Mortise: Auto-tuning Congestion Control to Optimize QoE via Network-Aware Parameter Optimization

Yixin Shen, Ruihua Chen, Bo Wang, Jing Chen, Haochen Zhang,
Minhu Wang, Yan Liu, Mingwei Xu, Zili Meng

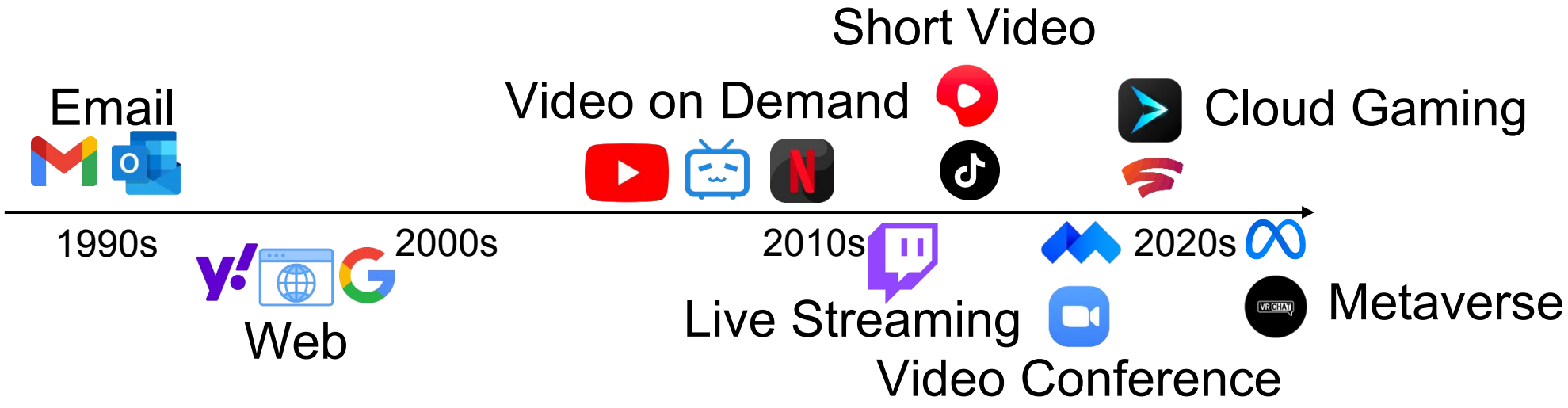
清華大學
Tsinghua University

ByteDance
字节跳动

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

Background

- **Quality of Experience (QoE)** is critical for content providers
  - Modeling from application level metrics, e.g. stall ratio
- Besides app-layer mechanisms, recent studies highlight the pivotal role of **Congestion Control Algorithms (CCAs)**



Background

- **CCAs** shape user experience through the tradeoff between transport layer Quality of Service (QoS) metrics:
  - **Throughput, Latency, Packet Loss**

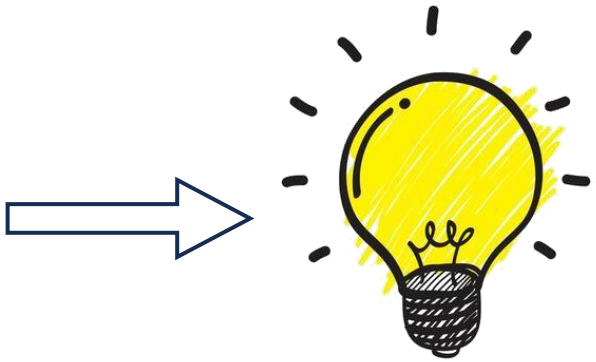
The Critical Question

How do we balance these competing transport layer QoS metrics to maximize application layer QoE?

A Common Case …

- Apparently, different apps want to choose the appropriate CCA or parameter values to match their preferences.
- **However**, operators can only intuitively choose upon their expertise

Video Conference



I need a low latency CCA …

GCC
NADA
Copa

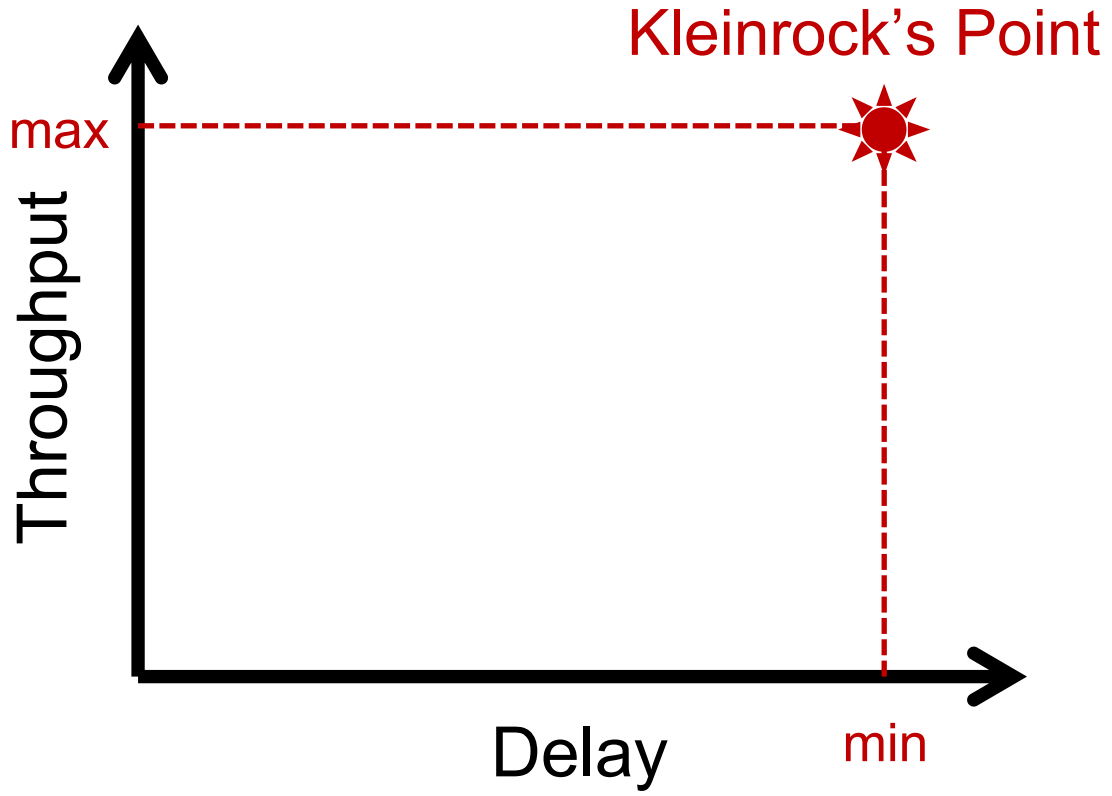
But which one is better?

