AUGUR: Practical Mobile Multipath Transport Service for Low Tail Latency in Real-Time Streaming

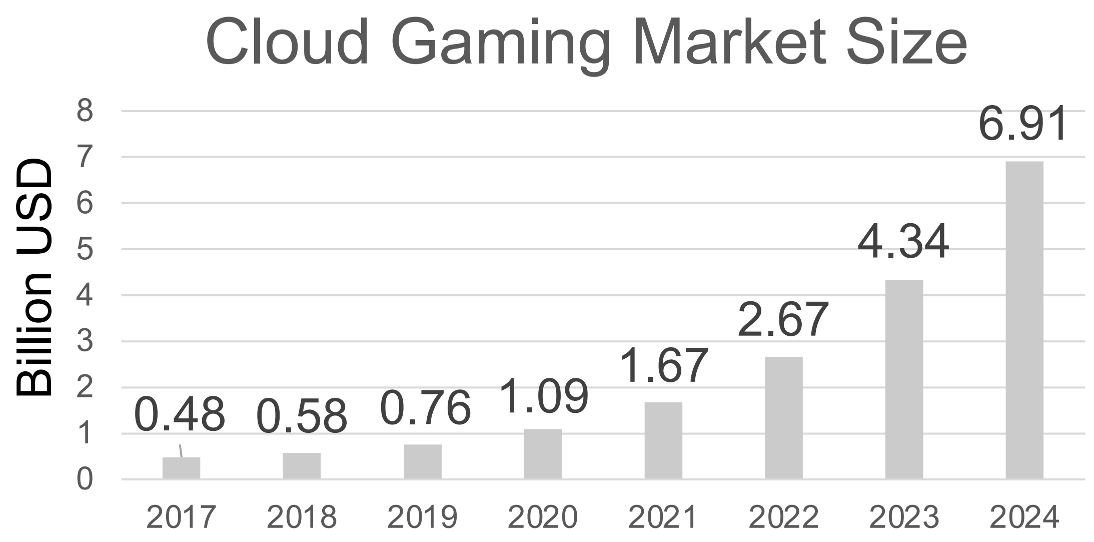
Yuhan Zh☺u, Tingfeng Wang, Liying Wang, Nian Wen, Rui Han, Jing Wang, Chenglei Wu, Jiafeng Chen, Longwei Jiang, Shibo Wang, Honghao Liu, Chenren Xu

USENIX NSDI 2024, SANTA CLARA, USA

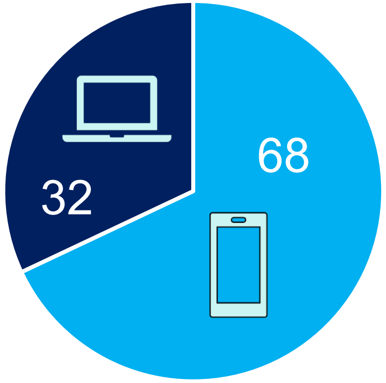
北京大學 PEKING UNIVERSITY

Tencent 腾讯

Interactive Mobile Real-Time Streaming: A Booming Market

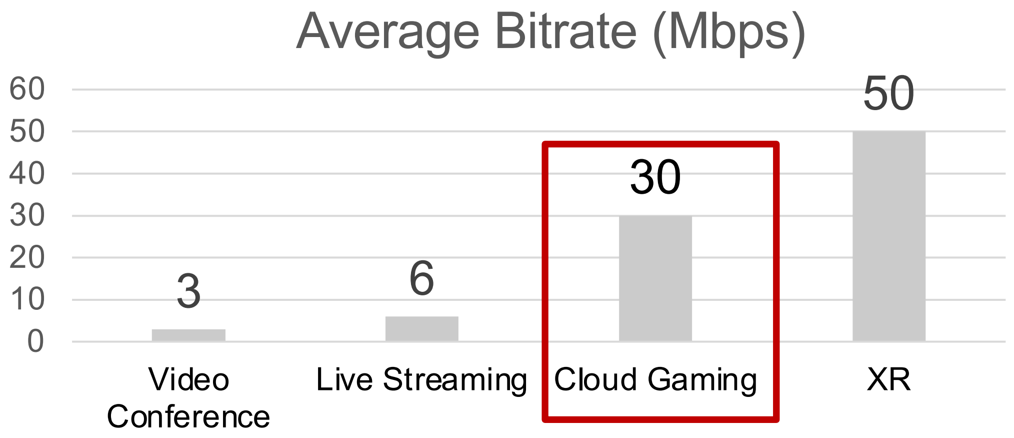


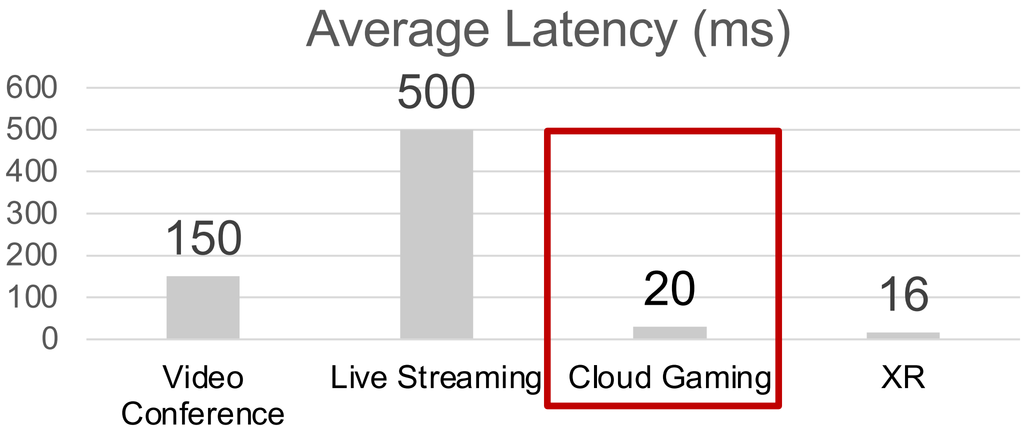
Source: Statista Market Insights



User with mobile devices are the majority in our cloud gaming service

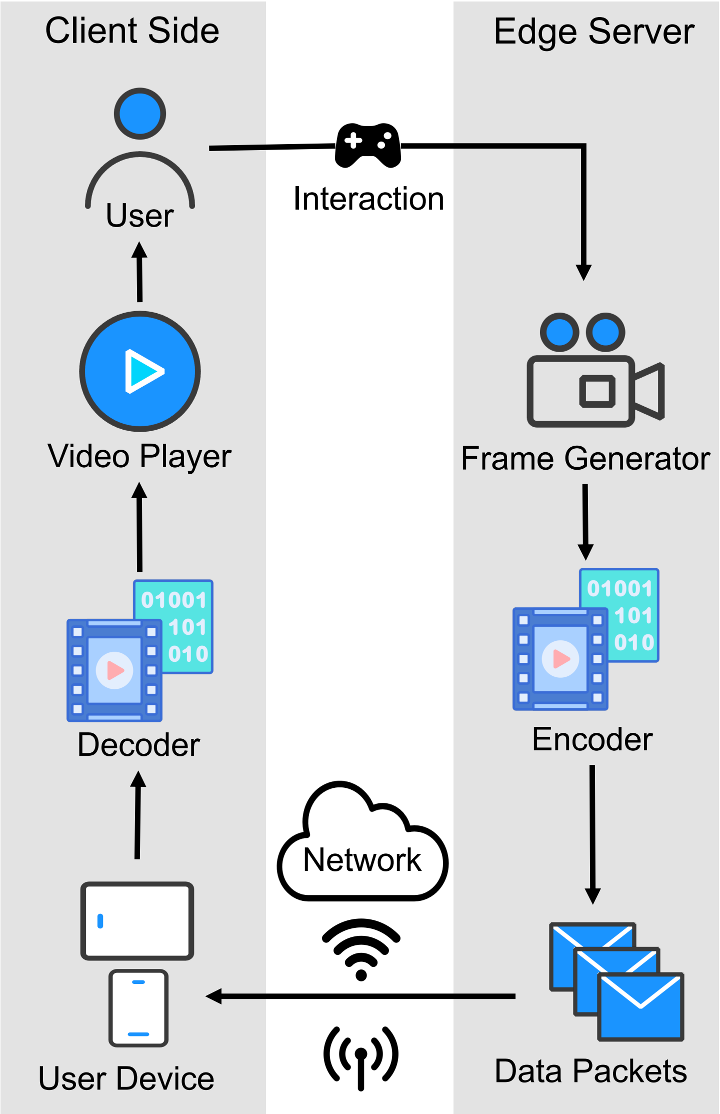
Interactive Mobile Real-Time Streaming: A New Challenge





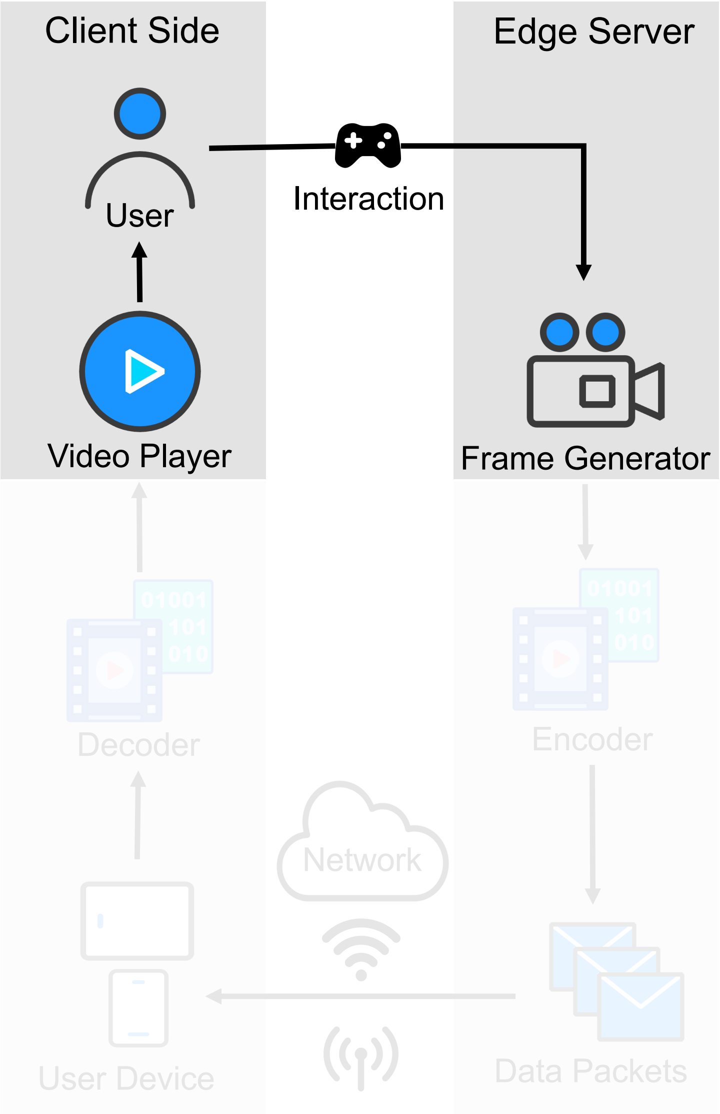
A brand new challenge for mobile network transport

How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

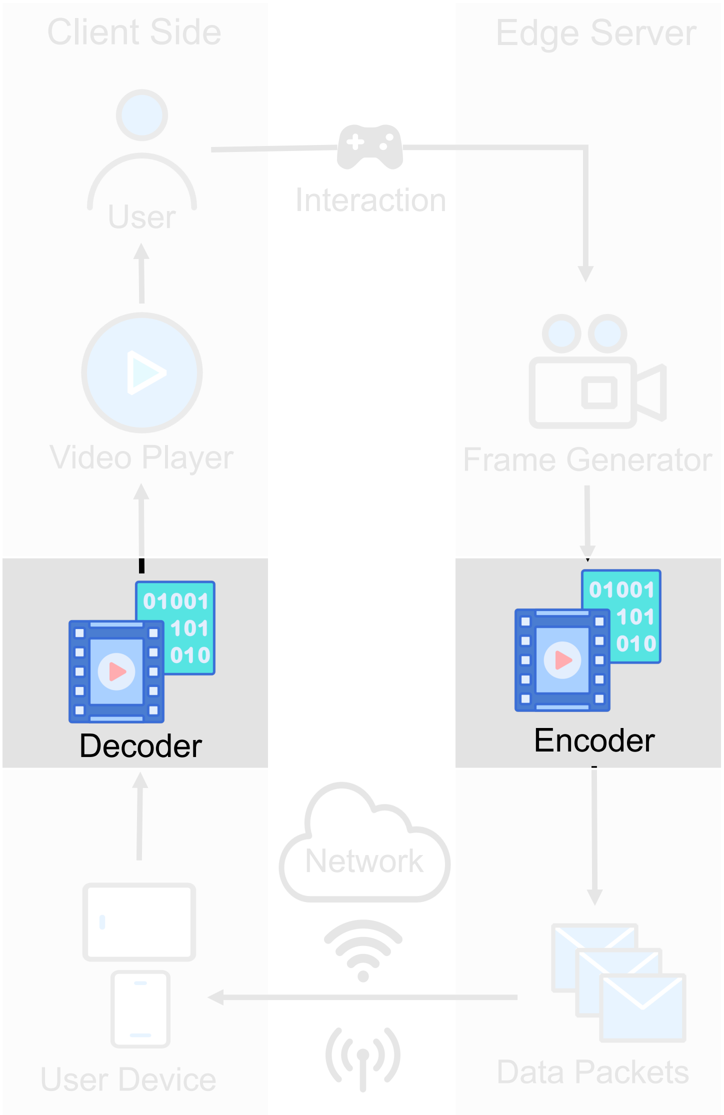
How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

Video Frame Generation

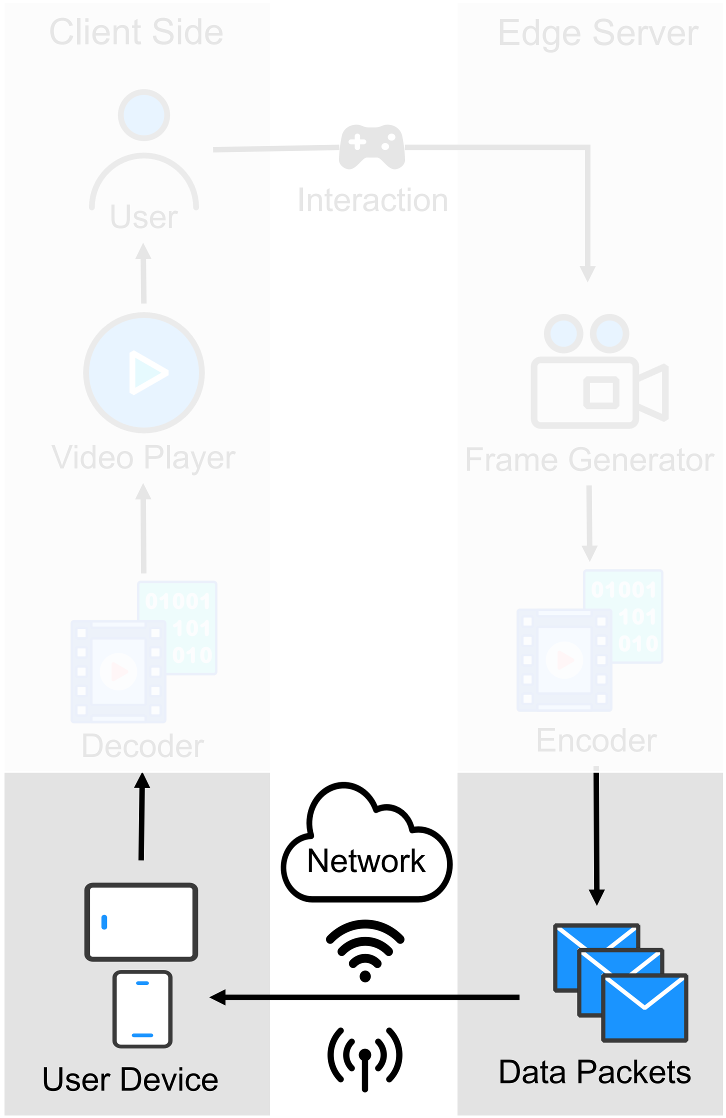
How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

Encoder & Decoder (< 10 ms)

