ORC: Online Reinforcement Learning for Congestion Control with Fast Convergence

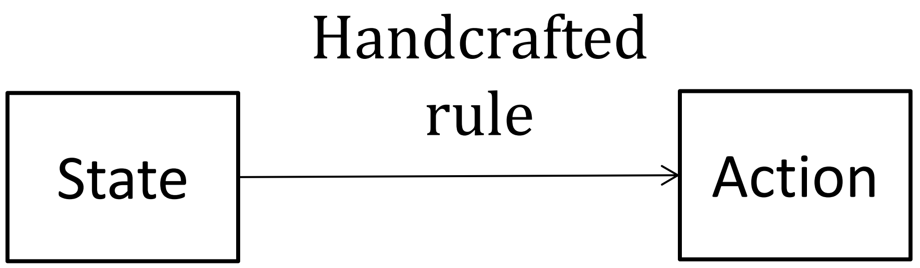
Yijun Li, Jiawei Huang, Chuliang Wu, Xiaojun Zhu, Jianxin Wang
Central South University

APNET 2025

Introduction

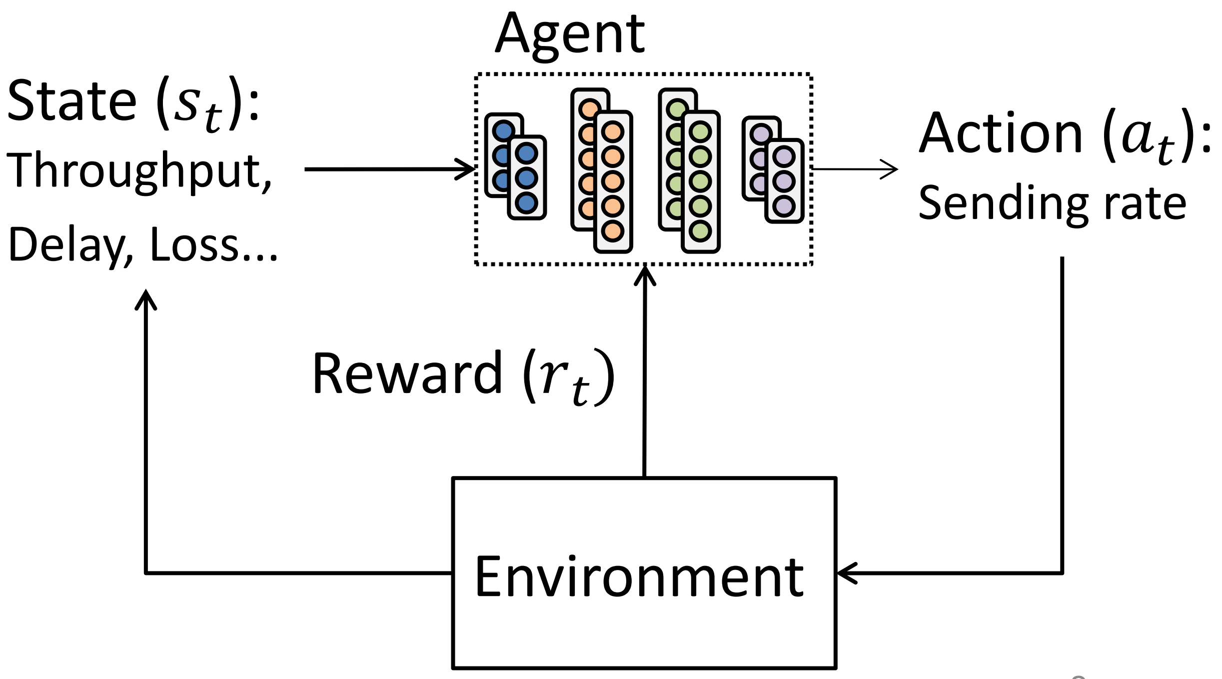
Heuristic congestion control

- Handcrafted rules.
- Specific network environment.



