SIGCOMM 2024
— SYDNEY —

RedTE: Mitigating Subsecond Traffic Bursts with Real-time and Distributed Traffic Engineering

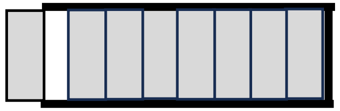
Fei Gui, Songtao Wang, Dan Li, Li Chen, **Kaihui Gao**,

Congcong Min, Yi Wang

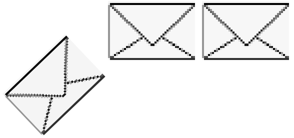
Traffic Burstiness Hurts User Experience

Internet traffic is bursty, causing:

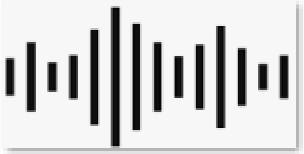
1. queue buildup



2. packet loss



latency jitter



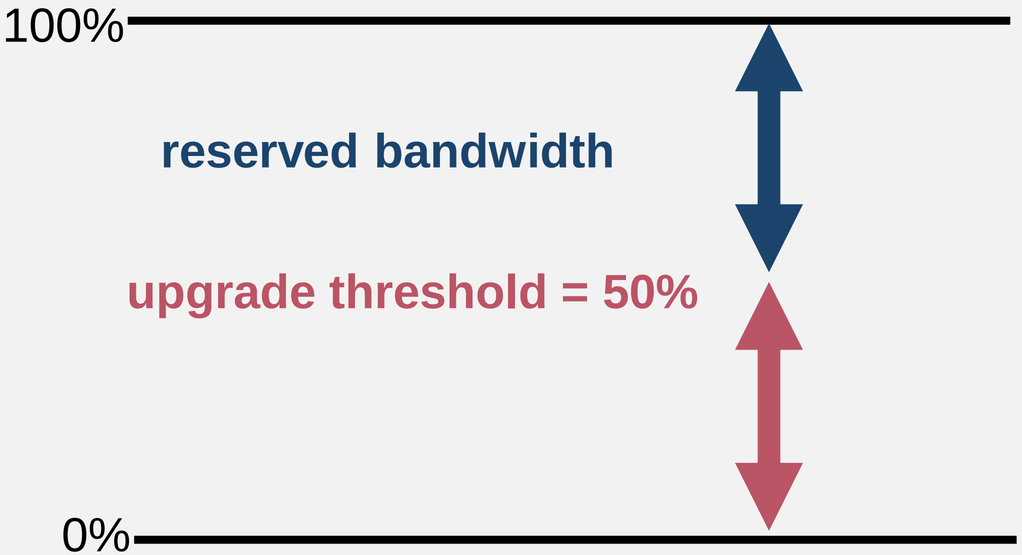
Hurting user experience of **latency-sensitive apps**

VR

Online Gaming

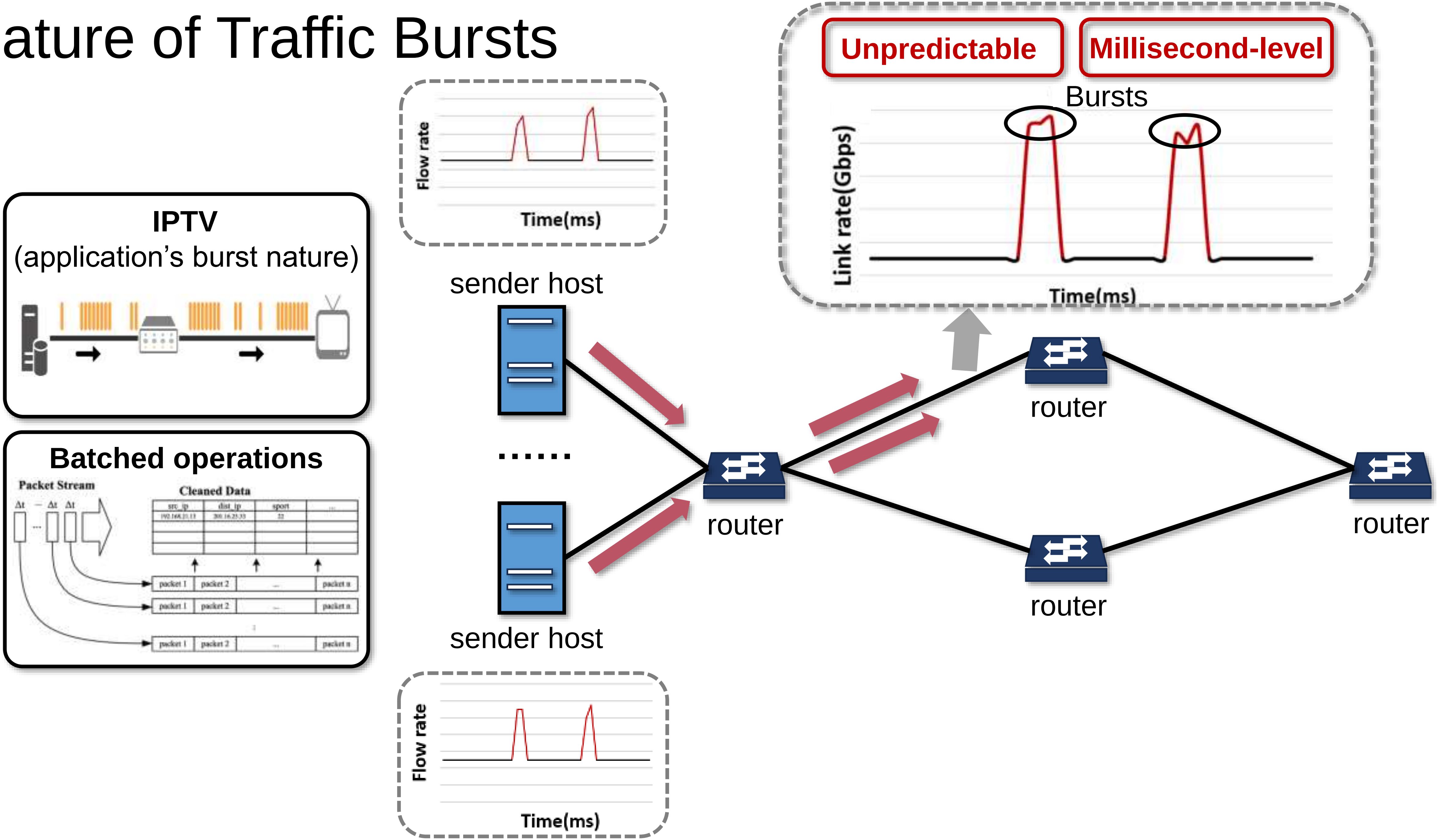
To mitigate the impact of bursts,

- **Over-provision**: ISPs upgrade bandwidth when link utilization **> 50%**[1]
  - ➢ coarse-grained
  - ➢ not cost-effective

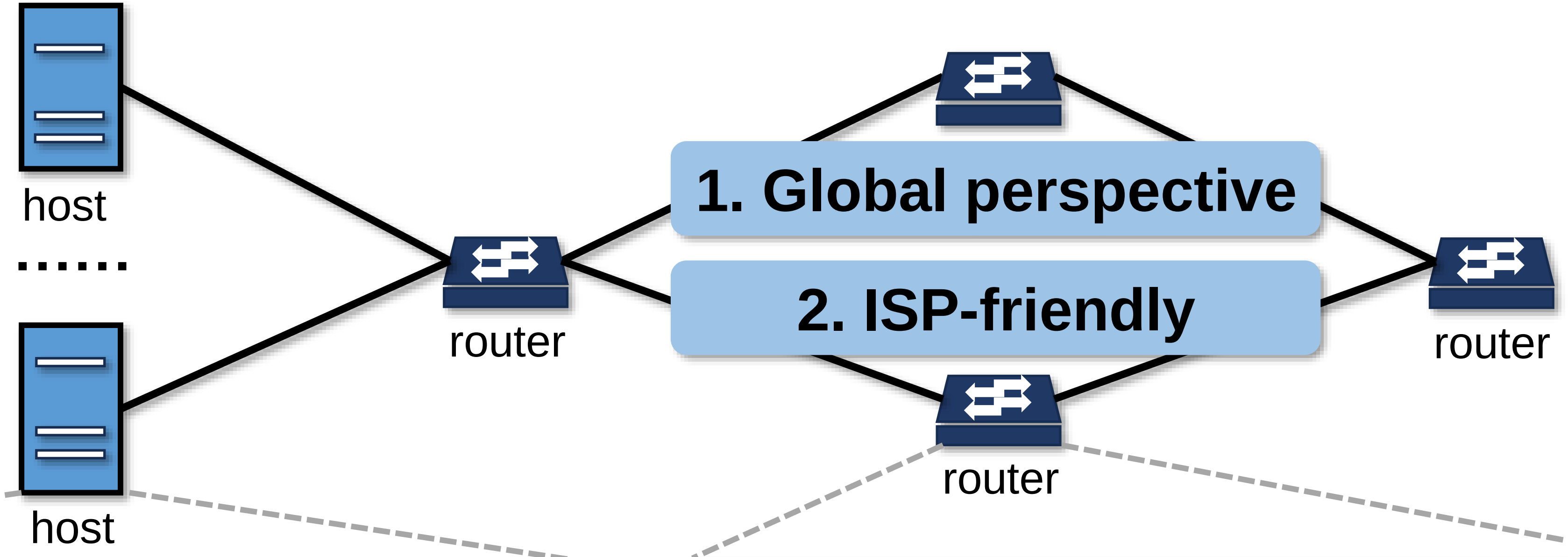


[1] Cisco. Best Practices in Core Network, Capacity Planning White Paper. 2020.