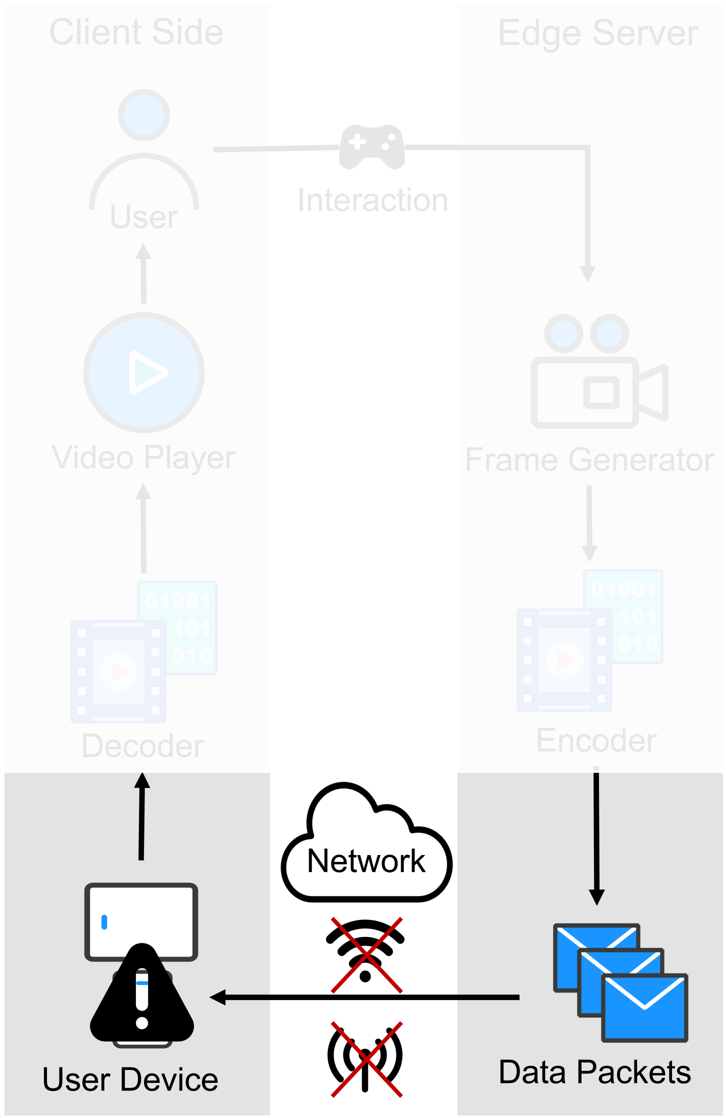
How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

Network Transport System

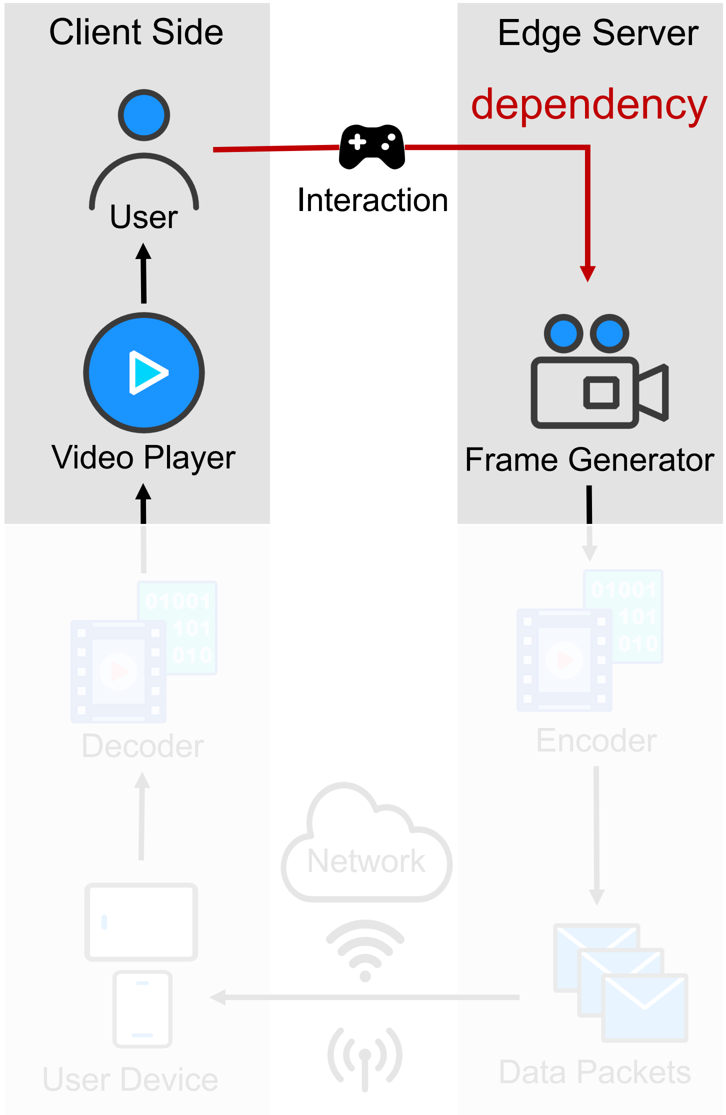
How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

Critical network latency demand: delayed frame delivery lead to video stall

How Does Interactive Mobile Real-Time Streaming Work?

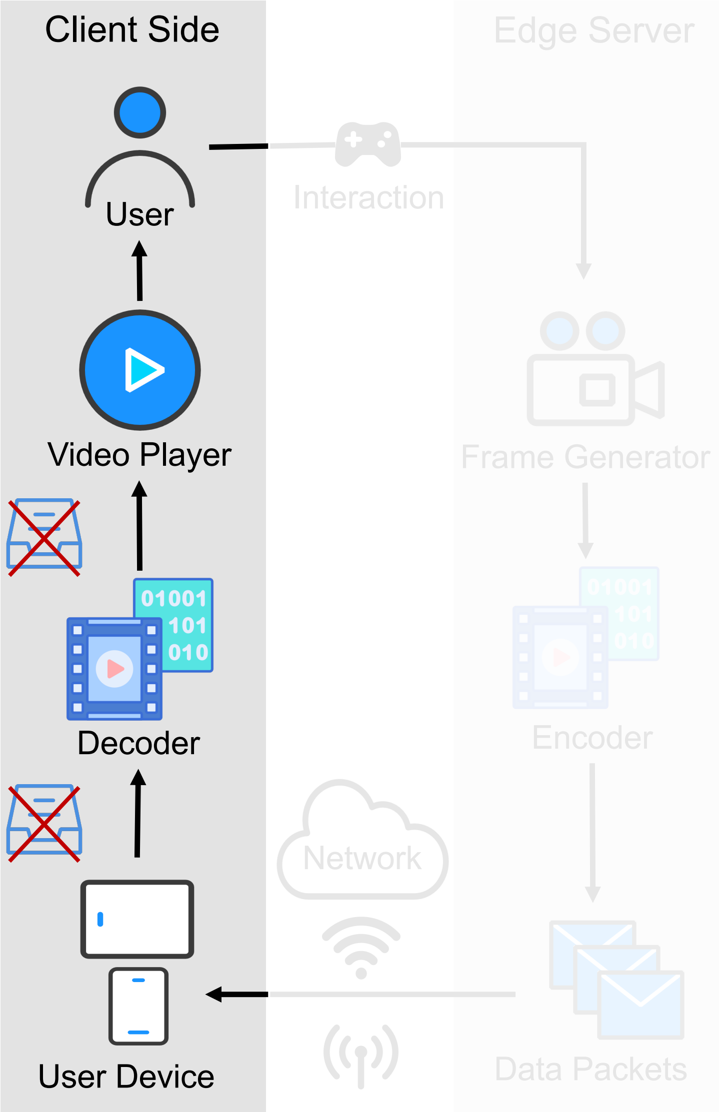


Pipelined framework: closed-loop streaming

Critical network latency demand: delayed frame delivery lead to video stall

Heavy interactivity: video content & size is unknown in advance

How Does Interactive Mobile Real-Time Streaming Work?



Pipelined framework: closed-loop streaming

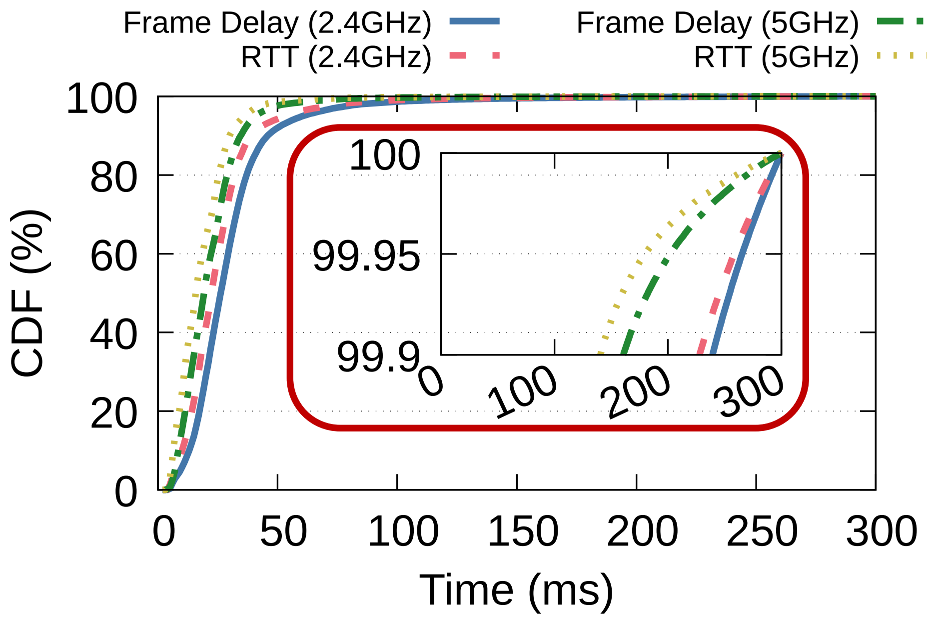
Critical network latency demand: delayed frame delivery lead to video stall

Heavy interactivity: video content & size is unknown in advance

Small playback buffer: insufficient to absorb network fluctuation

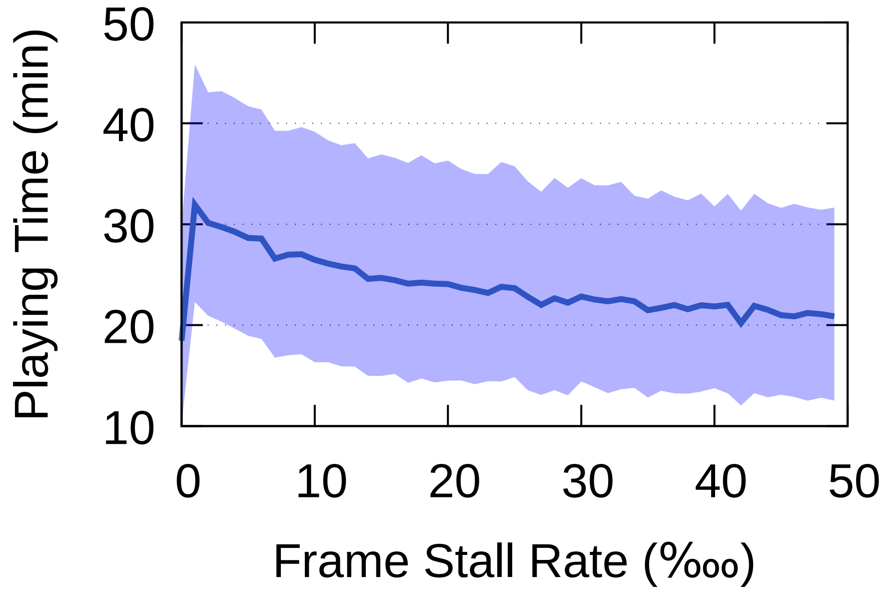
Problem: Long Tail Latency in Mobile Networks

- Frame delay in Wi-Fi network
  - Median < 30 ms 🥳
  - 99.5th percentile > 200 ms 💥
- Video stall degrades users' willingness
  - Stall rate + 0.5% 🤔
  - User playing time – 33% 😭



video stall

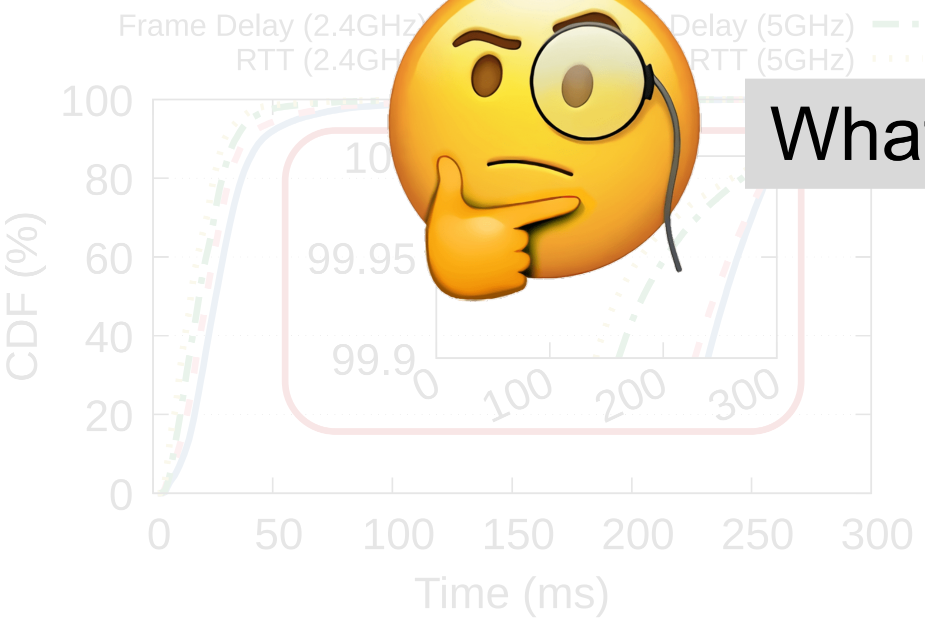
every ~16.7 s



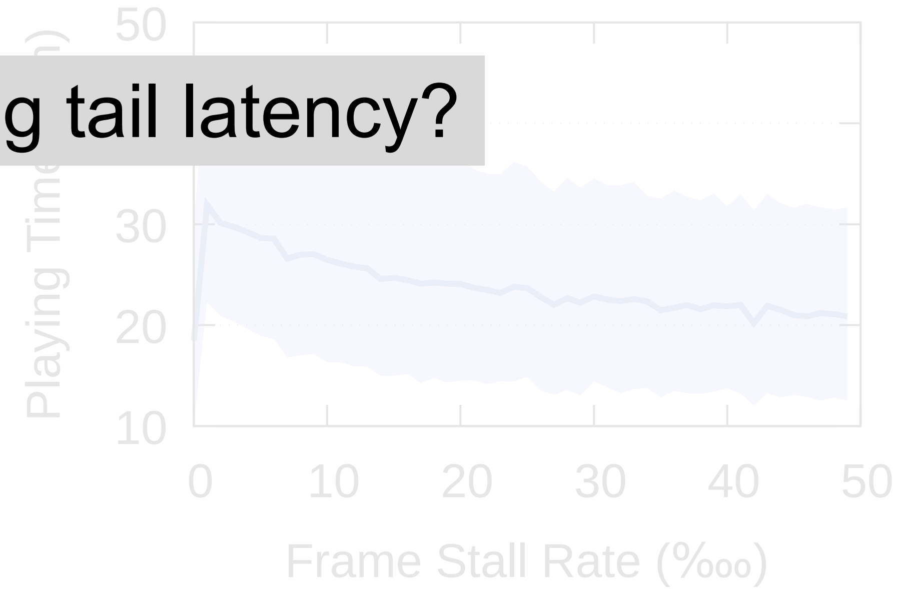
*Statistics from our cloud gaming platform

Problem: Long Tail Latency in Mobile Networks

- Frame delay in Wi-Fi network
  - Median < 30 ms
  - 99.9th percentile > 200 ms
- Video stall degrades users' willingness
  - Stall rate + 0.5%
  - User playing time – 33%



Video stall every ~16.7 s

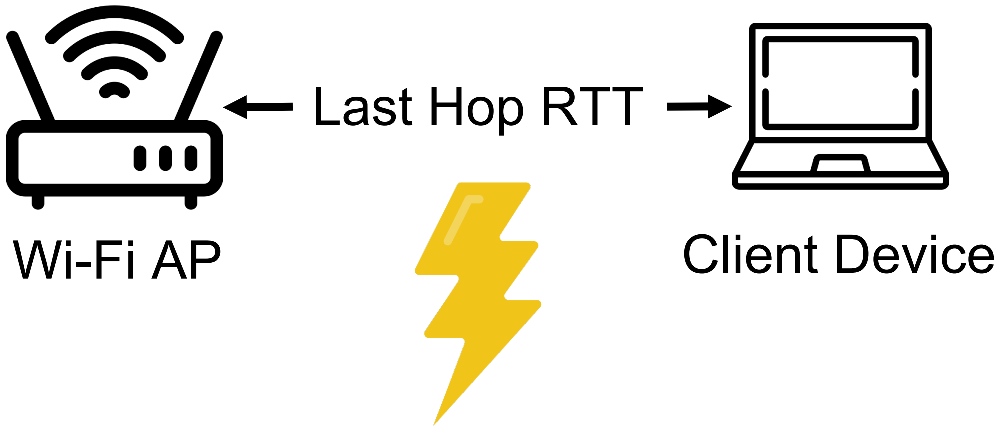


What leads to long tail latency?