Observation #1: The Inevitable Tradeoff

- **CCAs all face the tradeoff between QoS metrics**

The Ideal: Kleinrock's Point

- Perfect bandwidth utilization
- Empty network queues
- Zero packet loss



Observation #1: The Inevitable Tradeoff

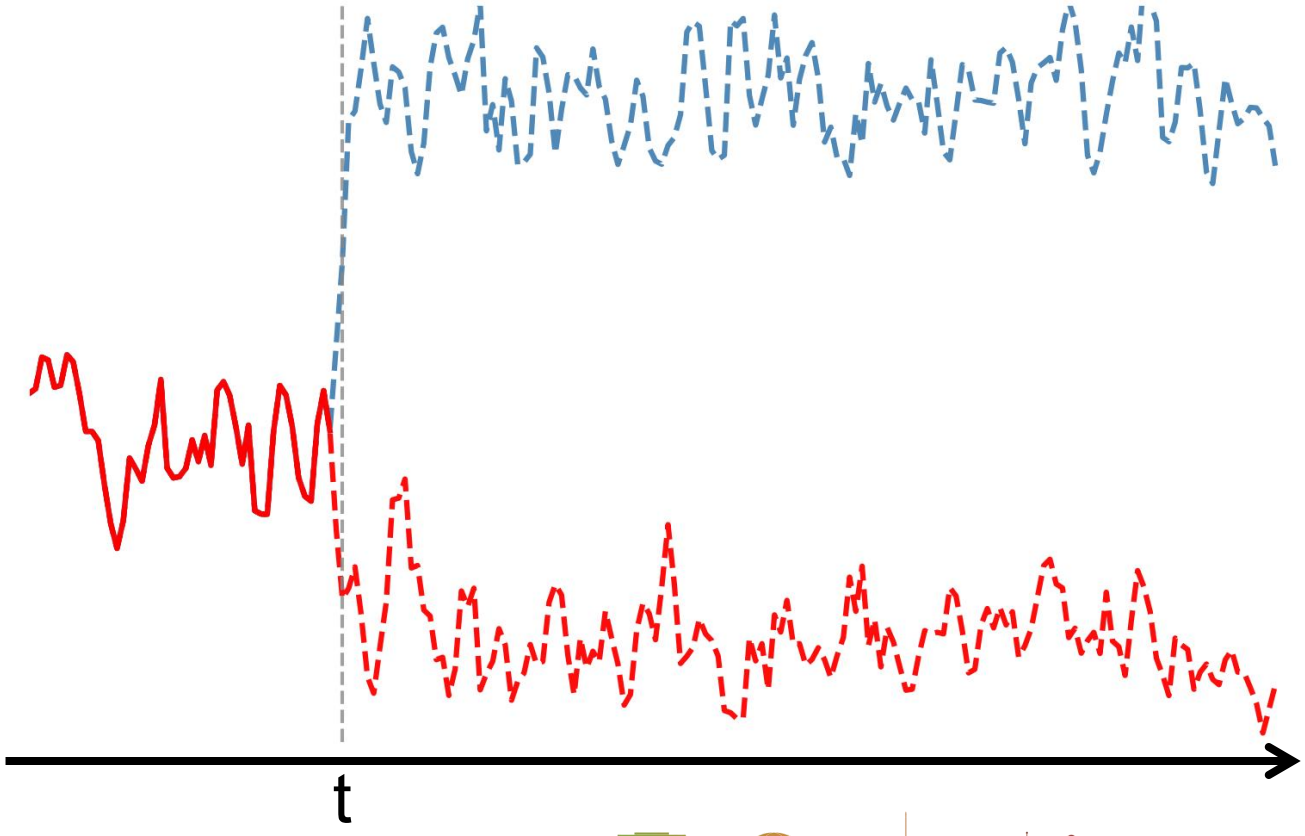
- **CCAs all face the tradeoff between QoS metrics**

The Ideal: Kleinrock's Point

- Perfect bandwidth utilization
- Empty network queues
- Zero packet loss

Practically Impossible!

- Unpredictable fluctuations



Observation #1: The Inevitable Tradeoff

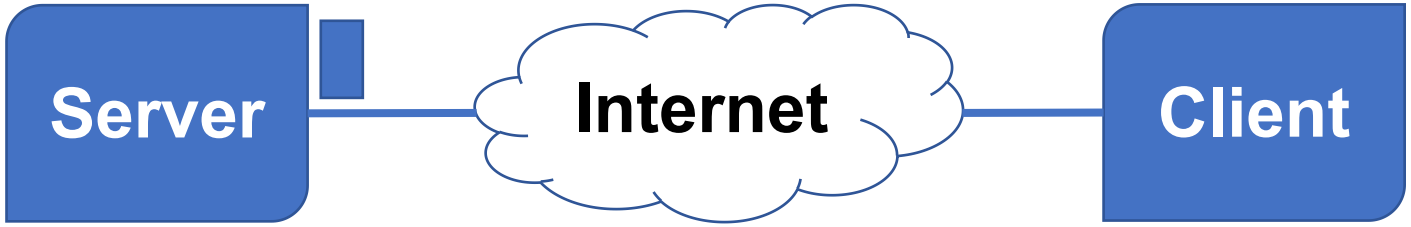
- **CCAs all face the tradeoff between QoS metrics**

The Ideal: Kleinrock's Point

- Perfect bandwidth utilization
- Empty network queues
- Zero packet loss

Practically Impossible!