Learning-based CC

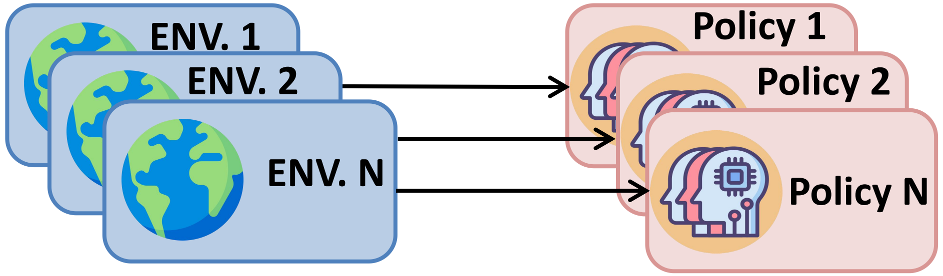
- Learn policy.
- Adapt to various conditions.



Introduction

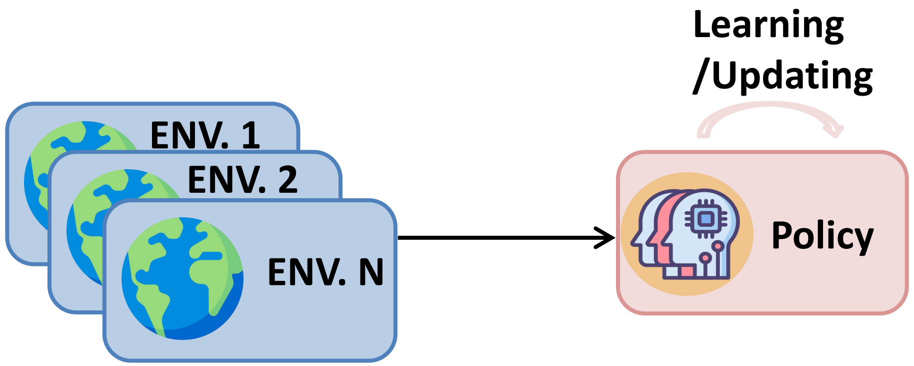
■ Offline methods

- Indigo [ATC ’18]
- Orca [SIGCOMM ’20]
- Degraded performance in unseen scenarios.



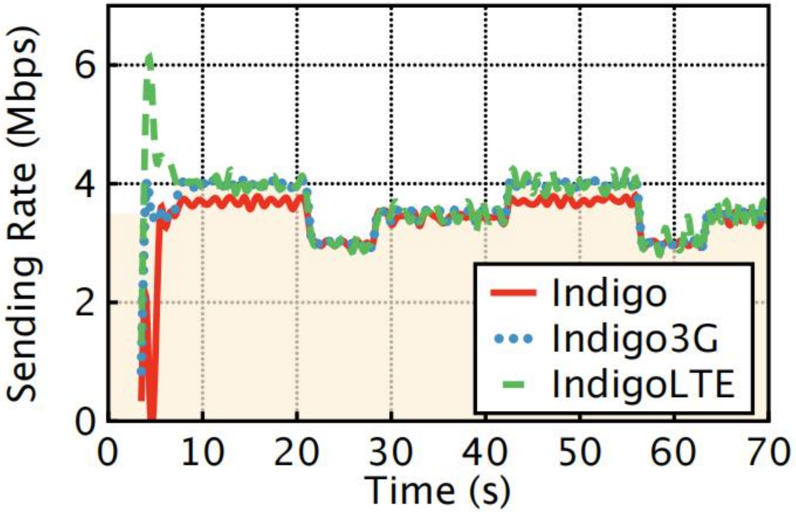
■ Online methods

- PCC [NSDI ’15]
- PCC Vivace [NSDI ’18]
- RL-based CC suffer from quickly update policy.

