SIGCOMM 2021

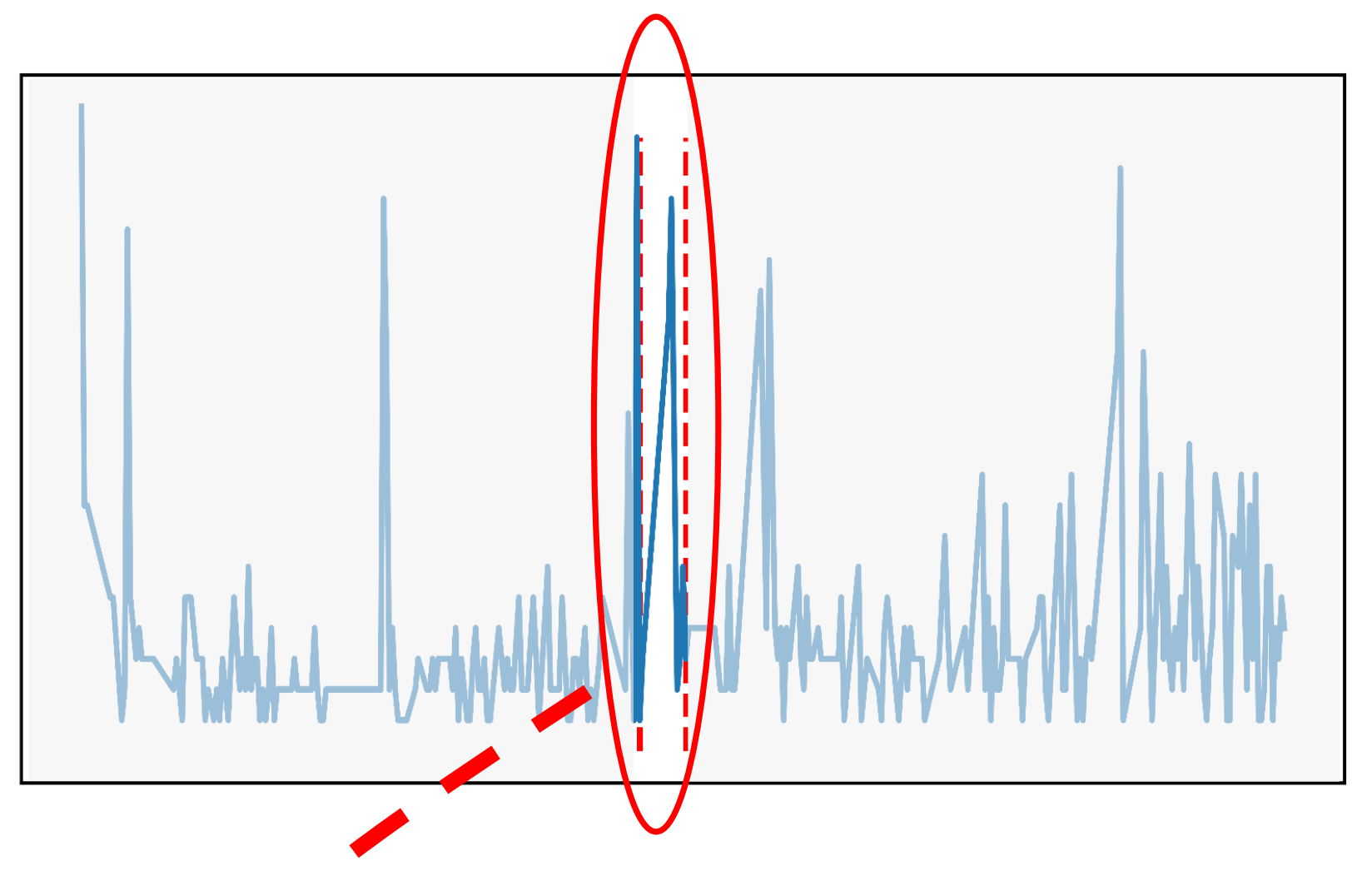
Quantifying the Transient Performance of Congestion Control Algorithms

Yixin Shen, Zili Meng, Jing Chen, Mingwei Xu (INSC, Tsinghua University)

Contact: shen-yx18@mails.tsinghua.edu.cn **Project Repo**: https://github.com/BobAnkh/TPCCA

