IEEE INFOCOM
IEEE International Conference on Computer Communications
29 April - 2 May 2019 // Paris, France
IEEE ComSoc IEEE Communications Society
IEEE

Delay-Optimal Traffic Engineering Through Multi-agent Reinforcement Learning

Pinyarash Pinyoanuntapong, Minwoo Lee, Pu Wang
University of North Carolina at Charlotte, USA

Presenter: Moinul Hossain

UNC CHARLOTTE

Introduction

Traffic Engineering (TE)

- Dynamically measuring and analyzing real-time network traffic
- Designing optimal forwarding/routing rules
- meet QoS requirements for a large volume of traffic flows.

Challenges

- Optimizing End-to-End QoS metrics (E2E delay & E2E throughput) is challenging for large-scale multi-hop networks

- Dynamics and non-stationarity in traffic flow patterns, network topologies, and link status

Existing TE Solutions

- Heuristic methods (Shortest path)
  - * OSPF (wired), IEEE 802.11s and BATMAN (wireless)
  - * sample but not provably-optimal
- Model-based Optimization Method: Network Utility Maximization (NUM)
  - * TE formulated as constrained maximization problem of utility function
  - * cannot minimize E2E delay: E2E delay cannot be mathematically modeled as explicit functions of TE control parameters

UNC CHARLOTTE

Why Multi-Agent RL?

- Why RL-based TE?
  - * experience-based & model-free => robust & resilient
  - * handle non-stationarity => adapt to time-varying network dynamics
  - * suitable for handling large & sophisticated state/action spaces

- Why distributed TE via Multi-agent RL?
  - * routers collectively learn optimal policy.
  - * suitable for large-scale multi-hop networks

- Existing solution: Q-routing and its variants:
  - * off-policy learning methods => converges slowly
  - * deterministic TE policy => single-path routing

UNC CHARLOTTE

Objective

- There exist a variety of RL algorithms & their extensions to address the limitations of Q-learning

  * **How** these algorithms can be generalized to a multiagent setting to enable distributed TE?

  * **What** is the actual performance of these algorithms when applied for TE problems?

  * **Which** of the extensions are complementary & can be combined to yield substantial performance improvements?

- **Objective:** develop a modular & composable multiagent learning system

  * provide modules and their extensions that can be assembled in various combinations to generate specific MA-RL algorithms