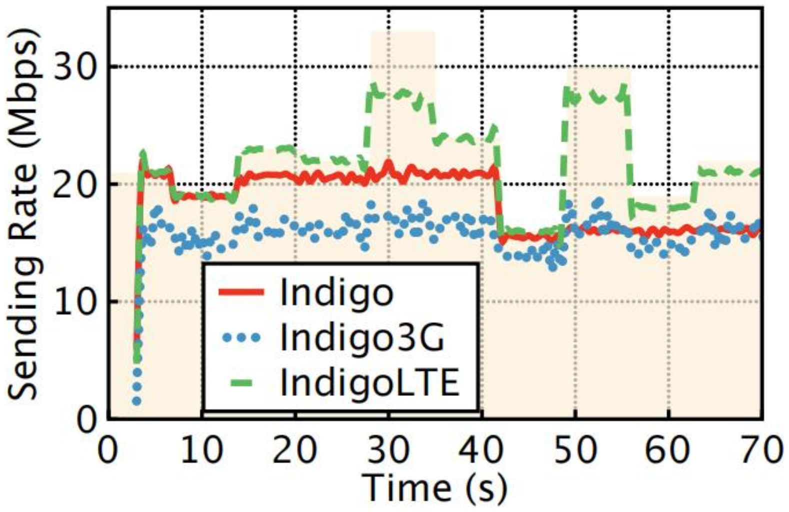


Motivation

- **Problem of Offline Learning:**
  - The deviations between realistic and trained networks result in degraded performance for offline learning-based Congestion control algorithms (CCAs).



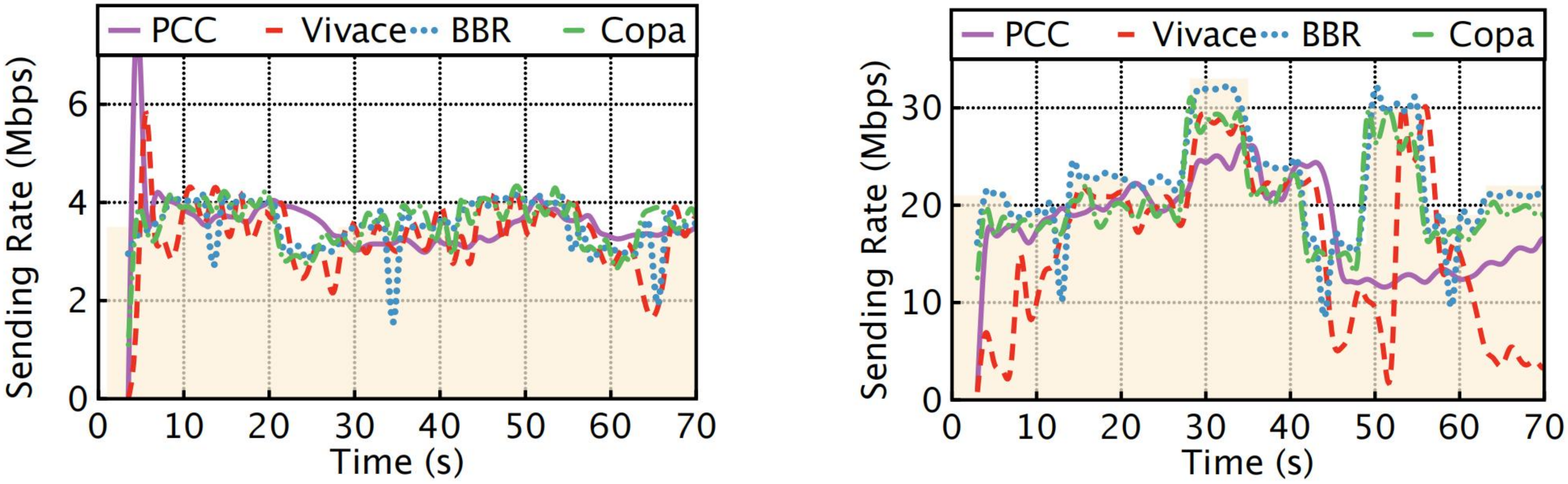
(a) 3G scenario



(b) LTE scenario

Motivation

- **Problem of Online Learning:**
  - Online learning-based CCAs struggle to converge perfectly.
  - Heuristic CCAs quickly explore the available bandwidth.