Nature of Traffic Bursts



Traditional Approaches to Mitigating Traffic Bursts

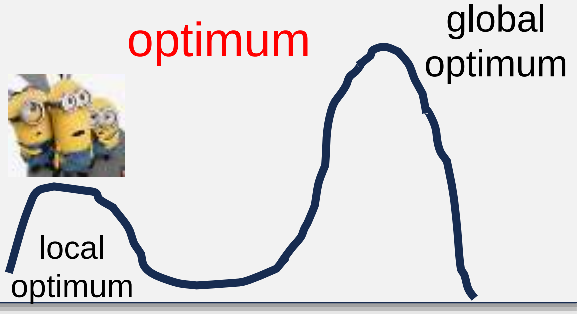


Mechanisms **in end-hosts**

- Application layer: QUIC
- Transport layer: Google BBR , **DCTCP**

Unfriendly to ISP deployment

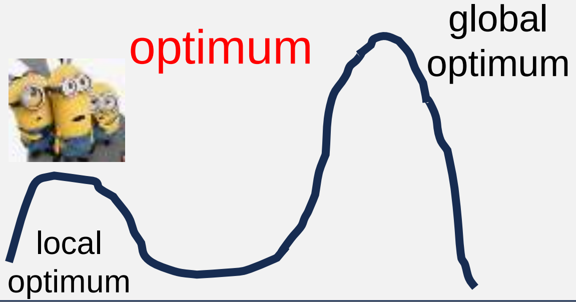
Stuck at local optimum



Device-local traffic managers **in network**

- E.g. Dynamic load balance of Cisco
- Distribute traffic among multiple paths

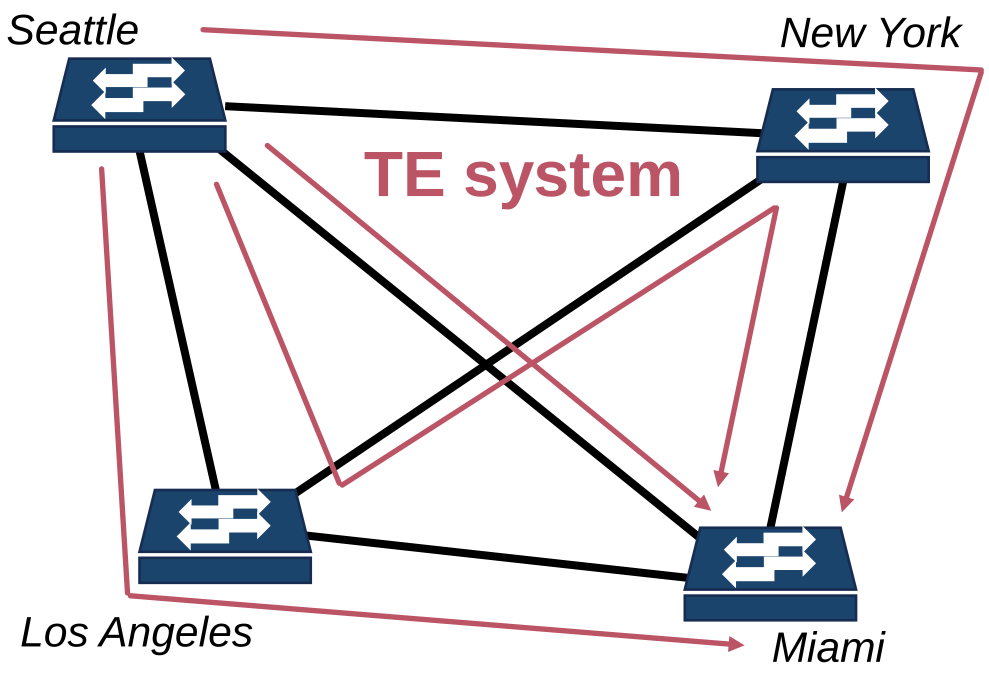
Stuck at local optimum



Traffic engineering (TE) for burst mitigation?

- Traffic Engineering (TE) has potential
  - Global perspective
  - ISP friendly

- But TE is ignored previously
  - Because of its slow decision-making speed
  - Compared to the duration of bursts, the control loop of TE operates on a larger time-scale



The Control Loop of Typical TE

1. Data collection

a. network topology

b. global traffic demand matrix

$$
\begin{array}{ccc}
 & \textit{Seattle} \quad\quad\quad\quad \textit{Miami} \\
\begin{array}{c} \textit{Seattle} \\ \\ \textit{Miami} \end{array} & \begin{bmatrix} & \cdots & 2.0\ \mathrm{TB} \\ \vdots & \ddots & \vdots \\ 1.5\ \mathrm{TB} & \cdots & \end{bmatrix}
\end{array}
$$

seconds-long

Centralized Controller

The Control Loop of Typical TE

1. Data collection

a. network topology

b. global traffic demand matrix

$$\begin{array}{c} \\ \textit{Seattle} \\ \\ \textit{Miami} \end{array} \begin{array}{c} \begin{array}{ccc} \textit{Seattle} & \quad & \textit{Miami} \end{array} \\ \begin{bmatrix} & \cdots & 2.0\text{ TB} \\ \vdots & \ddots & \vdots \\ 1.5\text{ TB} & \cdots & \end{bmatrix} \end{array}$$

seconds-long

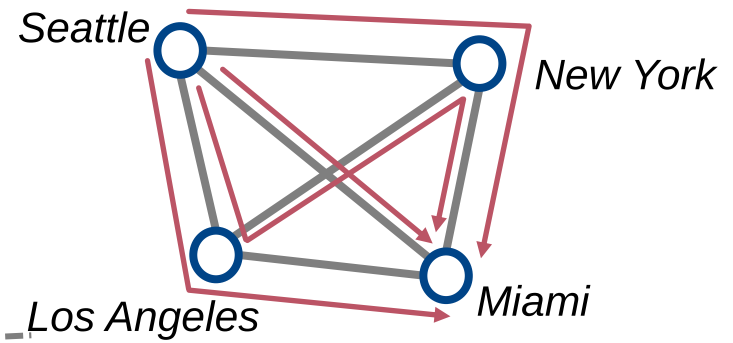
Centralized Controller

seconds/minutes-long

2. Computation

traffic split on predefined paths

(objective: *minimizing the max. link util.)*



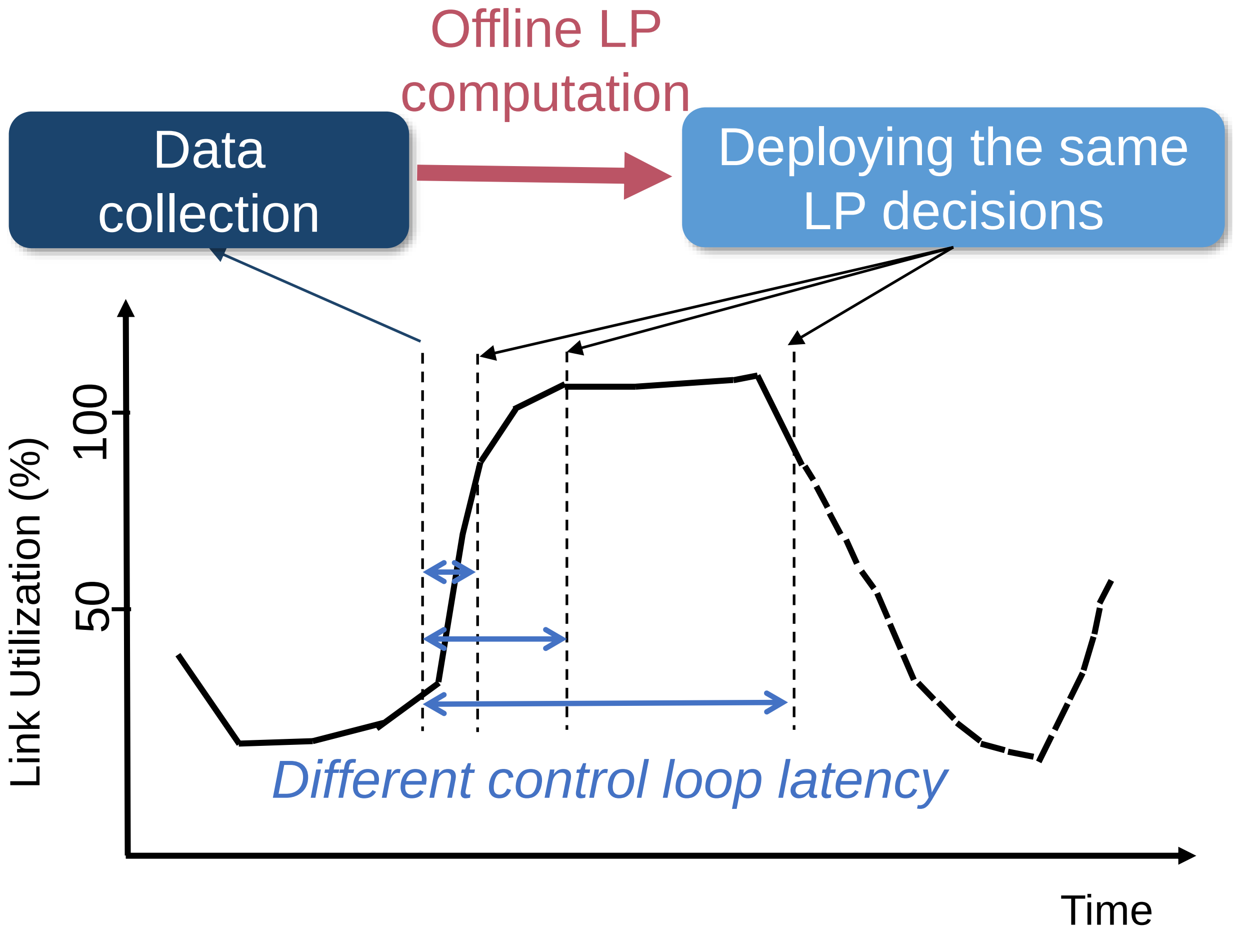
seconds-long

3. Decision deployment

Rule table in data plane		
dst. node	index	path identifier
edge router 1	1	path 1
	2	path 1
	3	path 2
edge router 2	…	…