Background

Network condition



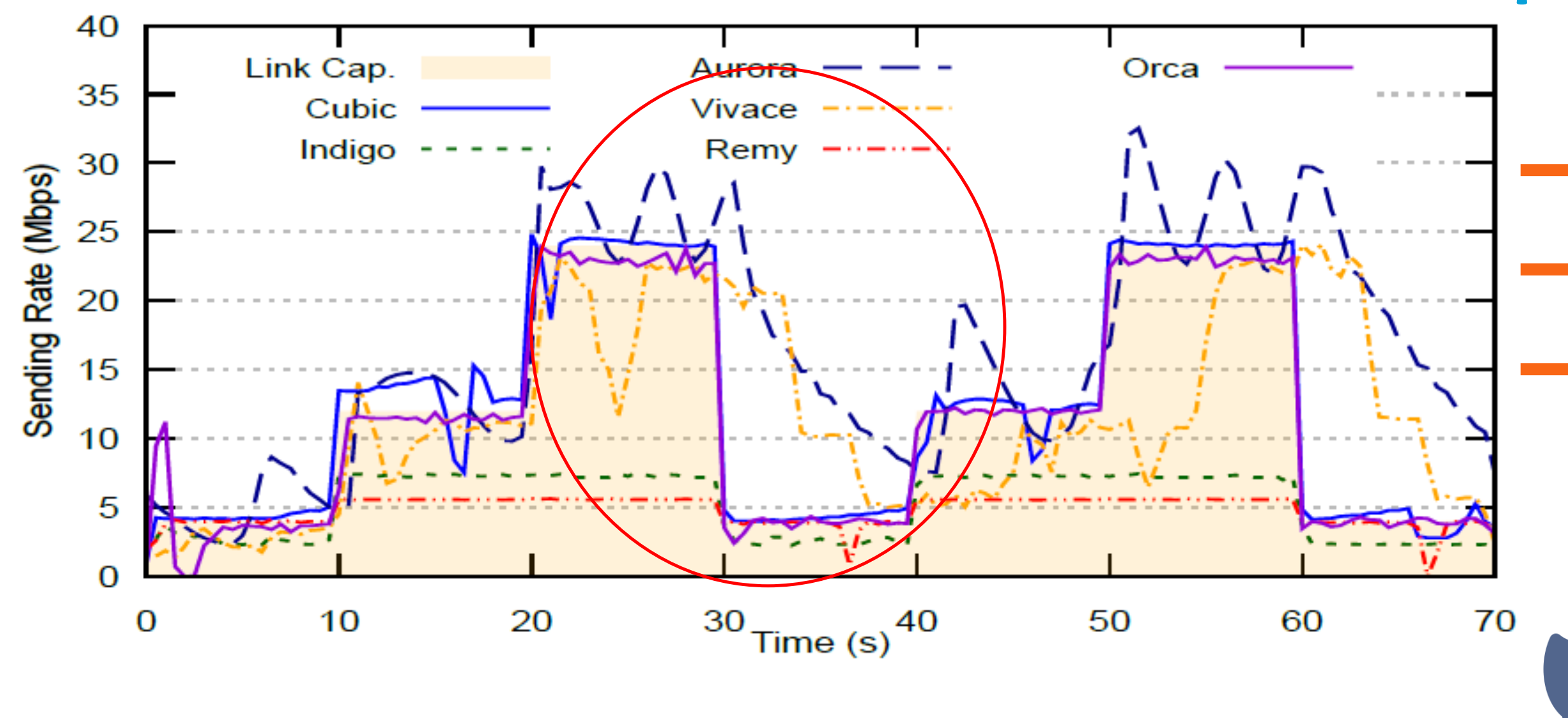
Plethora of different CCAs with complex mechanism