(a) 3G scenario (b) LTE scenario

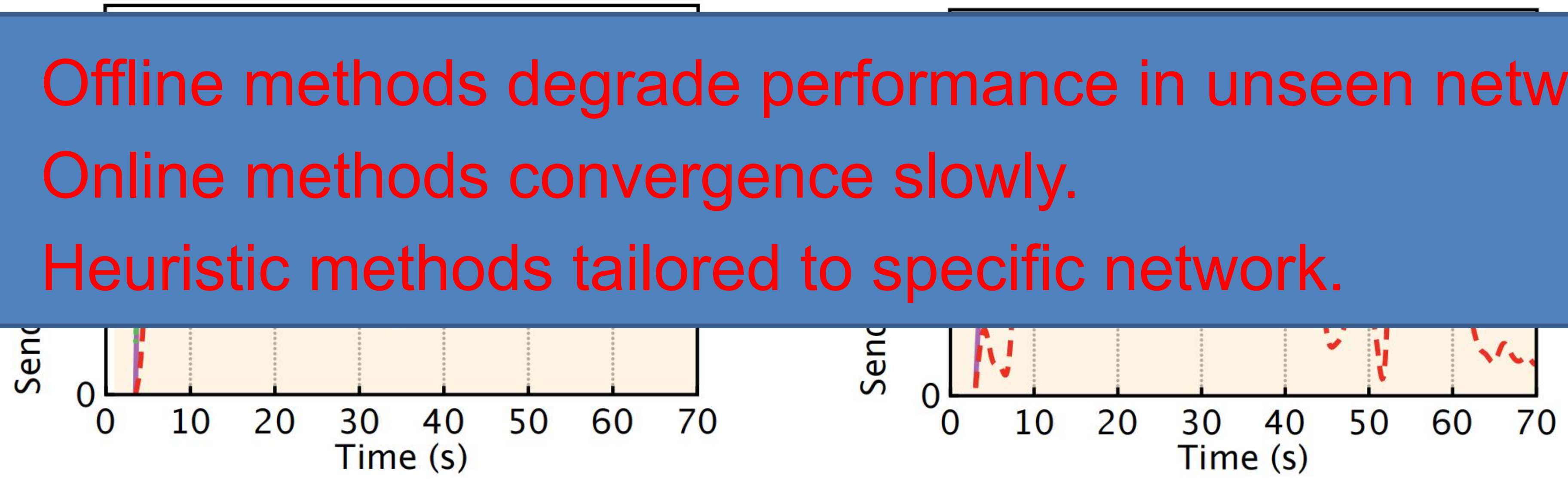
Motivation

- **Problem of Online Learning:**
  - Online learning-based CCAs struggle to converge perfectly.
  - Heuristic CCAs quickly explore the available bandwidth.

Offline methods degrade performance in unseen network.

Online methods convergence slowly.

Heuristic methods tailored to specific network.