  * study performance gain of MA-RL algorithms

  * automate E2E TE in large-scale multi-hop network

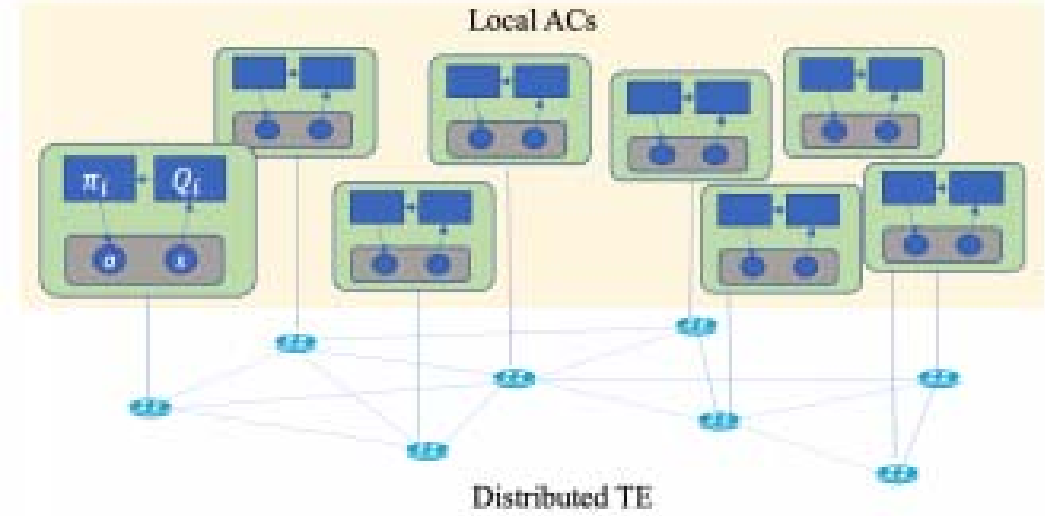
UNC CHARLOTTE

Multi-Agent Learning System

1. Multi-agent Markov decision process (MA-MDP) for Traffic Engineering

$$< S, O_1, \dots, O_N, A_1, \dots, A_N, P, r_1, \dots, r_N >$$

2. Distributed TE Learning Framework based on Multi-agent actor-critics executor (MA-ACE)

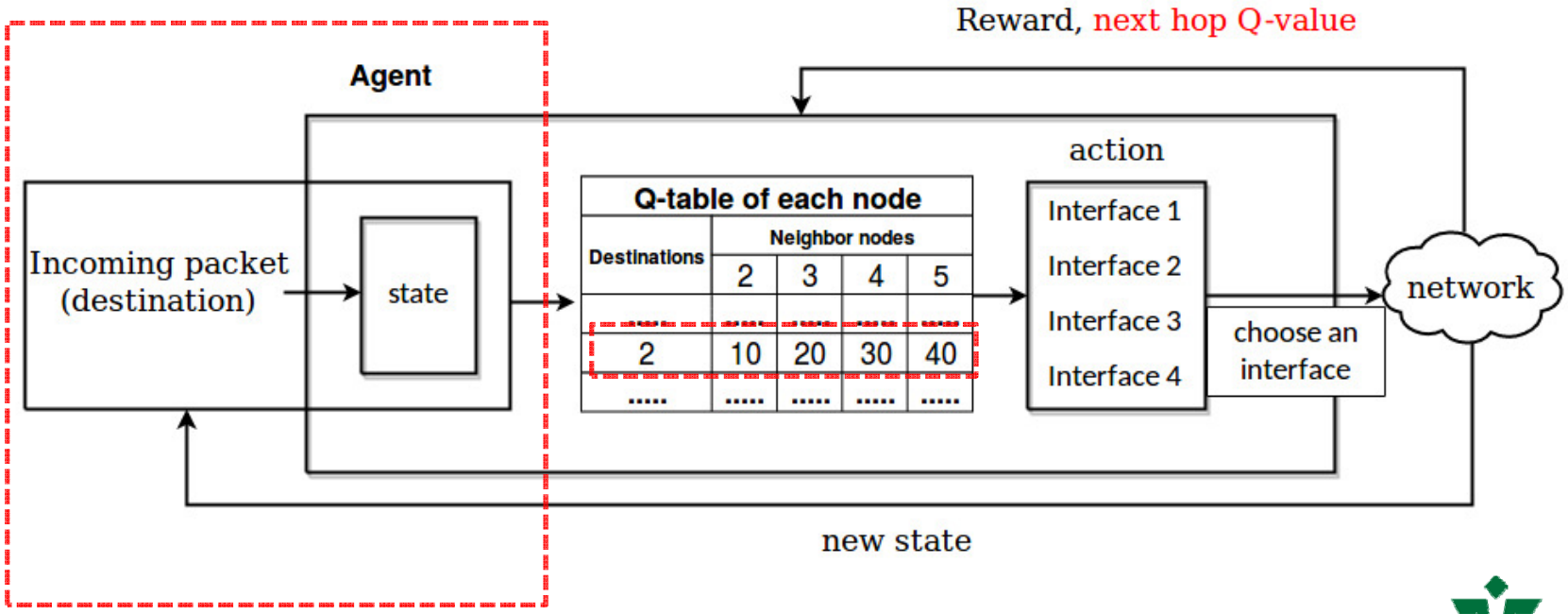


UNC CHARLOTTE

Distributed TE as a MA-MDP

- **Agent** (Each Router): when a packet arrives, observe local network state, chose an action, and receive a reward

- **Observation (local state)**: s_i = source/destination IP address, queue size, SINR, ….



