Eagle: Refining Congestion Control by Learning from the Experts

Salma Emara[1], Baochun Li[1], Yanjiao Chen[2]
[1] University of Toronto, {salma, bli}@ece.utoronto.ca
[2] Wuhan University, chenyj.thu@gmail.com

Internet Congestion Control

Video Streaming Applications

Before 2000	2005	2010	2015	2020
Vegas	Hybla	CUBIC	BBR	Indigo
	BIC	Illinois	PCC	Vivace

Internet Congestion Control

[Dong et al., 2015 & 2018]
- *Online learning*
- Utility framework

Before 2000 | 2005 | 2010 | 2015 | 2020

Vegas

Hybla

BIC

CUBIC

Illinois

BBR

Indigo

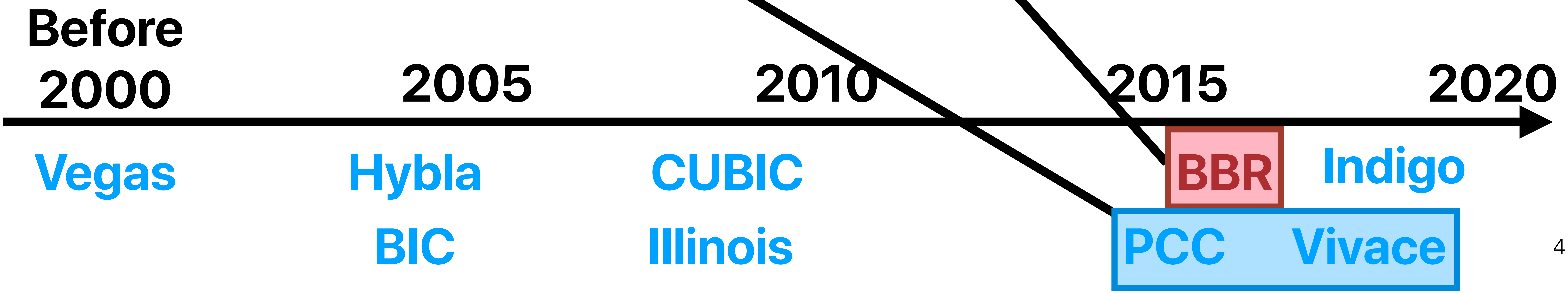
PCC

Vivace

Internet Congestion Control

[Dong et al., 2015 & 2018]
- *Online learning*
- Utility framework

[Cardwell et al., 2016]
- *Heuristic*
- Estimate bottleneck bandwidth and minimum RTT