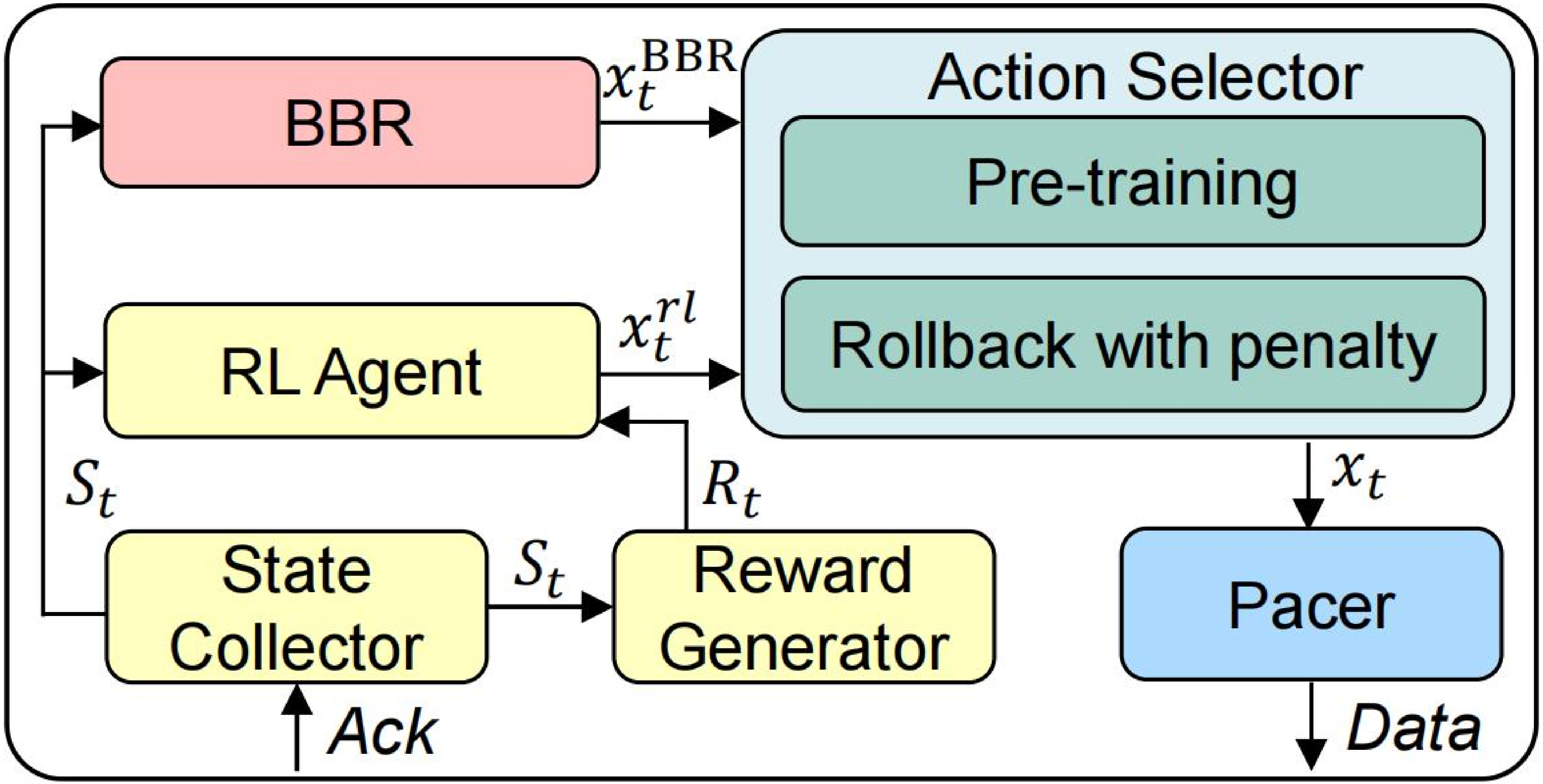
(a) 3G scenario (b) LTE scenario

ORC

Basic idea

- ORC combines online reinforcement learning (RL) and heuristic methods.
- In the start phase,ORC employs the heuristic CC to train the RL model .
- In the exploration phase, the RL-based CCA explore optimal actions. when falling into the bad situation, ORC will switch to the heuristic CC.

