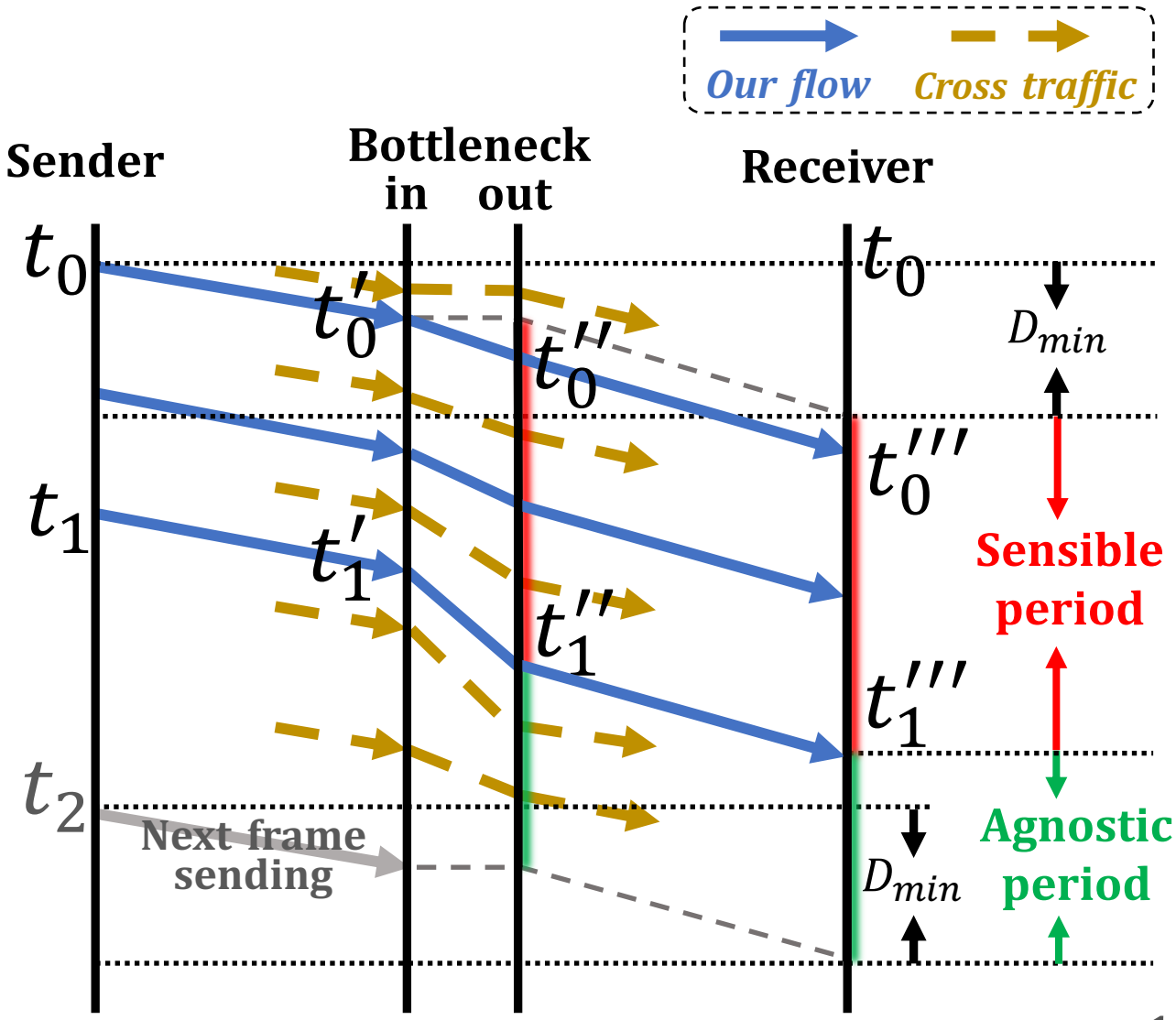
① **Assume w/o cross traffic:**

$$\widehat{BUR} = \frac{t_1'' - t_0'}{L} = \frac{t_1''' - t_0 - D_{min}}{L}$$

Frame-level queuing delay

② **Assume w/ cross traffic:**

$$\widehat{BUR} = \frac{t_1''' - t_0 - D_{min}}{L} + R_{Agnostic}$$



Pudica Design: Network Probing and Estimation

- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (= red shadow in right fig) for each frame period.