- Unpredictable fluctuations
- Inevitable control loop delay



Observation #1: The Inevitable Tradeoff

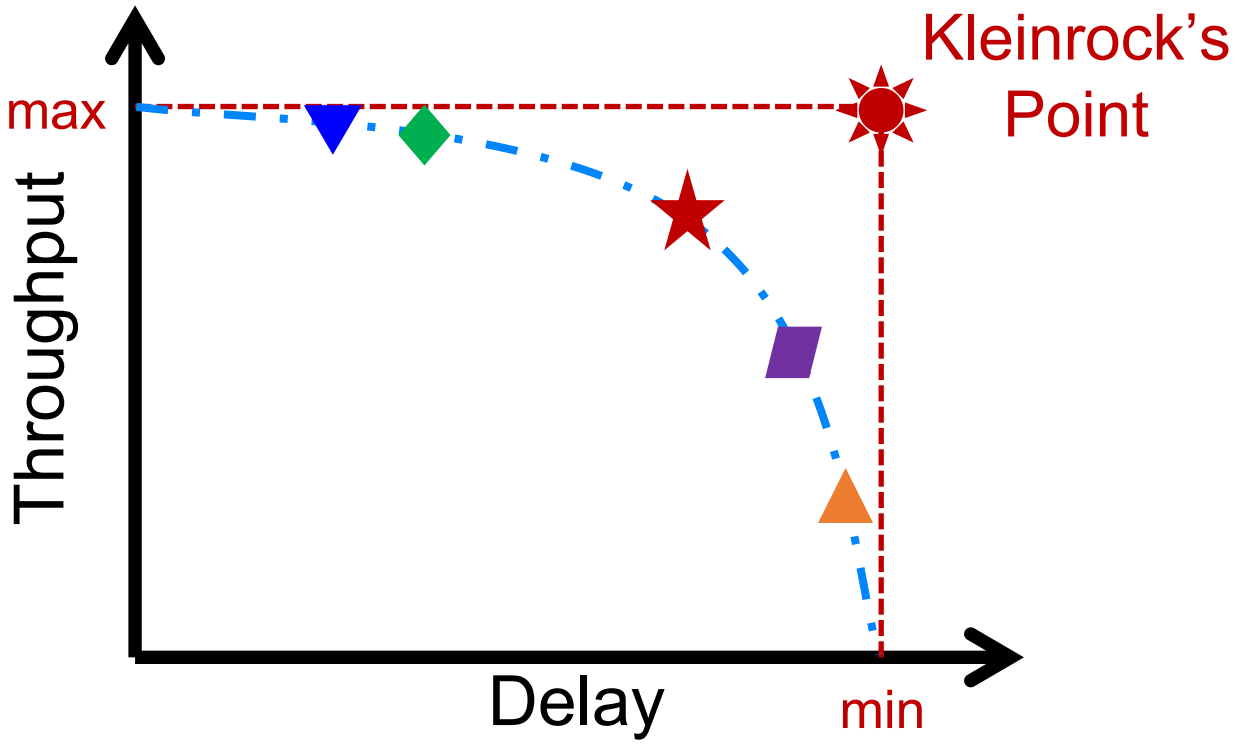
- **CCAs all face the tradeoff between QoS metrics**

The Reality: Pareto Frontier

- Actively probe for network state
  - Probing is uncertain!
  - Aggressive:

    high throughput & high delay
  - Conservative:

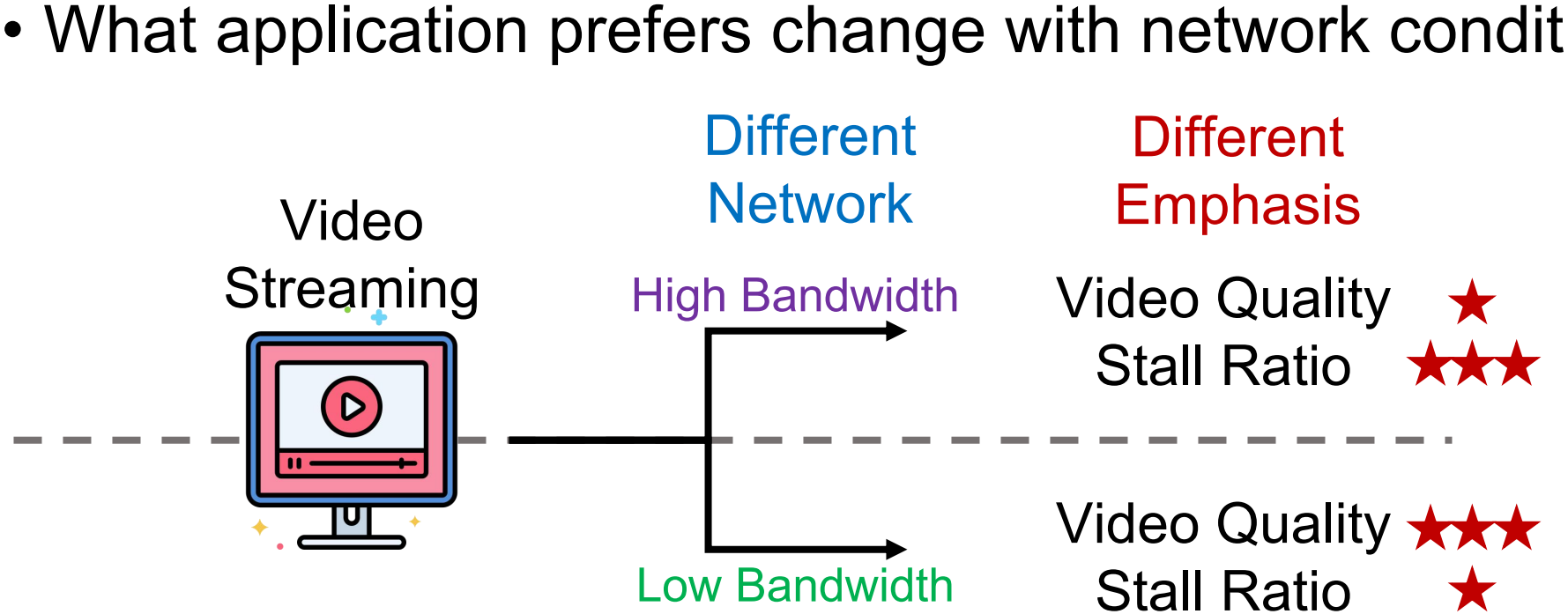
    low throughput & low delay

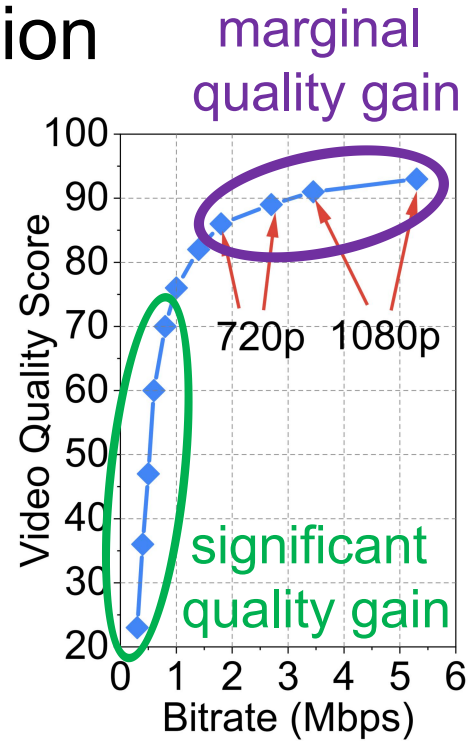


Each CCA operates at a different operating point along the tradeoff spectrum, depending on its internal strategy