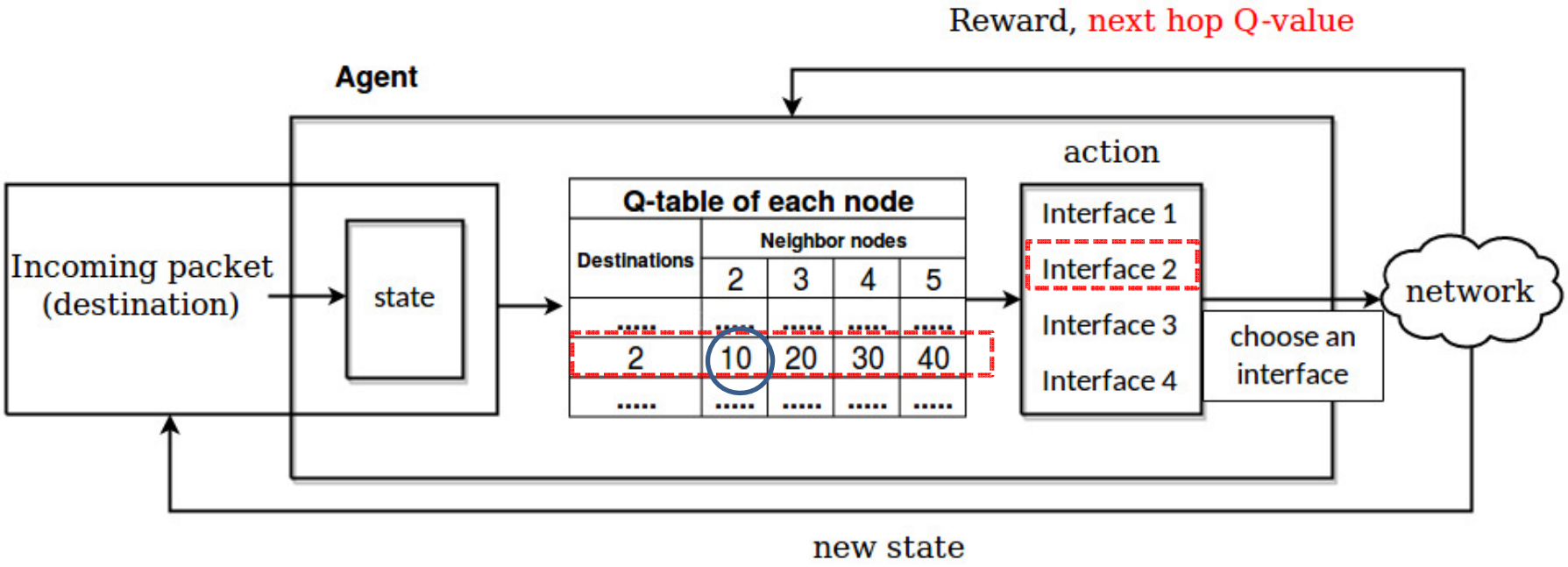
UNC CHARLOTTE

Distributed TE as a MA-MDP

- **Policy** π_i specifies how the router i chooses its action given the observation.
- **Reward:** r_i : negative one-hop delay
- **Goal**: find optimal policy for each router that maximizes the expected return (i.e., average E2E delay)

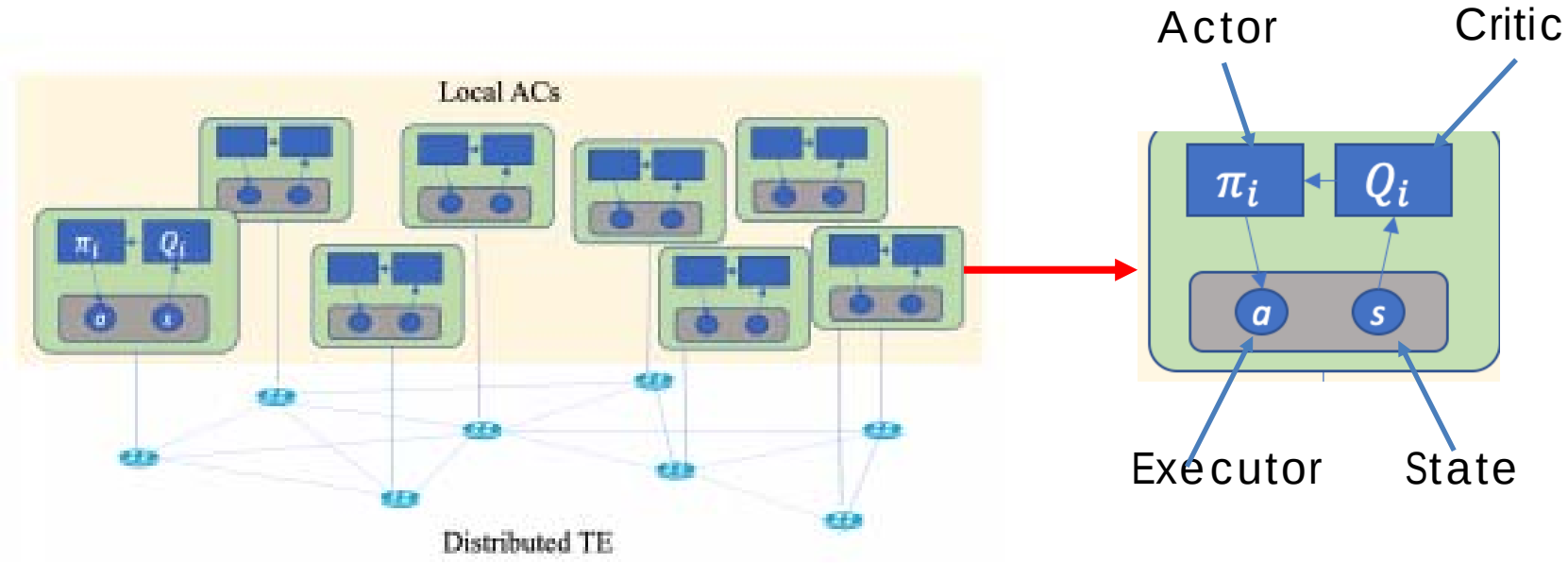
$$J(\pi) = E[G_i|\pi] = E[\sum_{i=1}^{T} r_i|\pi]$$

  - i = 1: source router of a packet (initial state),
  - i = T: destination router of a packet (termination state)



Multi-Agent Actor-Critic- Executor (MA-ACE) Architecture

- Local Critic for Policy Evaluation
- Local Actor for Policy Improvement
- Local Executor for Policy Execution



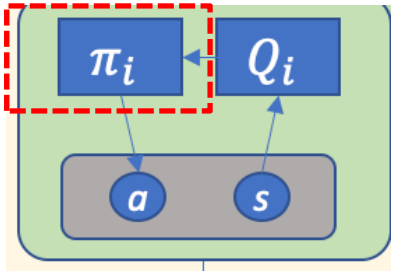
UNC CHARLOTTE

Local Actor: Action-value based Method

- **Policy Improvement** via action-value based methods

- Greedy policy: deterministic single-path routing

$$\pi_i(s) \leftarrow \arg\max_a Q_i^{\pi_i}(s,a).$$



- Near-Greedy policy: stochastic multi-path routing

  * Epsilon-greedy: select greedy action with prob of Epsilon

  * SoftMax: select action according to prob. P(a)