① **Assume w/o cross traffic:**

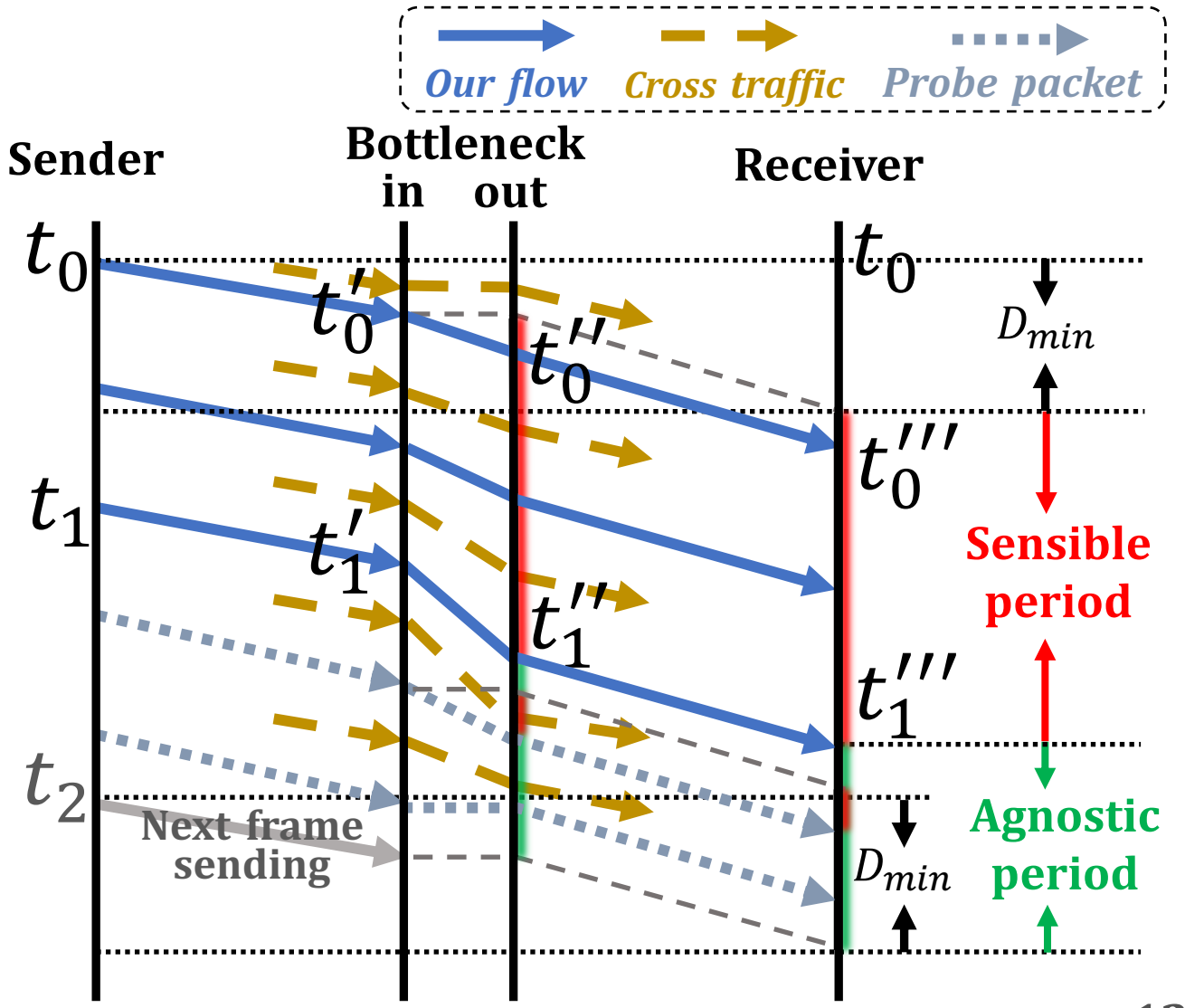
$$\widehat{BUR} = \frac{t_1''' - t_0'}{L} = \frac{t_1''' - t_0 - D_{min}}{L}$$

Frame-level queuing delay

② **Assume w/ cross traffic:**

$$\widehat{BUR} = \frac{t_1''' - t_0 - D_{min}}{L} + R_{Agnostic}$$

$$\boldsymbol{\widehat{BUR} = \frac{t_1''' - t_0 - D_{min}}{L} + R_{probe\_packet}}$$