Observation #2: Network-Dependent Optimal Operating Points

- What application prefers change with network condition

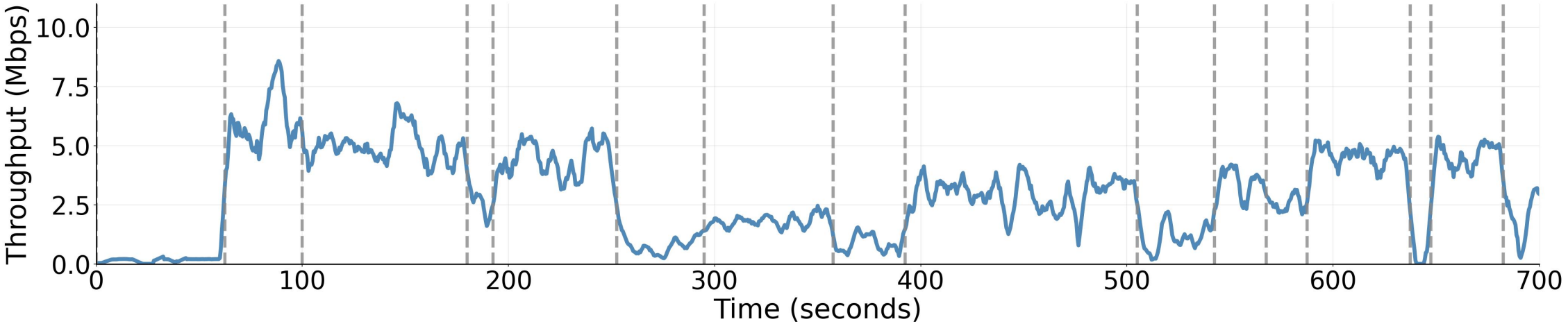




The optimal tradeoff among QoS metrics and thus the optimal operating point is inherently network-dependent

Observation #3: Rapid Network Changes

- Network conditions experience substantial changes with frequently within a session
  - Condition changes are significant and unpredictable



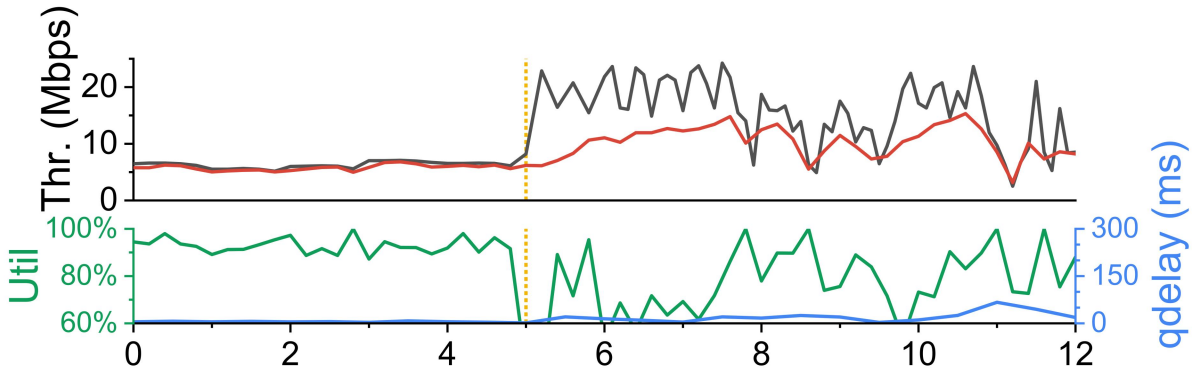
Real-time and continuous identification and achievement of optimal operating points are necessary

Key Insight

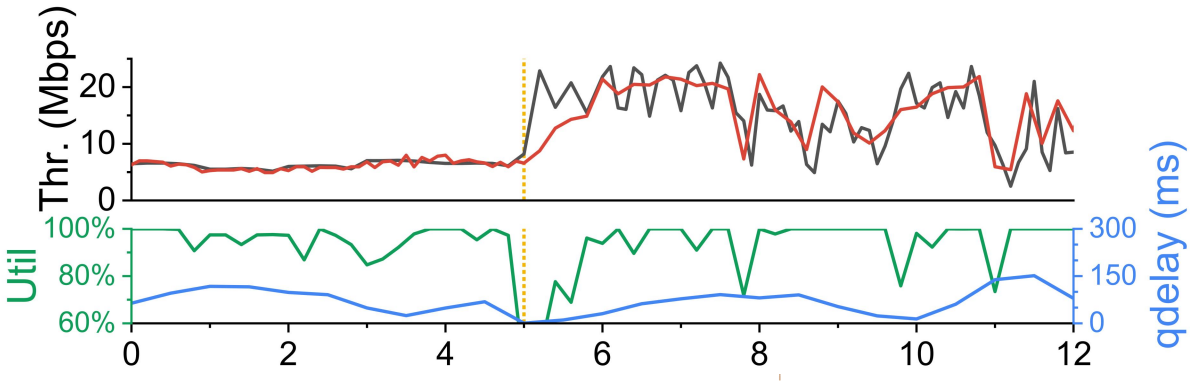
CCAs must adapt their parameter configurations (tradeoff) in real time to track the optimal operating point varied with the evolving network state

Why Current Solutions Fail?

- **Heuristic and Static Configuration**
  - Parameters are set qualitatively and fixed throughout session
  - Cannot maintain optimal QoE under dynamic conditions
- **Limited Flexibility and Poor Adaptability**
  - Transfer learning is intuitive but requires minutes to hours
  - CCA selection are limited to a few discrete operating points



Conservative Copa (δ = 0.5)



Aggressive Copa (δ = 0.04)