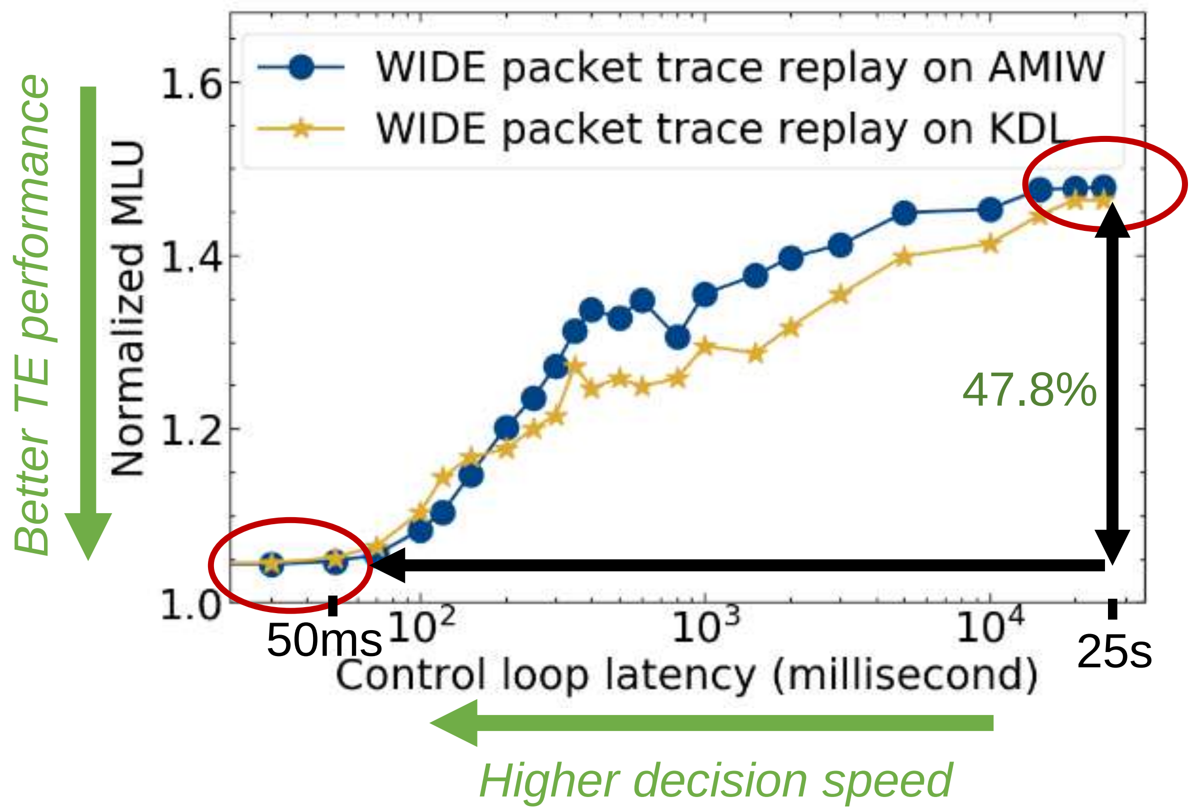
Control Loop Latency of TE Matters: Experiment Setting

- **Method:**
  - Linear-Programming-based TE
  - Simulating different Control Loop Latency
- **Traffic traces:**
  - $2k$ 15-minute packet trace segments from WIDE backbone network
  - The packet traces have 50ms-level bursts
- **Topologies:**
  - A WAN topo. from a major ISP (291 routers)
  - KDL (791 routers) from Internet Zoo



Control Loop Latency of TE Matters: Experimental Results

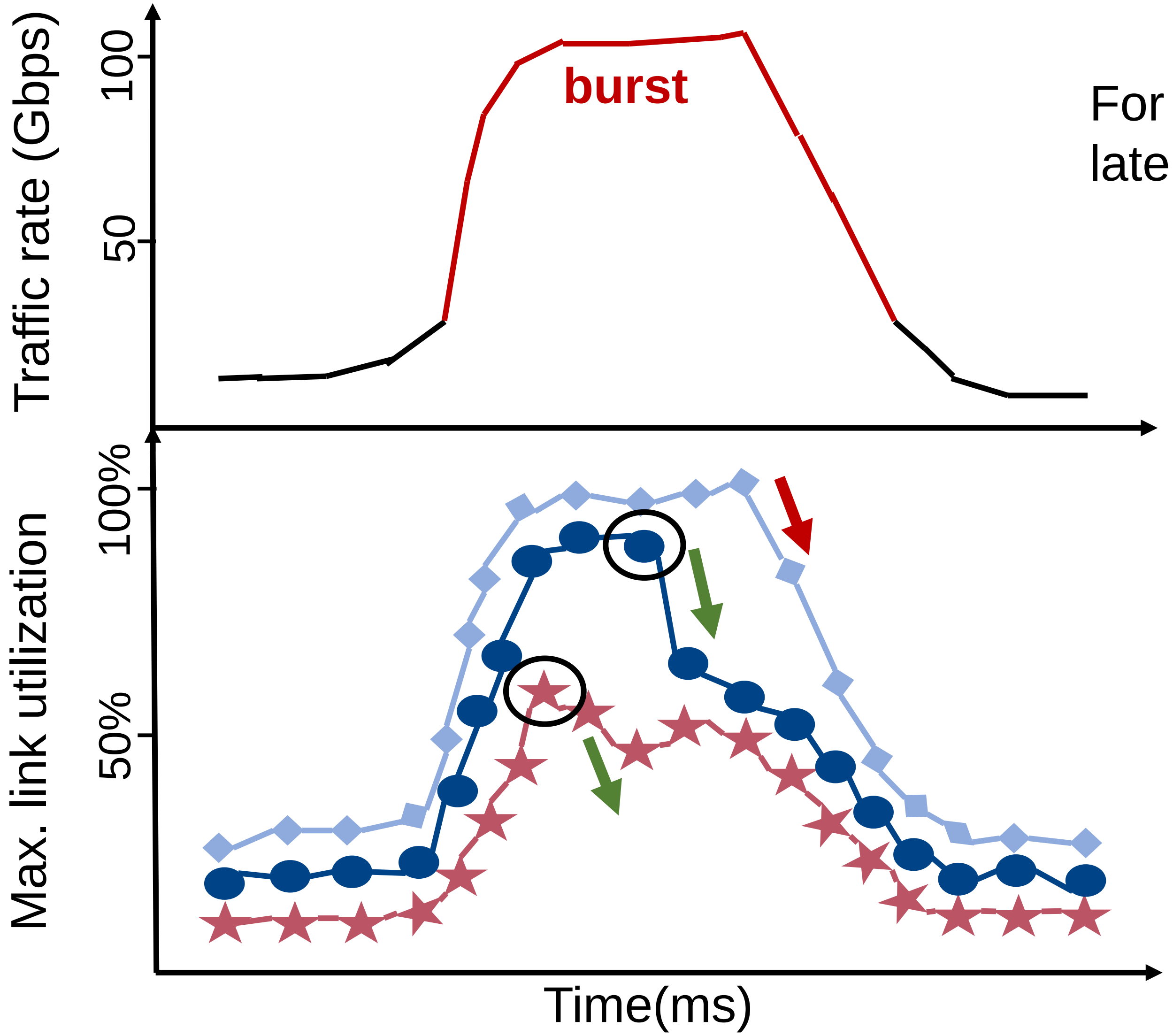
- **Method:**
  - ➢ Linear-Programming-based TE
  - ➢ Simulating different Control Loop Latency
- **Traffic traces:**
  - ➢ $2k$ 15-minute packet trace segments from WIDE backbone network
  - ➢ The packet traces have 50ms-level bursts
- **Topologies:**
  - ➢ A WAN topo. from a major ISP (291 routers)
  - ➢ KDL (791 routers) from Internet Zoo



If we reduce control loop latency, we can reduce up to **47.8%** of MLU

- The queuing length is reduced by **77.2%**
- The queuing delay is reduced by **75.9%**
- The number of events where MLU exceeds the capacity upgrade threshold (50%) is reduced by **38.3%**

Shorter Control Loop Latency Brings Better Performance



For the same burst, how different control loop latency can take effect on max. link utilization (MLU)?

- *Control loop latency > burst duration*
- *Control loop latency = 1/2 burst duration*
- *Control loop latency = 1/5 burst duration*

The control loop latency must **be close to the time scale of traffic bursts, or even smaller,** in order to effectively alleviate burst-induced congestion!!!

Question: How to reduce the control loop latency to the burst time-scale?