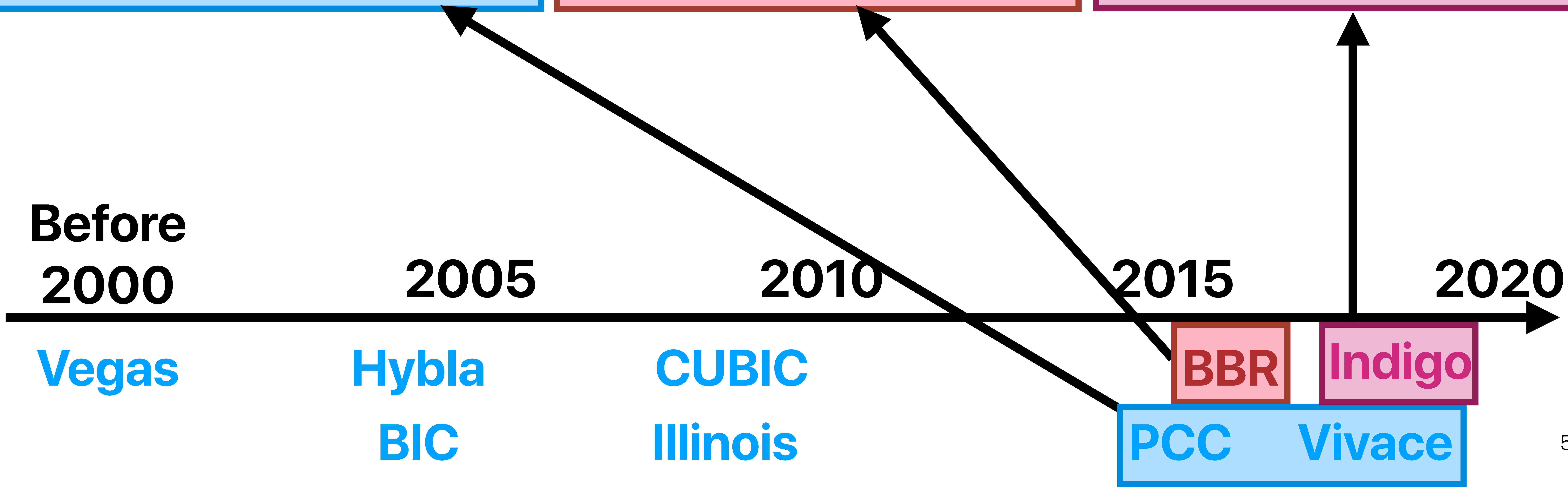


Internet Congestion Control

[Dong et al., 2015 & 2018]
- *Online learning*
- Utility framework

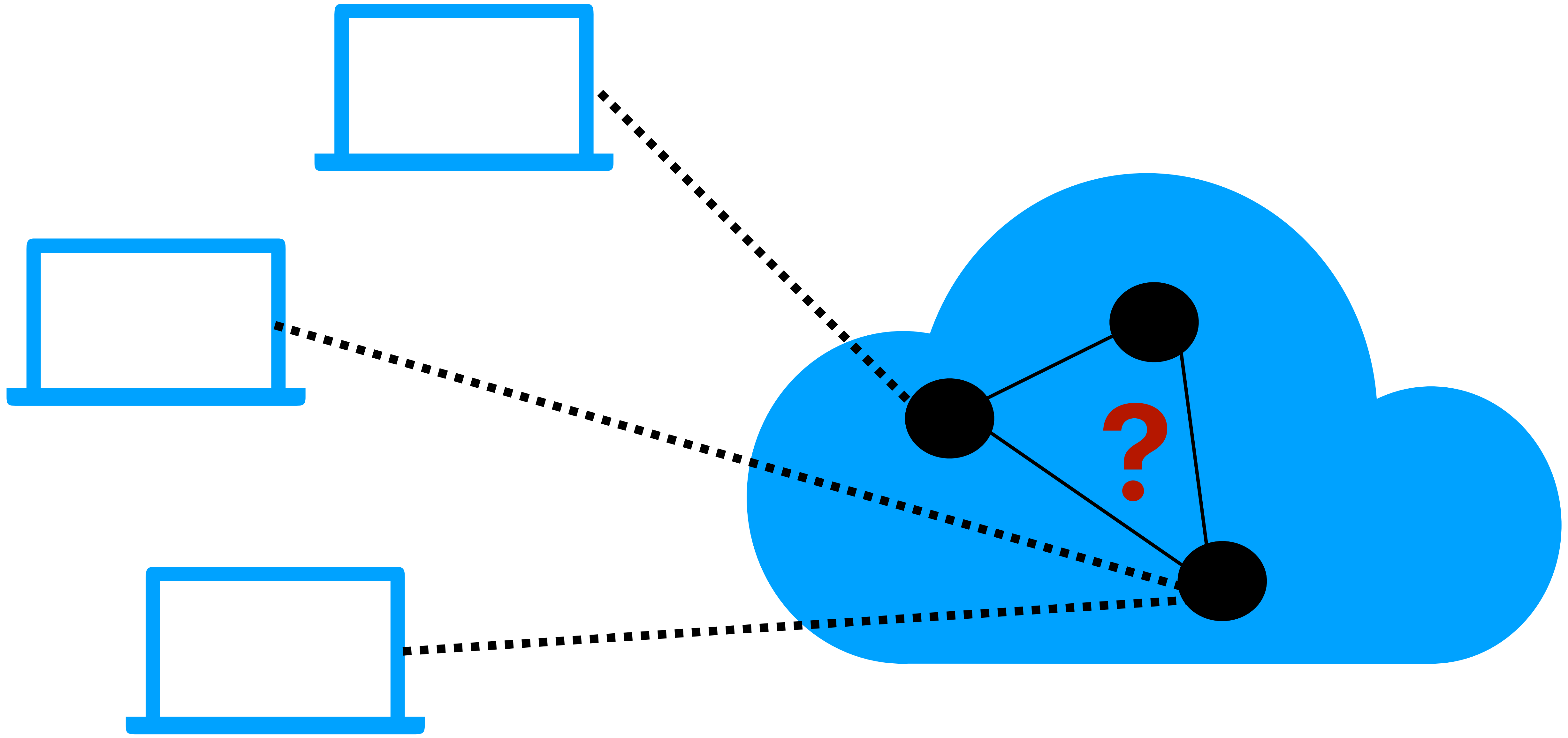
[Cardwell et al., 2016]
- *Heuristic*
- Estimate bottleneck bandwidth and minimum RTT

[Yan et al., 2018]
- *Offline learning*
- Map states to actions



Existing Congestion Control Algorithms

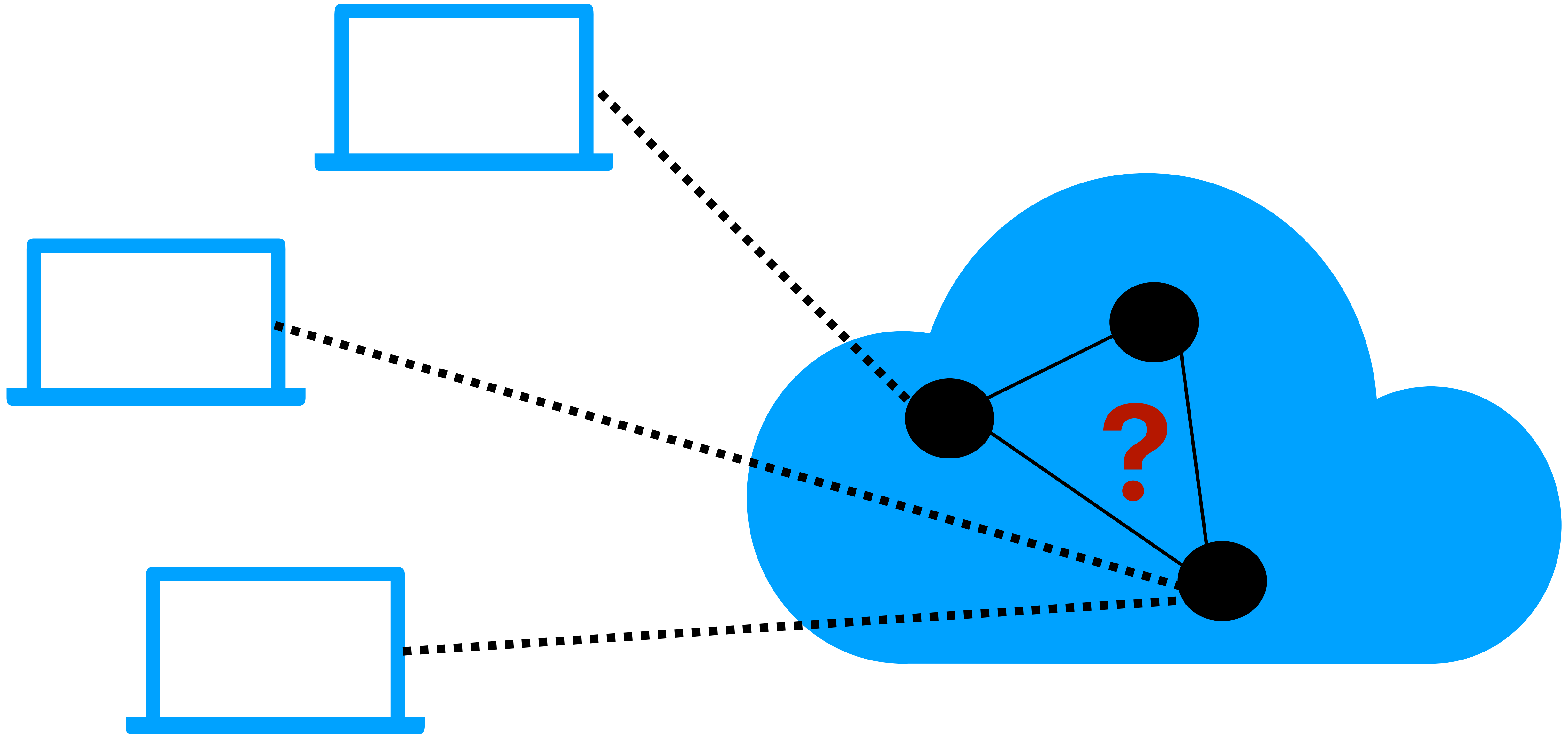
- Fixed mappings between events and control responses



Bandwidth is Dynamic or Stable?
Shared with other flows?
Lossy?