*Statistics from our cloud gaming platform

Observation #1: Bottleneck Lies in the Wireless Last-Hop

Wi-Fi path suffer from *RTT inflation*

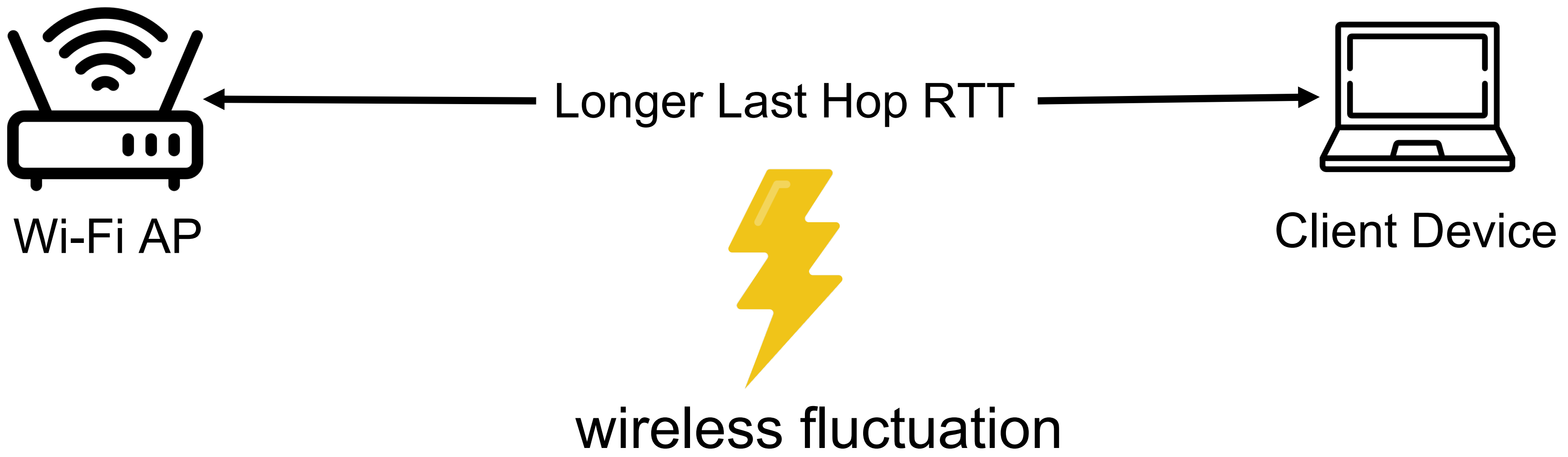


wireless fluctuation

signal attenuation
channel contention
non-line-of-sight
...

Observation #1: Bottleneck Lies in the Wireless Last-Hop

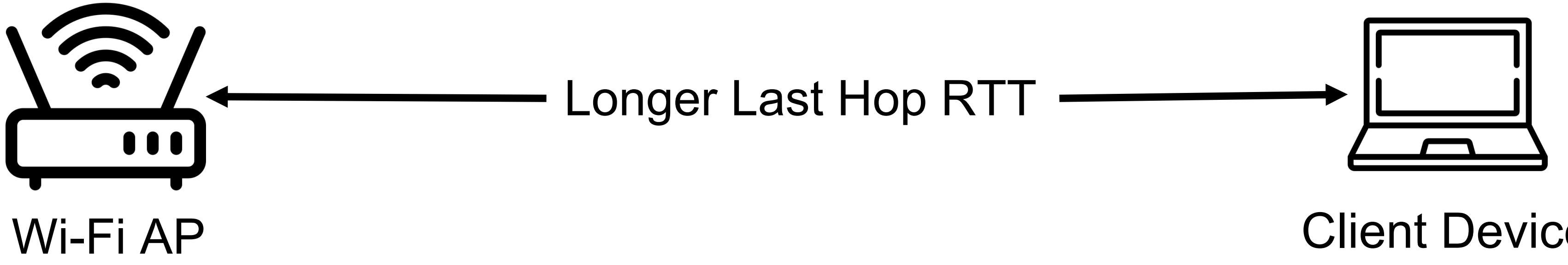
Wi-Fi path suffer from *RTT inflation*

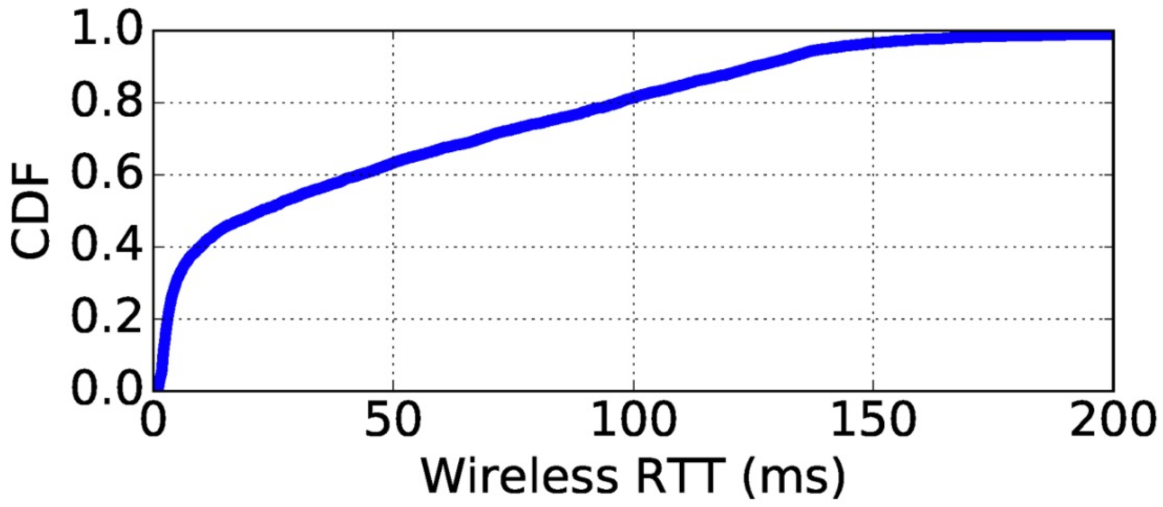


signal attenuation
channel contention
non-line-of-sight
...

Observation #1: Bottleneck Lies in the Wireless Last-Hop

Wi-Fi path suffer from *RTT inflation*



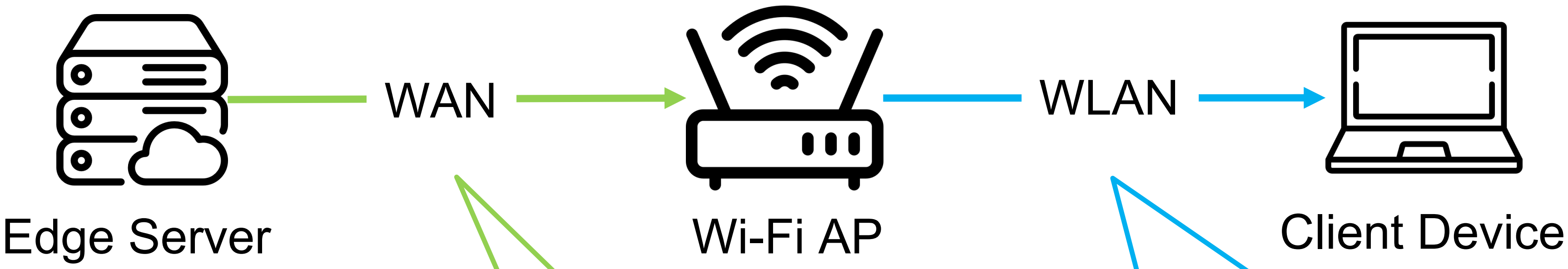


Last Hop RTT can inflate to > 200 ms[1]

[1] Latency Based WiFi Congestion Control in the Air for Dense WiFi Networks, IEEE IWQoS 2017

Observation #1: Bottleneck Lies in the Wireless Last-Hop

Can congestion control algorithm or adaptive bitrate solve it?

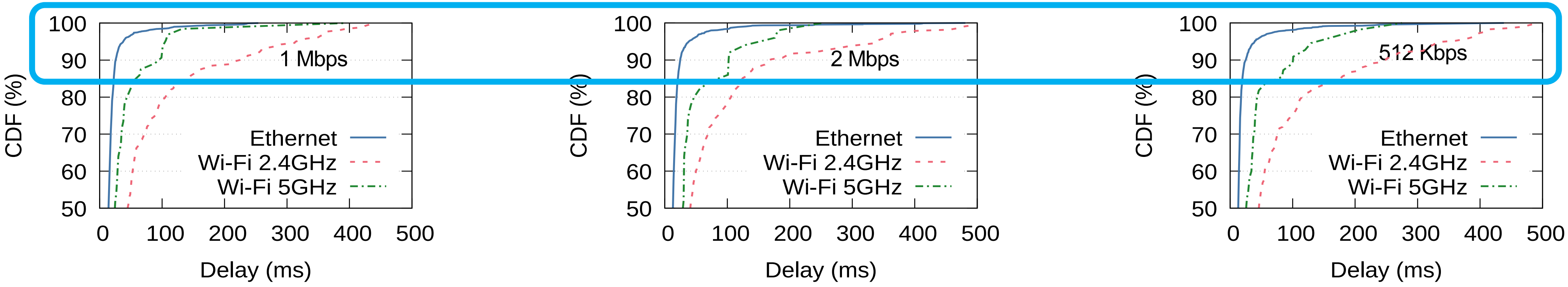


Maybe I can adjust bitrate timely?

No congestion occurred!

RTT inflation still exists because you cannot eliminate fluctuation!

Long tail latency occurs in wireless networks even with low bitrate



Observation #1: Bottleneck Lies in the Wireless Last-Hop

Can congestion control algorithm or adaptive bitrate solve it?

😫 RTT inflation is intrinsic to a wireless path

&

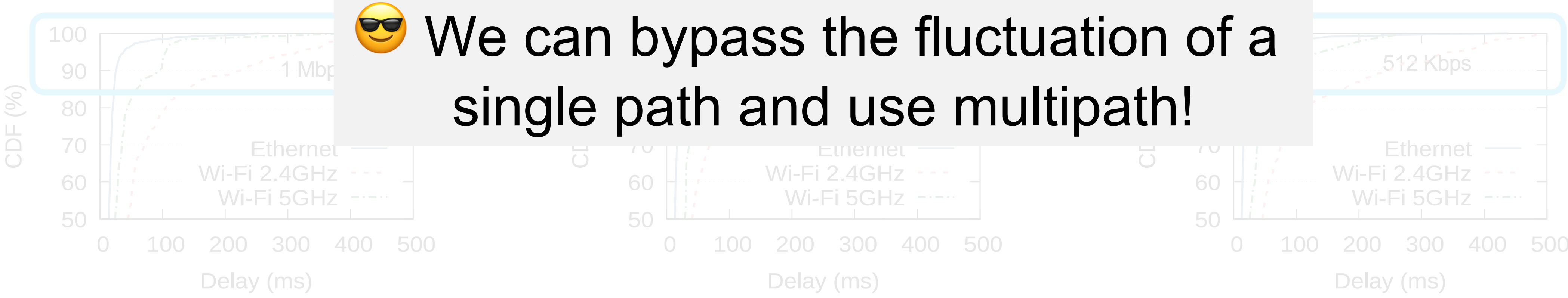
CCAs or ABRs cannot solve it

Maybe I can adjust bitrate timely?

No congestion occurred!

Still RTT inflation because you can not deal with fluctuation!

Long tail latency occurs in wireless networks even with low bitrate

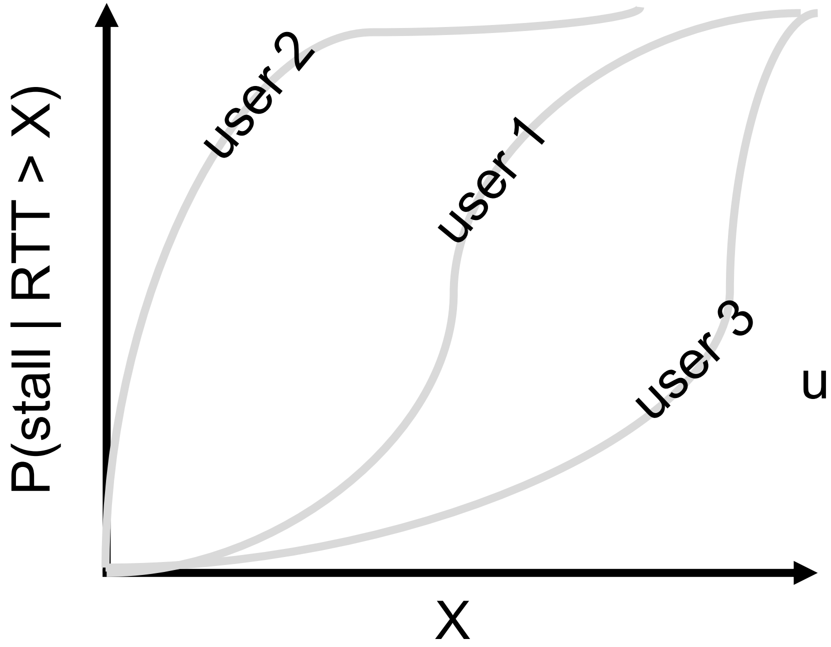


😎 We can bypass the fluctuation of a single path and use multipath!