Heuristic: Reno, Cubic, BBR, …

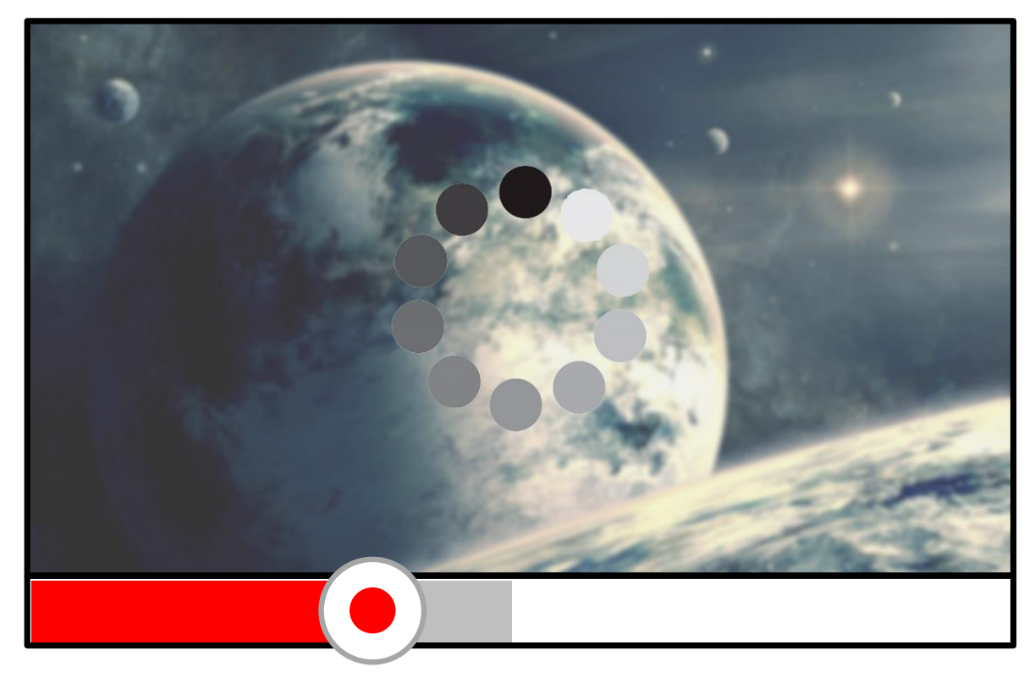
Congestion Control Algorithms

Learning-based: Indigo, Vivace, Aurora, …

- **Not converge within 500×RTT**