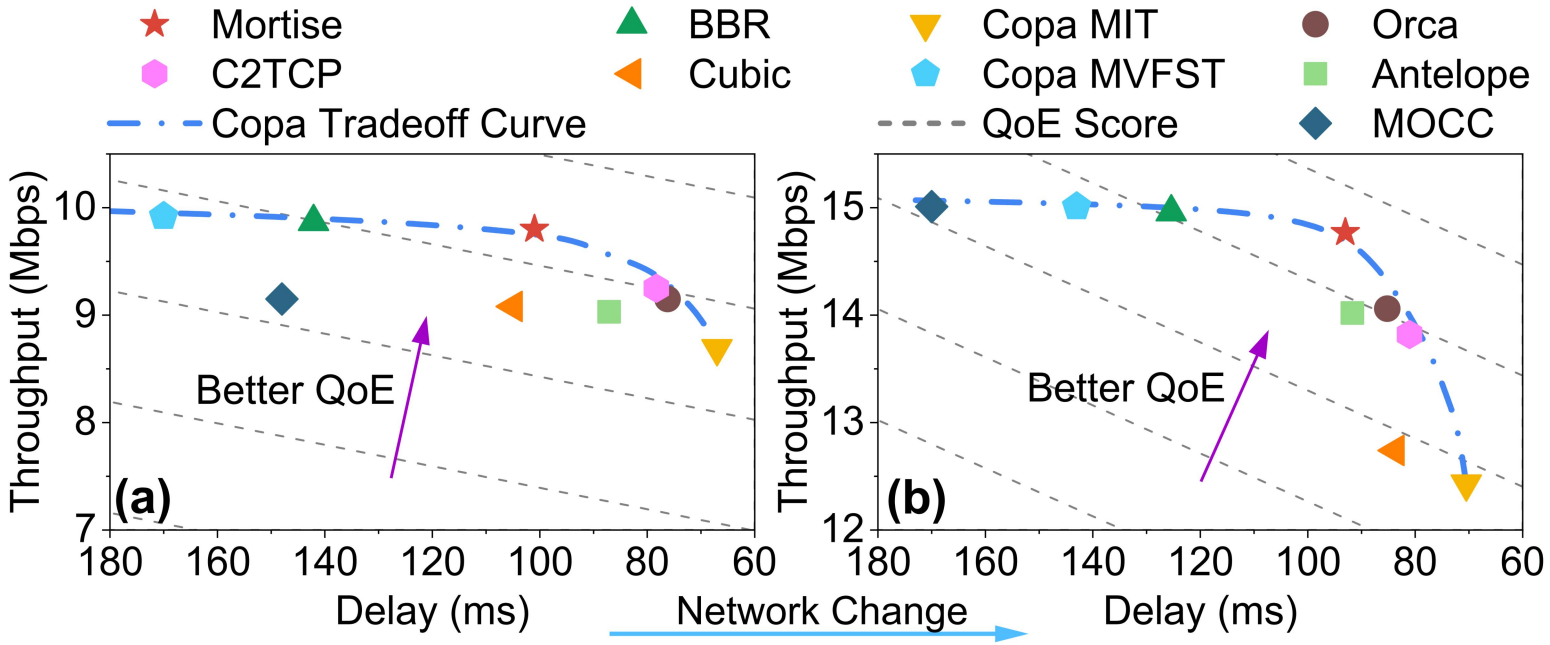
It is challenging …

CCAs must adapt their parameter configurations (tradeoff) in real time to track the optimal operating point varied with the evolving network state

- **Modeling Complexity** between parameters, network states and QoE
- **Metric Mismatch** of CCAs optimizing for QoS whereas QoE depends on app-level semantics
- **Feedback Latency** from QoE metrics typically only available upon session completion

Mortise: Our Choice

- Build on rule-based CCAs with explicit network models
  - Higher computational efficiency
  - Can achieve pareto-optimal performance competitively
  - Flexible and frequent parameter tuning for arbitrary tradeoff



Mortise: QoS Tradeoff as Proxy

- Introduce the QoS tradeoff as a proxy

QoE Model —Direct Mapping→ **CCA Parameters**

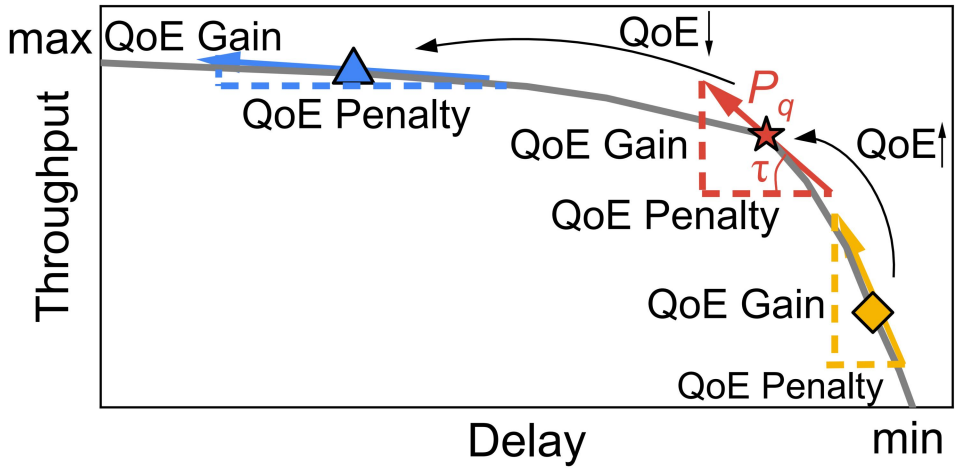
Mortise: QoS Tradeoff as Proxy

- Introduce the QoS tradeoff as a proxy

QoE Model → **QoS Tradeoff** (Proxy) → **CCA Parameters**

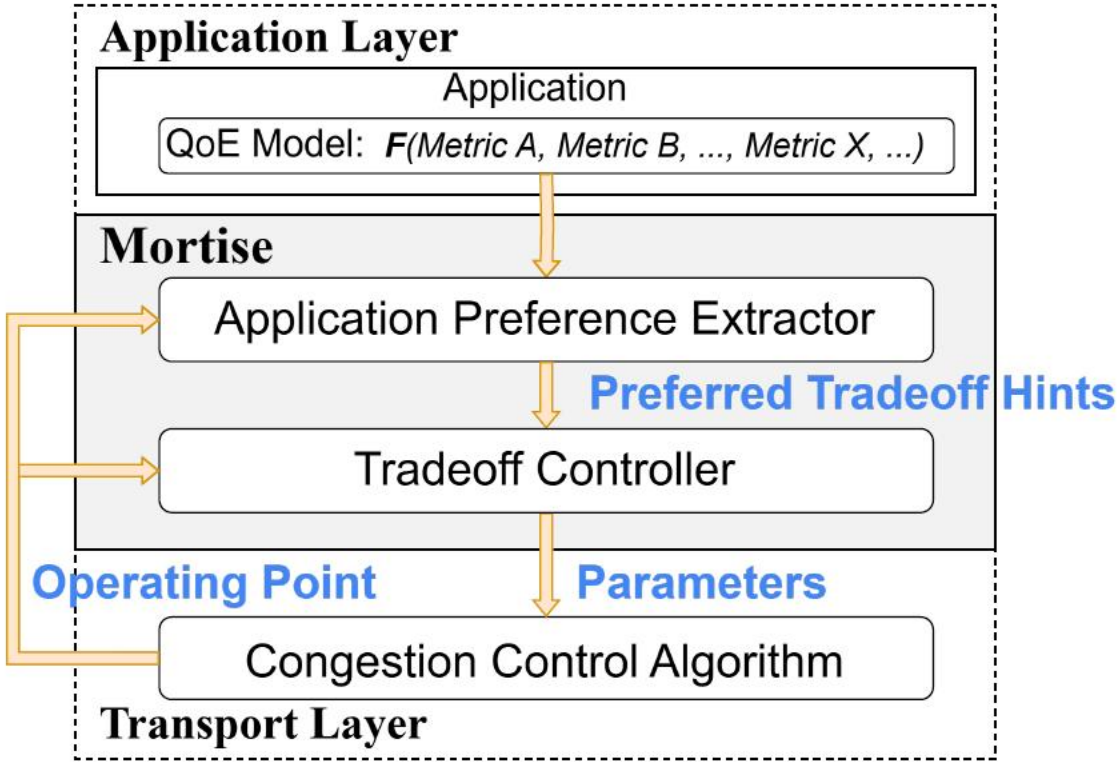
- Decompose the complex problem into two tractable tasks