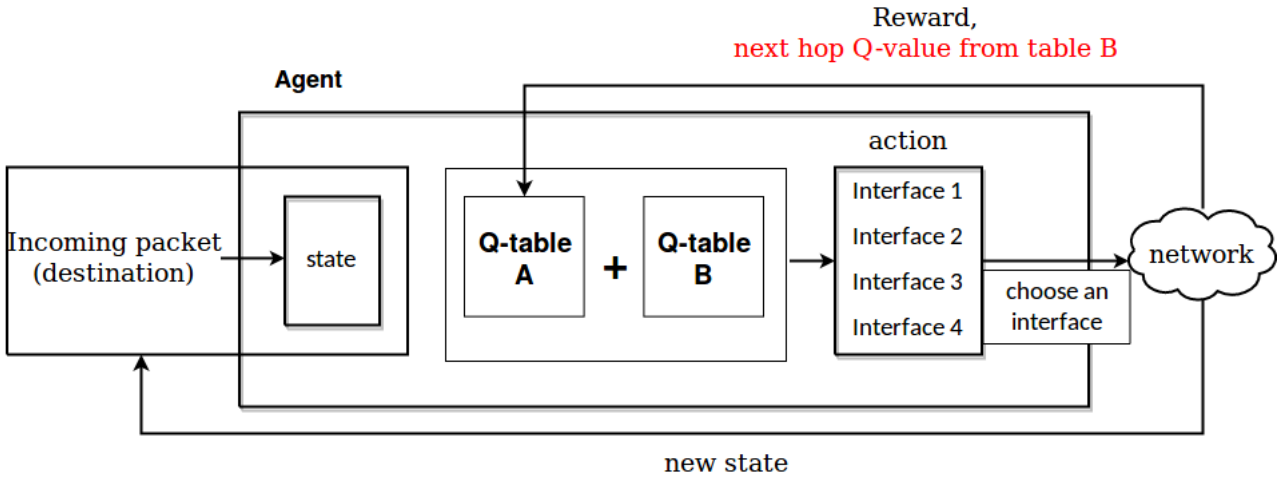
$$P(a) = \frac{\exp(Q_i^{\pi_i}(s,a)/\tau)}{\sum_{b\in\mathcal{A}_i}\exp(Q_i^{\pi_i}(s,b)/\tau)}$$

UNC CHARLOTTE

Local Actor: Double Learning

- **Motivation:** All action-value algorithms involve maximization
  - overestimation bias => degraded learning gain

- **Solution:** Double Learning
  - Decouple action section and evaluation via 2 independent estimators
    * one selects the action
    * one estimates the action value
  - Implementation: **double Q tables**



UNC CHARLOTTE

Local Critic: Spatial Difference Prediction

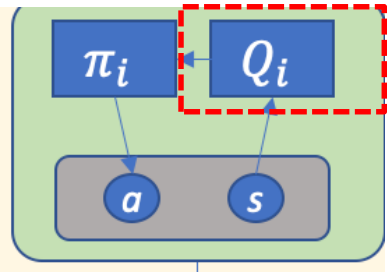
- Local **Critic** for **policy evaluation** via spatial difference prediction

$$q_i^{\pi_i}(s,a) = E\left[r_i + q_{i+1}^{\pi_{i+1}}(s',a')\right].$$

- 1-hop action-value estimation

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i + Q_{i+1}^{\pi_{i+1}}(s',a') - Q_i^{\pi_i}(s,a)]$$

1- hop exponential weighted average



- n-hop action-value estimation: smaller est. bias & higher est. variance

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i^n + Q_{i+n}^{\pi_{i+n}}(s',a') - Q_i^{\pi_i}(s,a)]$$

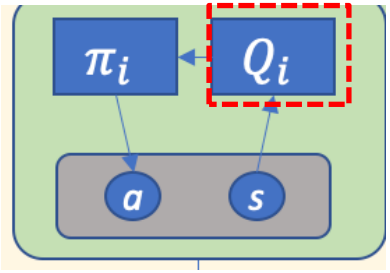
$r_i^n = \sum_{k=i}^{i+n-1} r_k$: accumulated reward when a packet arrives at the n-hop router