Existing Congestion Control Algorithms

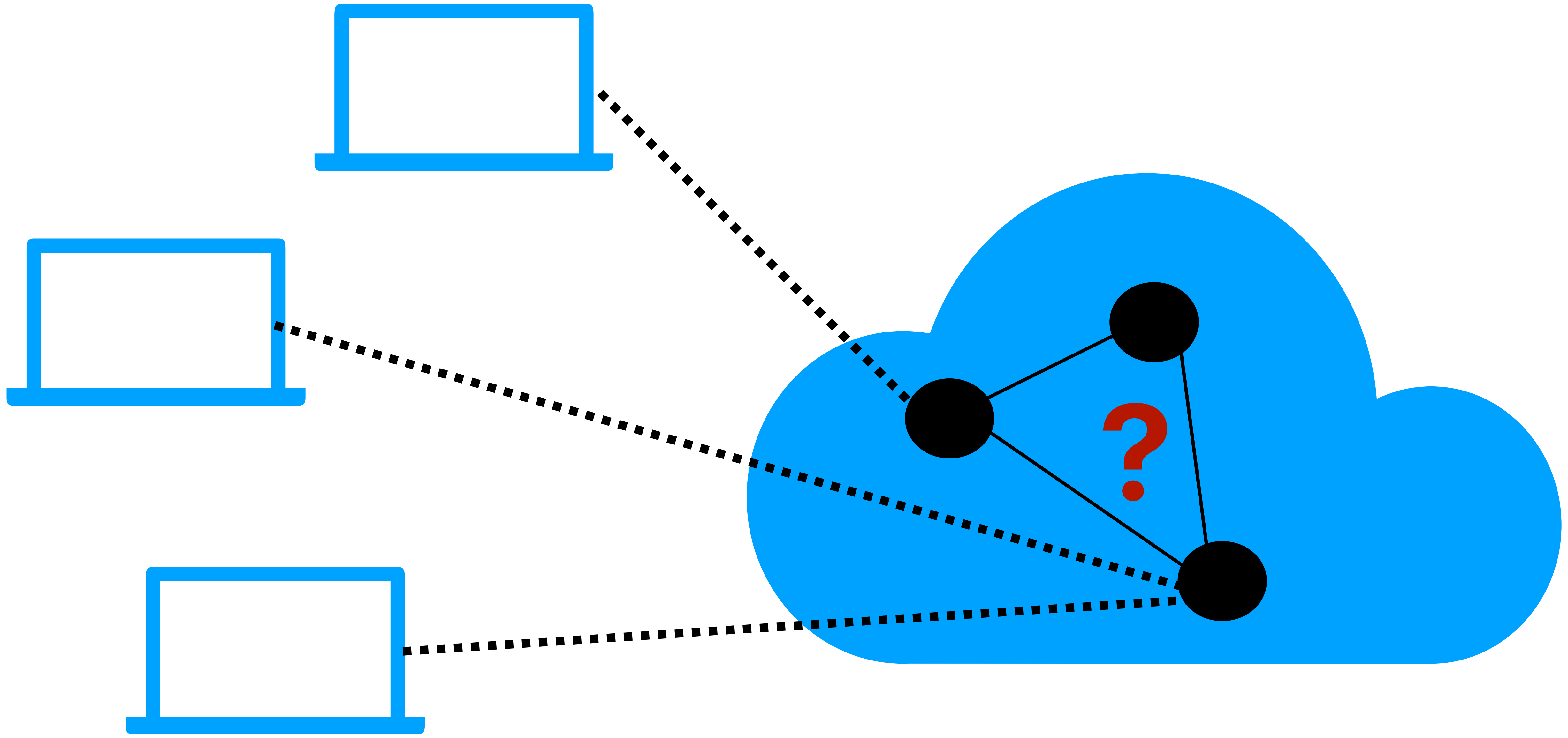
- Fixed mappings between events and control responses
- Mappings are fixed on environments the model was trained on



Bandwidth is Dynamic or Stable?
Shared with other flows?
Lossy?

Existing Congestion Control Algorithms

- Fixed mappings between events and control responses
- Mappings are fixed on environments the model was trained on
- Oblivious to earlier traffic patterns



Bandwidth is Dynamic or Stable?
Shared with other flows?
Lossy?

Think of Congestion Control as a ***Game***

Think of Congestion Control as a ***Game***

1
No fixed way to play the game

2

3

Think of Congestion Control as a ***Game***

1
No fixed way to play the game

2
Based on **changes in the game**, you make a move

3

Think of Congestion Control as a ***Game***

1
No fixed way to play the game

2
Based on **changes in the game**, you make a move

3
Use **history** to understand your game environment

A Sender/Learner/Agent can be ***trained*** to play the Congestion Control Game

Earlier Success Stories of Training for Games

- In 2016, AlphaGo was the first to beat human expert in Go game
- It was trained using supervised and reinforcement learning

Contributions

- Eagle is designed to
    - Train using **reinforcement learning**
    - Learn from **an expert** and **explore** on its own
    - Matching performance of expert and outperform it on average

What do we need to play the congestion control game?

Target Solution Characteristics

- **Consider**
  - Avoiding deterministic mappings between network states and actions by the sender

Target Solution Characteristics

- **Consider**
  - Avoiding deterministic mappings between network states and actions by the sender
  - Generalizing well to many network environments

Target Solution Characteristics

- **Consider**
  - Avoiding deterministic mappings between network states and actions by the sender
  - Generalizing well to many network environments
  - Adapting well to newly seen network environments

Target Solution Characteristics

- **Consider**
  - Avoiding deterministic mappings between network states and actions by the sender
  - Generalizing well to many network environments
  - Adapting well to newly seen network environments