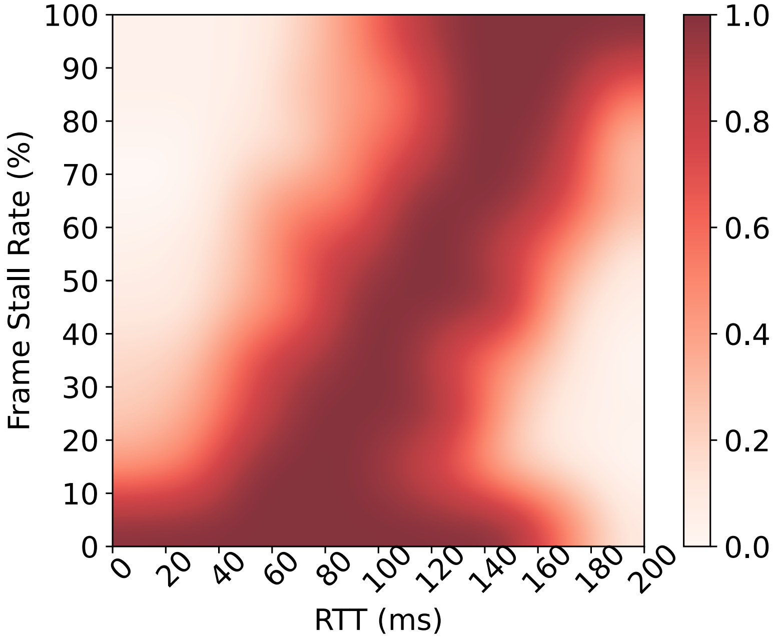
Observation #2: Network Characteristics Stability

Network characteristics vary among different user sessions

By using “characteristics”, we mean video stall rate
and the correlation between RTT inflation and frame stall
i.e. P(stall) and P(stall | RTT)



overlay all
user sessions

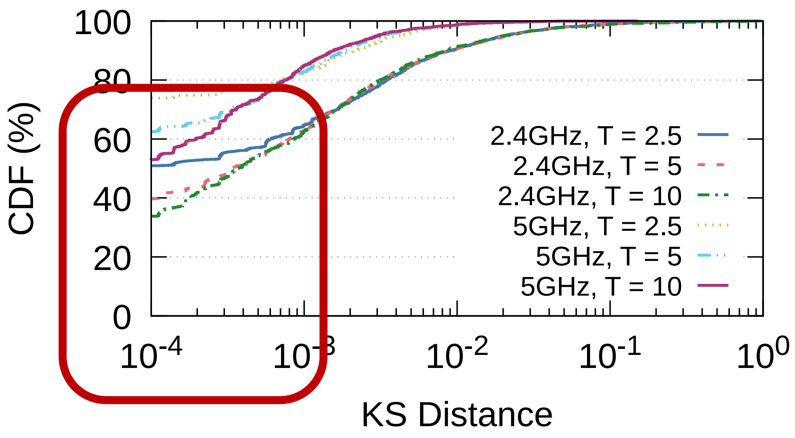


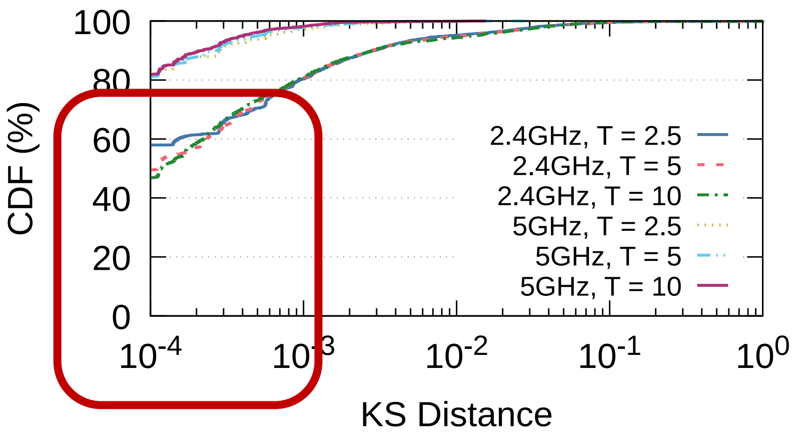
RTT inflation has different impacts on different users ⇒ No general solution for all

Observation #2: Network Characteristics Stability

Network characteristics remain stable
within a session for a time window

P(stall) and P(stall | RTT) distributions are stable:
the KS distances[1] of these distributions between adjacent time windows are tiny



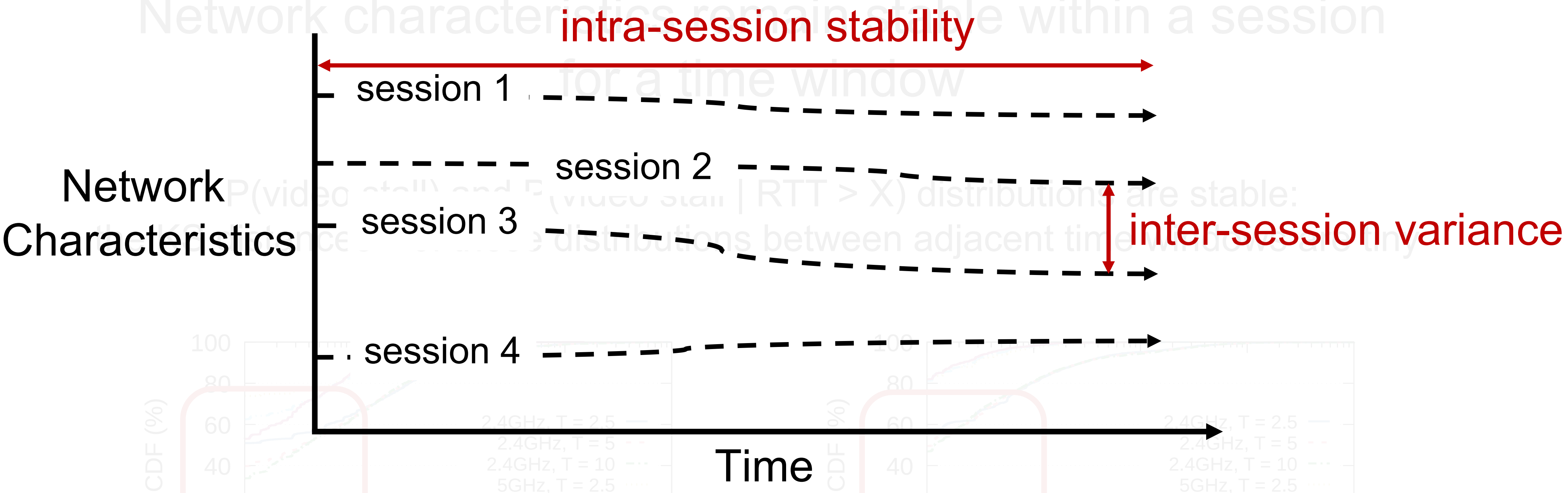


Difference between P(stall) and P(stall | RTT) distributions

In short, the distribution of P(stall) and P(stall | RTT) is stable

[1] Kolmogorov-Smirnov (KS) distance is a metric to examine the similarity between two distributions ranging from 0 to 1 and a KS distance of 0 indicates two identical distributions

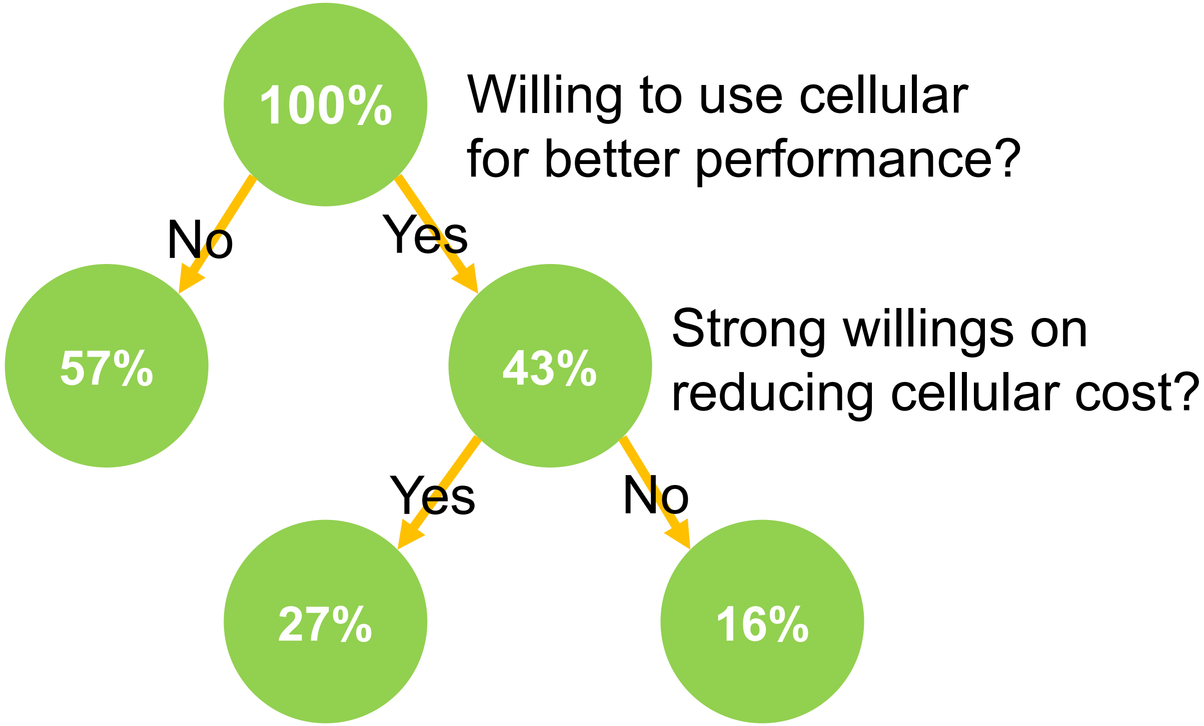
Observation #2: Network Characteristics Stability



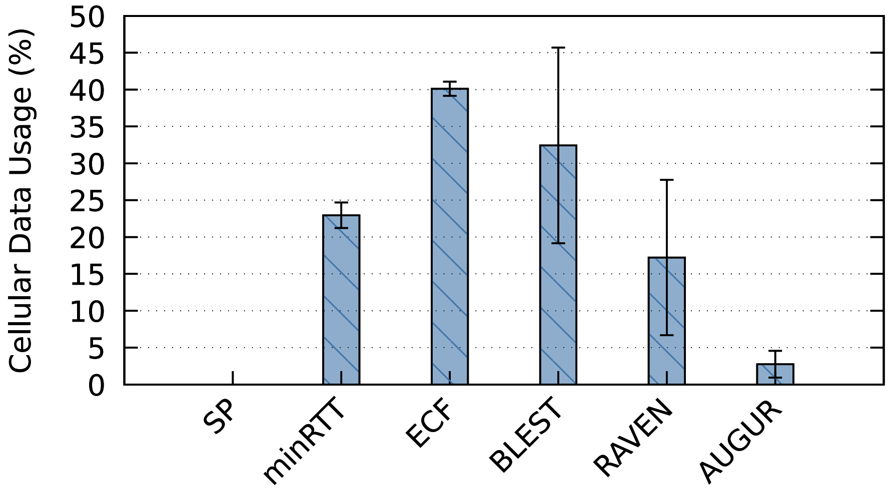
😎 We can capture and leverage the unique network characteristics of each user session

Observation #3: Cellular Usage Minimization is Crucial

Users express strong concern on cellular data usage and want to reduce it



Traditional multipath schemes ignore cellular cost and incur large cellular usage



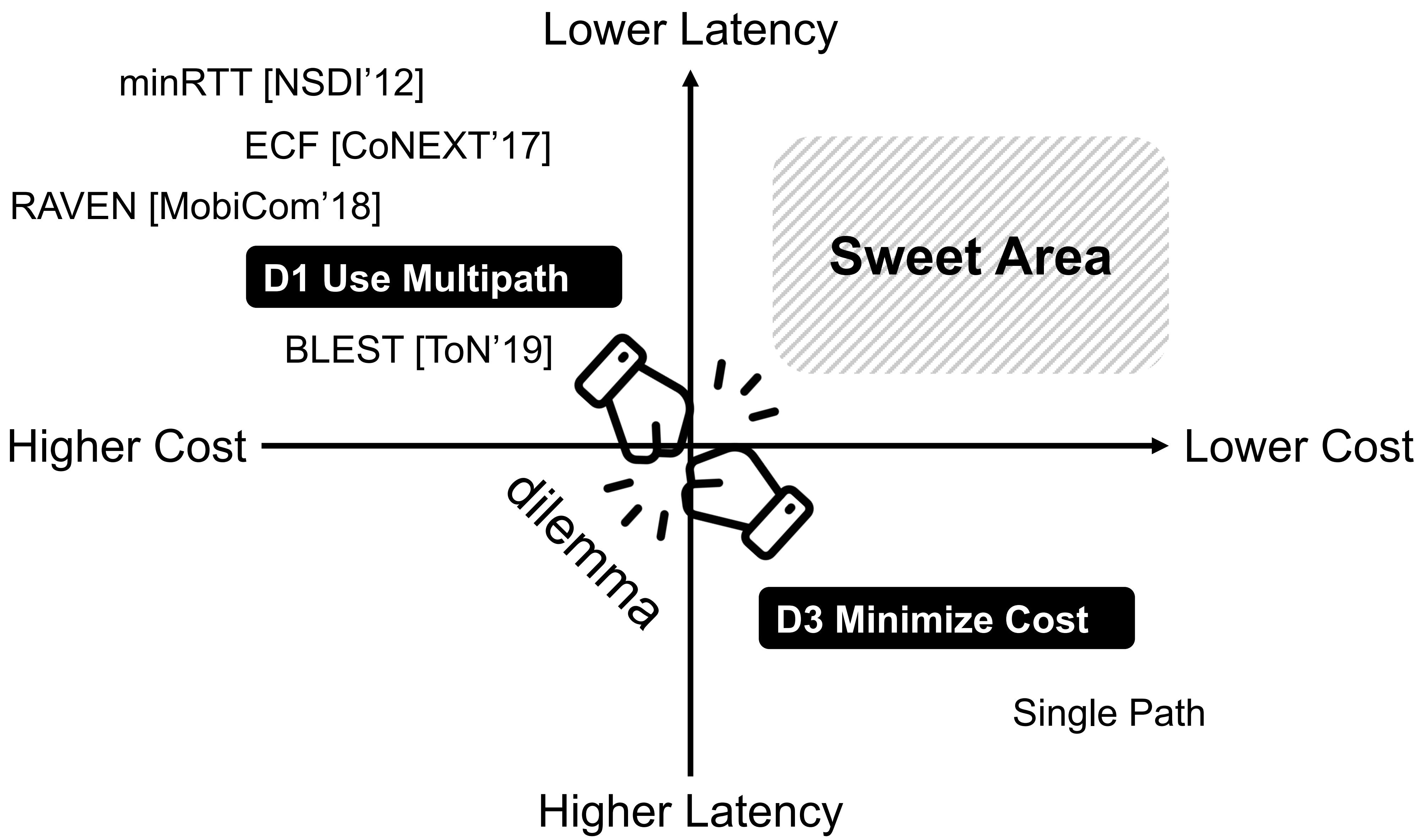
Traditional multipath schemes incurs up to 45% cellular data usage (bytes sent in cellular path / all bytes sent)

Cellular cost constraint should be a major concern instead of incidental

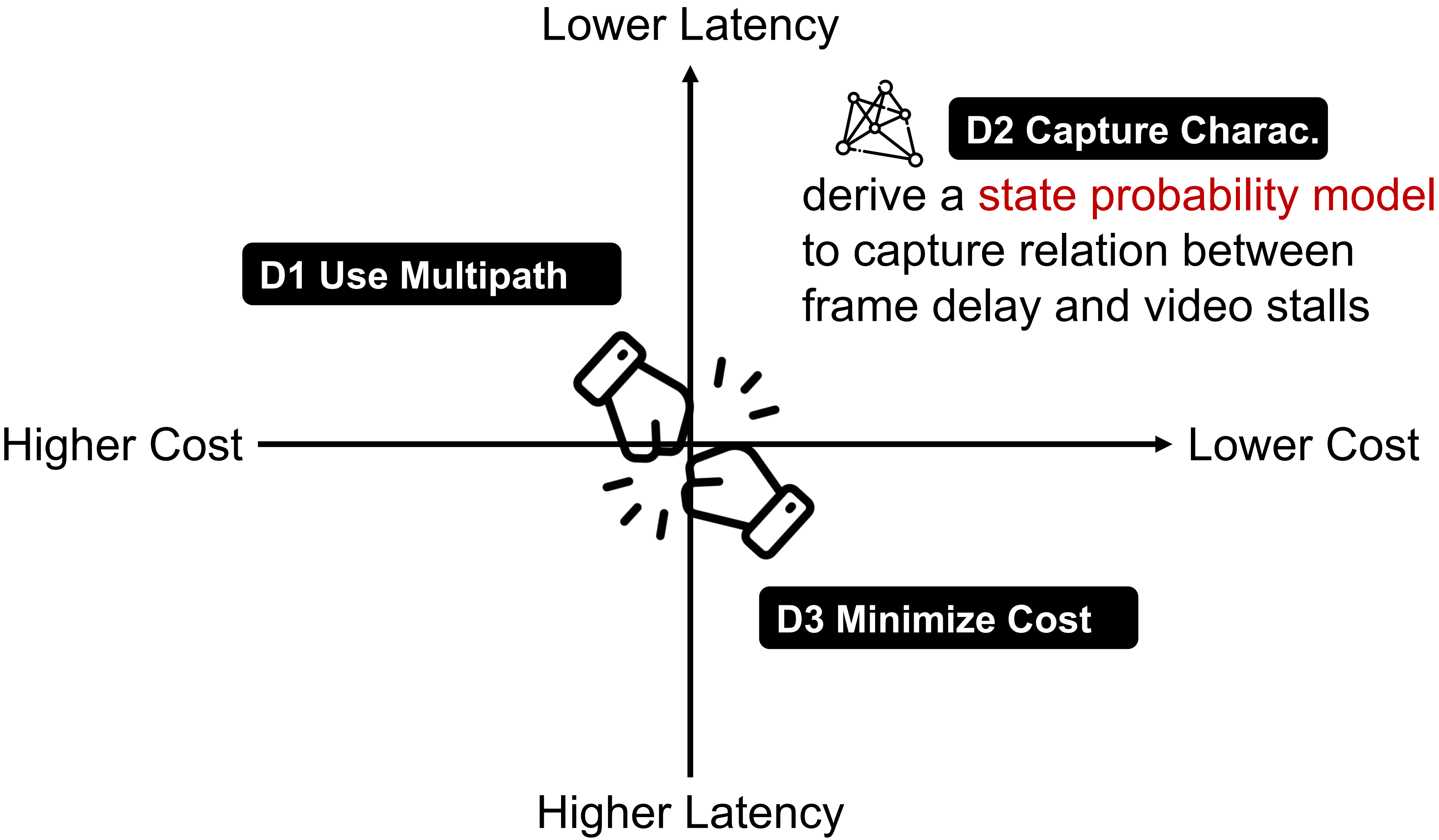
Recap: Our Design Considerations

Observations		Design Consideration
Single path Wi-Fi last-hop wireless fluctuation	→	**D1 Use Multipath**
Per-session network characteristics stability	→	**D2 Capture Charac.**
Strong concern on cellular data usage	→	**D3 Minimize Cost**