- Optimal tradeoff and optimal operating point are mutually transformed



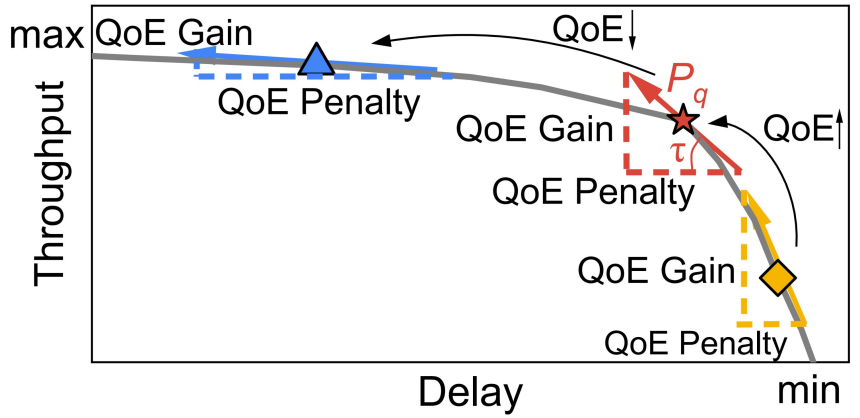
Design Overview

1. How to extract the tradeoffs preferred by QoE under current situation?
2. How to identify the CCA parameter configuration to achieve the preferred tradeoff?



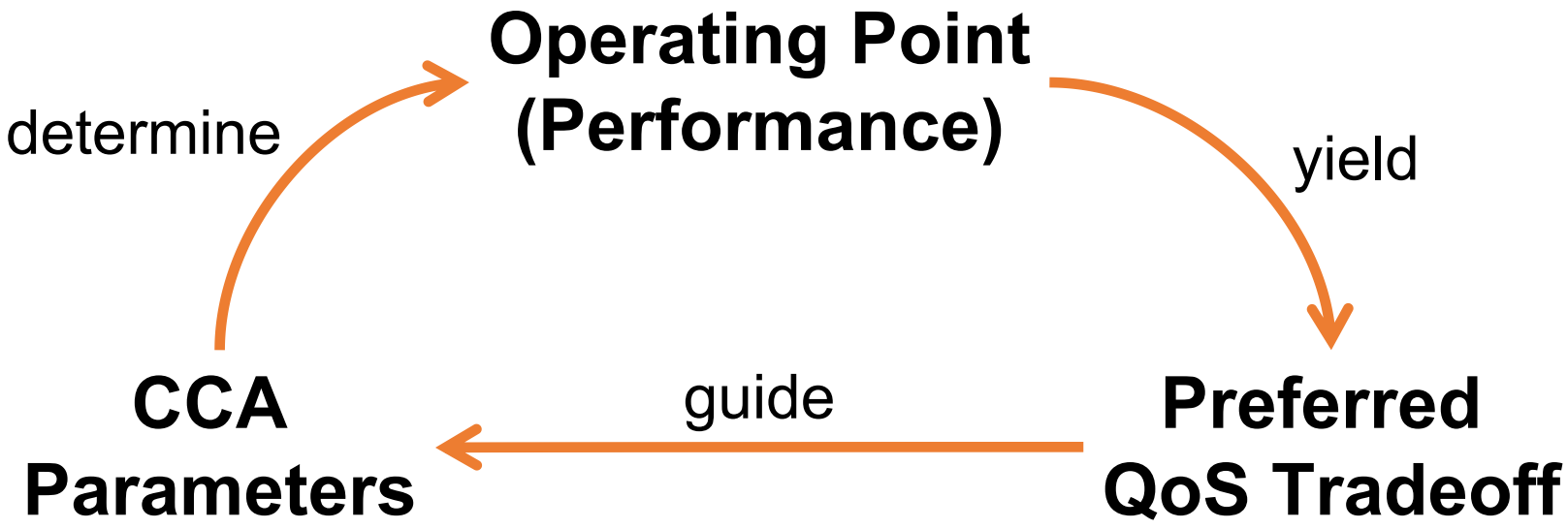
Design: Application Preference Extractor

- **Goal**: extract the application's preferred QoS tradeoff in real-time
- Explicit QoE Models $\mathrm{QoE} = F(M_1, M_2, \ldots, M_n)$
- Translated to QoS form $\mathrm{QoS} = f(T, D, L)$
- Solving the gradient $\nabla f = (\frac{\partial f}{\partial T}, \frac{\partial f}{\partial D}, \frac{\partial f}{\partial L})_{(T_c, D_c, L_c)}$
- Preference Hints
  - $\lambda = |\frac{\partial f}{\partial D} / \frac{\partial f}{\partial T}|$ (Delay-Throughput tradeoff)
  - $\beta = |\frac{\partial f}{\partial L} / \frac{\partial f}{\partial T}|$ (Loss-Throughput tradeoff)



Design: Tradeoff Controller

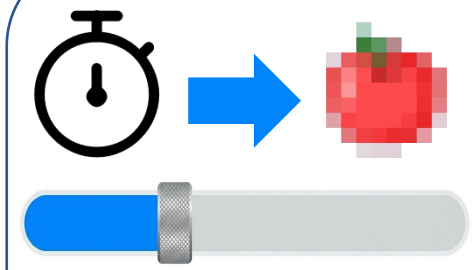
- Cross-dependency between parameters and preferred QoS tradeoffs