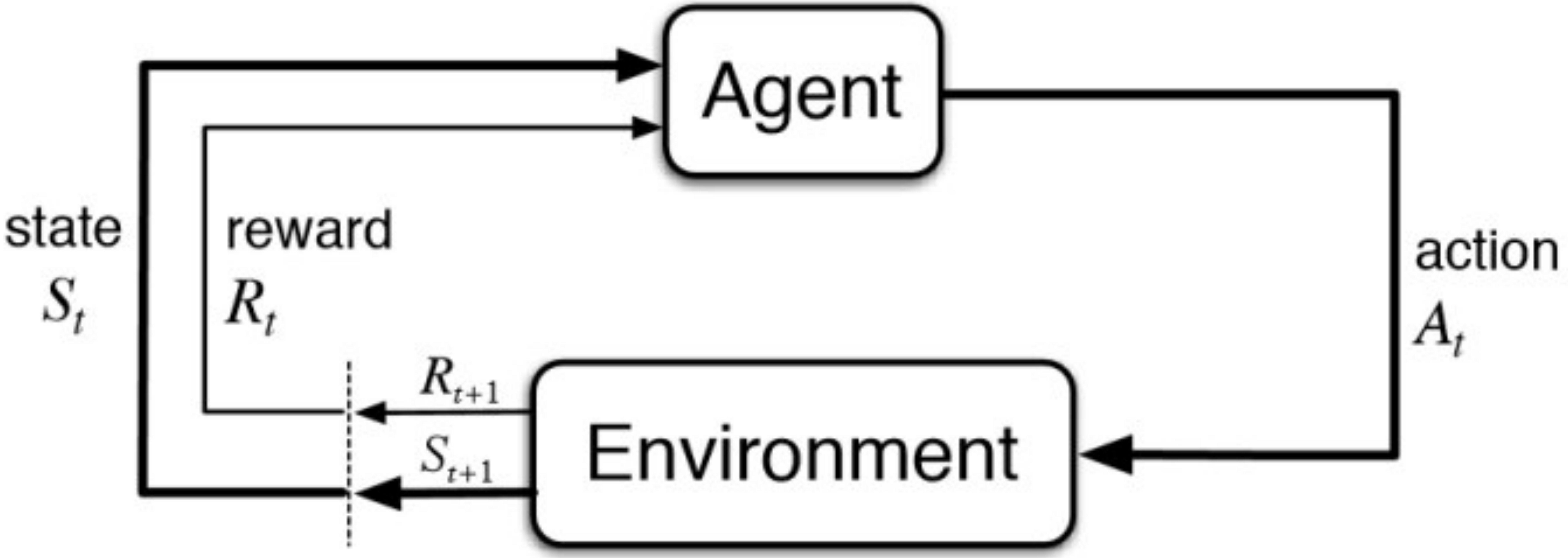
- **Areas of focus**
  - **Stochastic policy**
  - **A more general system design**
  - **Online learning**

Target Solution Characteristics

- **Consider**
  - Avoiding deterministic mappings between network states and actions by the sender
  - Generalizing well to many network environments
  - Adapting well to newly seen network environments

- **Areas of focus**
  - **Stochastic policy**
  - **A more general system design**
  - **Online learning**

General Framework of Reinforcement Learning



Challenges in using Deep Reinforcement Learning

First-Cut: GOLD

- *Deep Neural Network* with two hidden layers
- *Congestion window size (cwnd)* as the control parameter
- State space: [sending rate, loss rate, RTT gradient] in *past 4 steps*
- Action Space: [$\times 2.89, \ \times 1.5, \ \times 1.05, 0, \ \div 2.89, \ \div 1.5, \ \div 1.05$]
- Reward Function:

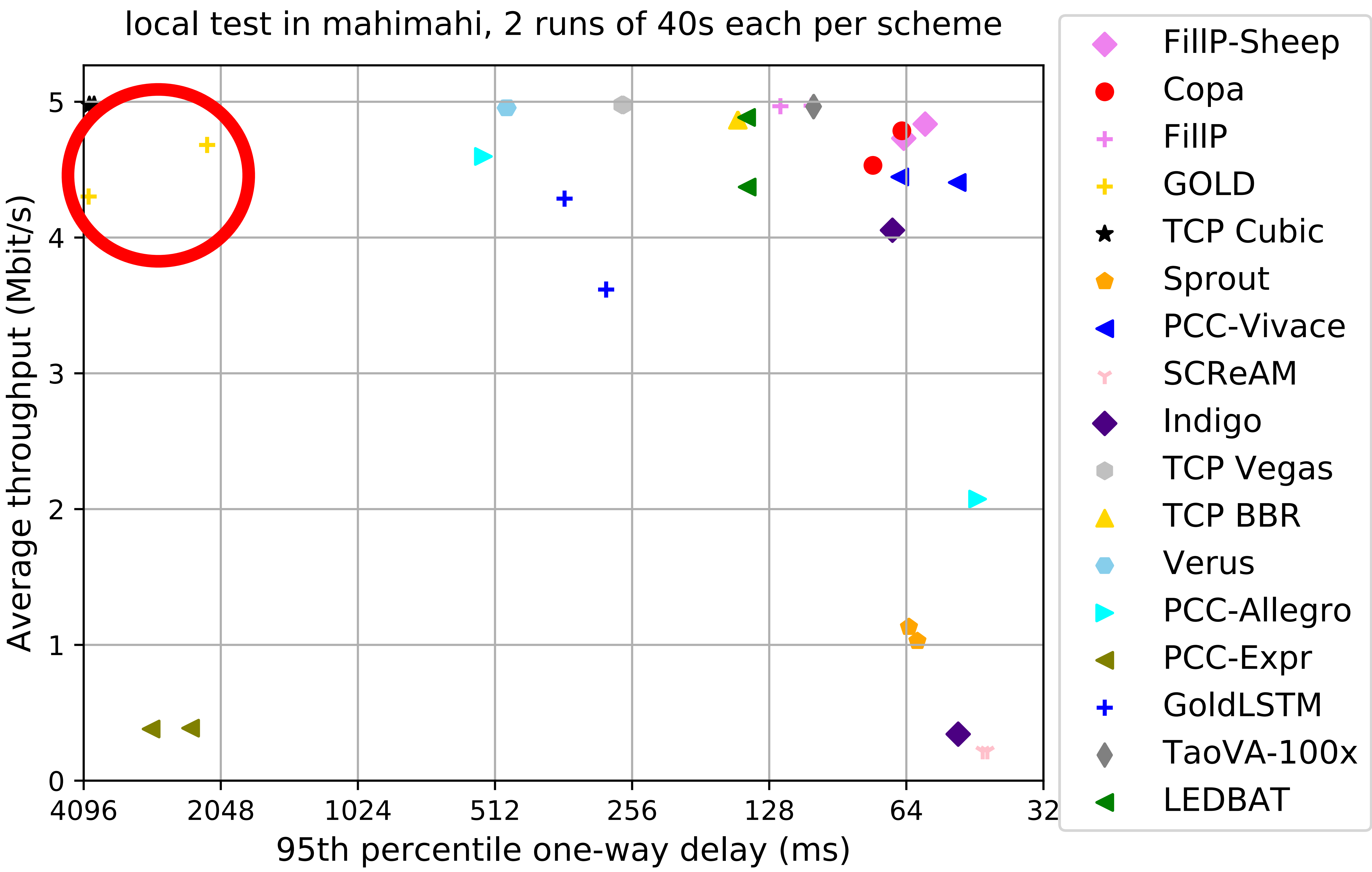
$$r_t = \text{goodness}^a - b \times \text{goodness} \times \frac{dRTT}{dT} - c \times \text{goodness} \times L_t$$

$$u_t = x_t - b \times x_t \times \frac{dRTT}{dT} - c \times x_t \times L_t$$

Issues with GOLD

- Overly aggressive action space taking so much time to drain queues
- Not considering delays in our reward function
- Hard coded the number of past steps to be considered to 4
- Slow training convergence, since step size was dependent on RTT