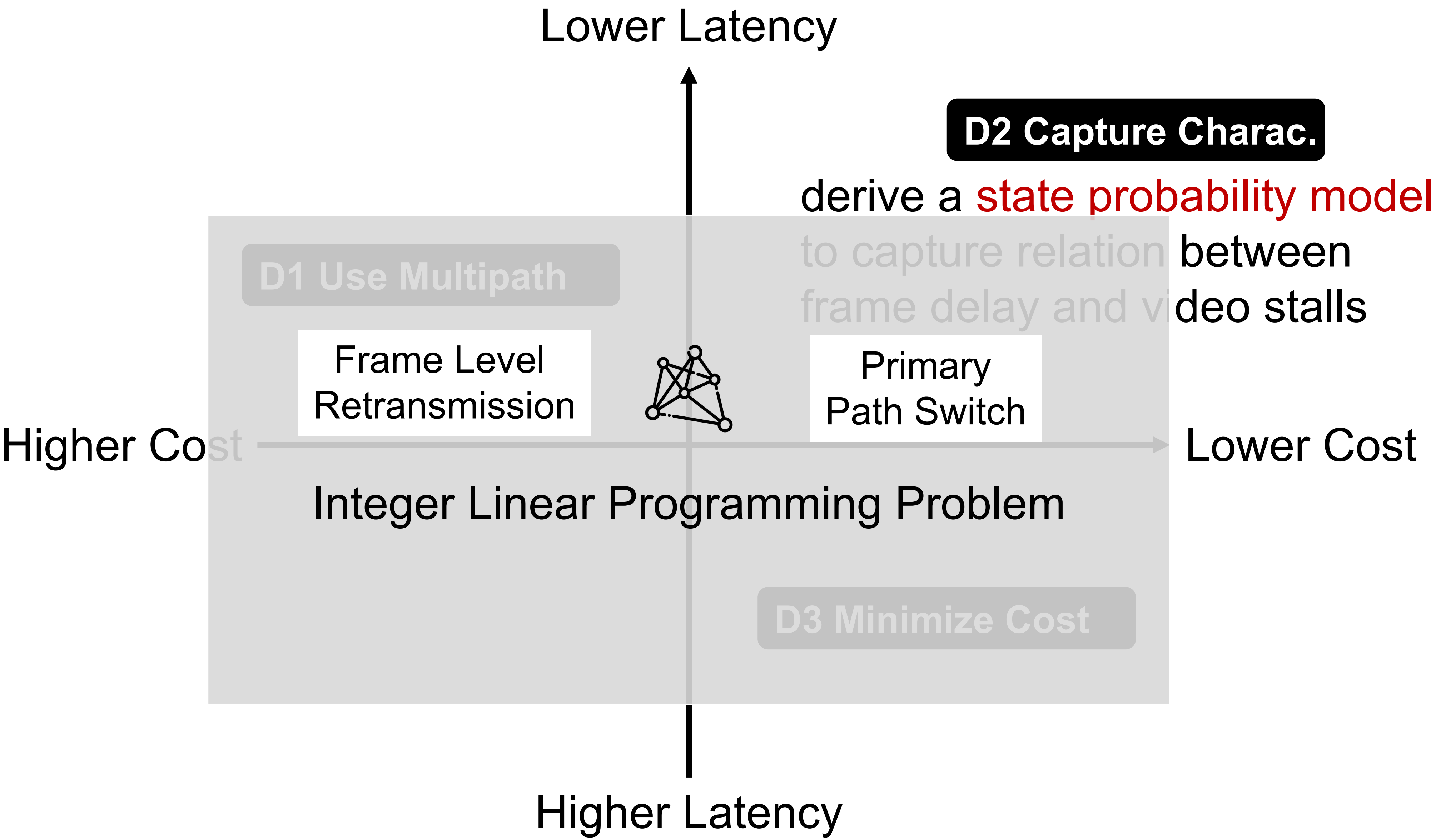
AUGUR Design Overview



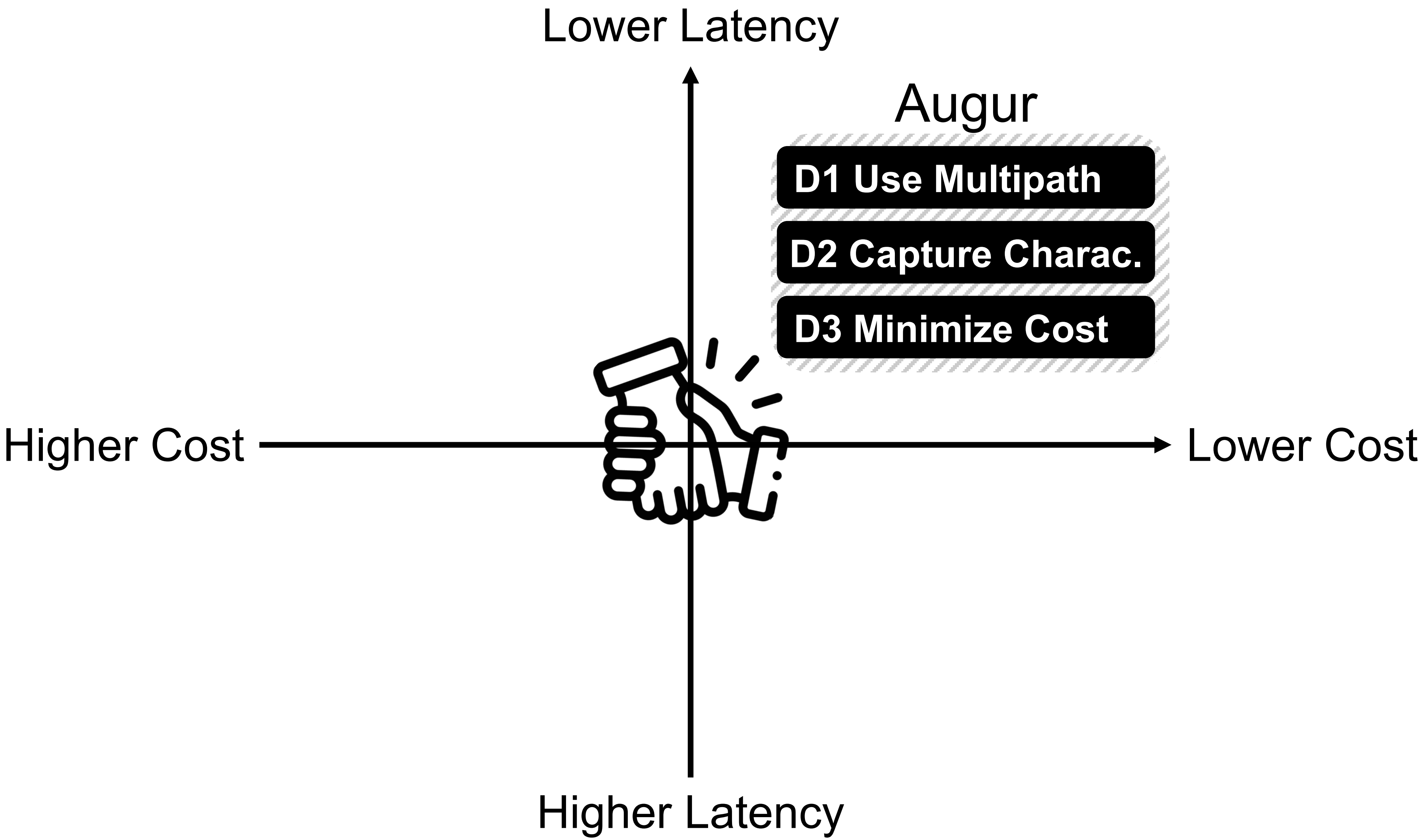
AUGUR Design Overview



AUGUR Design Overview



AUGUR Design Overview



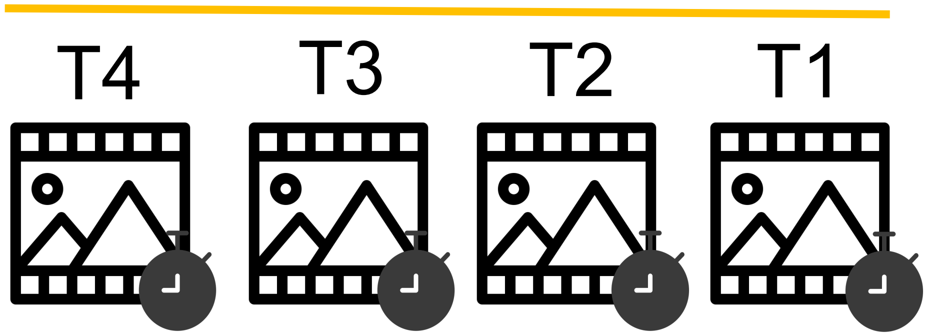
State Probability Model

- Step 1: Divide runtime network statistics into *states*

D2 Capture Charac.

  - Space $U_1 = \cup_i \{S_{1,i}\}$: in-flight time of all frames ⇒ capture *frame latency distribution*
  - Space $U_2 = \cup_i \{S_{2,i}\}$: in-flight time of earliest unACKed frame ⇒ capture *path latency distribution*

Server Sending Queue



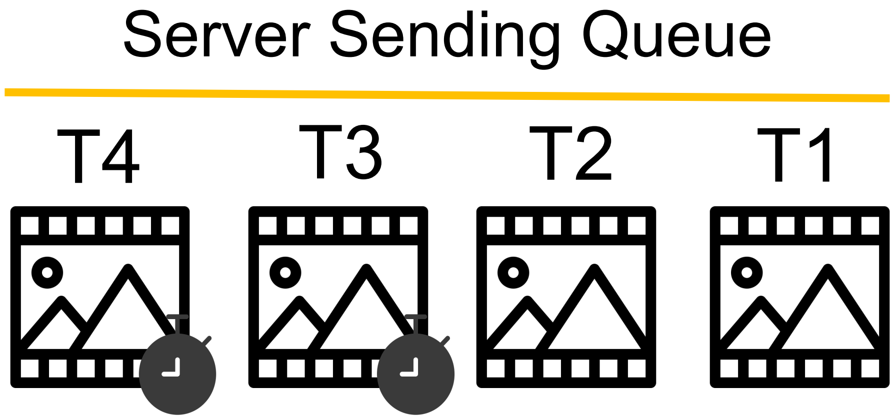
U1 = {}

U2 = {now – T1}

State Probability Model

D2 Capture Charac.

- Step 1: Divide session network statistics into *states*
  - Space $U_1 = \cup_i \{S_{1,i}\}$: in-flight time of all frames ⇒ capture *frame latency distribution*
  - Space $U_2 = \cup_i \{S_{2,i}\}$: in-flight time of earliest unACKed frame ⇒ capture *path latency distribution*



U1 = {now – T1, now – T2}

U2 = {now – T1, now – T3}

State Probability Model

- Step 2: Define state probability model

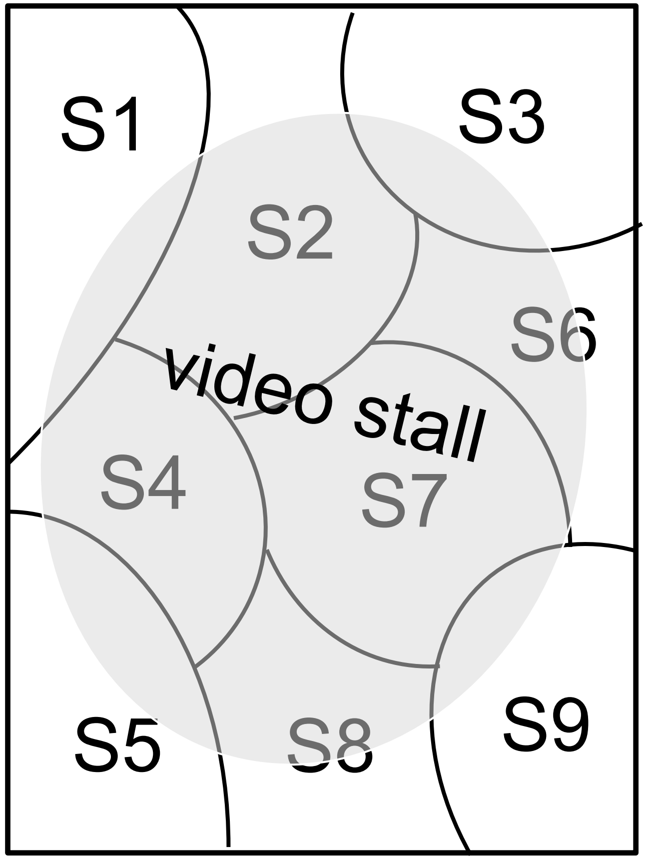
D2 Capture Charac.

  - Define overall frame stall probability for a user session as:

$$P(stall) = P(stall, U) = \sum_{S_i \in U} P(stall, S_i) = \sum_{S_i \in U} P(stall \mid S_i) P(S_i)$$

  - Therefore we define state probability model for a user session as:

$$M = \{P(stall), P(stall \mid S_i), P(S_i)\}, S_i \in U$$



$U = \bigcup_i \{S_i\}$

State Probability Model

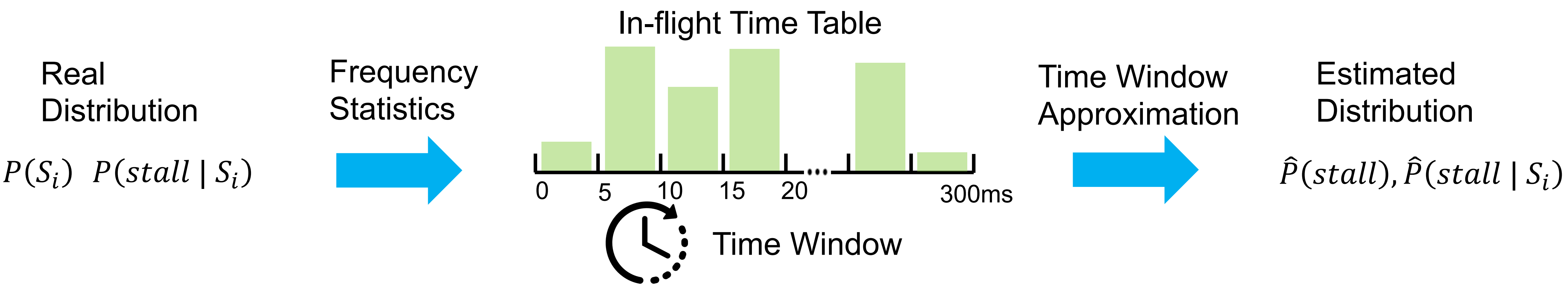
- Step 3: Update model at runtime

D2 Capture Charac.