Design: Tradeoff Controller

Measurement Noise

Longer Observation = **More Accurate Results**

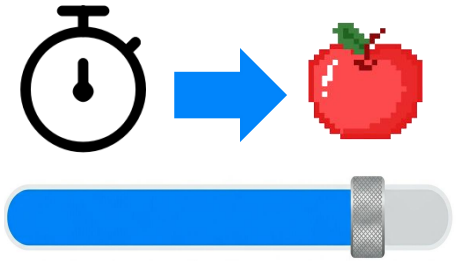


SHORT
Meaurement

Initially

Need: Directional guidance & fast adjustment

Can provide: Rough measurement



LONG
Meaurement

When closer

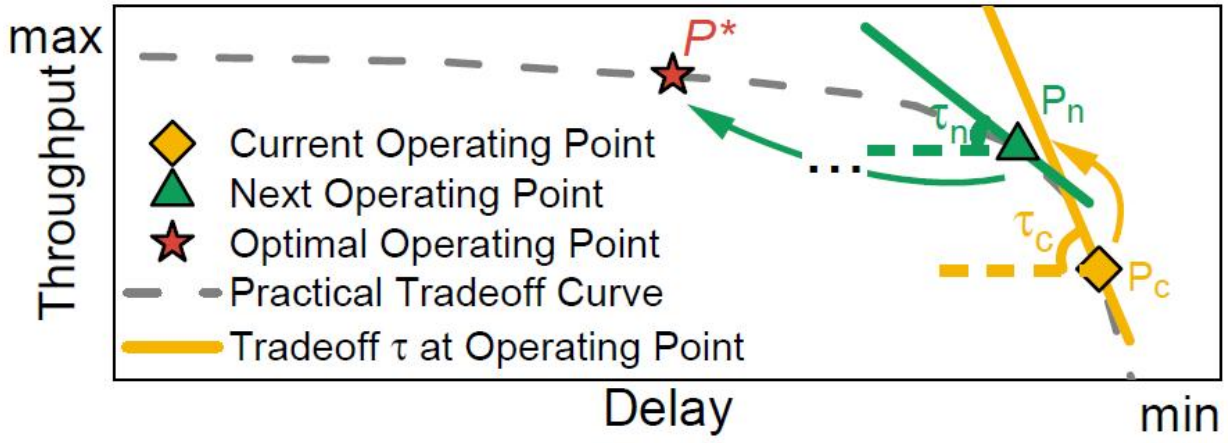
Need: Finer adjustment

Can provide: More accurate measurement by longer observation & historical data

Design: Tradeoff Controller

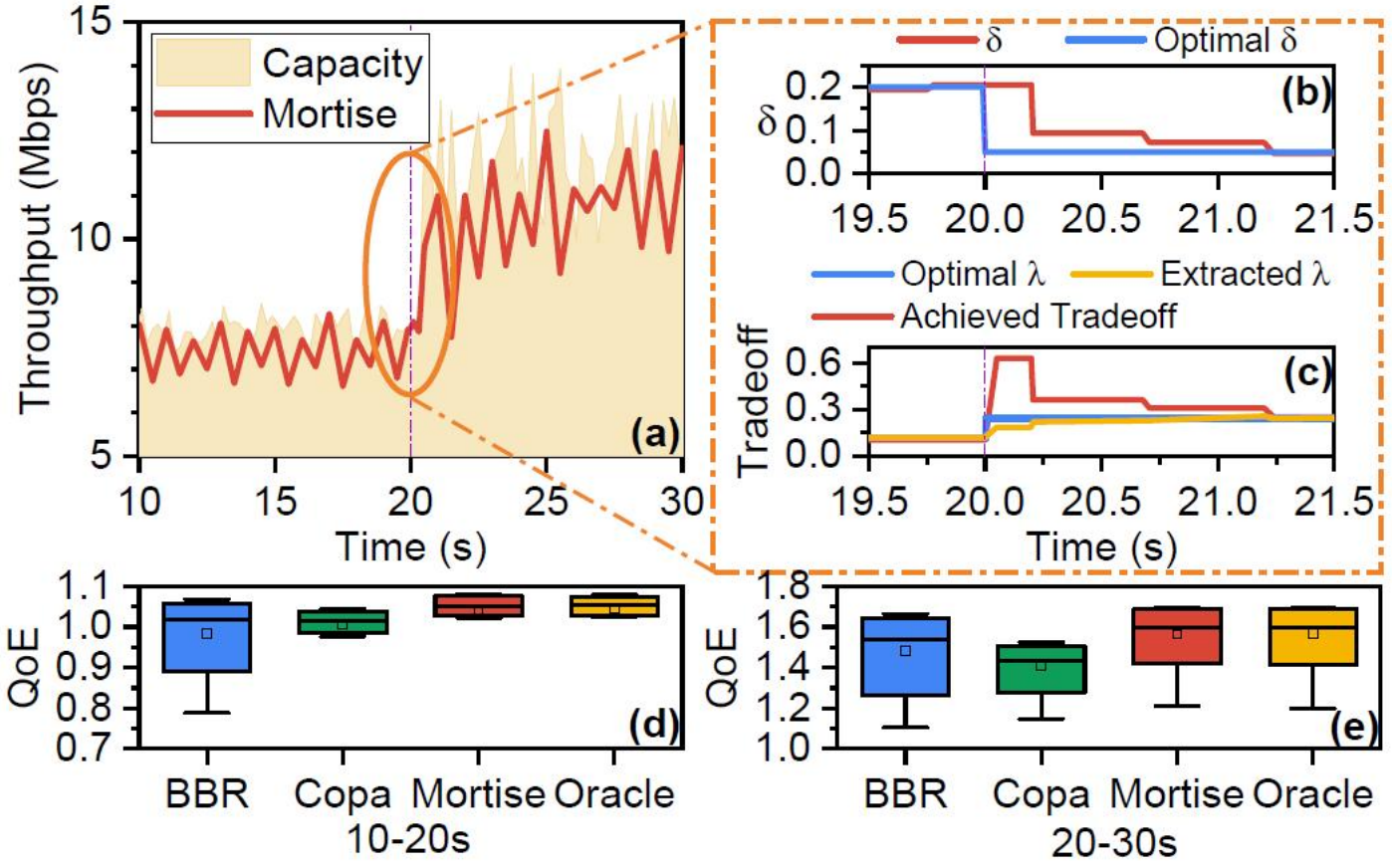
1. Measure Tradeoff (τ_c) of Current Operating Point (P_c)
   - Adaptive measure period
2. The Balancing Feedback Control
   - Move current tradeoff towards extracted preferred hint
3. Adjust the Parameters