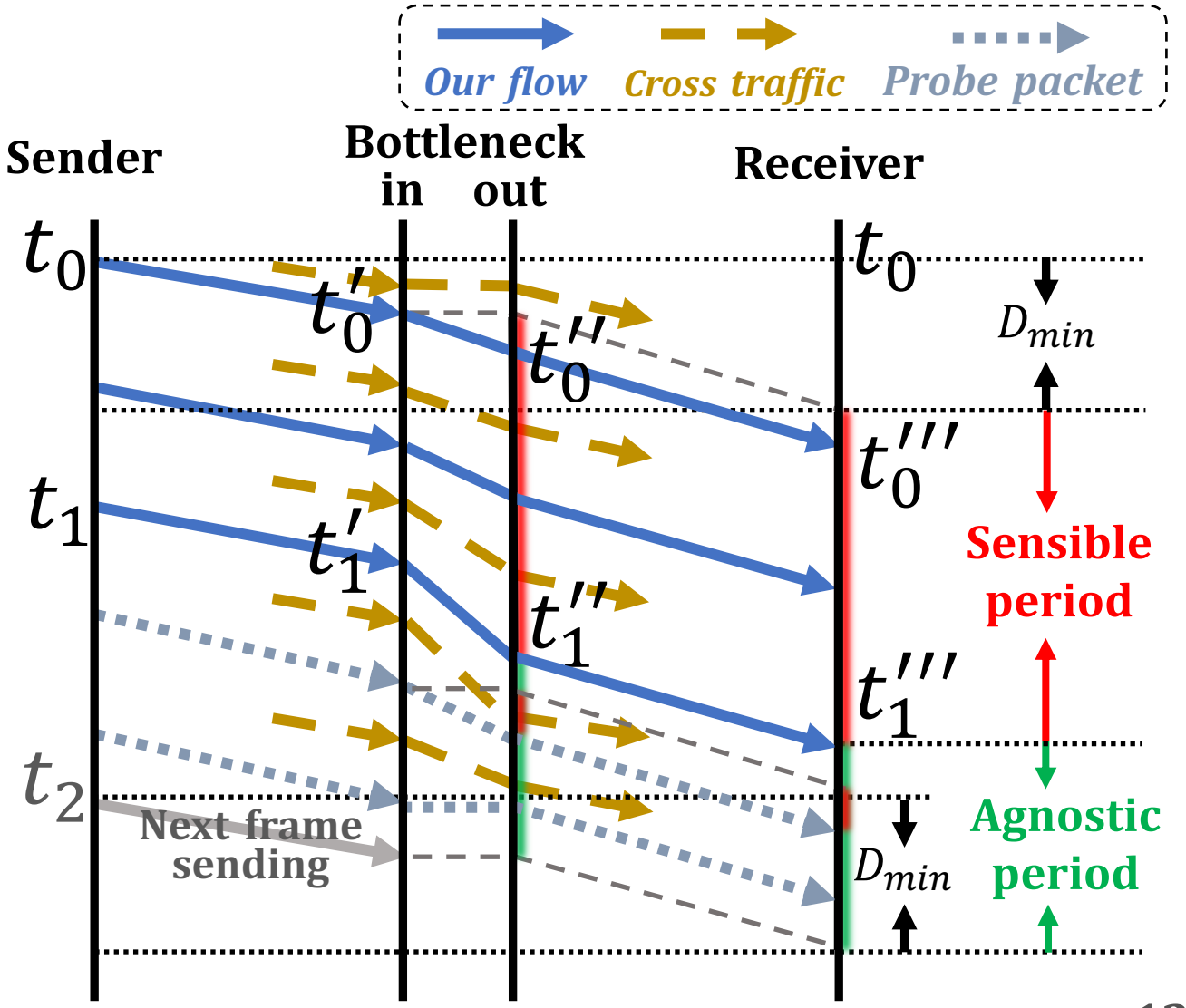
Pudica Design: Network Probing and Estimation

- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (= red shadow in right fig) for each frame period.

- **Pace adaptation scheme:**

- **Target:**
  - *Keep bursty in sensible period*
  - *Shrink the agnostic period*