The distribution of $P(S_i)$ and $P(stall \mid S_i)$ can be arbitrary and hard to describe

Motivated by characteristics stability of individual users
we use frequencies of each state in a time window to estimate the distribution

$$\widehat{M}_{1,2} = \{\hat{P}(stall), \hat{P}(stall \mid S_i), \hat{P}(S_i)\}, S_i \in U_{1,2}$$



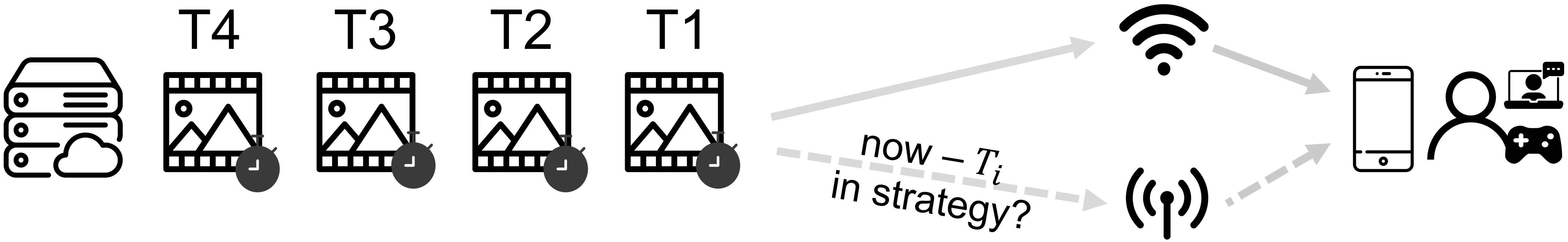
When to Use Multipath

Use Wi-Fi as primary path and use cellular as backup path

just slightly use the costly path if necessary

Define a strategy $X \subset U$ containing some states

at these states, backup paths should be used



Strategy Derivation: ILP Problem Formulation

D1 Use Multipath

Bypass Wi-Fi last-hop RTT inflation by using the backup path

Minimizing false negative rate $P(FN)$: I should have used the backup path, but I didn't

D3 Minimize Cost

Reduce the utilization of the backup path

Minimize false positive rate $P(FP)$: I shouldn't have used the backup path, but I did

D1: Minimizing $P(FN) \Leftrightarrow$ Maximizing $\sum_{S_i \in U} P(stall \mid S_i) P(S_i) P_{backup}(S_i) \cdot x_i$

$x_i = 0, 1$

D2 Capture Charac.

D3: Minimizing $P(FP) \Leftrightarrow$ Subject to $\sum_{S_i \in U} P(S_i) \cdot x_i \leq \sum_{S_i \in U} P(stall \mid S_i) P(S_i) \cdot \delta$

Integer Linear Programming Problem

$P_{backup}(S_i)$: Backup path capacity, measured in RTT

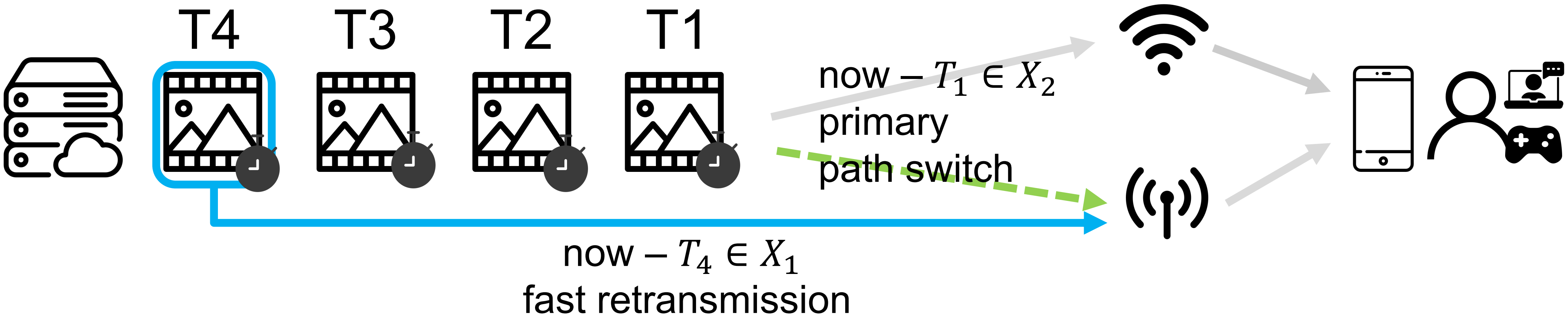
δ: Threshold factor representing the data usage limit of the backup path

How to Use Multipath

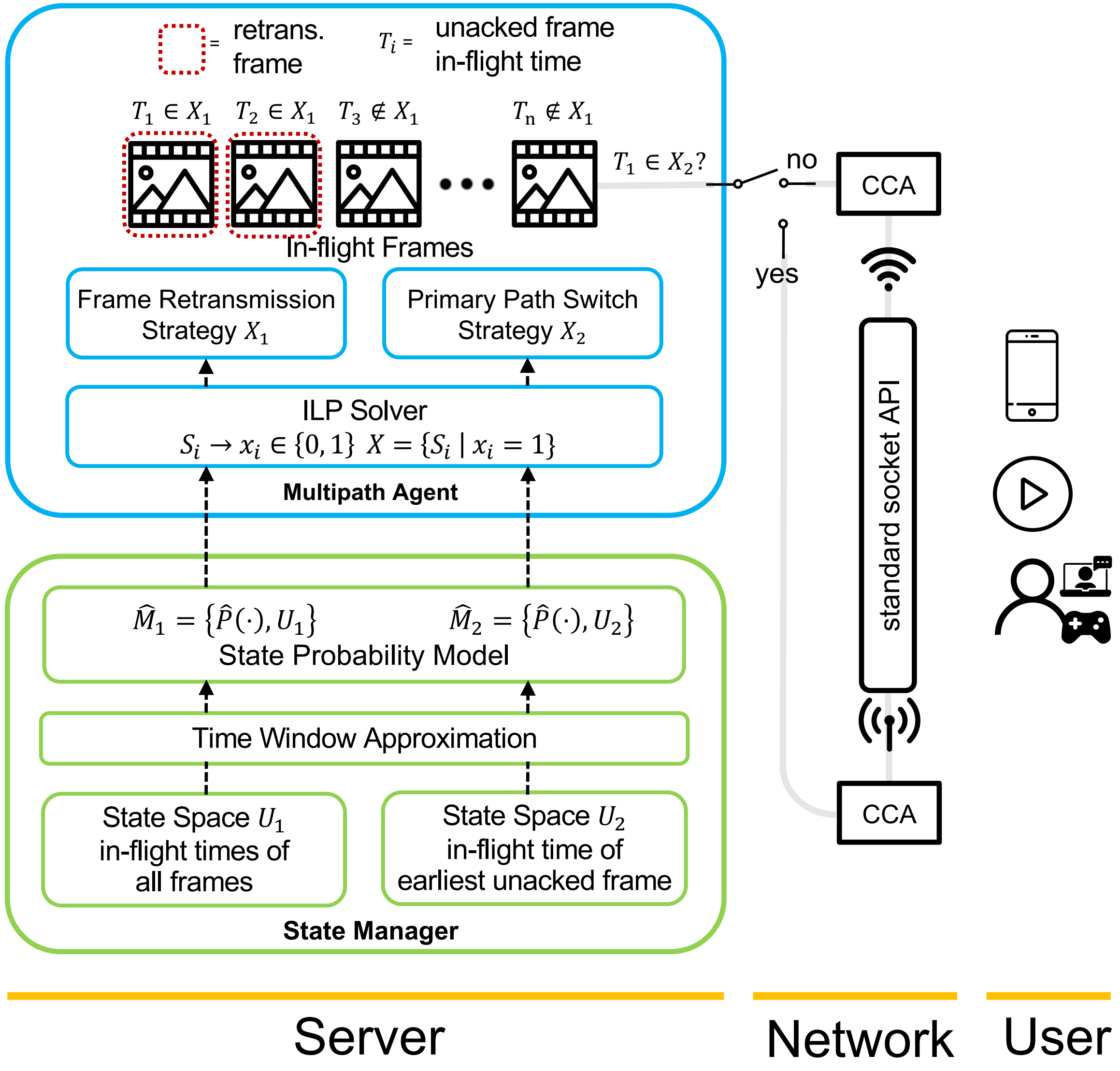
Use cellular path in two ways (two sets of strategies)

Frame retransmission strategy $X_1 \subset U_1$
actively retransmit in-flight frames on cellular path
before transmission timeout on Wi-Fi path

Path switch strategy $X_2 \subset U_2$
transiently switch the primary path to cellular
in case of severe capacity degradation of Wi-Fi path