- **Overshooting**



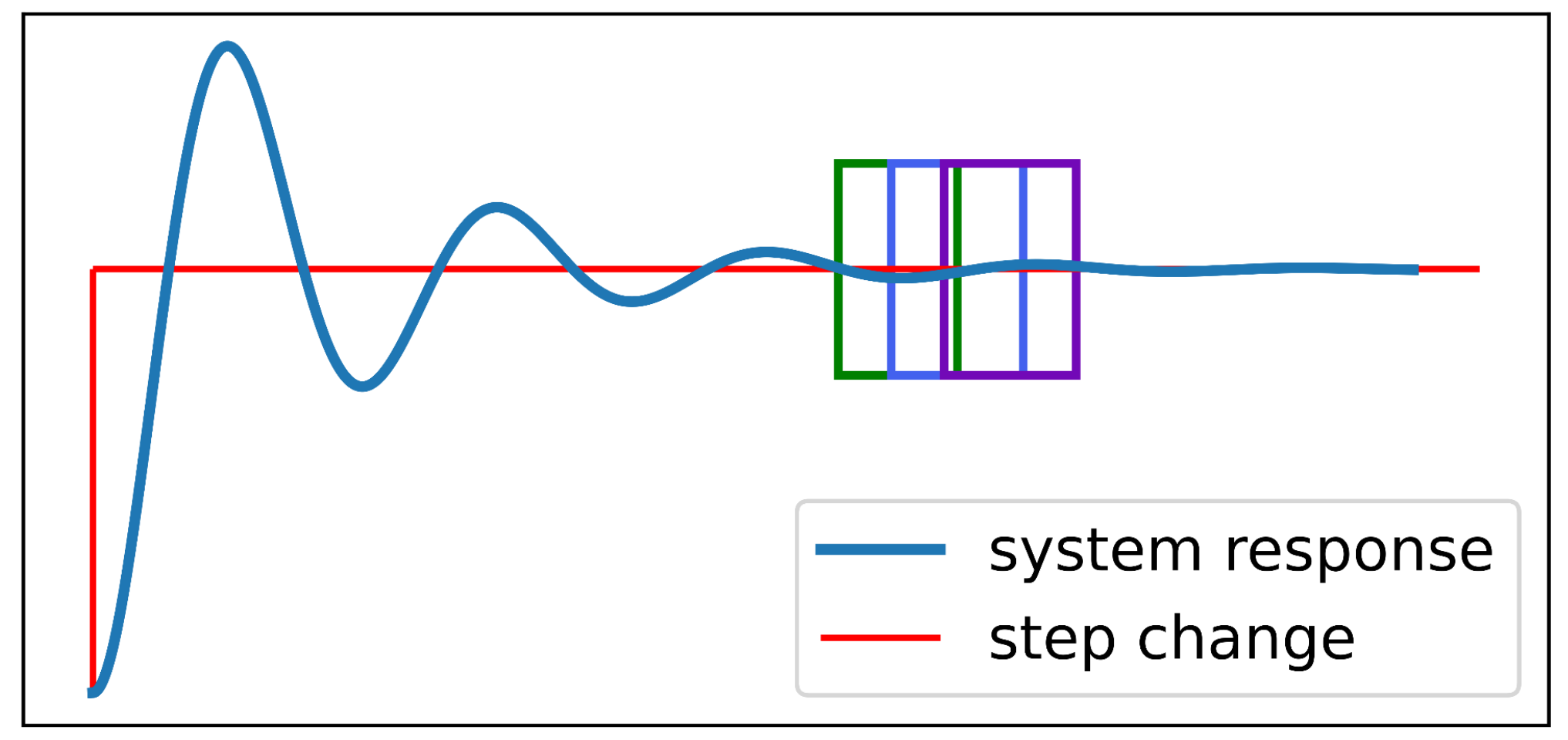
Result in App stalling



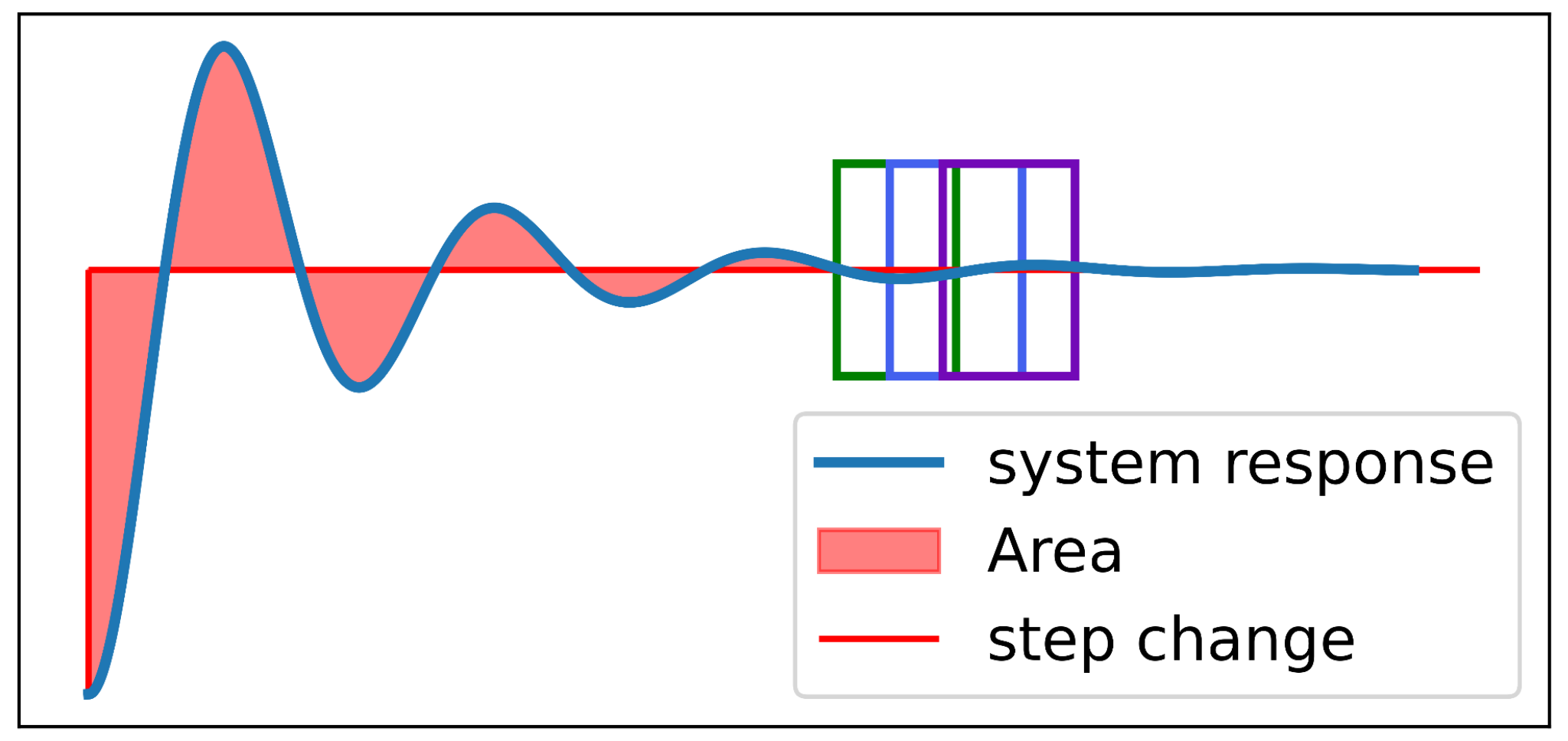
How to quantify the transient performance(e.g. responsiveness) of CCAs?