5 Mbps and 40ms one-way delay



Motivating Current System Design

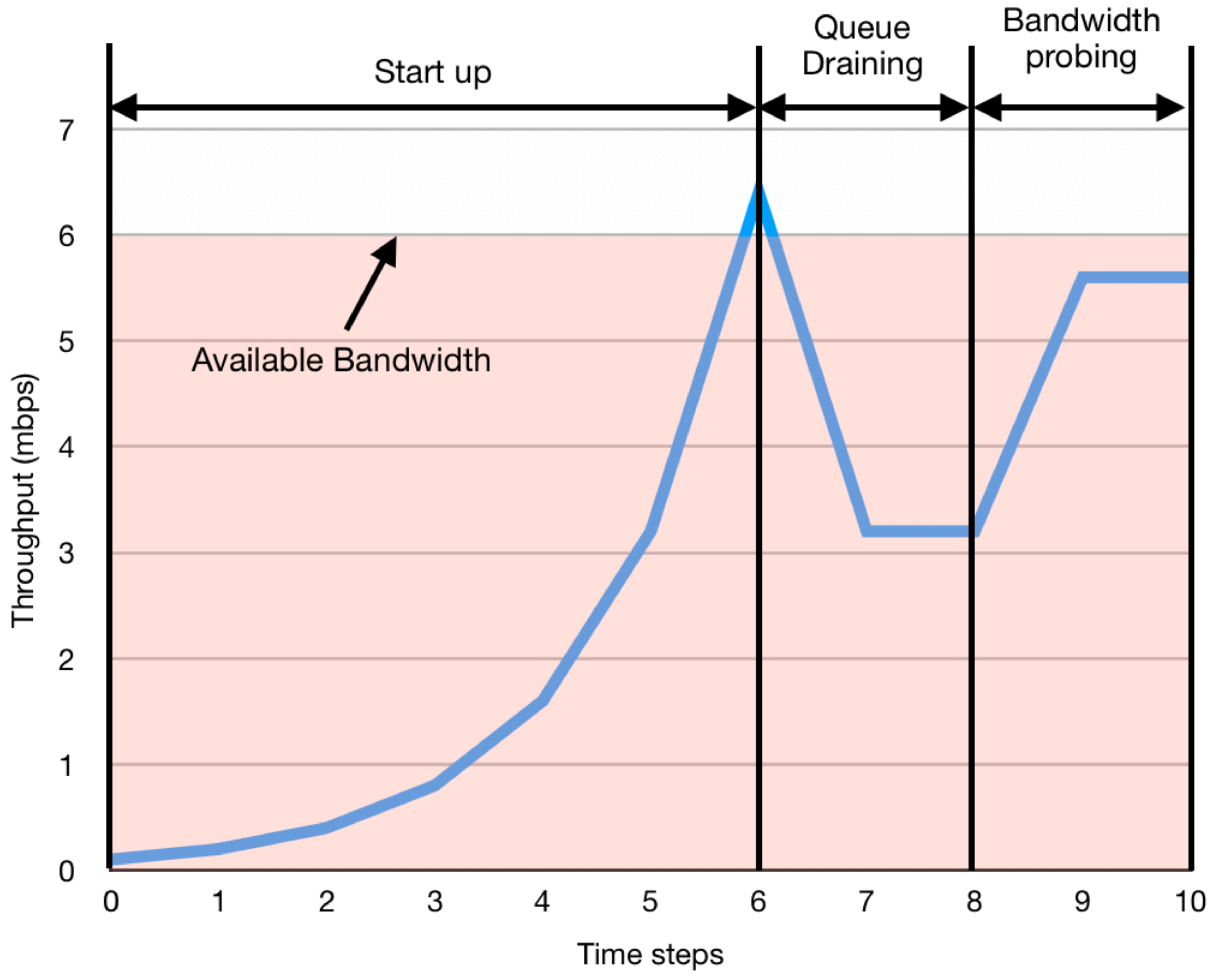
- Deep Reinforcement Learning:
  - Stochastic policy, hence we choose a **policy-based** algorithm
  - LSTM neural network to save weights **across time steps**
  - **Generalize** system design
    - state space across different environments
    - Tailor reward for ***different phases***