ORC Overview



ORC: Design Details

- **Model Architecture**

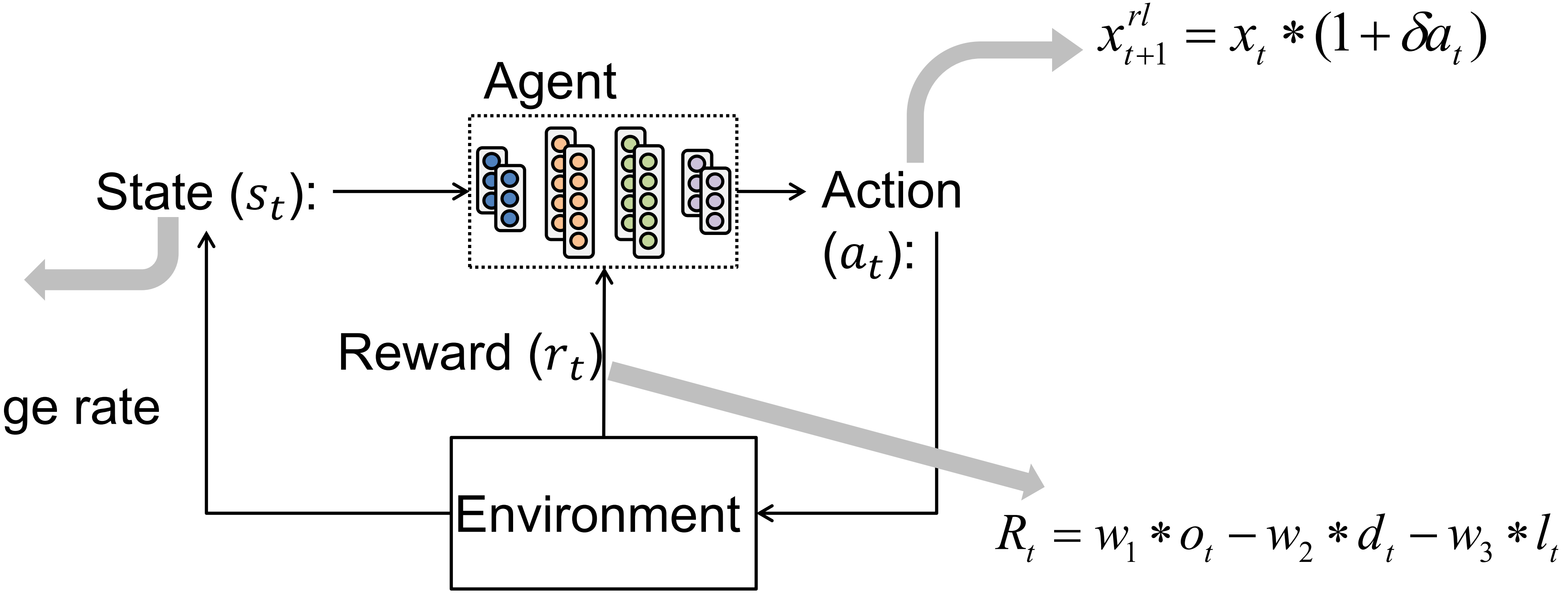
$s_t=(o_t,d_t,l_t,g_t)$,

o_t : throughput

d_t : latency

l_t : packet loss

g_t : latency change rate



ORC: Design Details

- **Pre-training**
  - RL agent mimic the behavior of heuristic CCAs when their difference Δx_t less than Δx_{th}.
  - Once the proportion of similar decisions exceeds 70%, switch to the exploration stage.

Algorithm 1: Pre-training in startup phase

Require:

current state: $s_t = (o_t, d_t, l_t, g_t)$;

training_phase: $training\_phase = startup\_phase$;

reward's weights: w_1, w_2, w_3;

exit threshold: Δx_{th};

for each time step t **do**

    if $training\_phase == startup\_phase$ **then**

        $\Delta x_t = \frac{|x_t^{BBR} - x_t^{lr}|}{B_t}$;

        $R_t = (1 + \Delta x_t)(w_1 * o_t - w_2 * d_t - w_3 * l_t)$;

        if Over 70% of steps satisfy $\Delta x_t < \Delta x_{th}$ exceeds **then**

            $training\_phase = explore\_phase$;

            $x_{t+1} = \text{RL\_agent}(s_t)$;

        else

            $x_{t+1} = \text{BBR}(s_t)$;

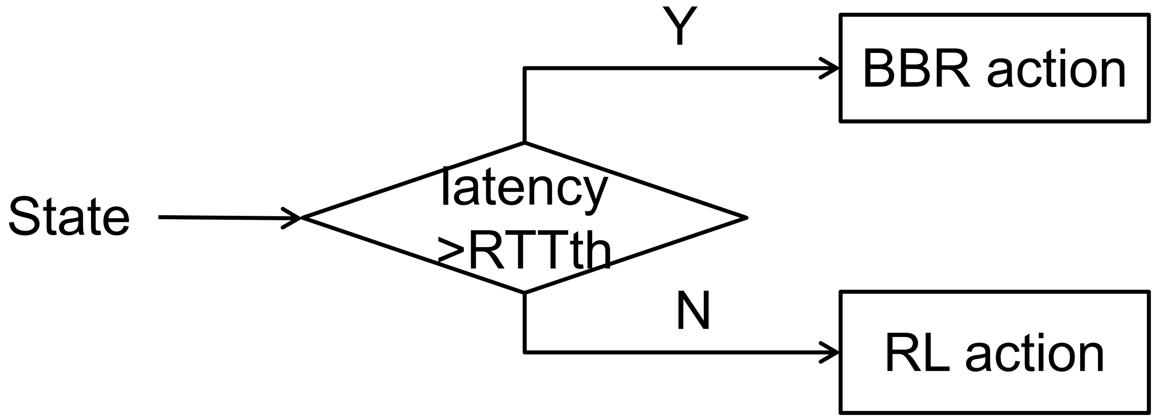
        end if

    end if

end for

ORC: Design Details

- **Rollback with Penalty**
  - In exploration stage, if latency>RTTth, rollback to BBR action.