Design Overview

- Detect the steady state by **consecutive windows**

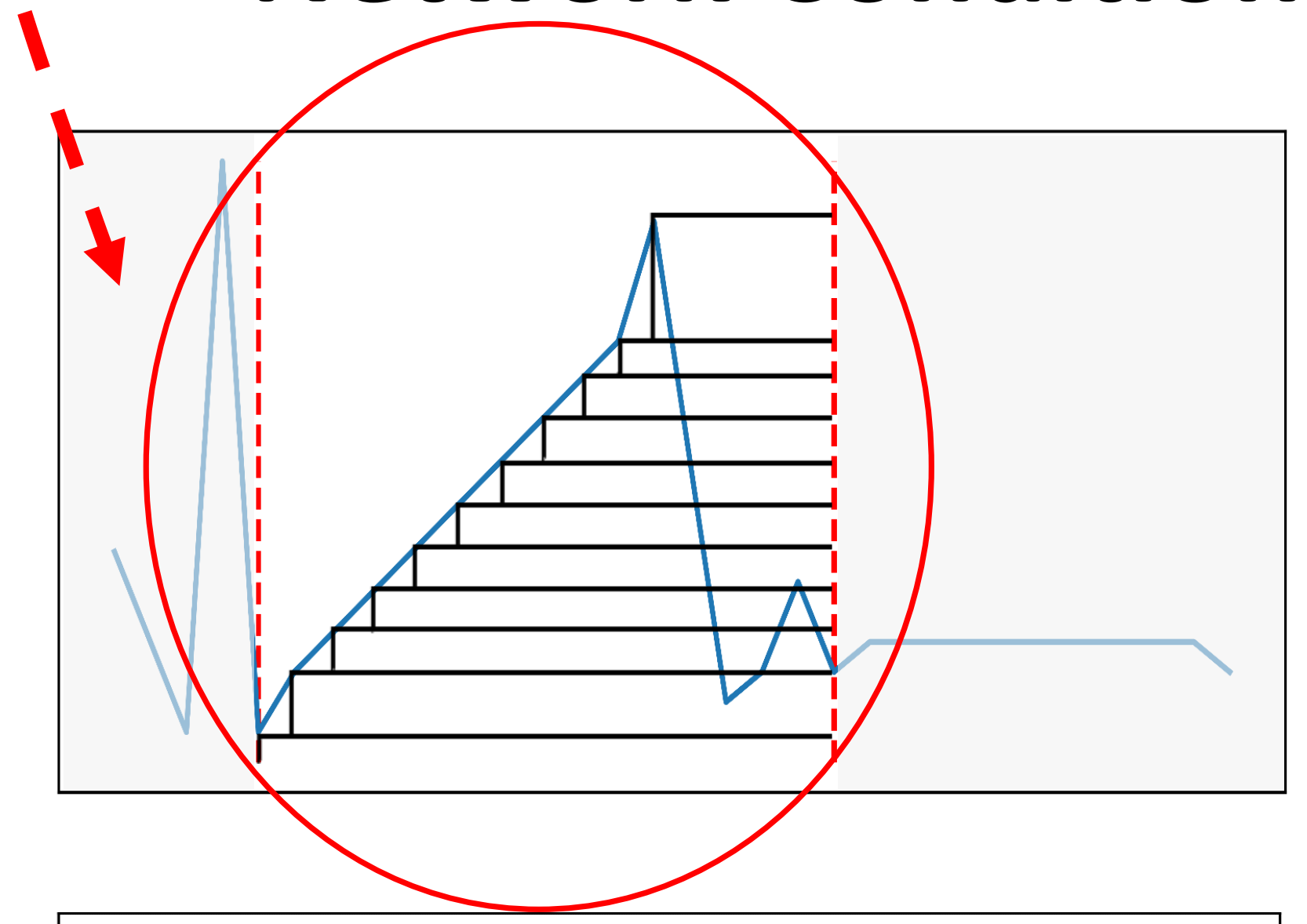


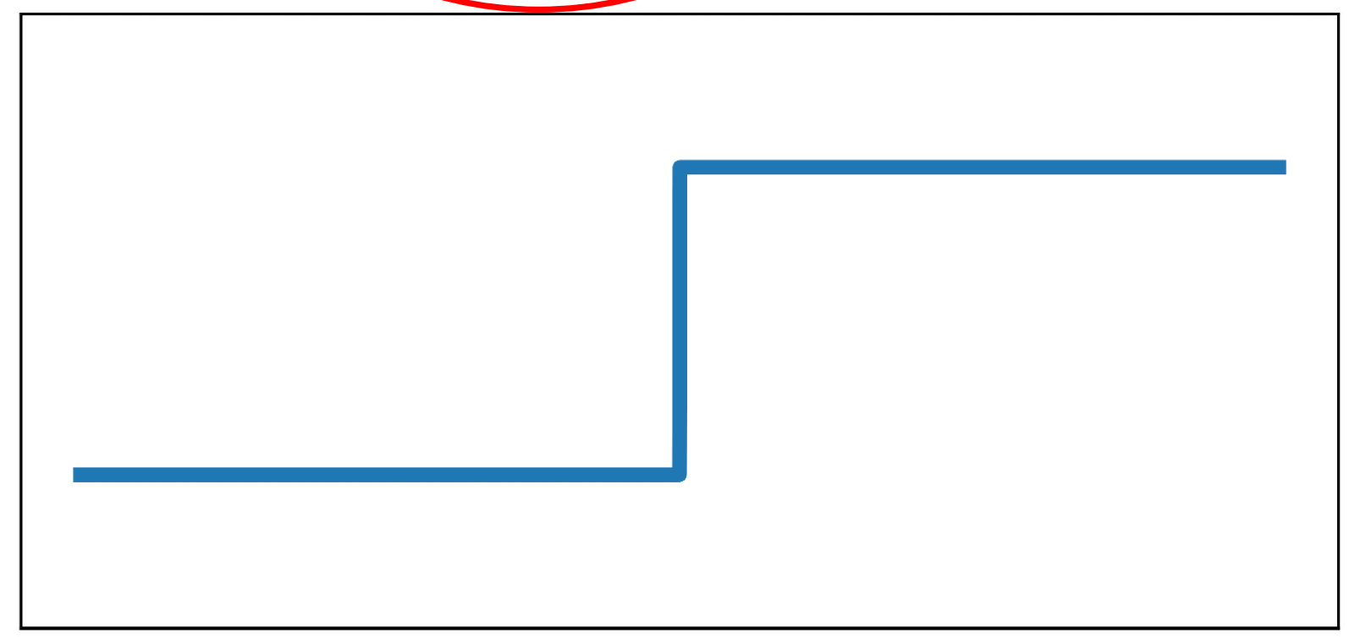
- Quantify the transient performance with the **area** between response and step change before steady state