Iterate until converge



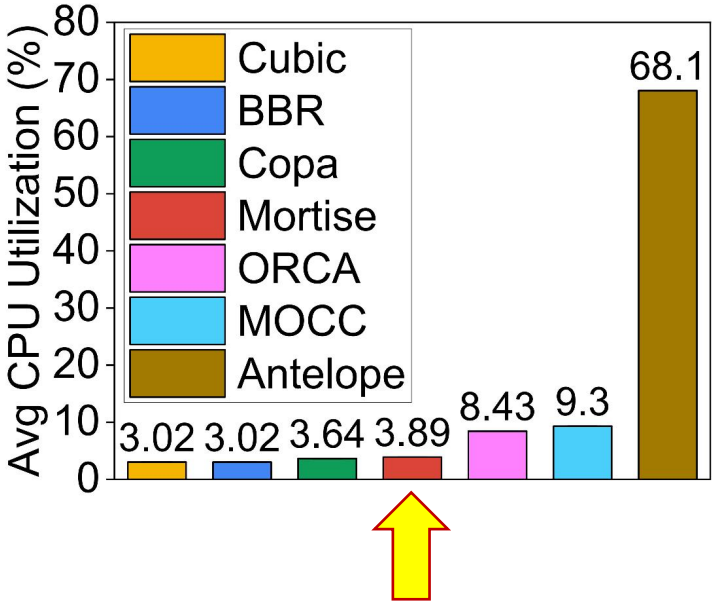
Design: Tradeoff Controller

- An illustrating demo



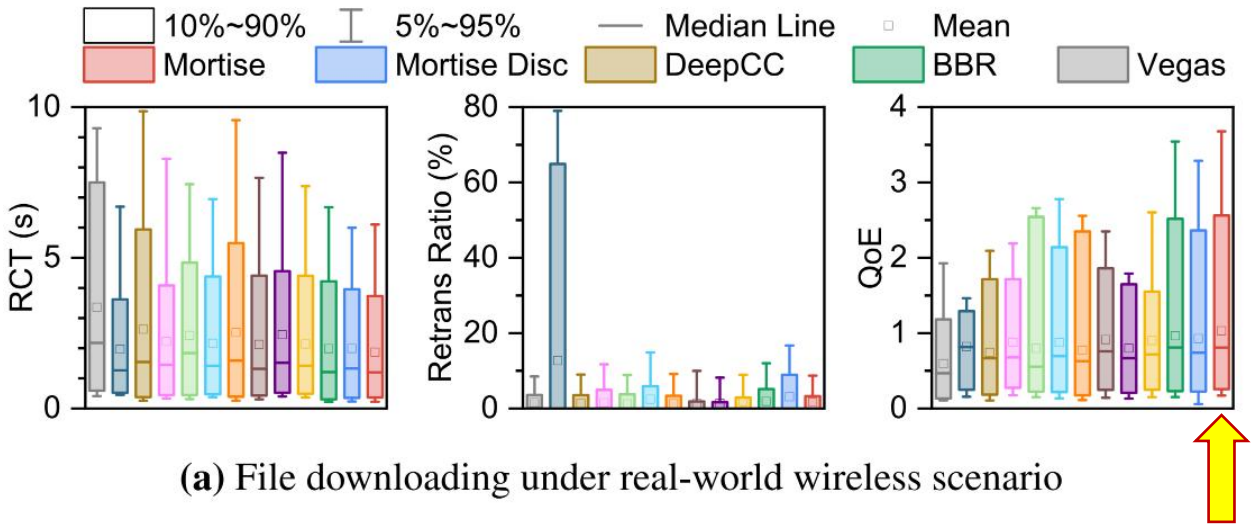
Implementation: Practical & Deployable

- Build on Linux TCP with eBPF
- Based on Copa to adjust its δ
- Low overhead
- Real-World A/B Testing at ByteDance
  - 128 cities worldwide over 3+ months
- Applications: File Downloading & Video Streaming
- Baselines
  - Throughput-oriented: Cubic, BBR
  - Delay-based: Vegas, Copa MIT, Copa MVFST
  - Learning-based: Orca
  - Adaptable Solutions: MOCC, Antelope, Antelope Copa, C2TCP, DeepCC
  - Ours: Mortise Discrete, Mortise

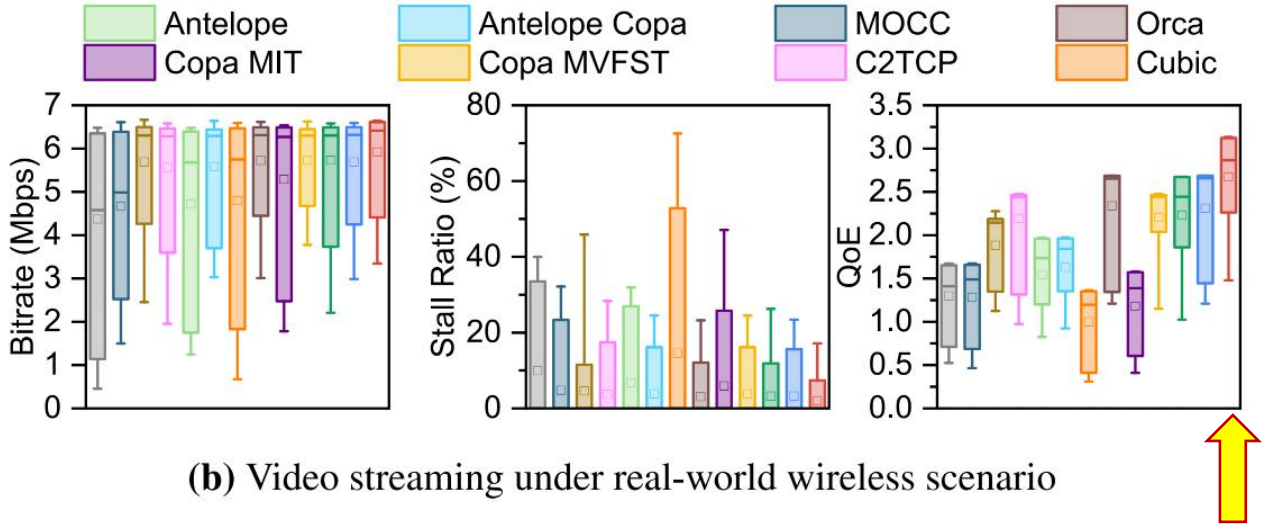


Results: Dramatic QoE Improvements

- Improve both on individual app-level metrics and overall QoE
- Elevate the QoE by 11.3%-73.5% in wireless file downloading and by 19.8%-167.2% in wireless video streaming



(a) File downloading under real-world wireless scenario



(b) Video streaming under real-world wireless scenario

Conclusions

- The optimal configuration is network-dependent and time-varying
- Heuristic and static parameter configuration are insufficient
- Fine-grained, continuous parameter adaptation is essential for optimal application QoE in fluctuating WANs

Mortise continuously and quantitatively adjusts CCA parameters to maintain QoE-optimal operating point as network evolves

Thank You!

Questions?

Paper: Mortise: Auto-tuning Congestion Control to Optimize QoE via Network-Aware Parameter Optimization

Code: https://github.com/BobAnkh/Mortise

Contact: syx@ieee.org zhc@ieee.org