Current TE Systems Still Have High Control Loop Latency

- **Centralized LP-based TE**
  - ➢ POP[SOSP'21] and NCFlow[NSDI'21]
  - ➢ Accelerate the LP computation by sub-problem decomposition and parallelly solving
- **Centralized ML-based TE**
  - ➢ DOTE[NSDI'23],TEAL[SIGCOMM'23]
  - ➢ Use deep learning to speed up the computation

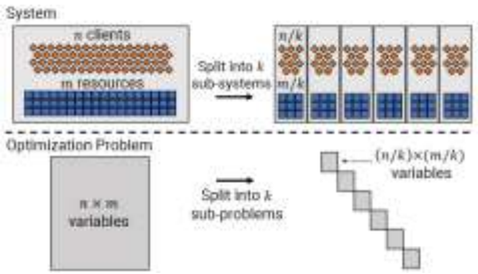
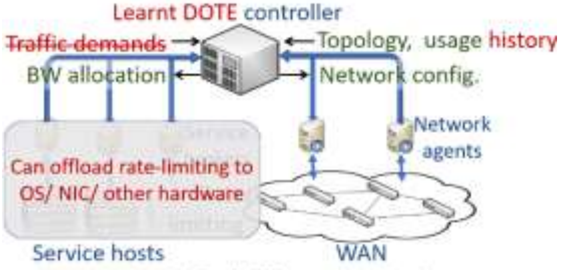
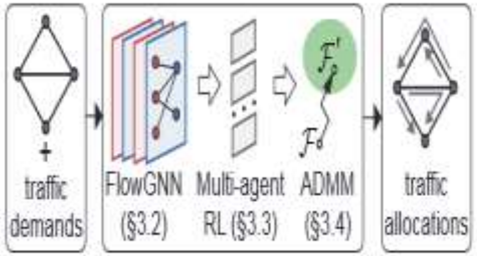


Figure 1: NCFlow's workflow.

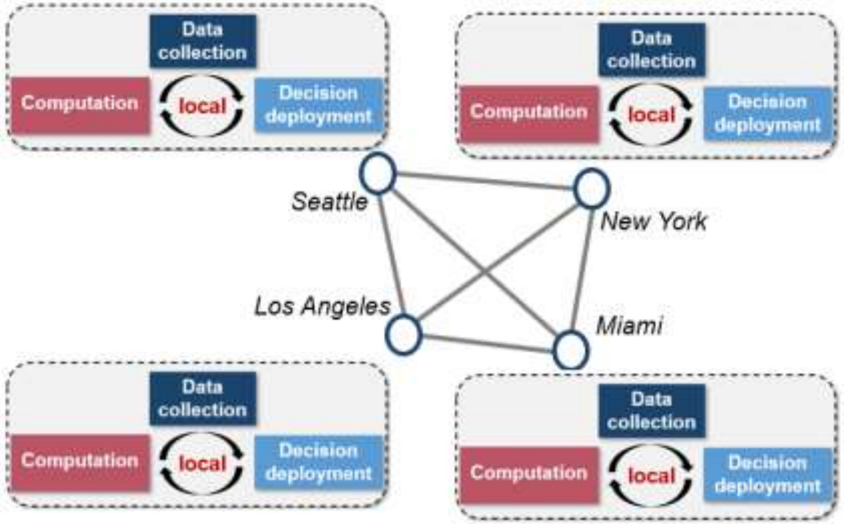


(b) DOTE with differences shown in red.



Only focus on accelerating the computation, still spend long time in data collection and decision deployment

- **Distributed TE**
  - ➢ TeXCP[SIGCOMM' 05], Halo[TON' 14], MATE[INFOCOM' 01]
  - ➢ Local input collection and local decision deployment
  - ➢ Progressively refine based on local feedback

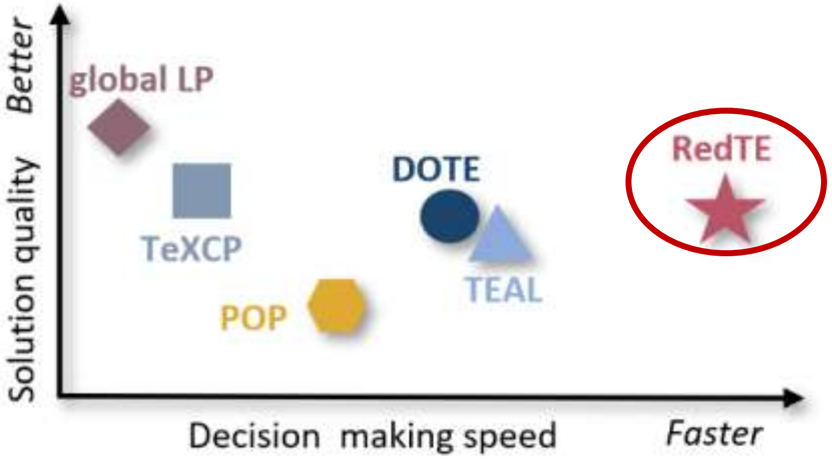


Slow multi-step convergence: at least seconds

Our Idea

Goals:

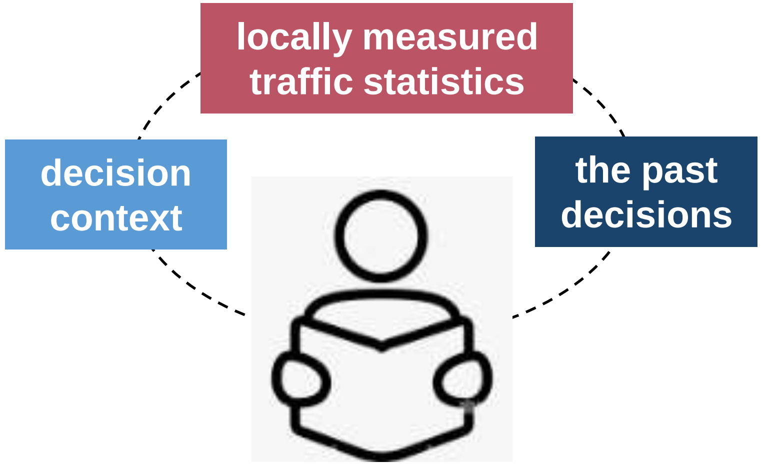
Reducing the control loop **latency** to the burst-scale while maintaining **performance** comparable to that of centralized TE systems



Answer: RedTE

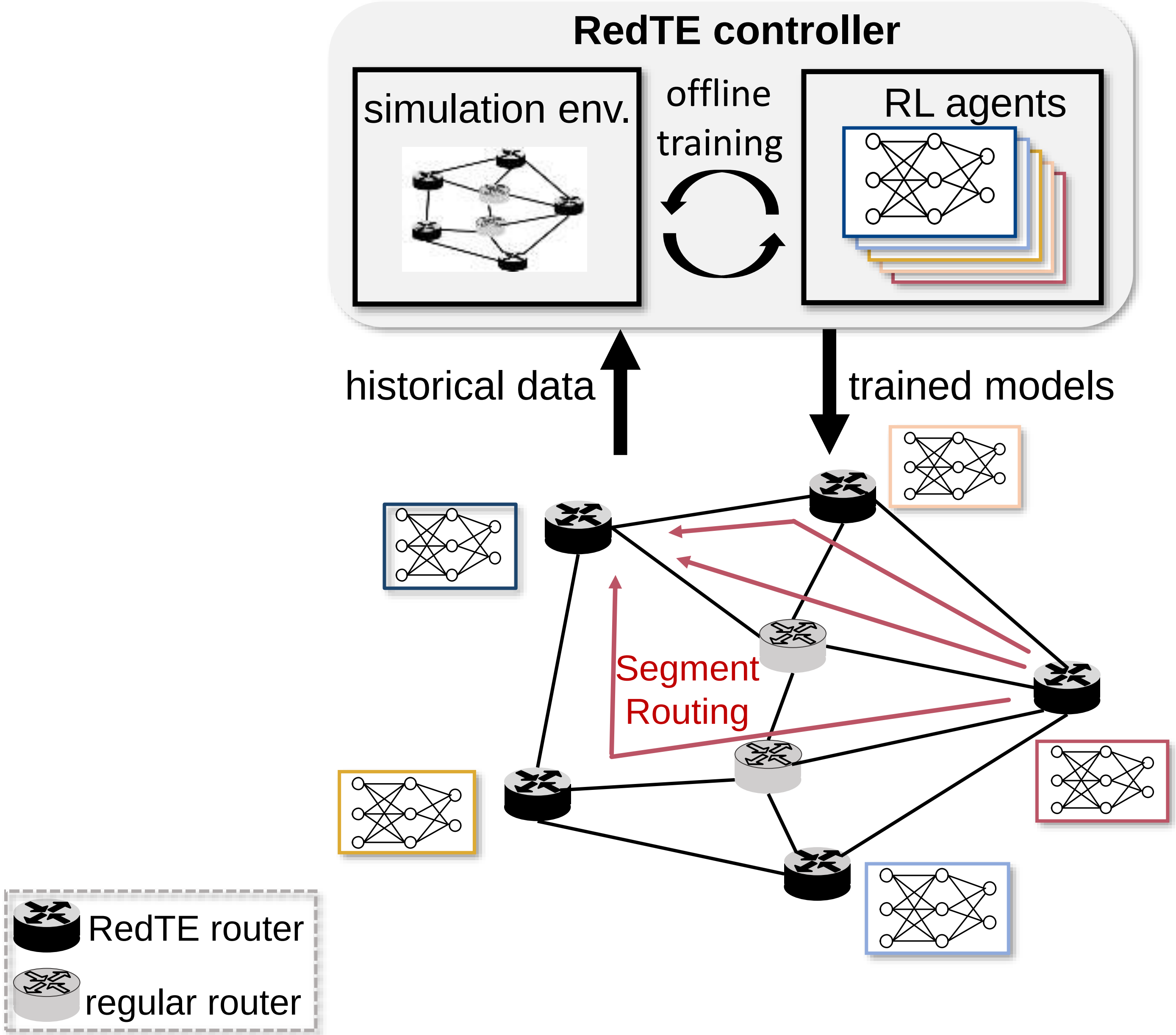
① Distributed TE, decision-making based on local info., and local decision deployment

② Multi-agent Reinforcement Learning (MARL), a router can **learn from past experience**, attempting to make the **global-informed decisions with locally information** (avoid a slow multi-step convergence)