Pudica Design: Network Probing and Estimation

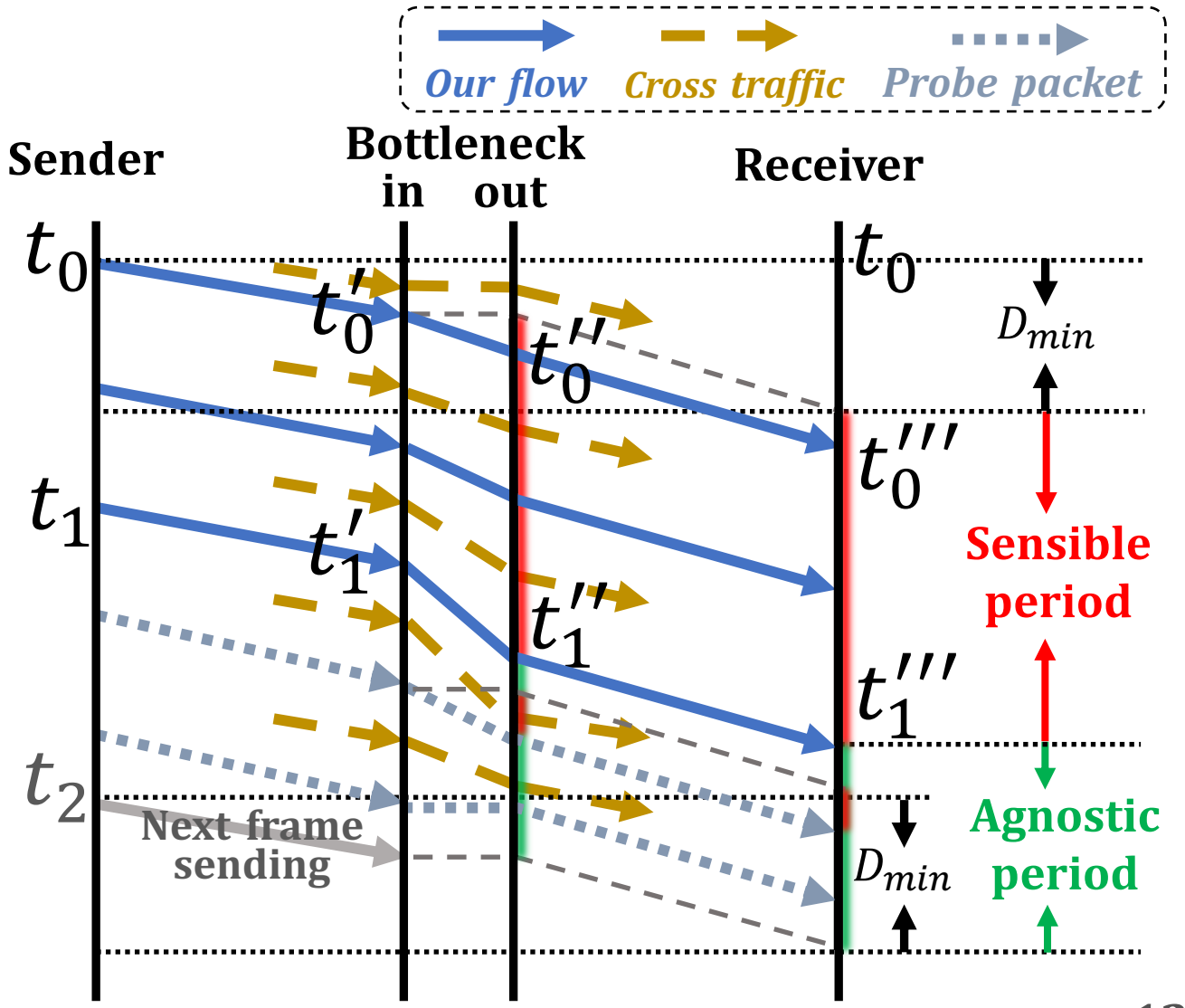
- **How to probe?**

- Estimates the BUR by probing the **bottleneck busy time** (= red shadow in right fig) for each frame period.

- **Pace adaptation scheme:**
- **Target:**
  - *Keep bursty in sensible period*
  - *Shrink the agnostic period*

$$\textbf{Pace multiplier} = \frac{\gamma_\rho}{\min(\widehat{BUR},\ 1)}$$

1.25



Pudica Design: Bitrate Adaptation over the Smoothed BUR

Pudica Design: Bitrate Adaptation over the Smoothed BUR

- Utilize **a window of historical BUR samples** to obtain the **smoothed BUR estimation.**

Pudica Design: Bitrate Adaptation over the Smoothed BUR

- Utilize **a window of historical BUR samples** to obtain the **smoothed BUR estimation.**
- **Control strategy:**

- When **smoothed BUR** $\leq \alpha$,

  multiplicative increase (MI) *for efficiency convergence*

- When **smoothed BUR** $> \alpha$,

  simultaneous additive increase (AI) and multiplicative decrease (MD) *for fairness convergence and the maintenance of near-zero queuing*

Pudica Design: Bitrate Adaptation over the Smoothed BUR

- Utilize **a window of historical BUR samples** to obtain the **smoothed BUR estimation.**
- **Control strategy:**