UNC CHARLOTTE

Local Critic: Spatial Difference Prediction

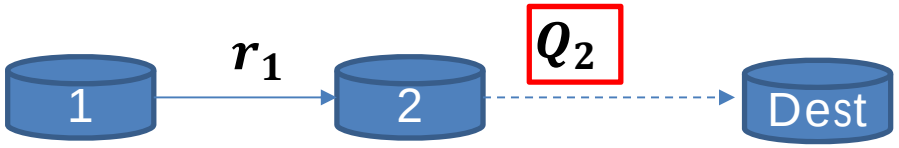
- Local **Critic** for **policy evaluation** via spatial difference prediction

$$q_i^{\pi_i}(s,a) = E\left[r_i + q_{i+1}^{\pi_{i+1}}(s',a')\right]$$



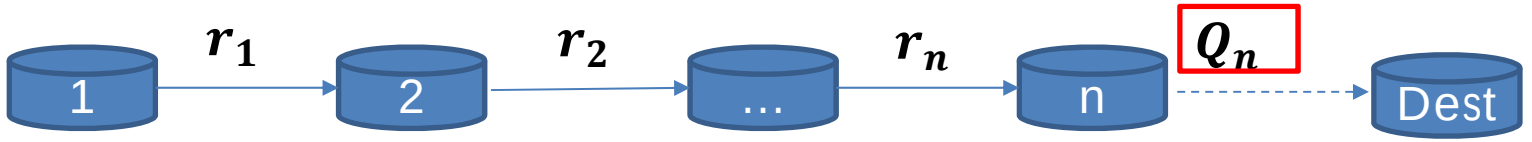
- 1-hop action-value estimation

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i + Q_{i+1}^{\pi_{i+1}}(s',a') - Q_i^{\pi_i}(s,a)]$$



- n-hop action-value estimation: smaller est. bias & higher est. variance

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i^n + Q_{i+n}^{\pi_{i+n}}(s',a') - Q_i^{\pi_i}(s,a)]$$



$r_i^n = \sum_{k=i}^{i+n-1} r_k$: accumulated reward when a packet arrives at the n-hop router

UNC CHARLOTTE

Local Critic: Expected Action-value Estimation

- **Motivation:** action change of next-hop or nth-hop router introduces high variance in action-value estimation
  * Low convergence speed

- **Solution:** Expected Action-value Estimation

$$Q_i^{\pi_i}(s,a) \leftarrow Q_i^{\pi_i}(s,a) + \alpha[r_i^n + E[Q_{i+n}^{\pi_{i+n}}(s',a')] - Q_i^{\pi_i}(s,a)]$$

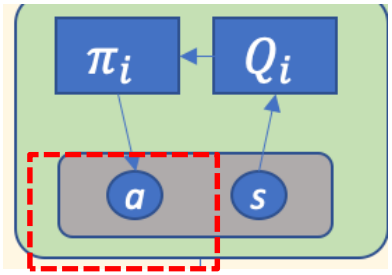
Expected action value of next-hop or n-th-hop router:

$$E[Q_{i+n}^{\pi_{i+n}}(s',a')] = \sum_{a' \in A_{i+n}} \pi_{i+n}(s',a') \tilde{Q}_{i+n}^{\pi_{i+n}}(s',a')$$

UNC CHARLOTTE

Local Executor: On-policy vs. Off-policy

- **Behavior policy** b_i **:** generates actual actions for learning agent
  - yields actual experiences for improving target policy π_i



- **On-policy:** behavior policy is **same** as target policy
  * target/behavior policy is generally near-greedy

- **Off-policy:** behavior policy is **different** from target policy
  * target policy is generally greedy
  * behavior policy is near-greedy to encourage explorations.

UNC CHARLOTTE

Experiments Setup

Discrete-time simulation

Traffic pattern

- packets periodically generated with a randomly selected source router and destination router
- Packets arrival follows Poisson distribution (λ)

Queuing and link delay

- Queue length of each router = 1000 packets
- Link delay = 1

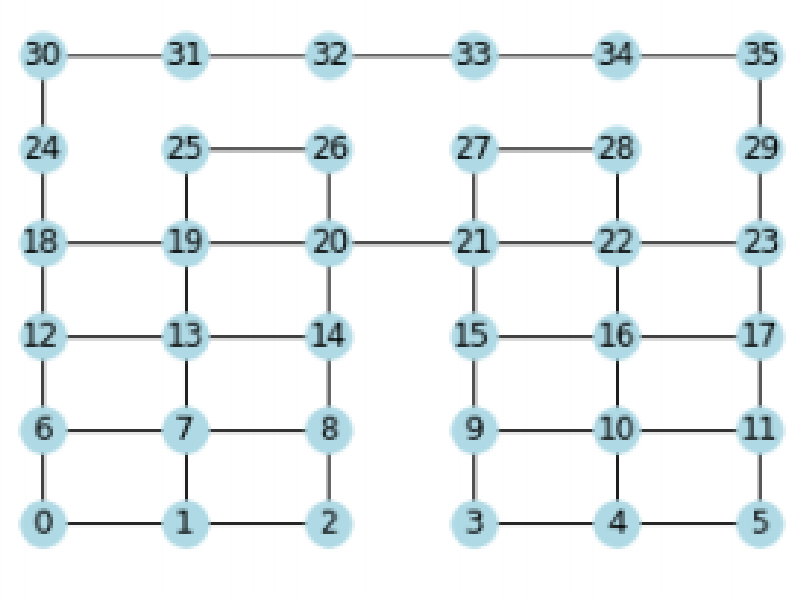


Fig. 2. Network topology in the simulations [16]