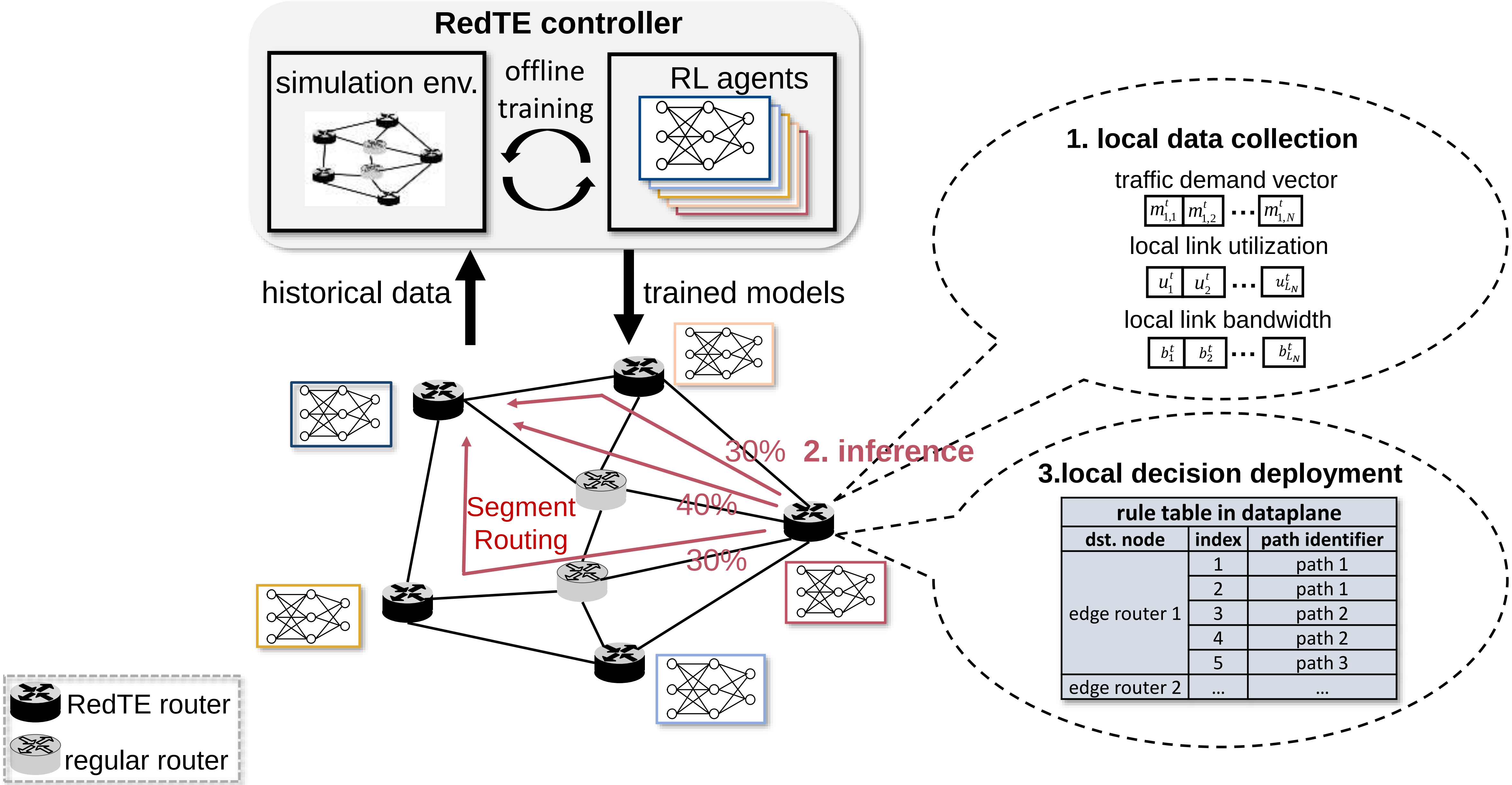


Learn from past experience

RedTE Architecture



RedTE Workflow



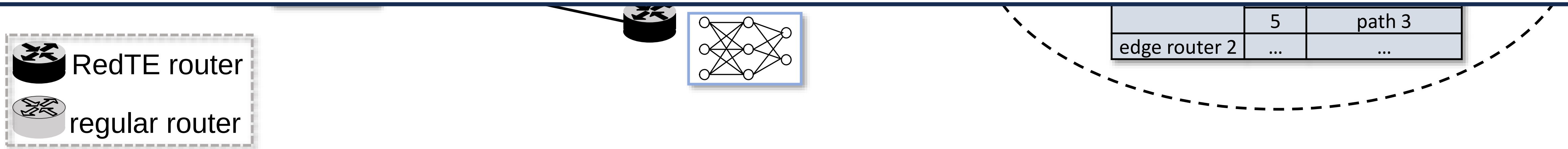
rule table in dataplane

dst. node	index	path identifier
edge router 1	1	path 1
	2	path 1
	3	path 2
	4	path 2
	5	path 3
edge router 2	…	…

RedTE Workflow

Challenges

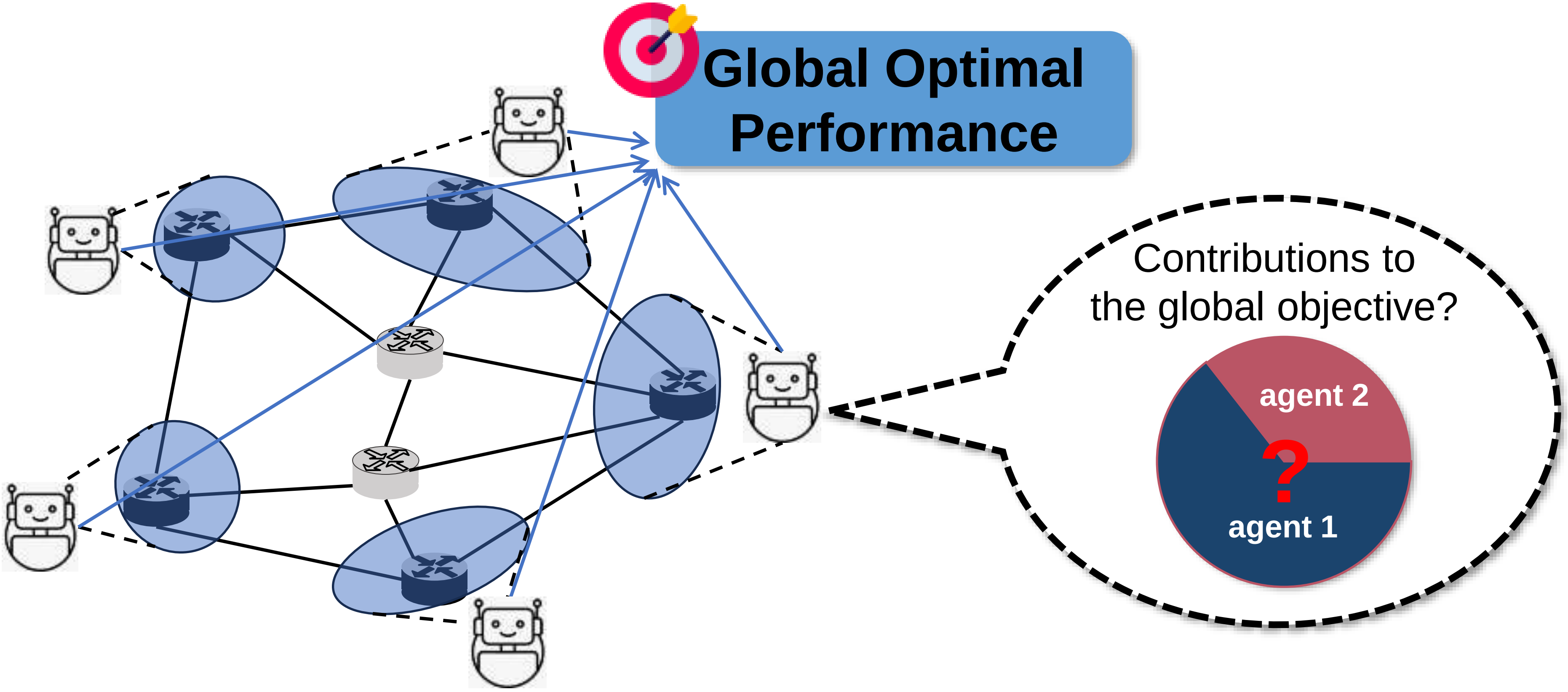
- Ensuring Collaboration Among All Agents Towards Global Optimum
- Ensuring Fast Convergence of Agent Training
- Reducing Decision Deployment Time