- When **smoothed BUR** $\leq \alpha$,

  multiplicative increase (MI) *for efficiency convergence*

- When **smoothed BUR** $> \alpha$,

  simultaneous additive increase (AI) and multiplicative decrease (MD)

  for fairness convergence and the maintenance of near-zero queuing

Pudica Design: Bitrate Adaptation over the Smoothed BUR

- Utilize **a window of historical BUR samples** to obtain the **smoothed BUR estimation.**
- **Control strategy:**

- When **smoothed BUR ≤ α,**

  multiplicative increase (MI) *for efficiency convergence*

- When **smoothed BUR > α,**

  simultaneous additive increase (AI) and multiplicative decrease (MD)
  for fairness convergence and the maintenance of near-zero queuing

For more details, please check out our paper.

Pudica Design: Bitrate Adaptation over the Short-Term BUR

- Smoothed BUR fails to respond promptly to abrupt bandwidth decreases.

Pudica Design: Bitrate Adaptation over the Short-Term BUR

- Smoothed BUR fails to respond promptly to abrupt bandwidth decreases.
- **Control strategy:**

- When the BUR of **recent one frame** exceeds one,

  Temporary and slight bitrate fallback: $bitrate \leftarrow bitrate \times (1 - \xi)$

- When the BURs of **consecutive three frames** exceed one,

  Active queue draining: $bitrate \leftarrow (\alpha \times rate_{recv} - rate_{draining})$

Pudica Design: Bitrate Adaptation over the Short-Term BUR

- Smoothed BUR fails to respond promptly to abrupt bandwidth decreases.
- **Control strategy:**

- When the BUR of **recent one frame** exceeds one,

  Temporary and slight bitrate fallback: $bitrate \leftarrow bitrate \times (1 - \xi)$

- When the BURs of **consecutive three frames** exceed one,

  Active queue draining: $bitrate \leftarrow (\alpha \times rate_{recv} - rate_{draining})$

- When the BUR of **recent one frame** recovers from the queue-draining phase,

  One-step bitrate recovery: $bitrate \leftarrow rate_{recv}$

Pudica Design: Bitrate Adaptation over the Short-Term BUR

- Smoothed BUR fails to respond promptly to abrupt bandwidth decreases.
- **Control strategy:** ⟸ ***Next delay***

- When the BUR of **recent one frame** exceeds one,

  Temporary and slight bitrate fallback: $bitrate \leftarrow bitrate \times (1 - \xi)$

- When the BURs of **consecutive three frames** exceed one,

  Active queue draining: $bitrate \leftarrow (\alpha \times rate_{recv} - rate_{draining})$

- When the BUR of **recent one frame** recovers from the queue-draining phase,

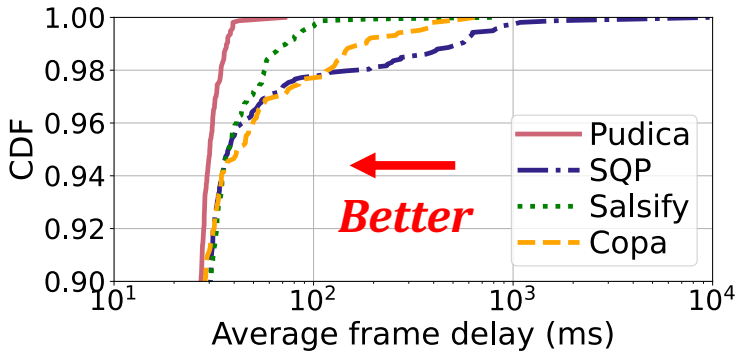
  One-step bitrate recovery: $bitrate \leftarrow rate_{recv}$