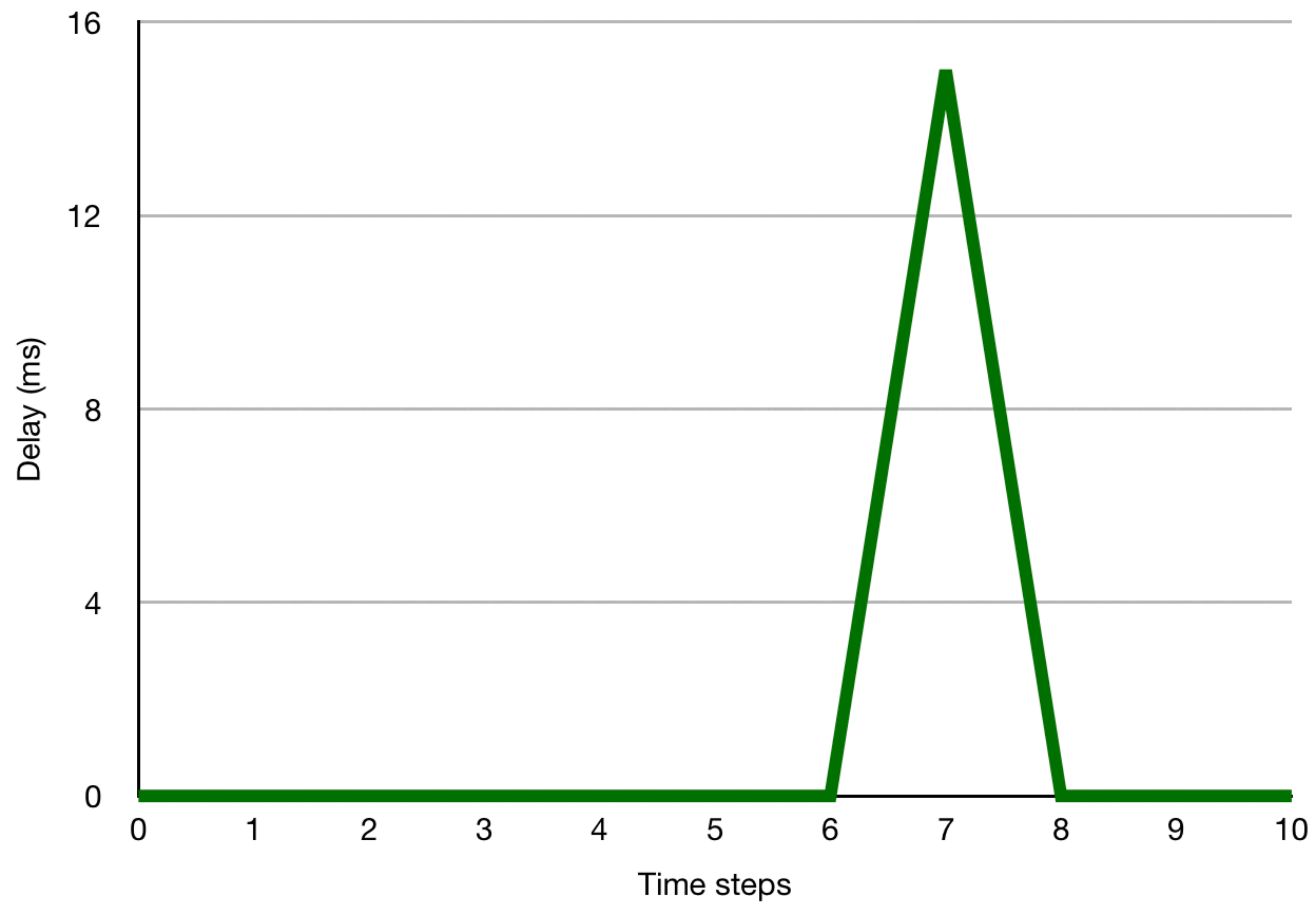
Motivating Current System Design

- Why do we need an expert?
  - Get out of bad states that slows training time, since step size depends on RTT
  - No need to try very bad actions when we can learn easy tasks quickly from expert
  - Avoid local optima

Expert BBR Mechanism

- *Start-up phase:* aggressive increase in sending rate until delay is seen
- *Queue draining phase:* decrease sending rate to the last sending rate before delay
- *Bandwidth probing phase:* increase sending rate slowly until delay is seen