Our Contributions

C#1: Complex & Changeable Network Condition



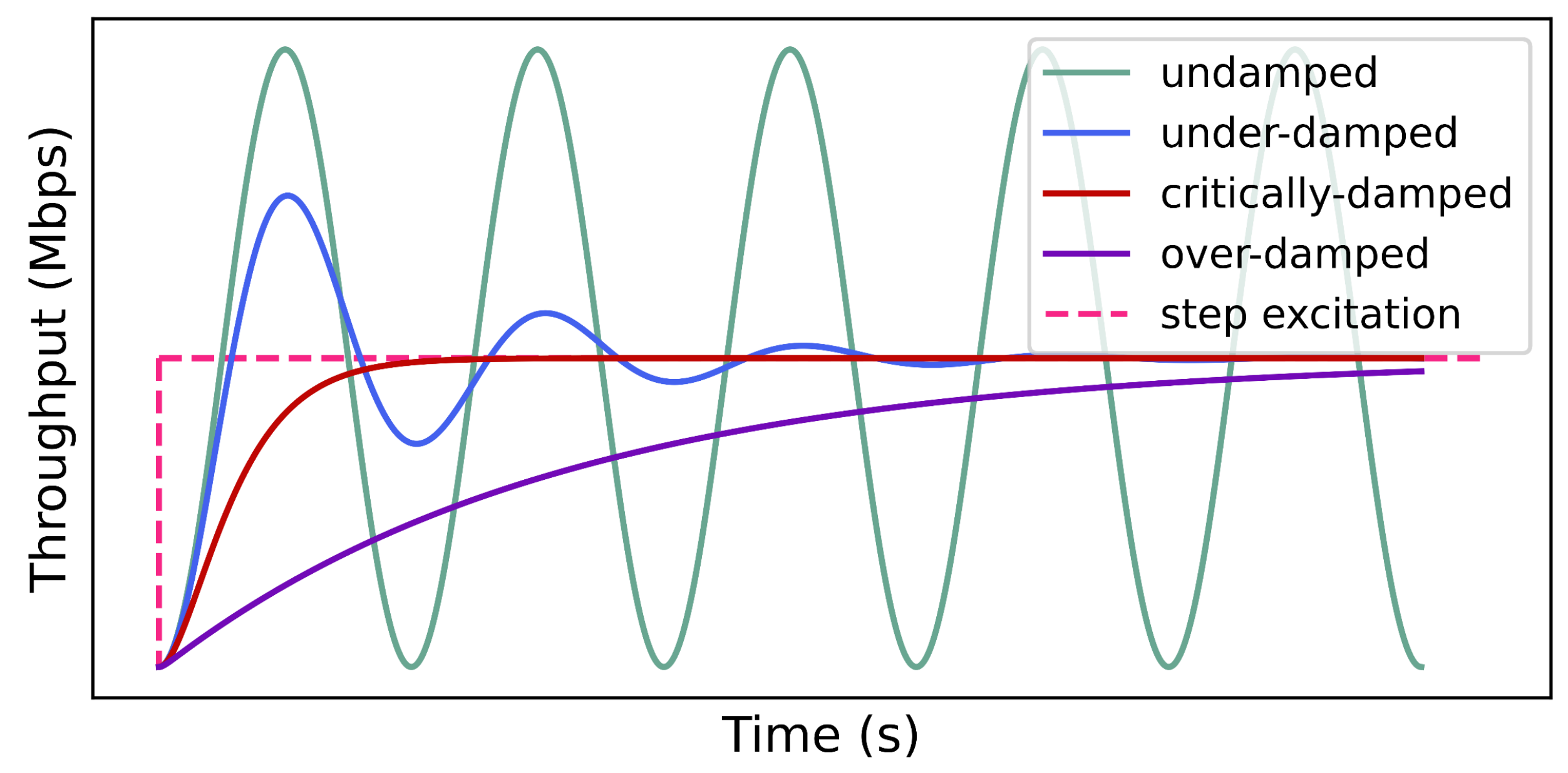


S#1: Treat the network change as the combination of **step changes**