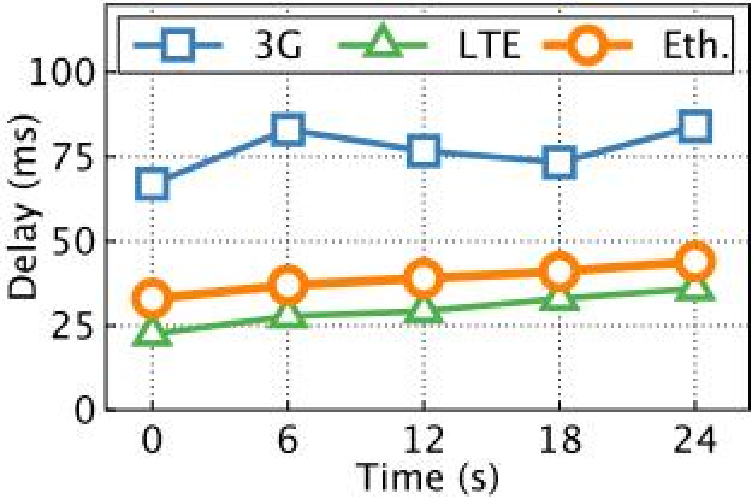


  - The penalty factor P is applied to the reward function to avoid frequent rollback

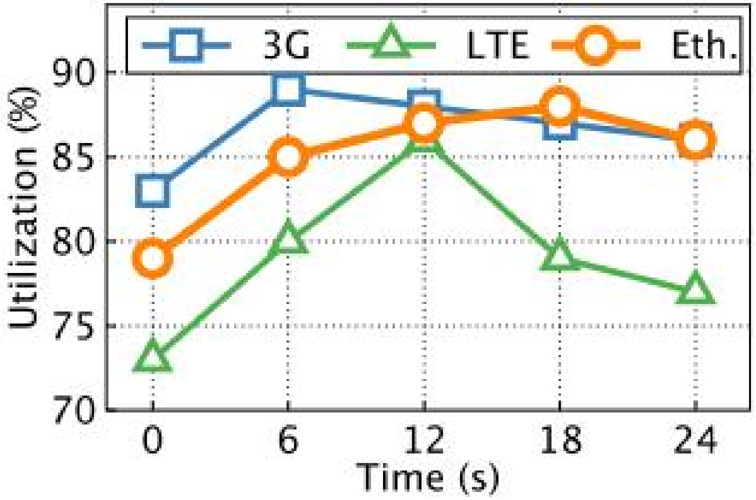
$$R_t = \frac{1}{P} * w_1 * o_t - P * w_2 * d_t - w_3 * l_t$$

Evaluation

- **Implementation:** We implement ORC at Pantheon
- **Effectiveness of Pre-training**
  - ORC avoids performance degradation due to long pre-training time by adjusting the policy similarity threshold Δx_{th} .