Design Decisions

- ***Reward function***: accurate feedback to the agent

  - *Start-up phase:*

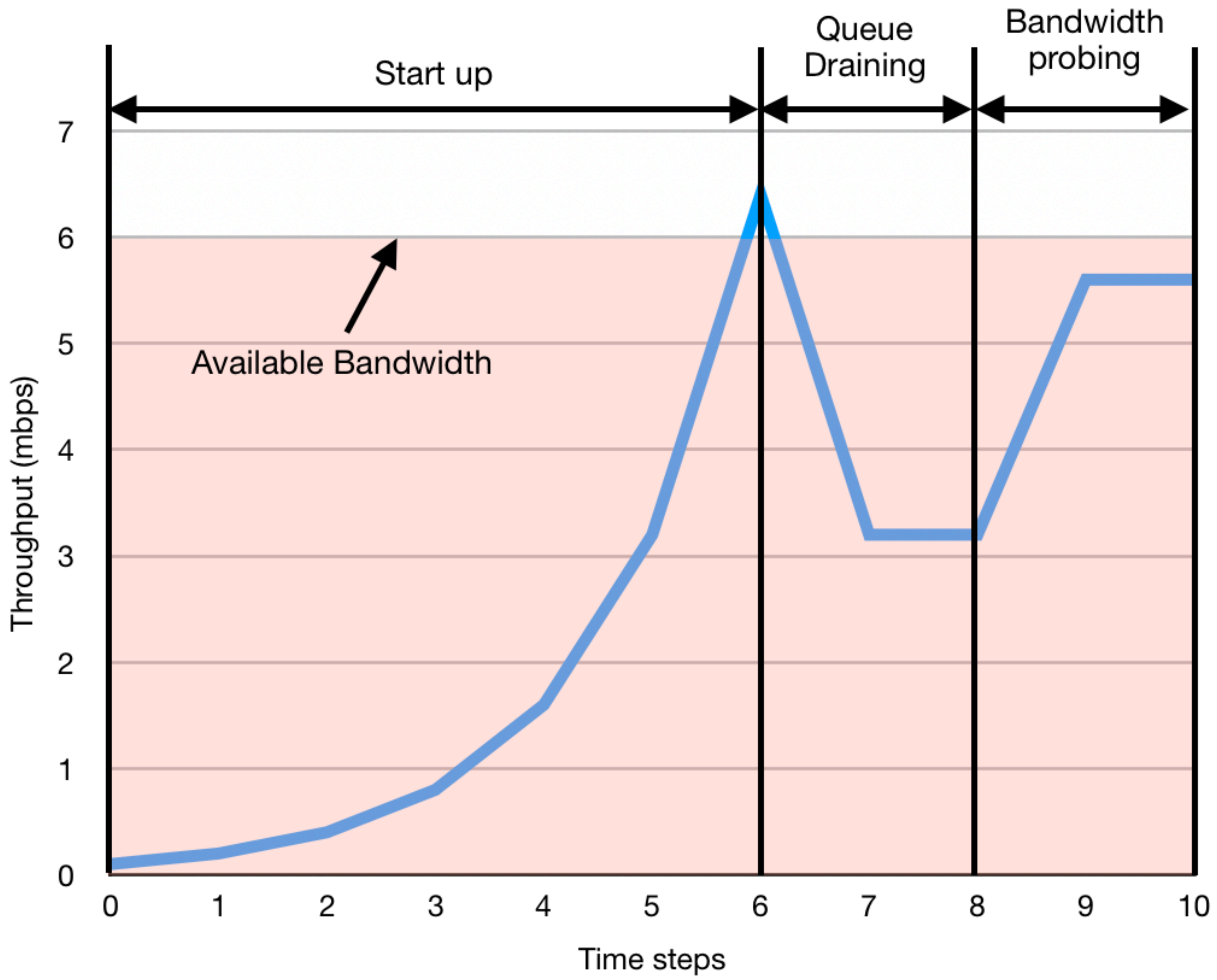
    $r_t \propto \Delta$delivery rate

  - *Queue draining phase:*

    $r_t \propto -\Delta$queueing delay

  - *Bandwidth probing phase:*

    $r_t \propto (\Delta$delivery rate $- \Delta$queueing delay$)$

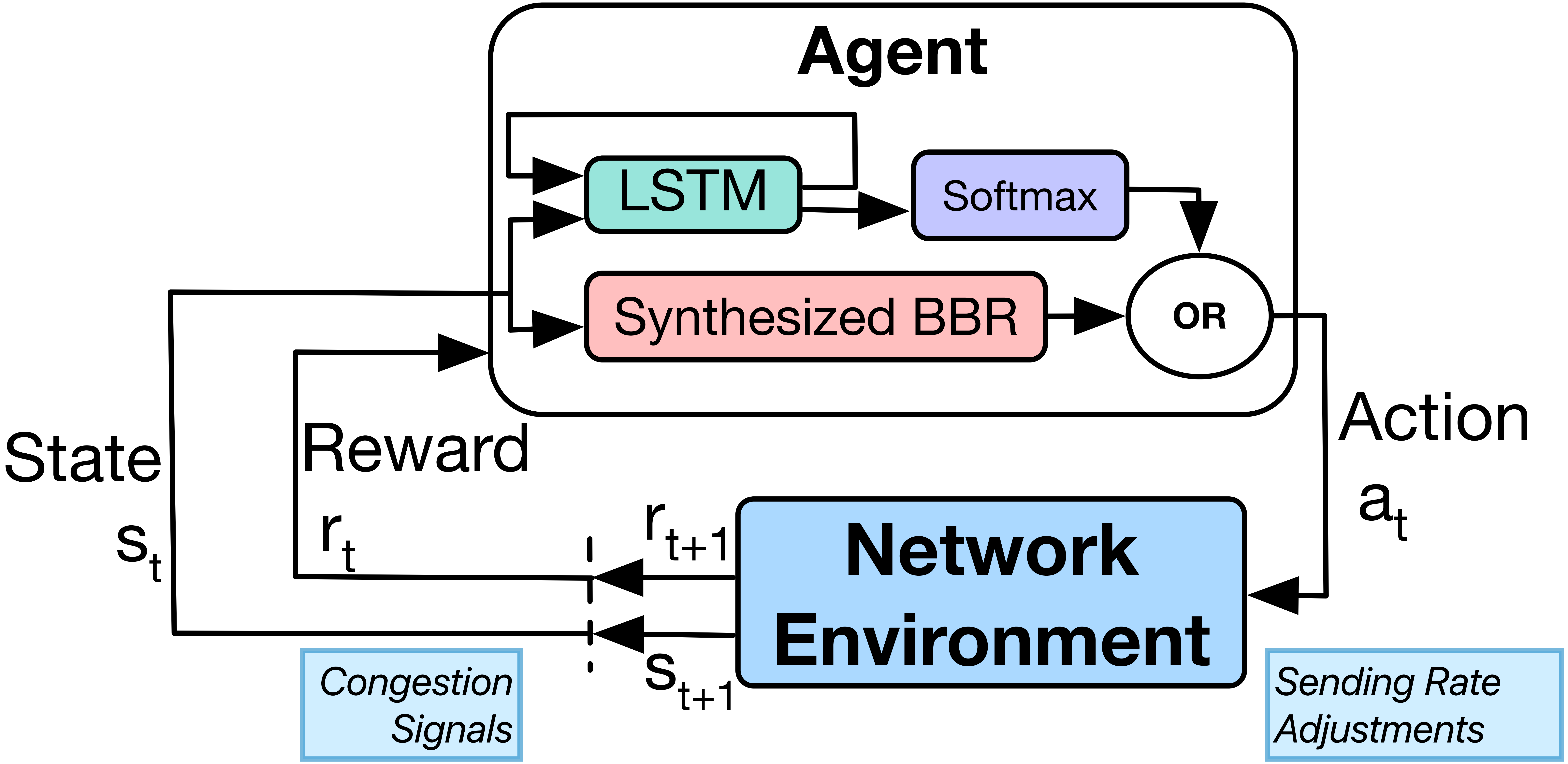


Design Parameters

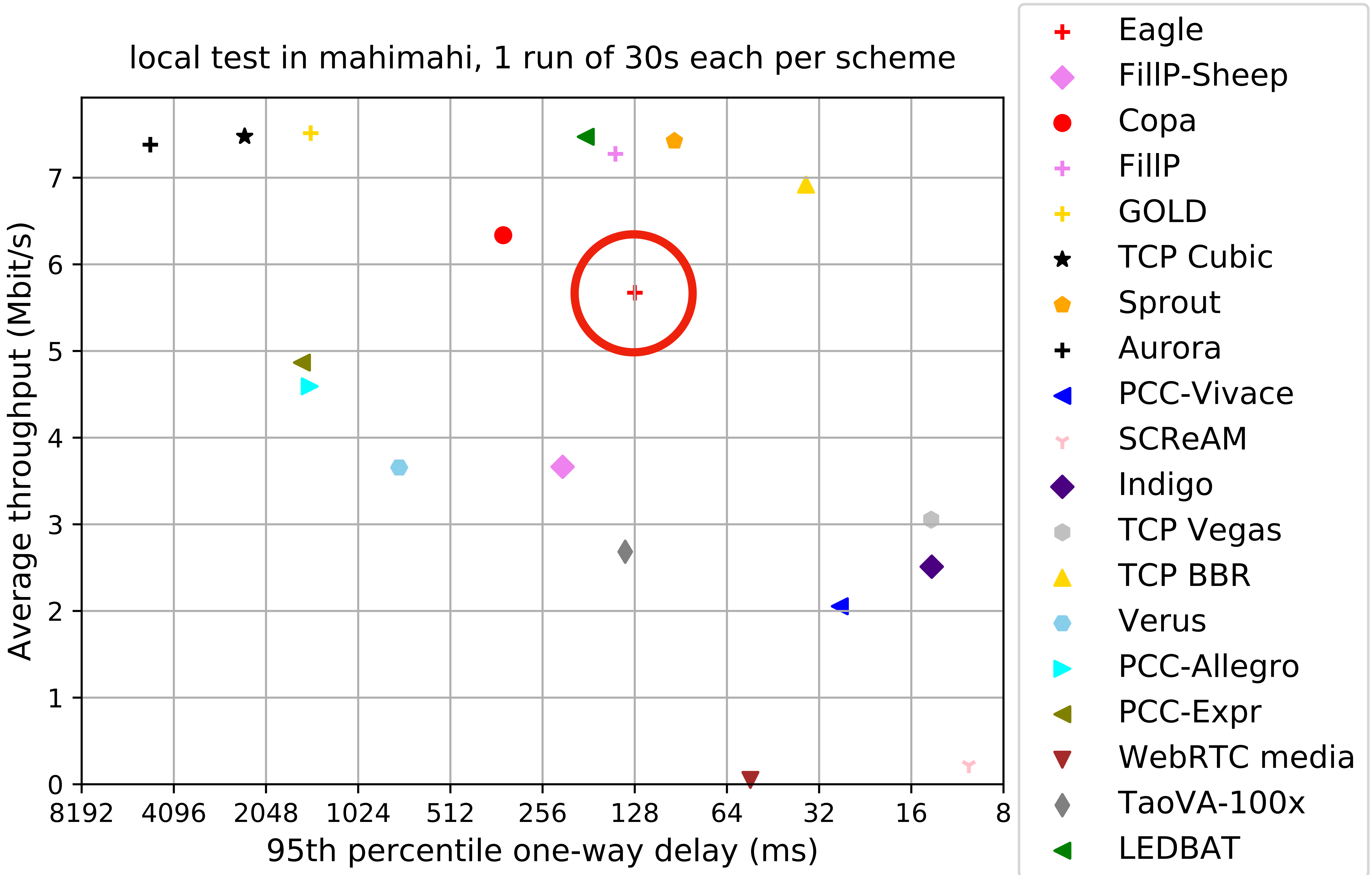
- **Step size:** $3 \times$ RTT
- **State space** (for past 4 steps)
  - Experienced Delay Before?
  - Increases - Decrease Multiples
  - Percentage Change in exponentially weighted moving average (EWMA) Delivery Rate
  - Loss Rate
  - EWMA of Queueing Delay