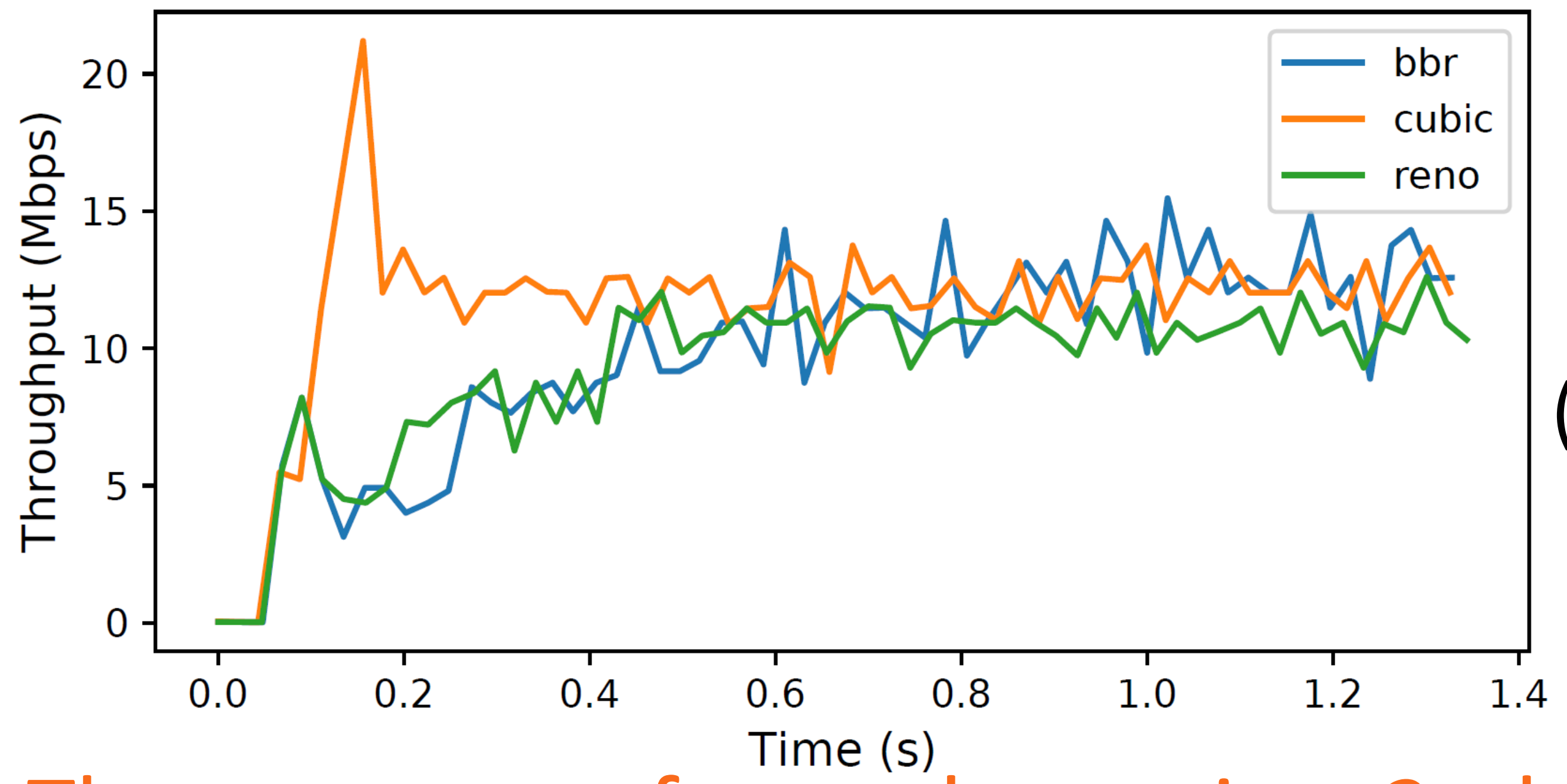
C#2: Complex Nature of CCAs, hard to be mathematically formulated

Second-order System
(measurement of higher-than-2nd-oreder moment in real time is not accurate enough)