Challenge #1: Ensuring Collaboration Among All Agents Towards Global Optimum

- We model distributed TE system as a **cooperative multi-agent system (CMAS)**

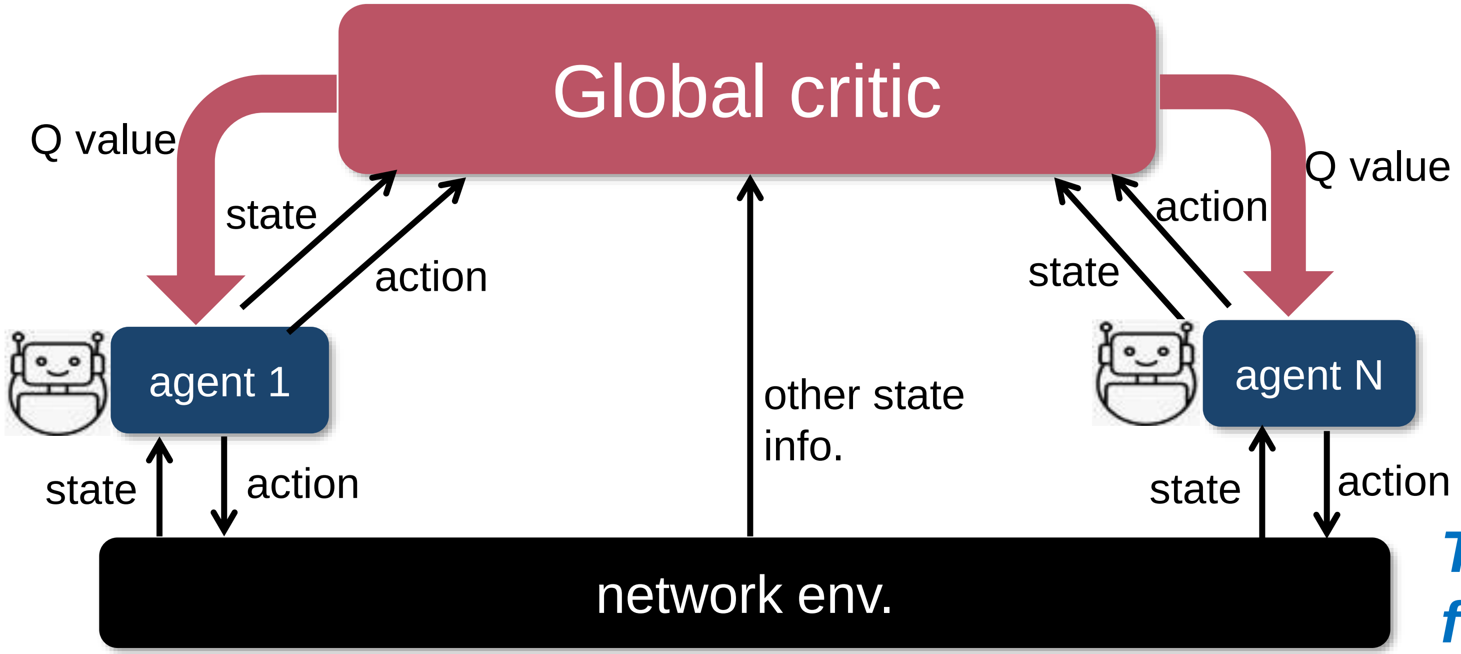
  But, since each router only has local information, how can they **cooperate towards the global optimum** instead of **each making greedy decisions**



Design #1: Enabling Cooperation by Introducing MADDPG

- Borrowing the idea from **Cooperative Game Field**, and applying the **MADDPG** (Multi-agent deep deterministic policy gradient)[2] to train RL models

MADDPG aggregates the policies of all agents into a **global critic model**, and figures out each agent's contribution to the global reward.

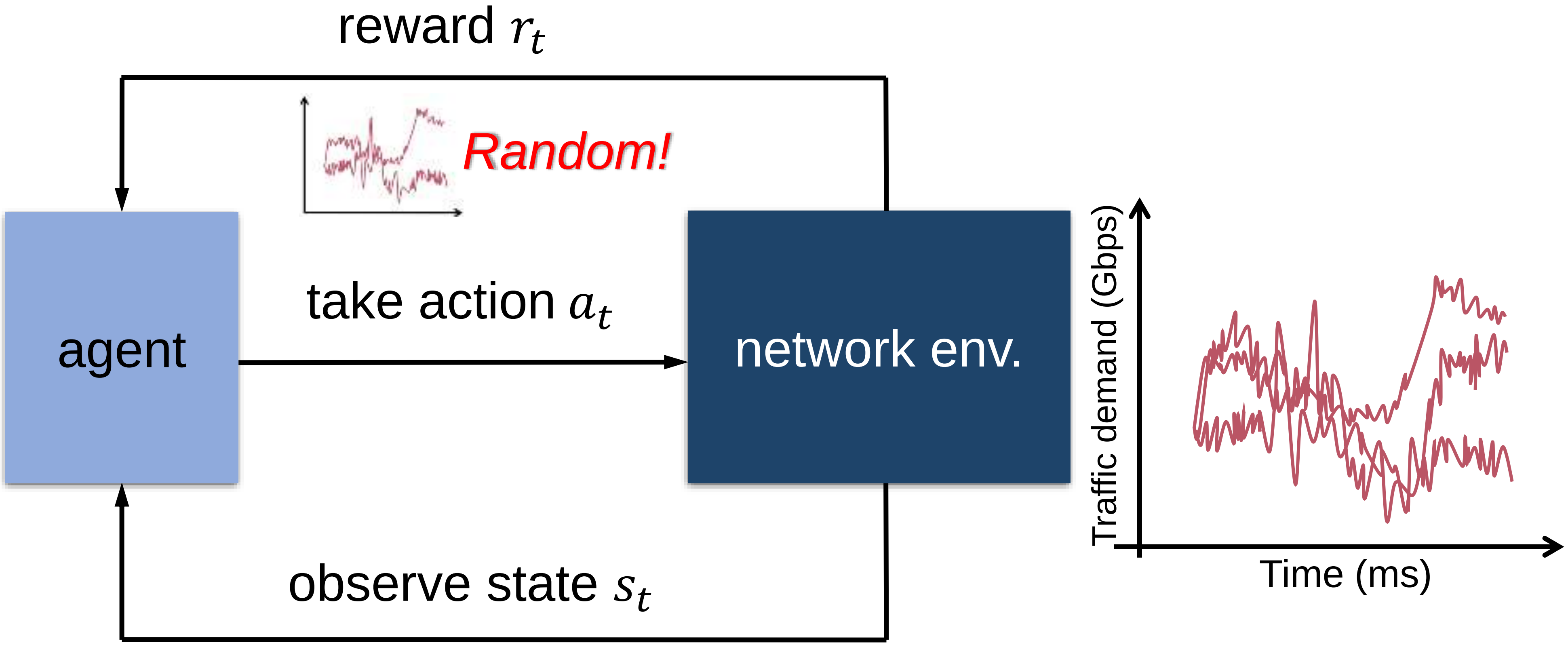


Through separate and precise feedback, agents learn to cooperate!

[2] Ryan Lowe, Jean Harb, Aviv Tamar, Pieter Abbeel, and Igor Mordatch. 2018. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *31st Conference on Neural Information Processing Systems (NIPS)*.

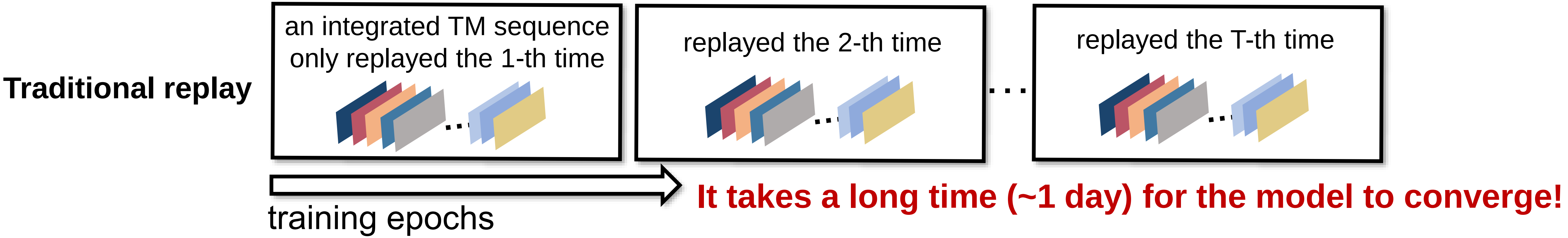
Challenge #2: Ensuring Fast Convergence of Agent Training

- Traffic-driven environment: state transition process is also affected by the arrival process of network traffic
- Problem: Randomness of network traffic → good action may receive a small reward due to a new TM sequence with a high load arrived, training takes a long time to converge