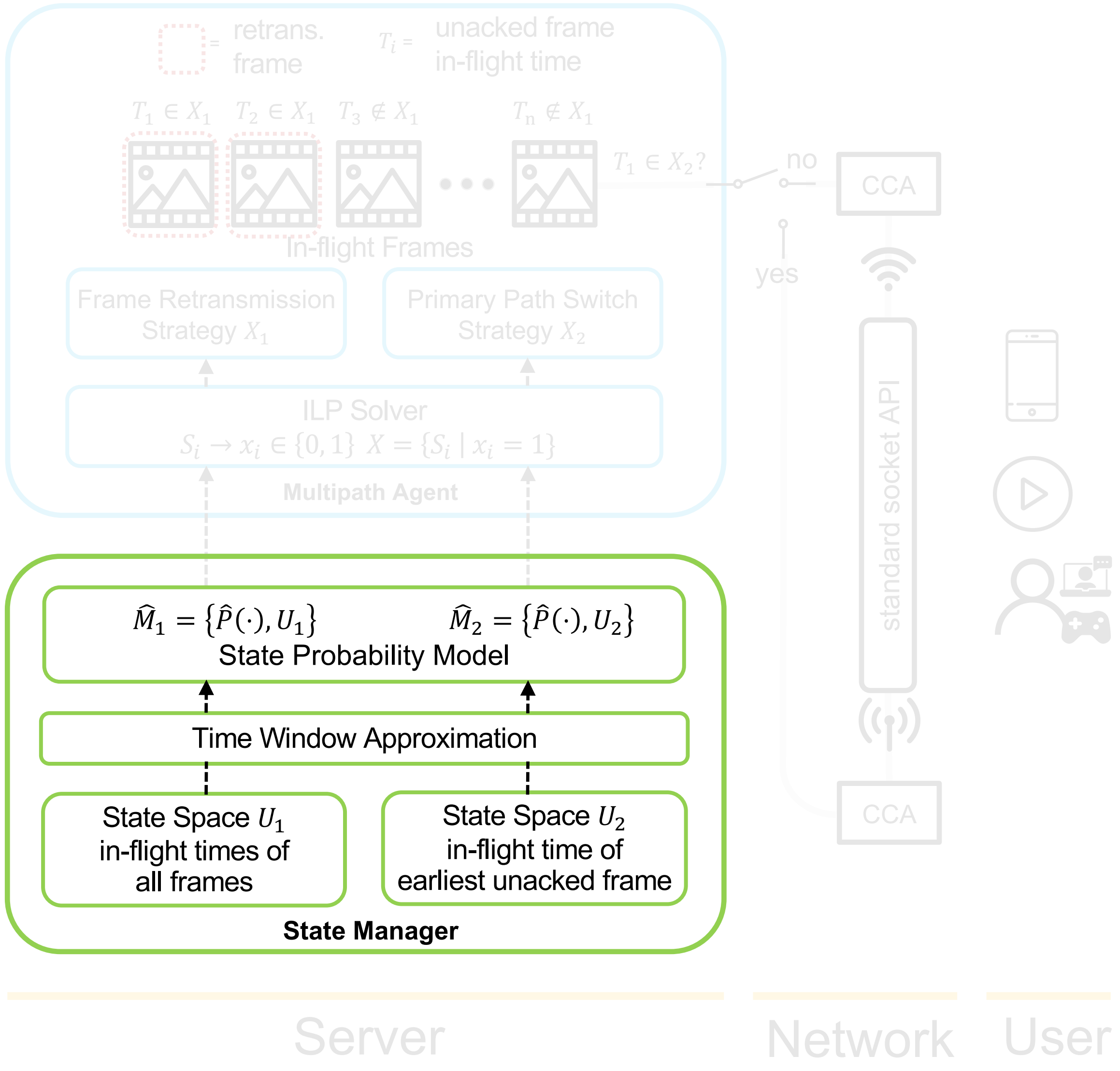


Put Everything Together



Put Everything Together

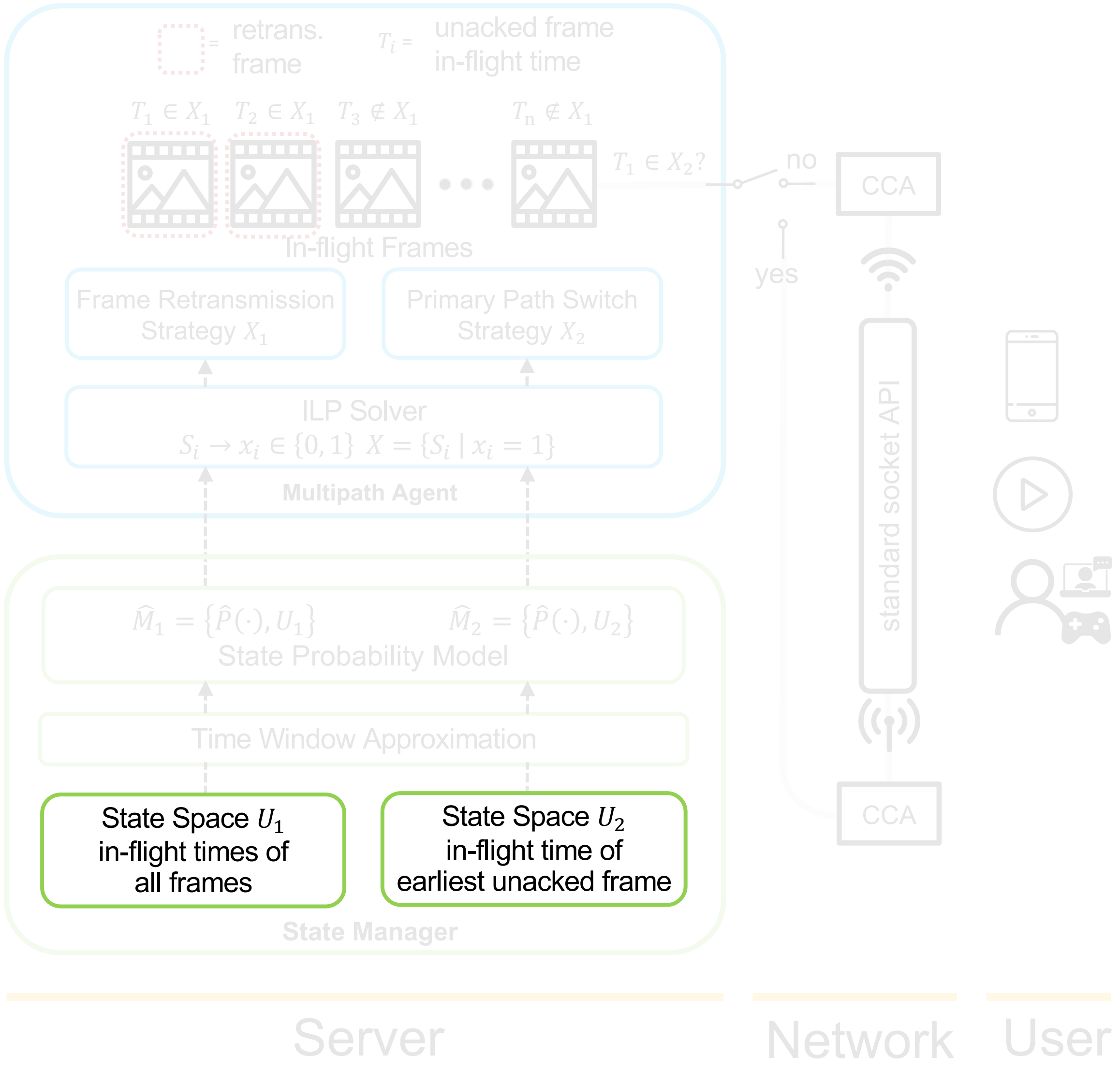
State Manager



Put Everything Together

Monitors and records states

State Manager

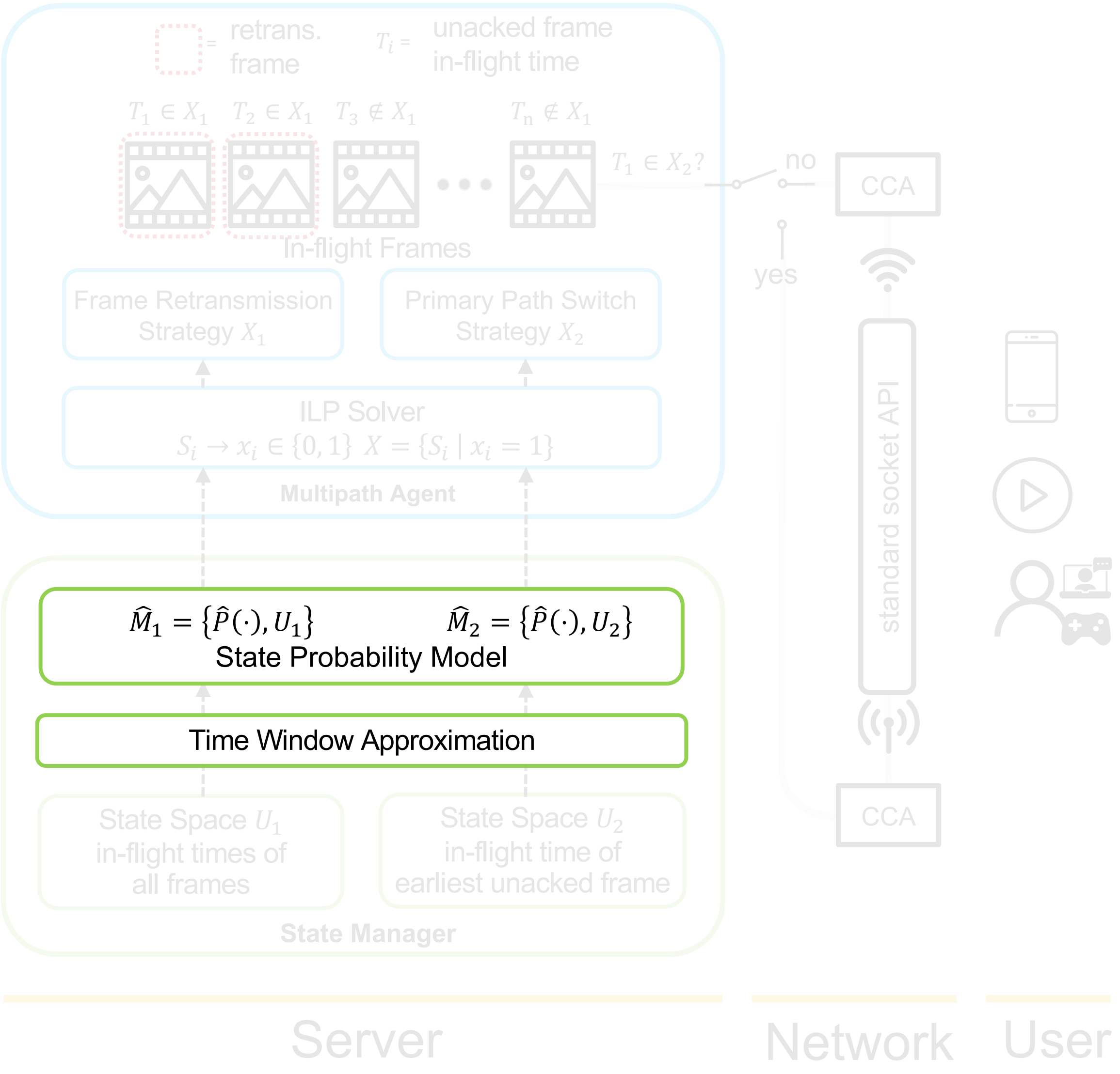


Put Everything Together

Update probability models
in each time window

Monitors and records states

State Manager



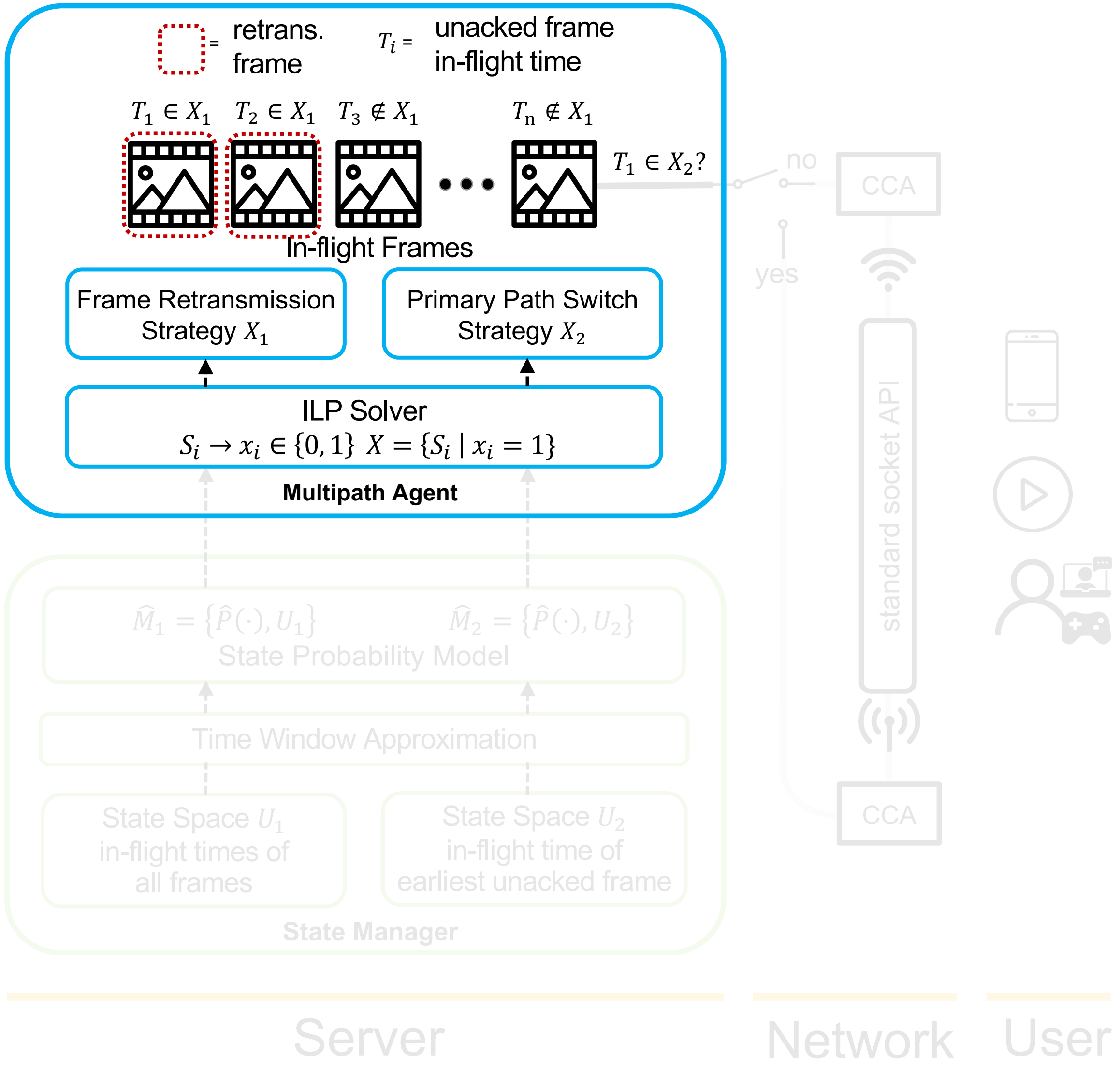
Put Everything Together

Multipath Agent

Update probability models
in each time window

Monitors and records states

State Manager



Put Everything Together

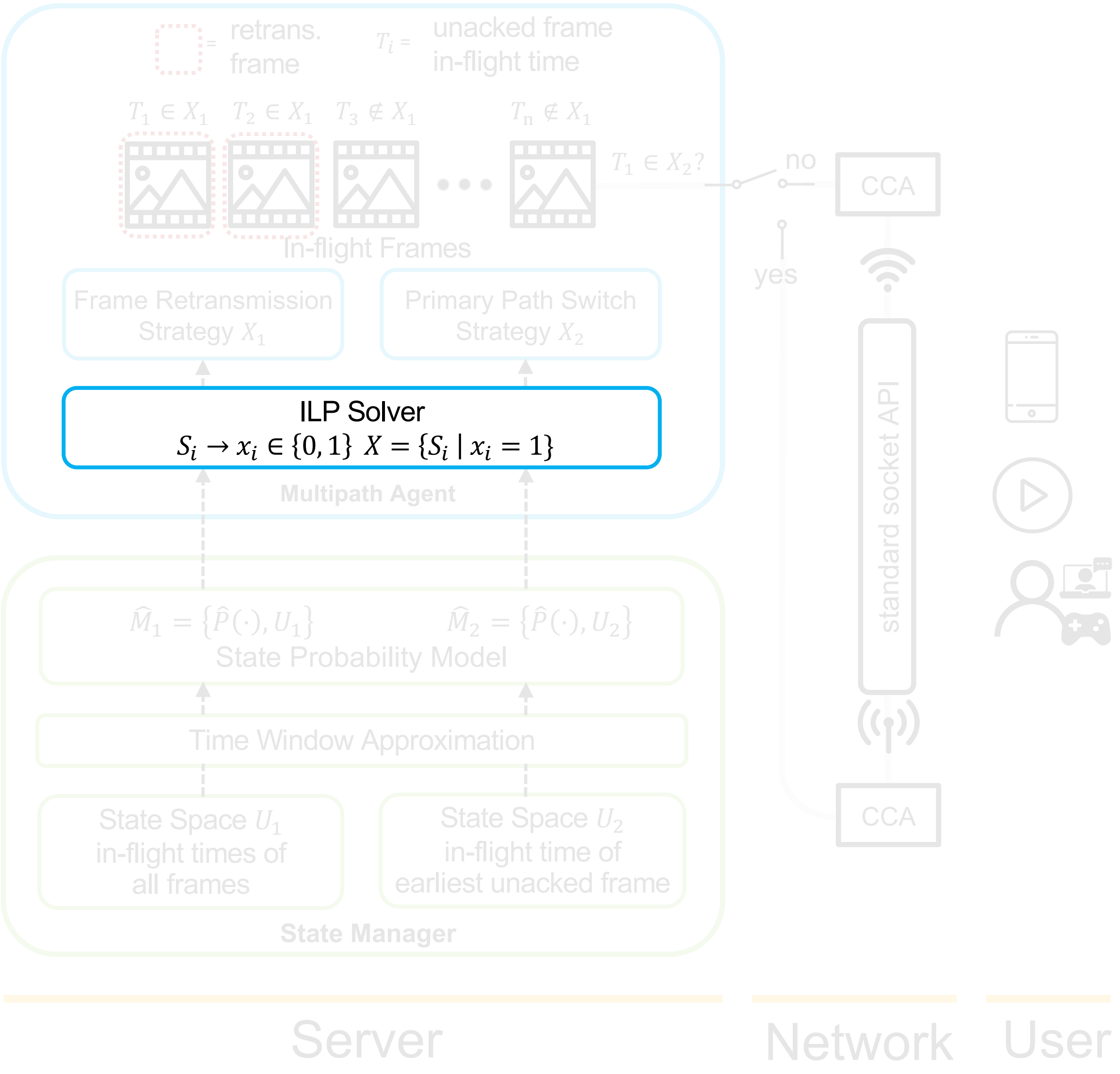
Solve ILP problem continuously

Multipath Agent

Update probability models
in each time window

Monitors and records states

State Manager



Put Everything Together

Perform frame retransmission based on strategy X_1

Perform path selection based on strategy X_2

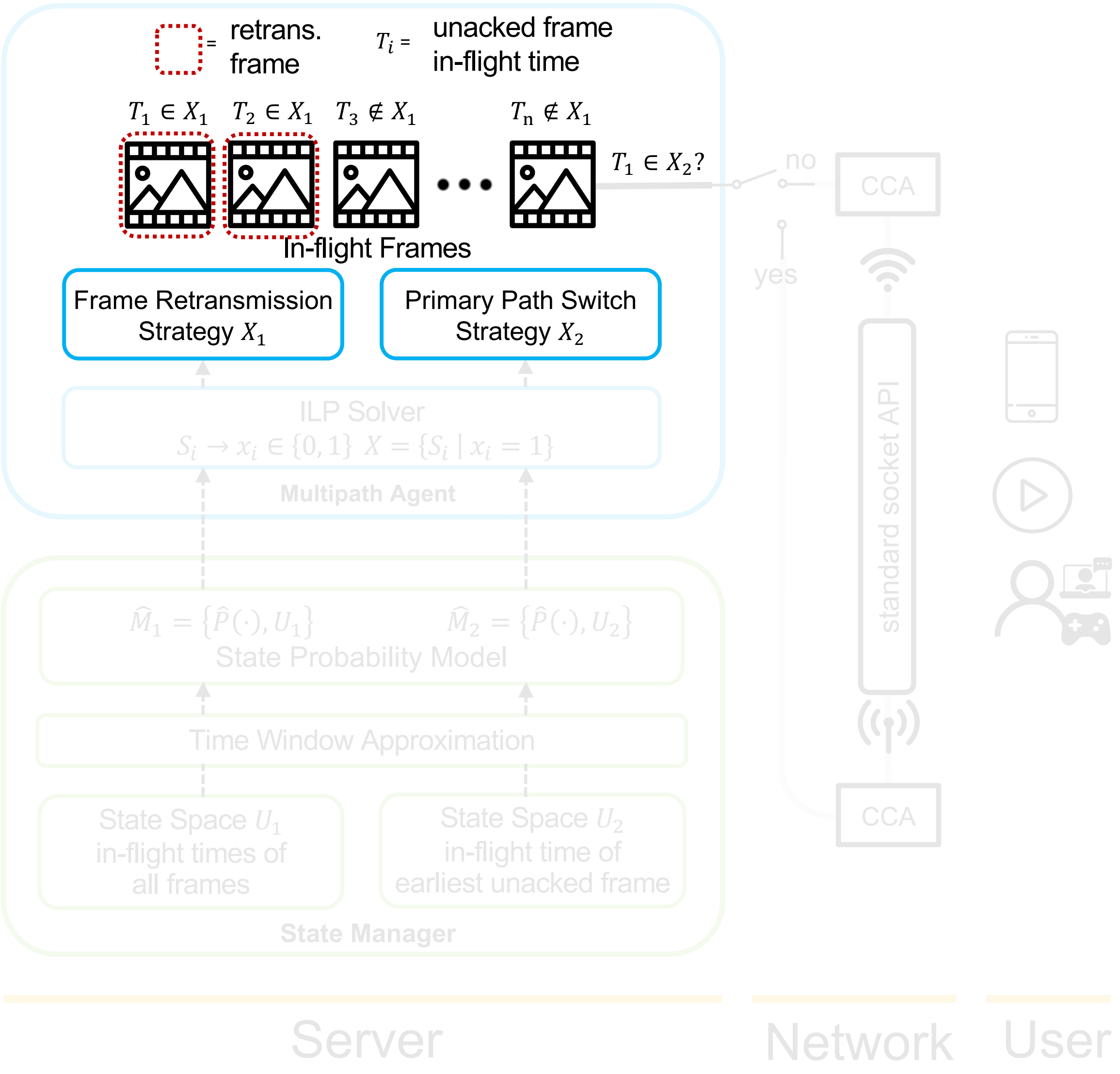
Solve ILP problem continuously

Multipath Agent

Update probability models in each time window

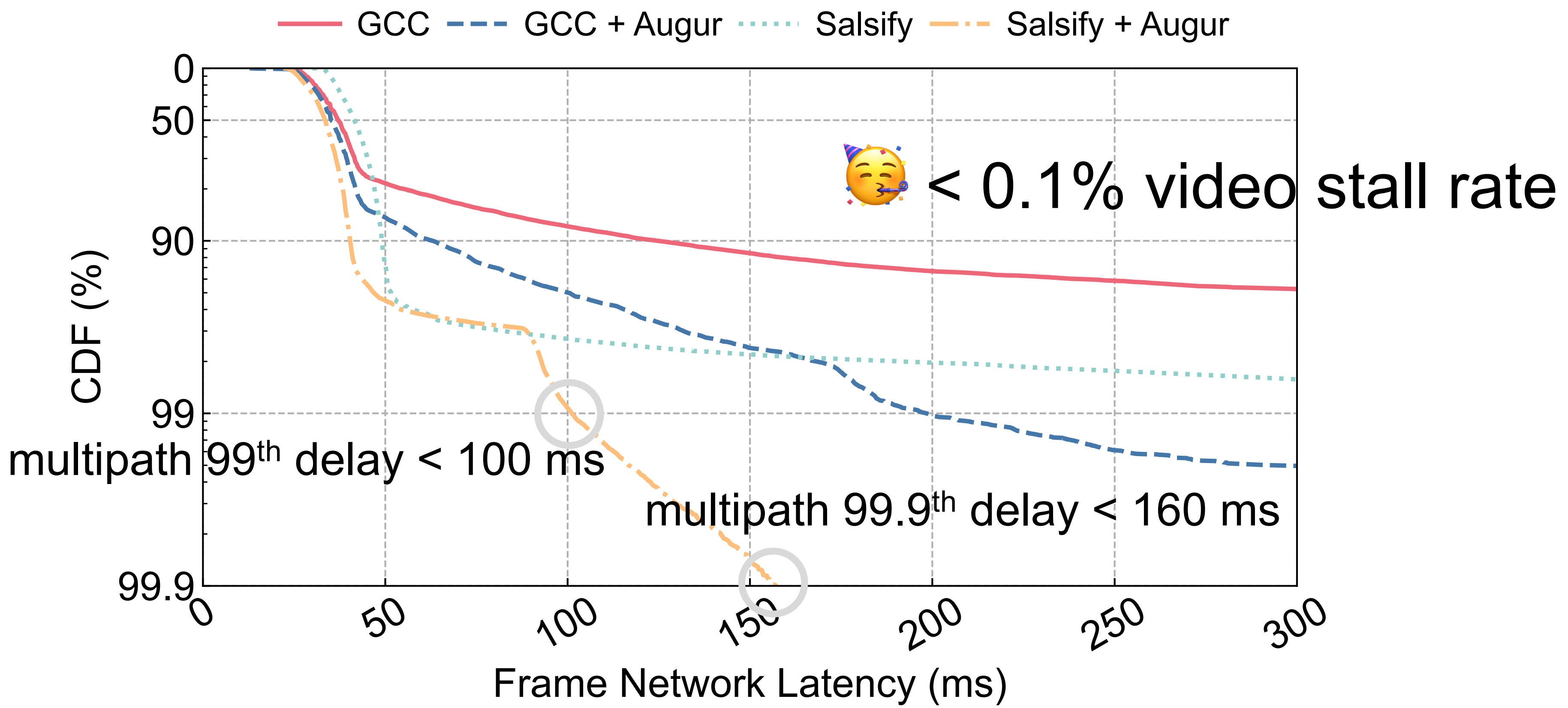
Monitors and records states

State Manager



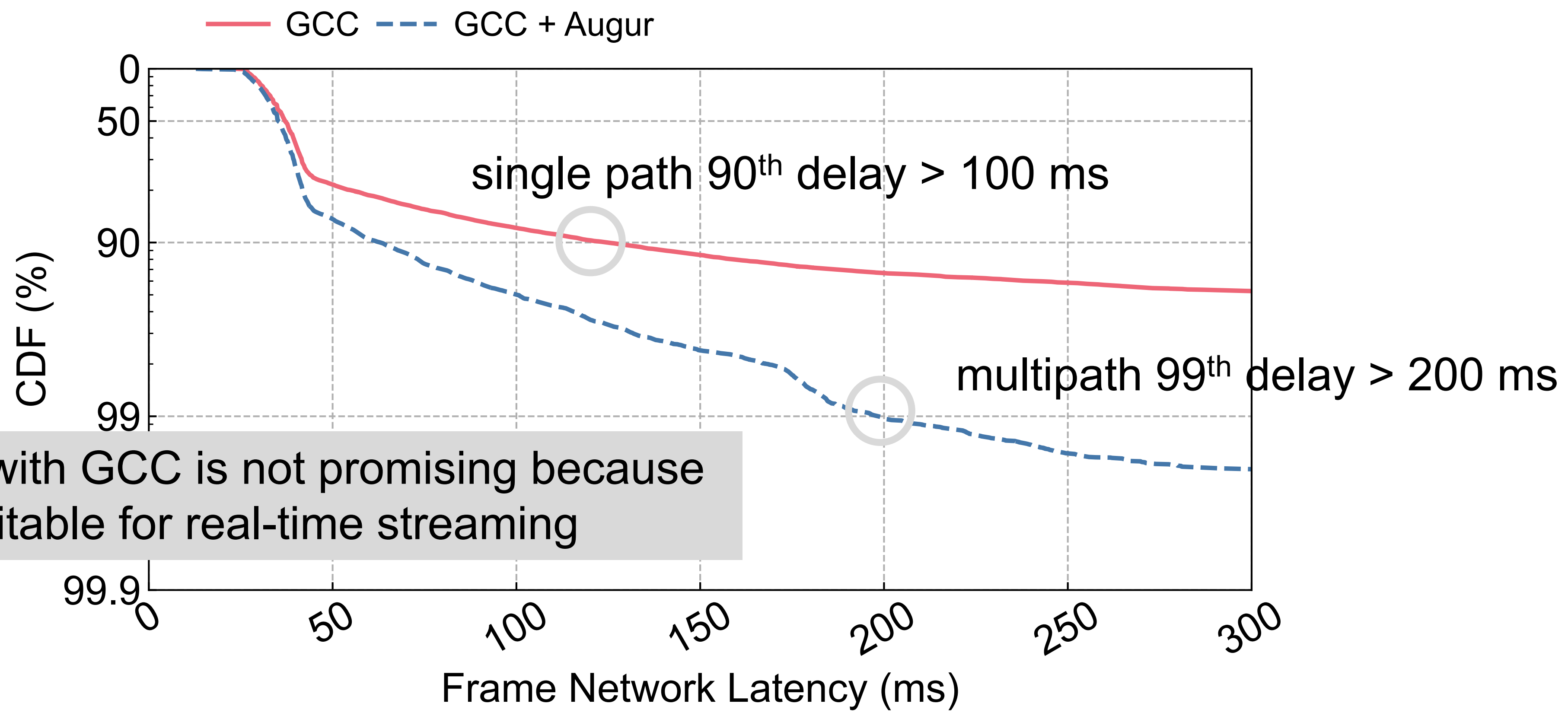
Trace-Driven Emulation

- Run AUGUR with Wi-Fi and cellular traces collected from user sessions
  - Cooperate with different congestion control algorithms (CCAs): GCC, SQP, Salsify



Trace-Driven Emulation

- Run AUGUR with Wi-Fi and cellular traces collected from user sessions
  - Cooperate with different congestion control algorithms (CCAs): GCC, SQP, Salsify, Pudica[1]



Performance with GCC is not promising because GCC is *not* suitable for real-time streaming

[1] Pudica: Toward Near-Zero Queuing Delay in Congestion Control for Cloud Gaming, USENIX NSDI, 2024

Trace-Driven Emulation

- Run AUGUR with Wi-Fi and cellular traces collected from user sessions
  - Cooperate with different congestion control algorithms (CCAs): GCC, SQP, Salsify, Pudica[1]

Performance with GCC is not promising because GCC is ...

AUGUR should cooperate with a CCA specially designed for real-time streaming (No congestion in WAN)

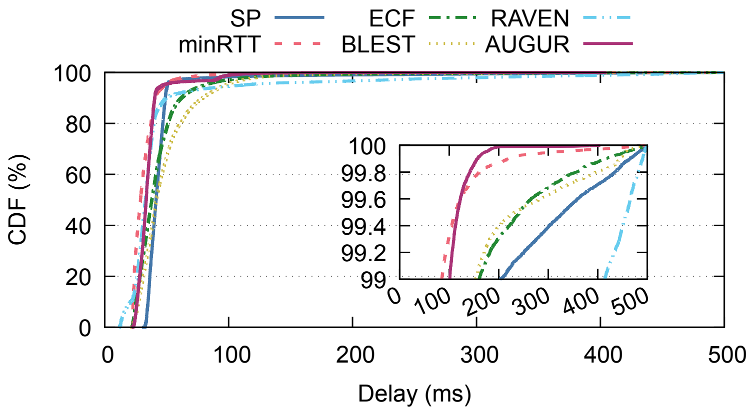
Frame Network Latency (ms)