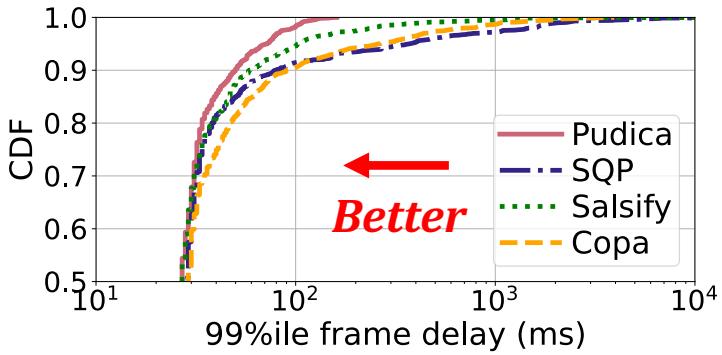
Evaluation: Methodology

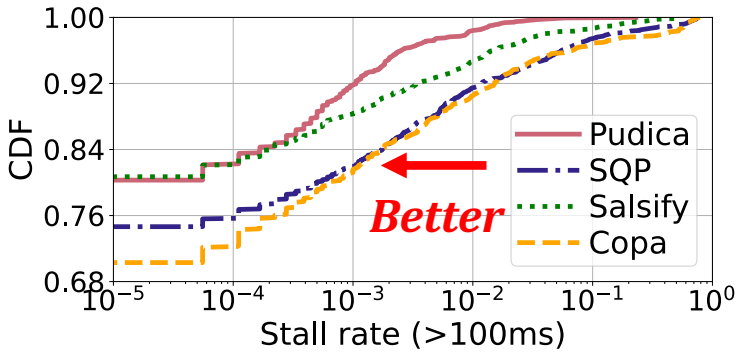
- Deploy four CC methods, including **Pudica, Salsify, Copa, and SQP**, on **Tencent START** cloud gaming platform for large-scale A/B tests.
- The evaluation involved more than **57,000** gaming sessions across **15** cities, **two** network types (**Ethernet and WiFi**), and **three** ISPs over **five** weeks.
- We set the frame rate as **60** and the maximal bitrate as **50 Mbps** for all algorithms.
- **Metrics** (per gaming session)**:**
  - Average, 95%ile, and 99%ile round-trip **frame delay.**
  - **Stall rates** for frame delays exceeding 100 *ms* and 200 *ms.*
  - **Average bitrate** and **valid bitrate** (i.e., average bitrate of frames with delay<50 *ms*).

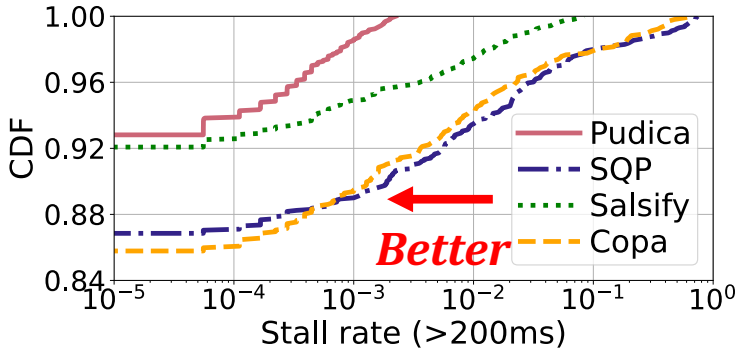
Evaluation: System-Level Performance at Scale

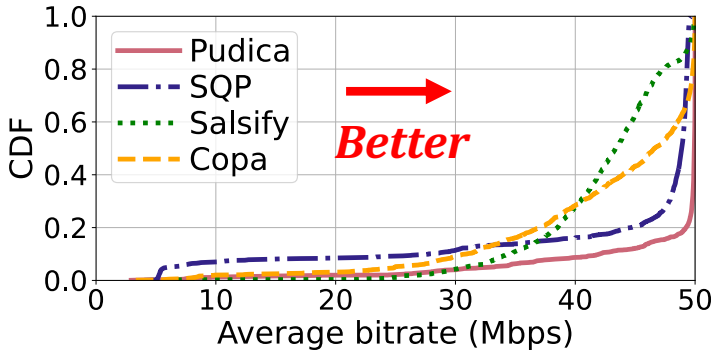
- For ***Ethernet*** networks, Pudica **(1)** reduces the average and 99%ile **frame delay** by **1.5×** and **3.2×,** respectively; **(2)** reduces the **stall rate** of 100 *ms* and 200 *ms* by **16.3×** and **22.5×**, respectively; **(3)** achieves a slight **frame bitrate** enhancement.