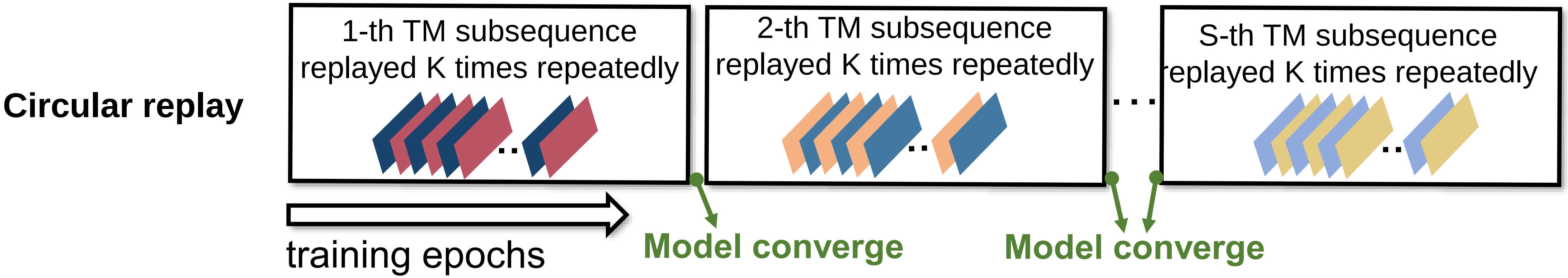


Design #2: Training with Circular Traffic Matrix (TM) Replay

Short-term memory of the RL model ⇨ **Learning the TM evolution patterns**

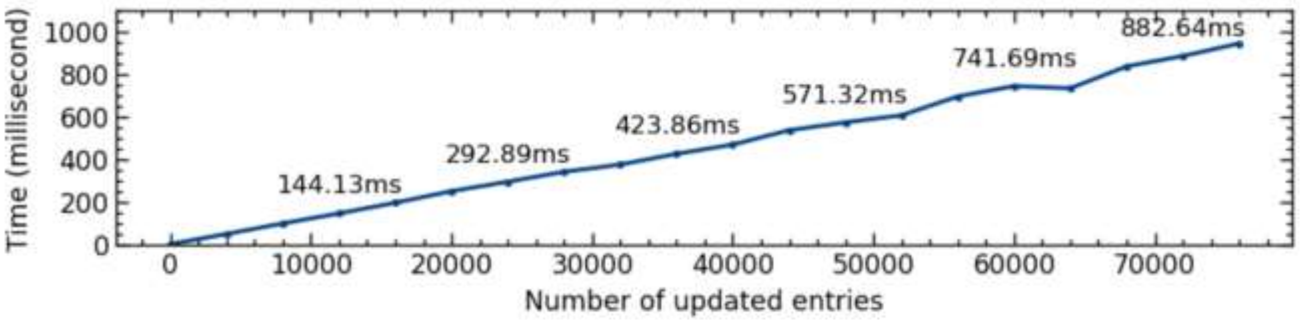


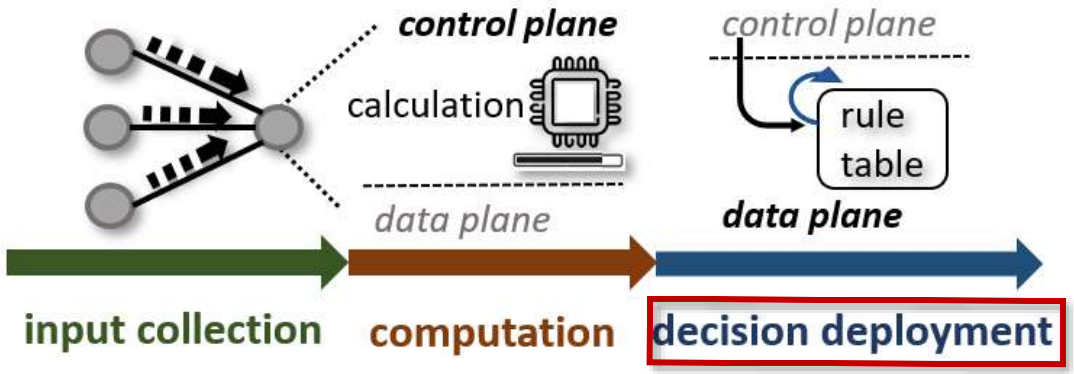
- Fix a TM subsequence and replay it multiple times repeatedly, till convergence
- Then switches to the next TM subsequence and conduct the same procedure again and again



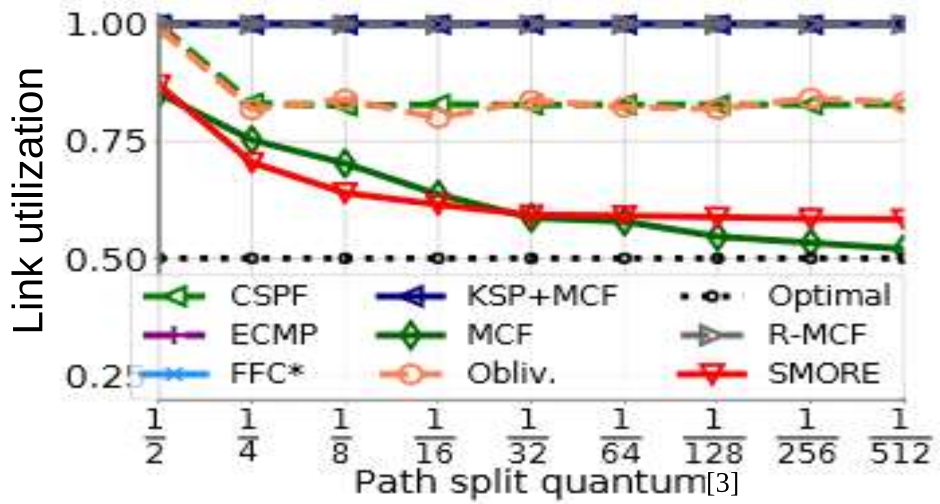
Challenge #3: Reducing Decision Deployment Time

- In the **decision deployment phase**: forwarding rule tables with many entries need to be updated
- Take **~1 second** for a rule table with 80k entries, accounting for **60%** of the total control loop latency



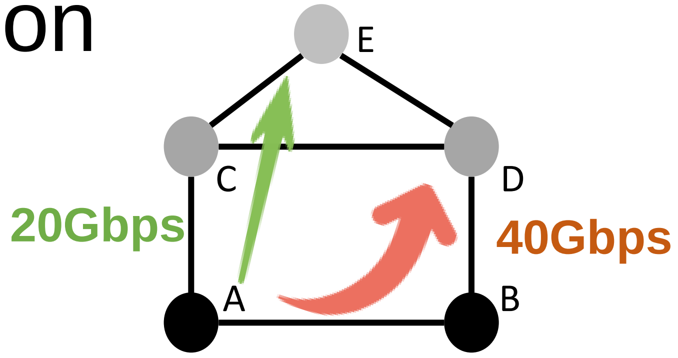


- Naïve solution: reduce the number of entries in the table
  - ➢ **sacrifice the TE performance**



[3] Semi-oblivious traffic engineering: the road not taken, NSDI 2018

Design #3: Deployment-aware Reward Function



- Insights:
  - ➢ Many **equivalent update policies**, each of which brings different time cost
  - ➢ We can **avoid unnecessary path adjustments** without sacrificing the performance

- Remove unnecessary TE adjustment by carefully engineering a **Deployment-aware Reward Function**

$$r_i = -\boxed{u_{\max}} - \boxed{\alpha * \max_{i \in (1,N)} \left\{ \sum_{j=1}^{N} f(d_{i,j}) \right\}}$$

TE performance: Max. Link Util.

Number of updated entries

Rule table		
dst. node	**index**	**path**
D	1	A->B->D
	2	A->B->D
	3	A->B->D
	4	A->B->D
	…	…
E	1	A->C->E
	2	A->C->E
	3	A->C->E
	4	A->C->E

Update policy #1:
Modifying 50% entries
→ 30% MLU

3	A->C->D
4	A->C->D

3	A->B->D->E
4	A->B->D->E

Rule table		
dst. node	**index**	**path**
D	1	A->B->D
	2	A->B->D
	3	A->B->D
	4	A->B->D
	…	…
E	1	A->C->E
	2	A->C->E
	3	A->C->E
	4	A->C->E

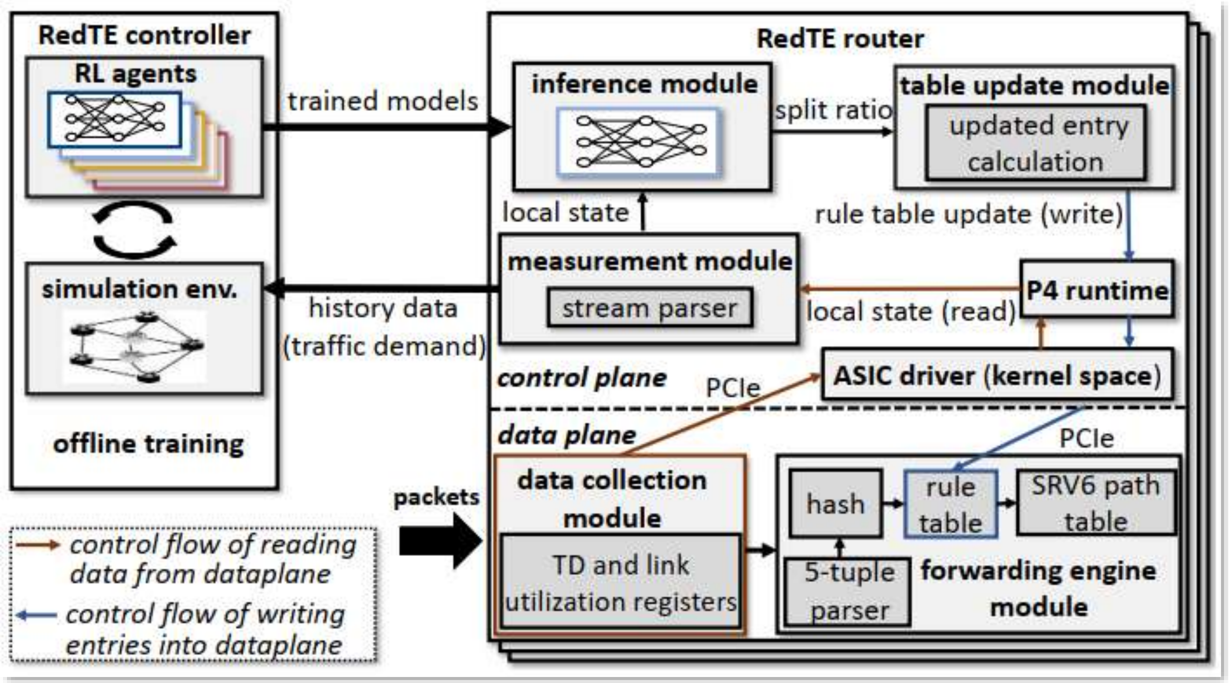
Update policy #2:
Modifying 12.5% entries
→ 30% MLU