[16] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach" NIPS'94

UNC CHARLOTTE

Experiments Setup

Results evaluation with different combined algorithms

Local Actor:

- ❑ greedy
- ❑ epsilon-greedy
- ❑ Softmax
- ❑ double learning

Local Critic:

- ❑ 1-hop action-value estimation
- ❑ n-hop action-value estimation
- ❑ Expected n-hop action-value estimation

Local Executor:

- ❑ Off-policy
- ❑ On-policy

HYPERPARAMETERS

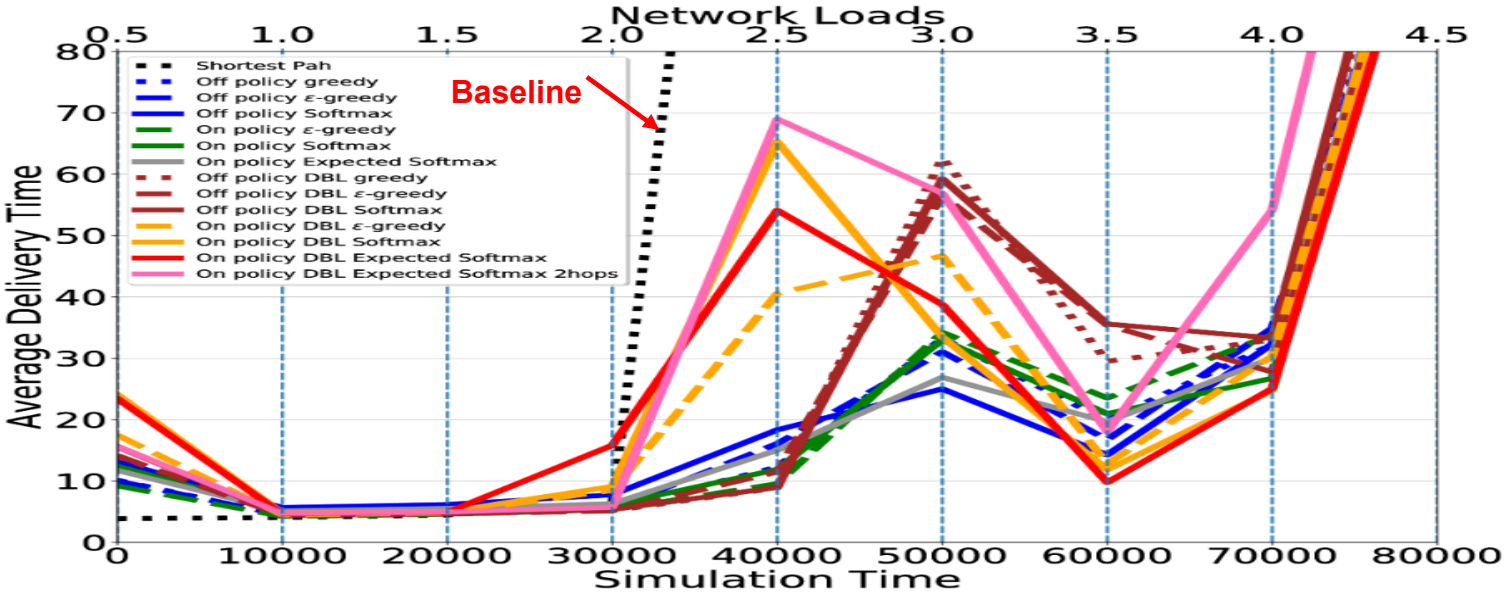
Parameter	value
Initial Q-values	0
Learning rate (α)	0.9
ε-greedy (ε)	0.1
Softmax Temperature (τ)	1
Multi-hop returns n	2

Baseline Algorithms

- ❑Shortest path
- ❑Off-policy greedy (classic Q0-routing)