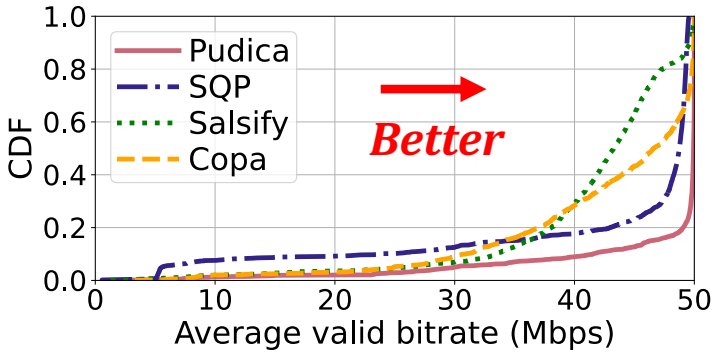






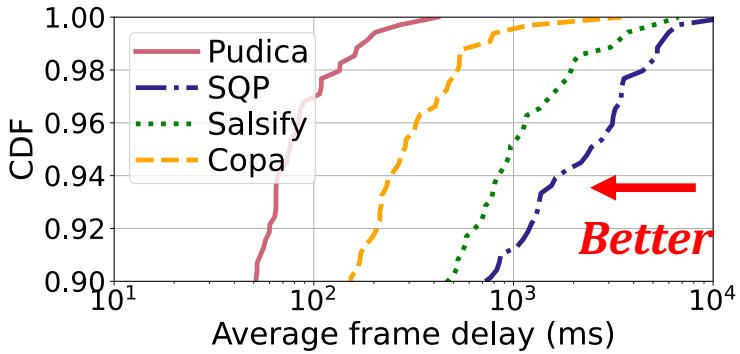


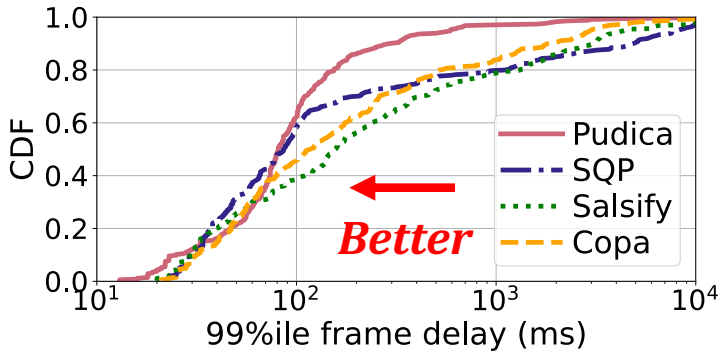


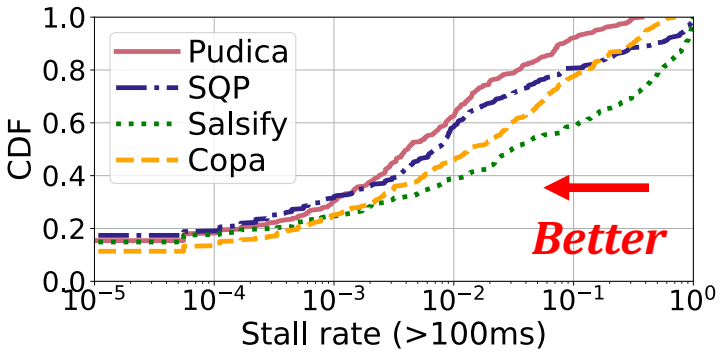


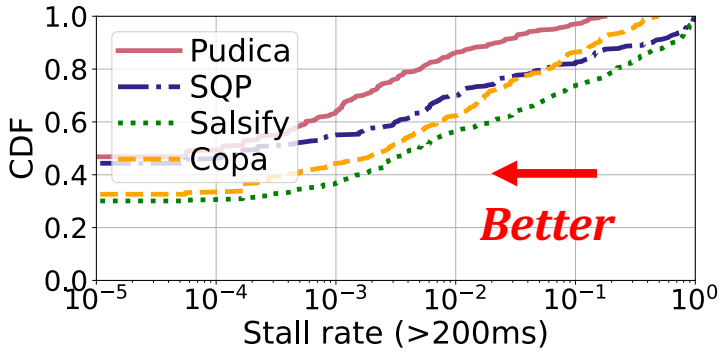
Evaluation: System-Level Performance at Scale

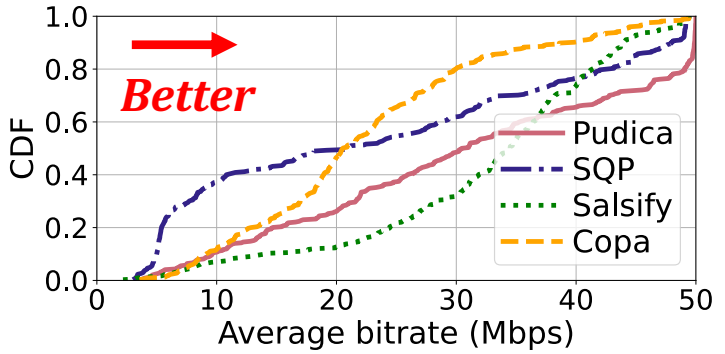
- For ***WiFi*** networks, Pudica **(1)** reduces the average and 99%ile **frame delay** by **5.7×** and **5.5×**, respectively; **(2)** reduces the **stall rate** of 100 ms and 200 ms by **5.5×** and **12.1×**, respectively; **(3)** achieve the comparable **frame bitrate.**