S#2: Treat the control system of a CCA as a **second-order system**



Theoretical Response

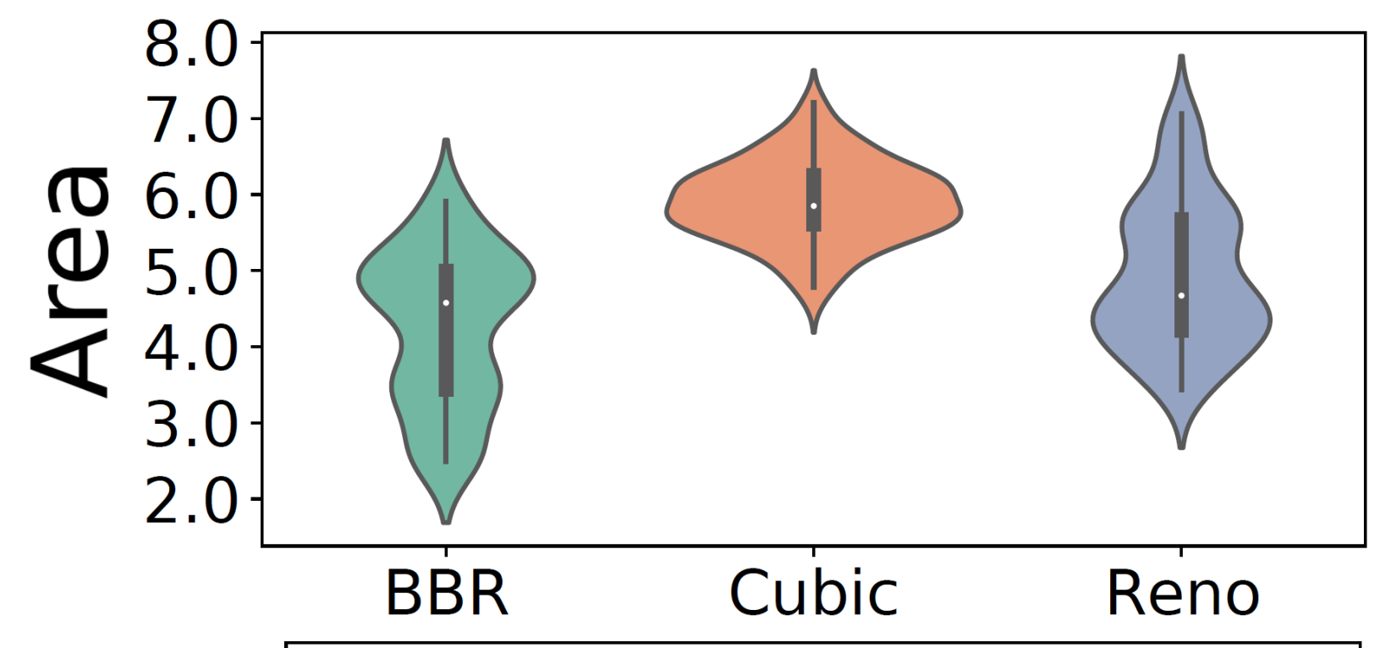


Practical Response (emulation)

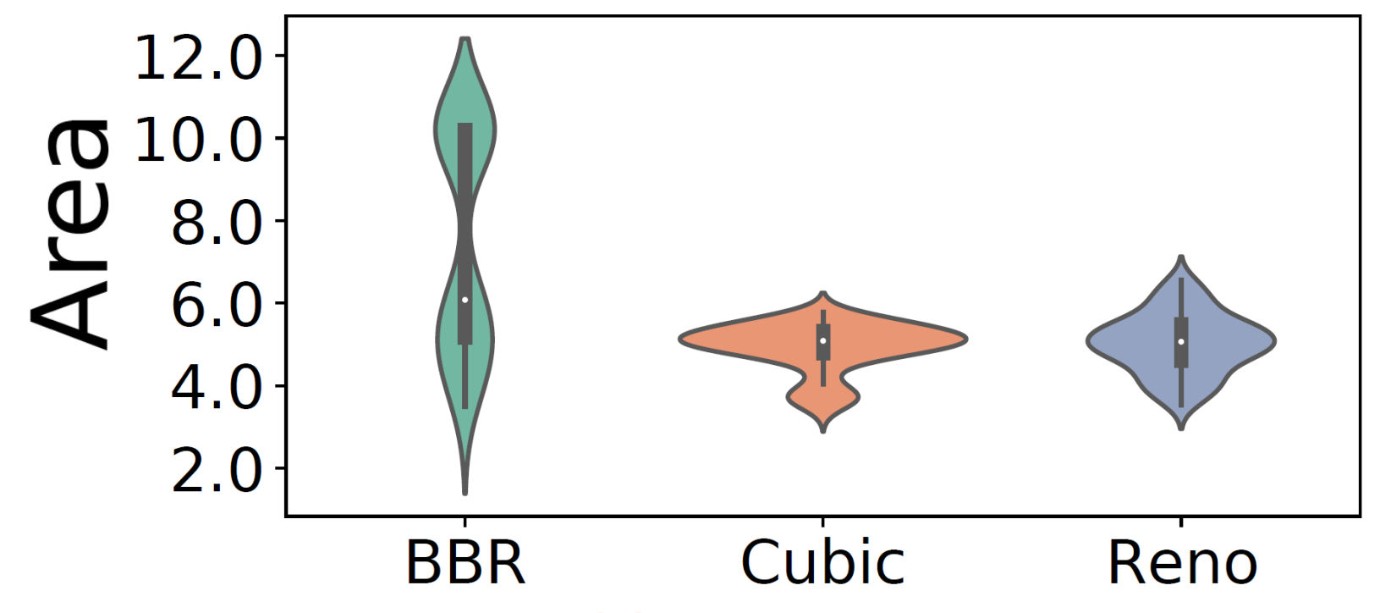
The response of step change in a 2nd-order system is **damped oscillation**. By comparing the **gap to critically-damped oscillation**(optimal), it can quantify the transient performance.

Preliminary Evaluation

We use *Mahimahi* and *CCP* for emulation and the figures show the **area** of 3 CCAs.



Smaller Buffer (40×MTU)



Larger Buffer (100×MTU)

The CCAs differ in transient performance (better with smaller area) with different buffer size, which is intuitive.

* This figure is partially borrowed from *Classic Meets Modern: a Pragmatic Learning-Based Congestion Control for the Internet*(sigcomm2020)