UNC CHARLOTTE

Varying Traffic Loads: Low to High

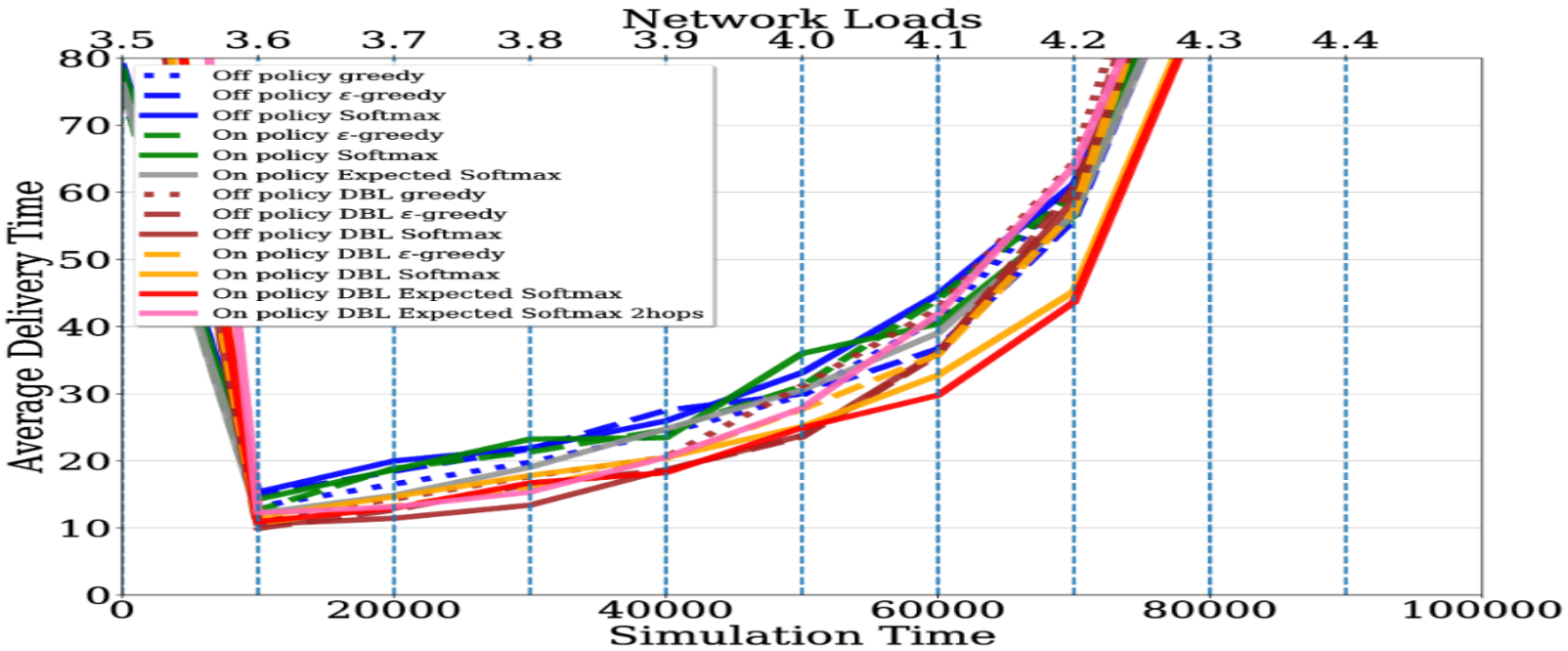


Methods	Network Loads (λ) 0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
Shortest Pah (baseline)	**3.88**	**4.05**	**4.46**	6.15	215.01	437.8	441.85	441.36	441.95
Off policy greedy (baseline)	10.14	4.43	4.78	5.51	12.27	33.25	19.87	32.24	114.89
Off policy ε-greedy	10.02	4.44	4.8	5.45	16.09	31.06	16.74	34.91	117.9
Off policy Softmax	13.14	5.69	6.17	[illegible]	[illegible]	**25.02**	14.26	32.34	111.35
On policy ε-greedy	9.22	4.24	4.59	[illegible]	[illegible]	34.22	23.54	33.85	113.43
On policy Softmax	12.43	5.06	5.38	[illegible]	[illegible]	33.02	20.96	26.74	112.85
On policy Expected Softmax	11.74	5.12	5.55	6.3	15.03	**26.89**	19.66	30.05	121.32
Off policy DBL greedy	14.05	4.39	4.64	**5.16**	**8.88**	62.68	29.53	33.0	131.63
Off policy DBL ε-greedy	14.15	4.39	4.64	**5.14**	11.5	56.57	35.[illegible]	[illegible]	[illegible].08
Off policy DBL Softmax	15.6	4.56	4.8	5.35	**8.97**	59.27	35	[illegible]	3.92
On policy DBL ε-greedy	17.48	4.46	4.79	8.47	40.55	46.68	12.[illegible]	[illegible]	19.9
On policy DBL Softmax	24.09	4.8	5.02	9.12	65.31	33.47	11.86	**24.79**	117.08
On policy DBL Expected Softmax	23.44	4.66	4.91	15.82	54.12	38.77	**9.76**	**25.07**	113.34
On policy DBL Expected Softmax 2hops	15.67	4.72	4.97	5.72	68.99	56.91	18.01	54.2	162.77