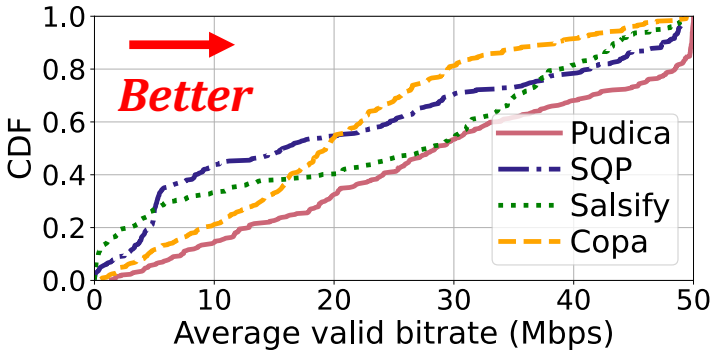






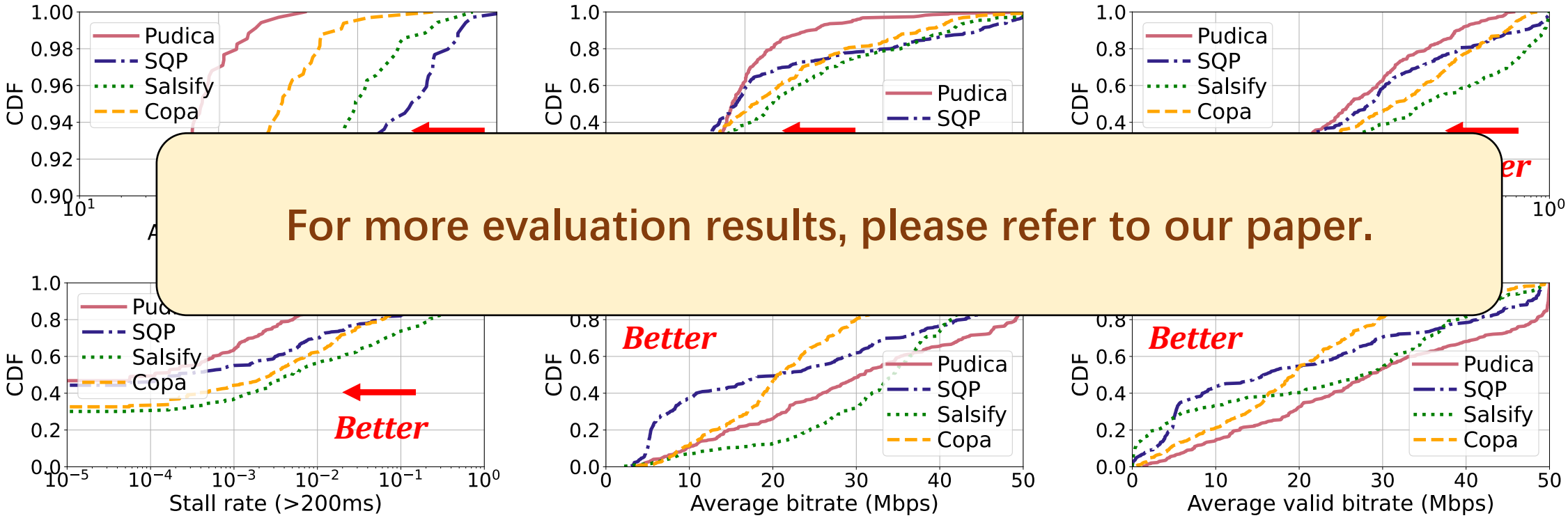






Evaluation: System-Level Performance at Scale

- For ***WiFi*** networks, Pudica **(1)** reduces the average and 99%ile **frame delay** by **5.7×** and **5.5×**, respectively; **(2)** reduces the **stall rate** of 100 ms and 200 ms by **5.5×** and **12.1×**, respectively; **(3)** achieve the comparable **frame bitrate.**



For more evaluation results, please refer to our paper.

Summary

- **We present *Pudica*, an Internet congestion control algorithm designed for cloud gaming:**

- ***Pudica*** proposes a BUR (i.e., *bandwidth utilization ratio*) probing approach and a holistic BUR-based control framework.
- ***Pudica*** achieves the convergence to efficiency and fairness under the constraint of near-empty bottleneck queues, and the agile adaptation to abrupt bandwidth decreases.
- By conducting large-scale A/B tests on Tencent START cloud gaming services, ***Pudica*** considerably reduces the frame delay and stall rate while preserving high bitrate.