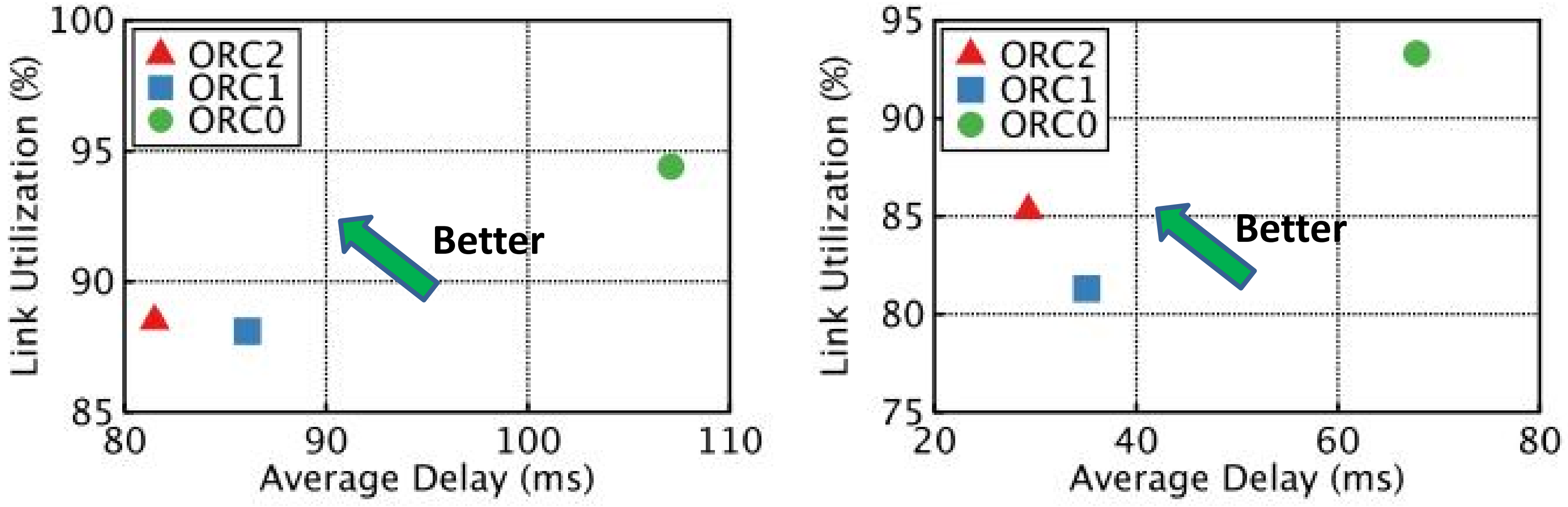
(a) Average delay



(b) Average link utilization

Evaluation

- **Effectiveness of Rollback with Penalty**
  - Evaluate the latency and link utilization of ORC (ORC 0), ORC with rollback (ORC 1), and ORC with rollback and penalty (ORC 2).

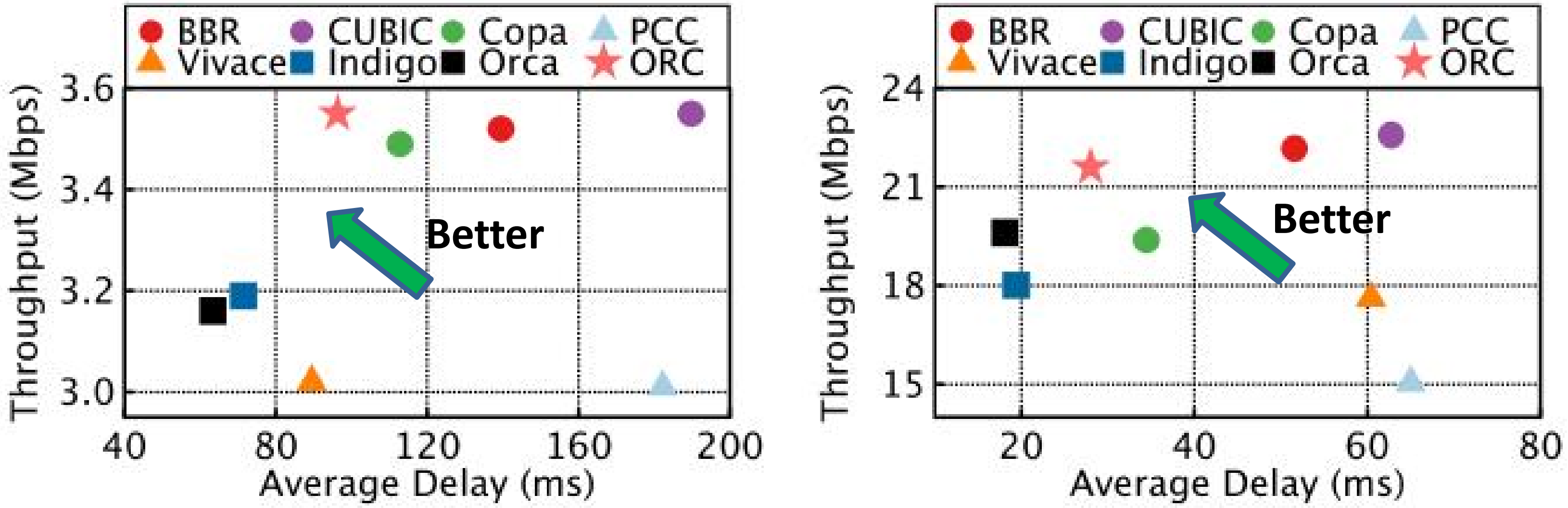


(a) 3G scenario

(b) LTE scenario

Evaluation

- **Performance in Real Networks**
  - ORC can still achieve better performance compare with heuristic and learning-based CCAs.



(a) 3G scenario

(b) LTE scenario

Summary

- ORC combines online learning-based CCA and heuristic methods.
- ORC employs a BBR-guided pre-training mechanism to accelerate initialization and introduces a rollback mechanism with penalty.
- ORC achieves good adaptability and fast convergence under different network conditions.

Thank you !

Q&A