Off-policy

On-policy

Adaptivity under High Traffic Load Region

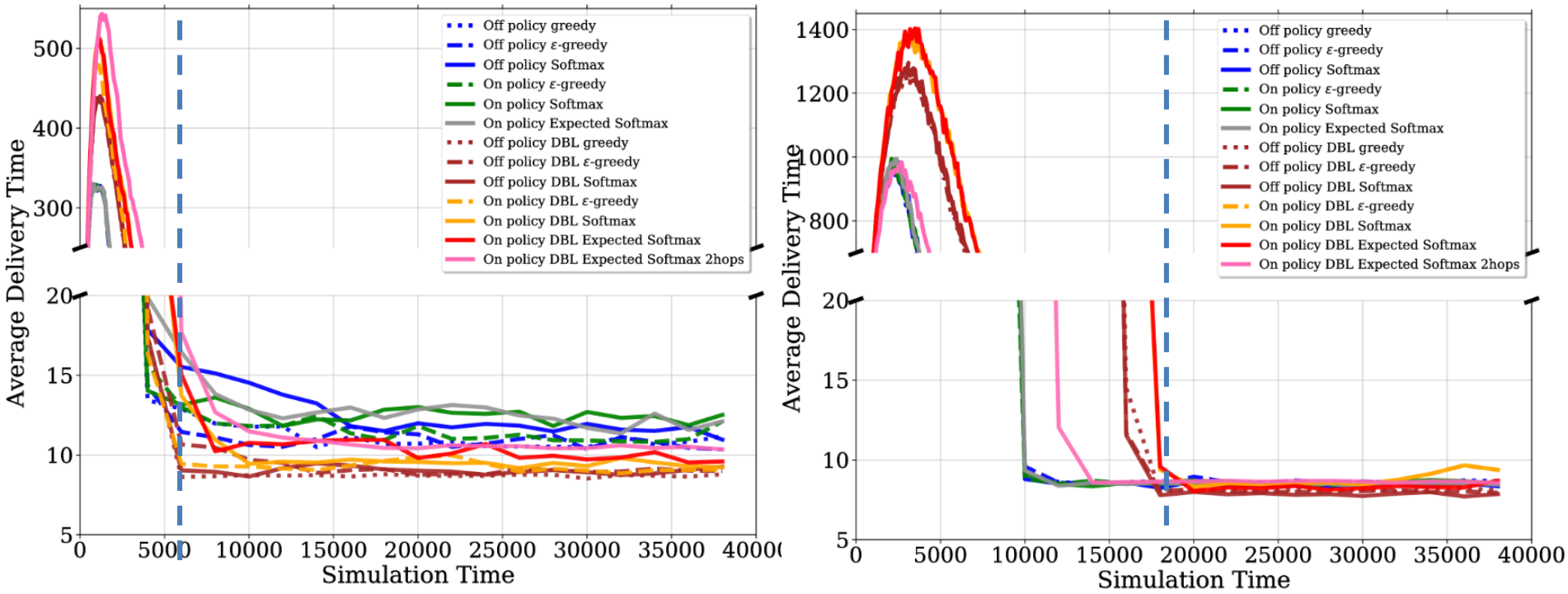


Methods	Network Loads (λ) 3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	4.3	4.4
Off policy greedy (baseline)	**75.47**	13.21	16.54	19.78	24.14	29.72	41.25	60.67	109.97	194.45
Off policy ε-greedy	77.51	14.93	18.46	21.75	27.53	29.98	36.69	55.91	103.57	193.79
Off policy Softmax	78.88	15.33	19.97	21.92	25.95	33.13	44.8	61.25	105.06	189.23
On policy ε-greedy	**75.33**	12.66	18.89	21.3	24.62	31.13	44.14	59.76	107.58	190.17
On policy Softmax	78.29	14.27	18.68	23.23	23.46	35.99	40.41	56.51	106.54	193.22
On policy Expected Softmax	76.0	12.21	14.85	19.05	24.75	30.54	39.04	56.79	102.35	189.33
Off policy DBL greedy	115.44	11.31	14.28	17.67	20.57	31.16	42.65	64.47	127.45	211.62
Off policy DBL ε-greedy	114.92	**9.92**	12.67	16.48	18.65	**23.53**	35.85	60.62	122.09	207.46
Off policy DBL Softmax	113.82	**10.53**	**11.43**	**13.39**	18.66	**23.76**	36.11	58.76	112.81	200.47
On policy DBL ε-greedy	119.95	10.72	13.26	16.03	20.36	27.69	35.84	57.07	115.98	202.2
On policy DBL Softmax	136.85	11.68	14.65	17.81	20.49	25.09	32.76	45.25	92.22	**189.29**
On policy DBL Expected Softmax	136.71	10.95	12.94	16.71	**18.23**	24.93	**29.77**	**43.66**	**90.99**	192.62
On policy DBL Expected Softmax 2hops	200.28	12.24	13.13	15.34	20.53	27.83	41.85	63.91	109.42	201.08

LOTTE

Convergence Analysis under a Stationary Case

High Network Load (λ) = 3.5



High learning rate (α = 0.9)

low learning rate (α = 0.1)

UNC CHARLOTTE

Conclusions and Key Observations

- We develop a modular & composable multiagent learning system to investigate the impact of variety of RL techniques on E2E TE
- Several techniques, i.e., Double learning, Expected value estimation, and Softmax on-policy, are complementary
  * combined learning algorithm helps to minimize E2E delay
- n-hop action-value estimation did not bring so much benefit
  * partially due to higher estimation variance and lower learning speed.
- Non-stationary network loads case
  * High learning rate ($\alpha = 0.9$) adapts faster
- The combined learning algorithms may not necessarily lead to improved performance
  * 2-hop On-policy DBL Expected Softmax did not work well

UNC CHARLOTTE