[1] Pudica: Toward Near-Zero Queuing Delay in Congestion Control for Cloud Gaming, USENIX NSDI, 2024

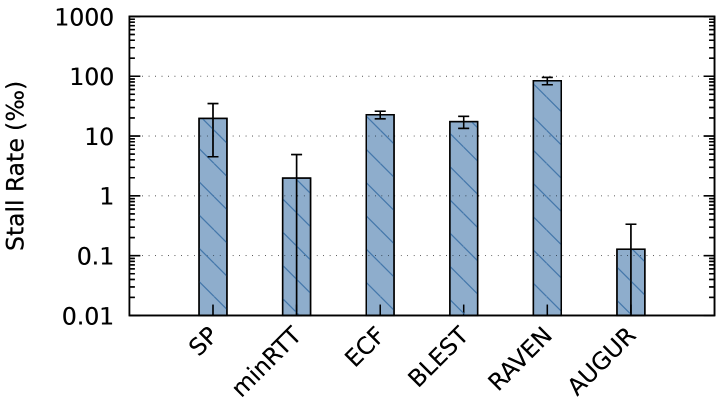
Trace-Driven Emulation

- Run AUGUR with real-time streaming CCAs against other multipath schemes

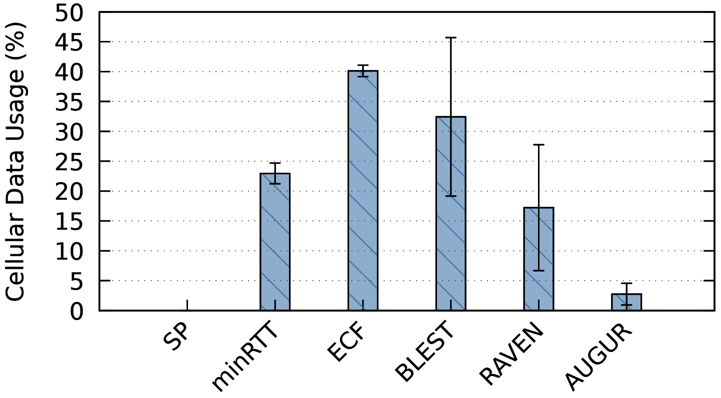
Salsify



Latency: 99th percentile ~100 ms

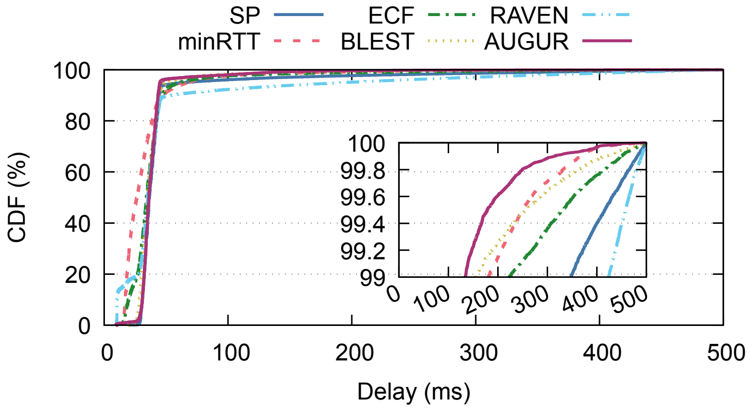


Stall rate: ~0.01%

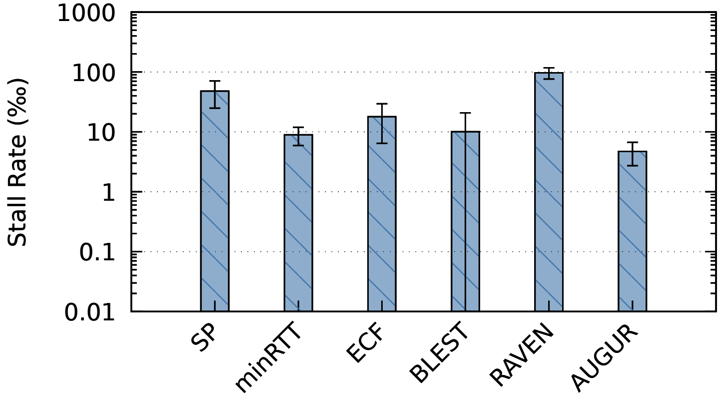


Cellular data usage: ~5%

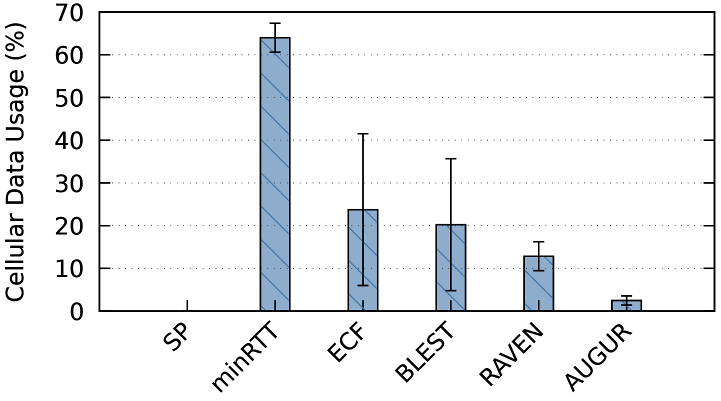
SQP



Latency: 99th percentile ~140 ms



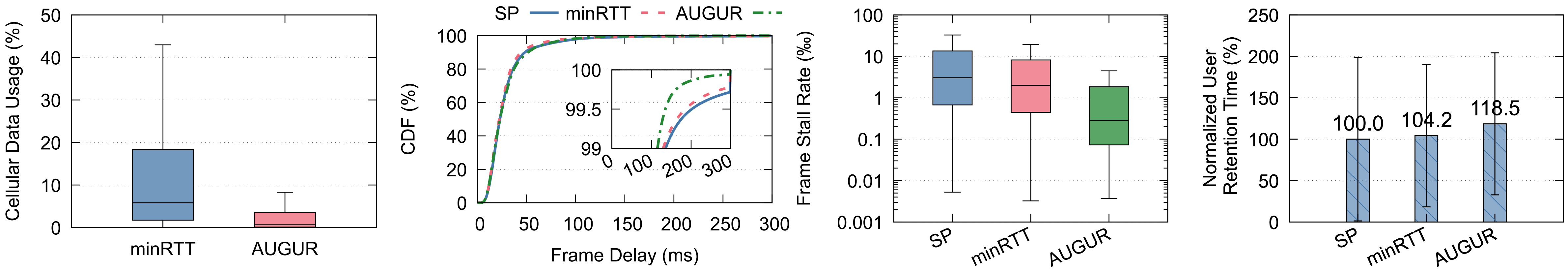
Stall rate: ~0.4%



Cellular data usage: ~2%

Deployment in the Wild

- Deployed in Tencent Start cloud gaming for over 1 year with over 1 million users
  - Pure server-side modification & easily deployable
  - User space implementation for simplicity
- Large-scale A/B test covering 11840 user sessions for 2 weeks with Pudica[1] CCA



- Latency: 99th percentile ~110 ms
- Stall rate: ~0.03%, reduced by 85.7%
- Cellular cost: < 5%, reduced by 3.7x
- User retention time: increased by 18.5%

[1] Pudica: Toward Near-Zero Queuing Delay in Congestion Control for Cloud Gaming, USENIX NSDI, 2024

Takeaways

- Why this work?
  - Wi-Fi wireless fluctuation causes last-hop RTT inflation and long tail latency in real time streaming
  - Users want to reduce cellular data cost as much as possible
- What problem do you solve?
  - Use multipath video frame transmission for real-time streaming with minimal cellular cost
- How do you solve it?
  - Capture sessions' unique network characteristics and deriving state probability models
  - Formulate and solve Integer Linear Programming problems
- How well is it?
  - Reduce 85.7% video frame stalls with < 5% cellular data cost in production
- Any Vision?
  - What about devices without multiple wireless network interfaces, e.g. My laptop?

Thanks for Your Listening & Any Questions?

Acknowledgement

- Collaborators:
  - Tingfeng Wang, Liying Wang, Nian Wen, Rui Han, Jing Wang, Chenglei Wu, Jiafeng Chen, Longwei Jiang, Shibo Wang, Honghao Liu, Chenren Xu

北京大學 PEKING UNIVERSITY

Tencent 腾讯

WAIS LAB

WIRELESS AI FOR SCIENCE