- **Algorithm:** Cross-entropy method
- **Neural Network**: LSTM with 64 hidden units and 2 layers
- **Action space on sending rate**
  - $\times 2.89$
  - $\times 1.25$
  - Do nothing
  - $\div 1.25$
  - $\div 2.89$

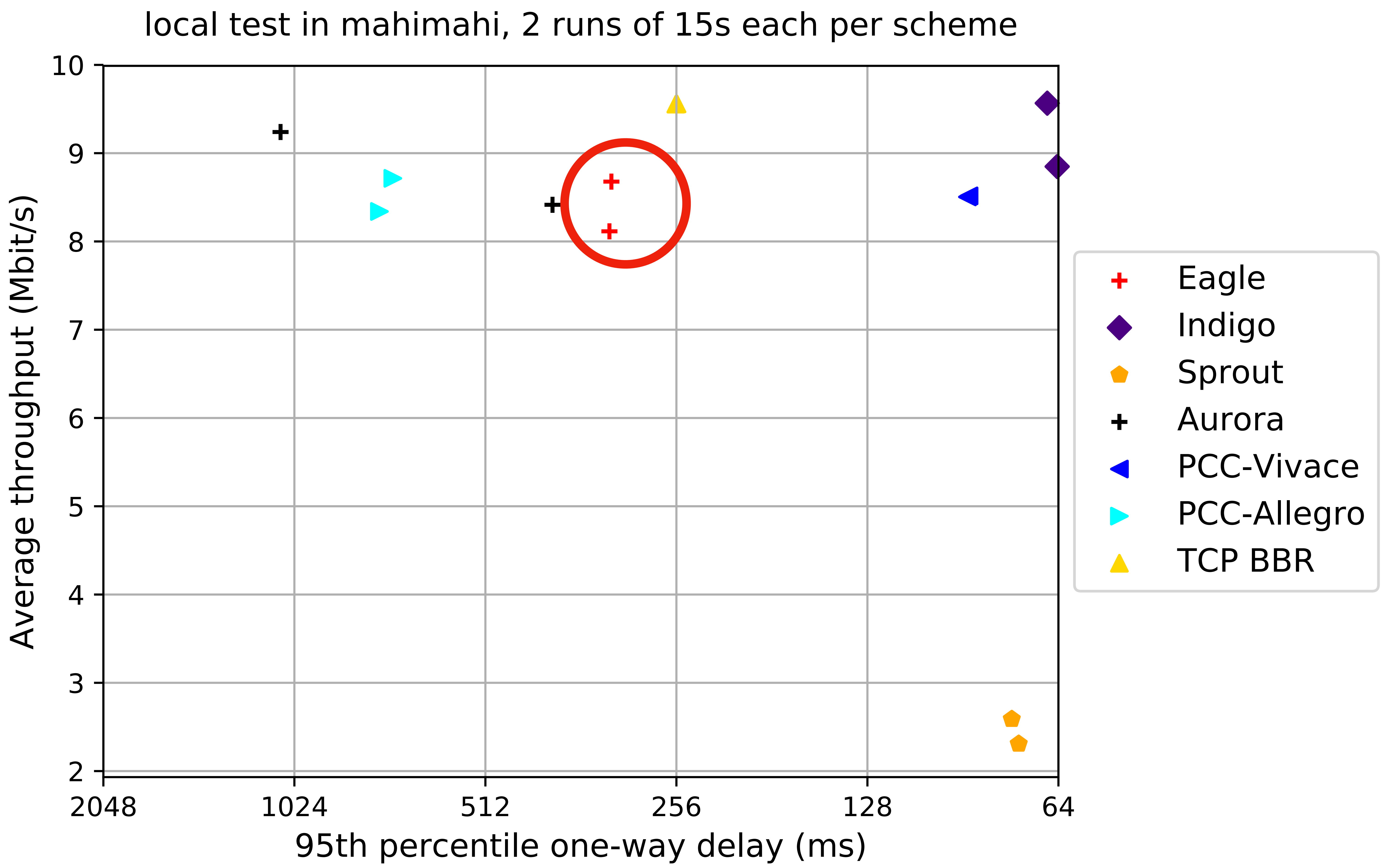
System Design



Results: Pantheon LTE Environment



Results: Pantheon Constant Bandwidth Environment



Concluding Remarks

- ***Eagle:*** *Congestion Control Algorithm powered by Deep Reinforcement Leaning and a teacher — BBR*
    - Generalize well
    - Performed well on newly seen environments
    - Step forward to self-learning congestion control
- ***Future work:***
    - Test the performance in online-learning phase
    - Test fairness with other flows

Thank you!