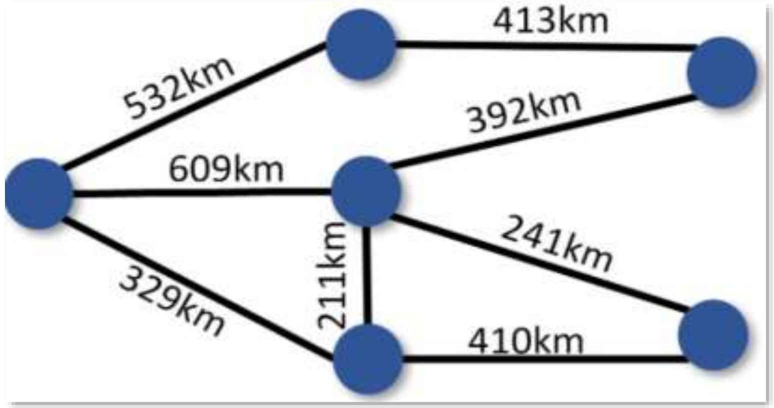
4	A->C->D

Implementation and Deployment

- The implementation of RedTE system
  - ➢ **Centralized controller** for data collection and model training
  - ➢ Implementing RedTE Router on **Barefoot Tofino platform**

- Deployment in a **real WAN**, consisting of 6 city nodes and spans 6 datacenters
  - ➢ The furthest distance between 2 nodes > 600KM

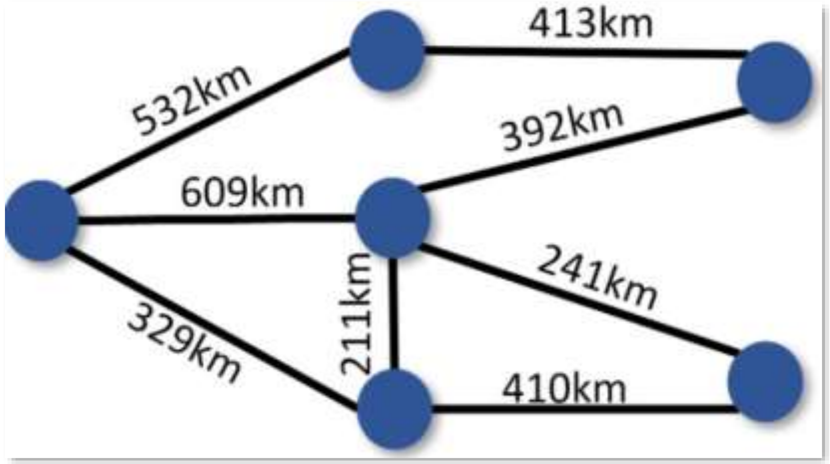




Evaluation

- **Metric**
  - TE performance
    - ✓ MLU (maximal link utilization)
    - ✓ MQL (maximal queue length)
  - Number of update entries
  - Control loop latency
- **Topology**
  - Our WAN testbed
  - 4 real WAN topologies in simulation (ns-3)
- **Packet trace**
  - Open-sourced packet traces from WIDE
- **Baseline**
  - Global LP-based TE, TeXCP[SIGCOMM'05], POP[SOSP'21], DOTE[NSDI'23], TEAL[SIGCOMM'23]

Our WAN testbed



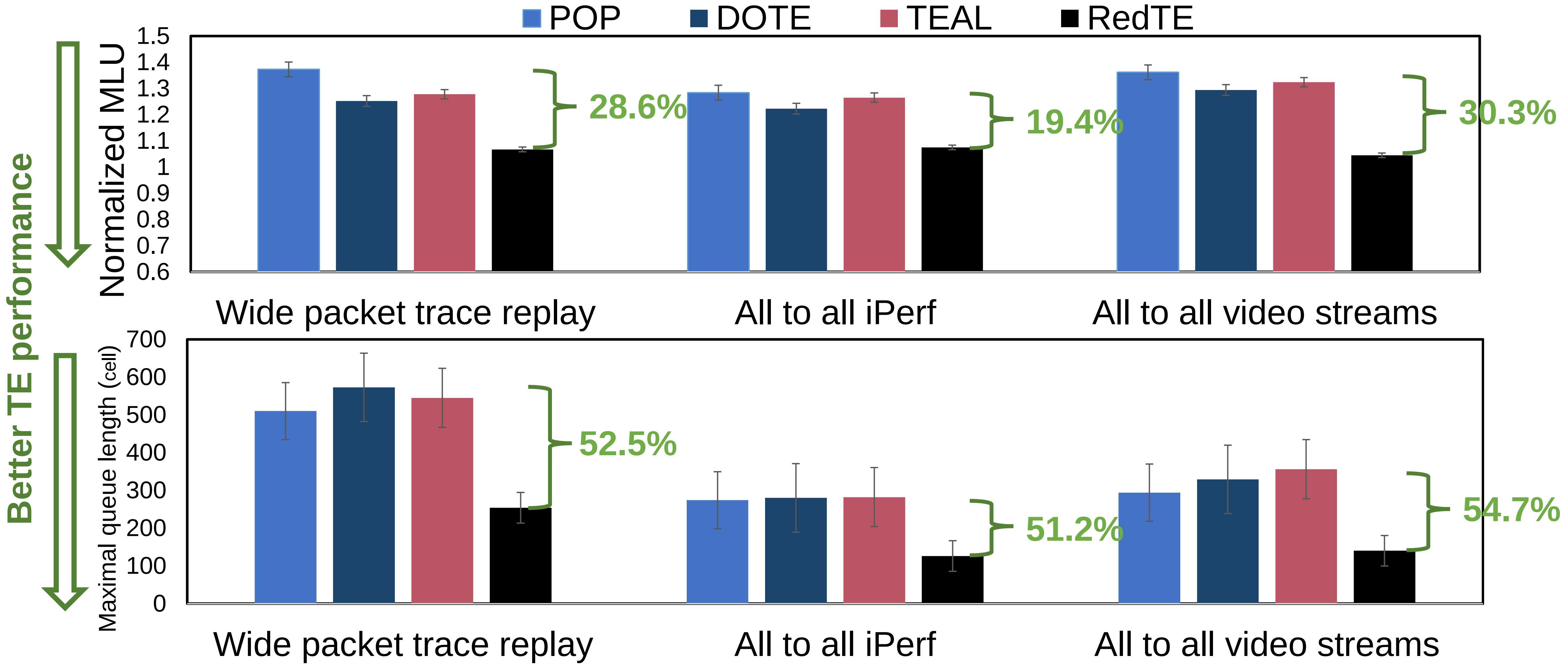
Topologies in simulation

Topo.	Scale (# router, # link)
Viatel	88, 184
Colt	153, 354
AMIW	291, 2248
KDL	754, 1790

Practical TE Performance in our WAN testbed

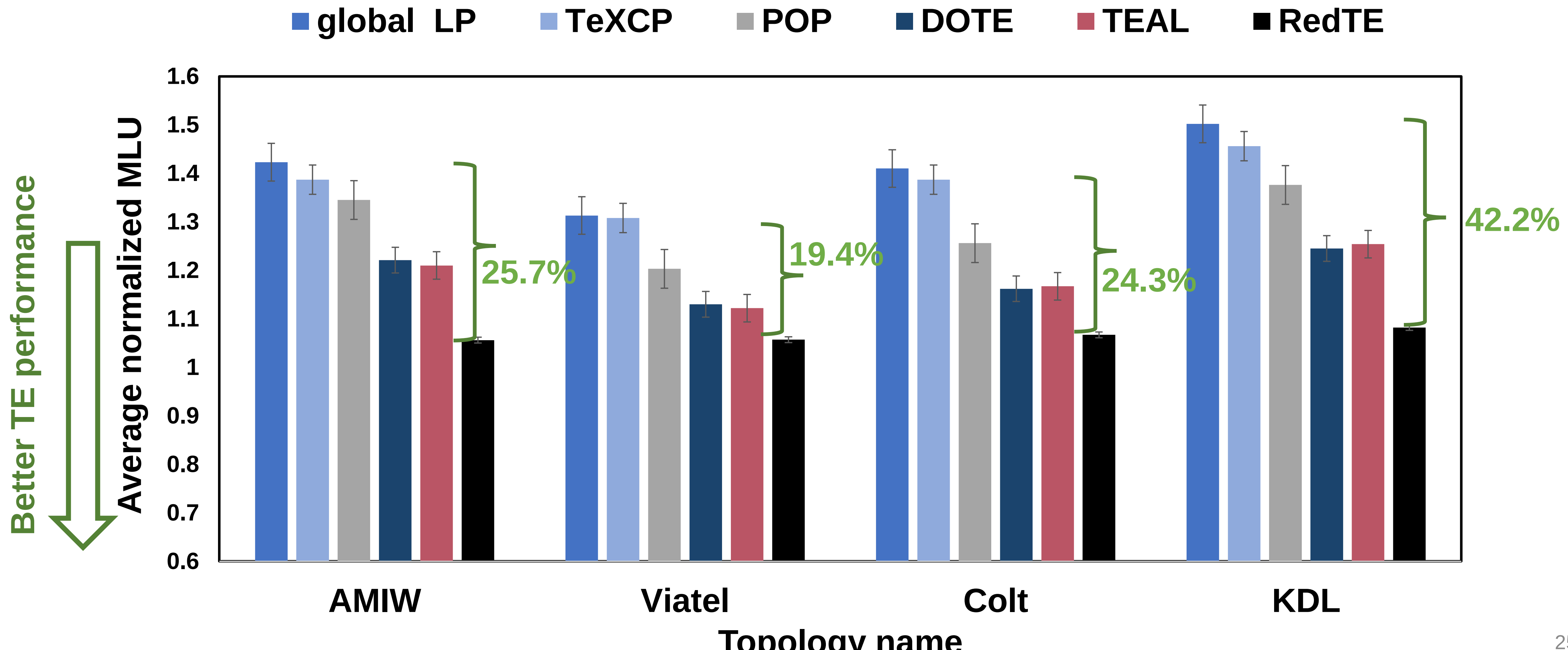
RedTE reduces ①up to **30.3%** of average normalized MLU compared to other TE methods,

② up to **54.7%** of maximal queue length compared to other TE methods



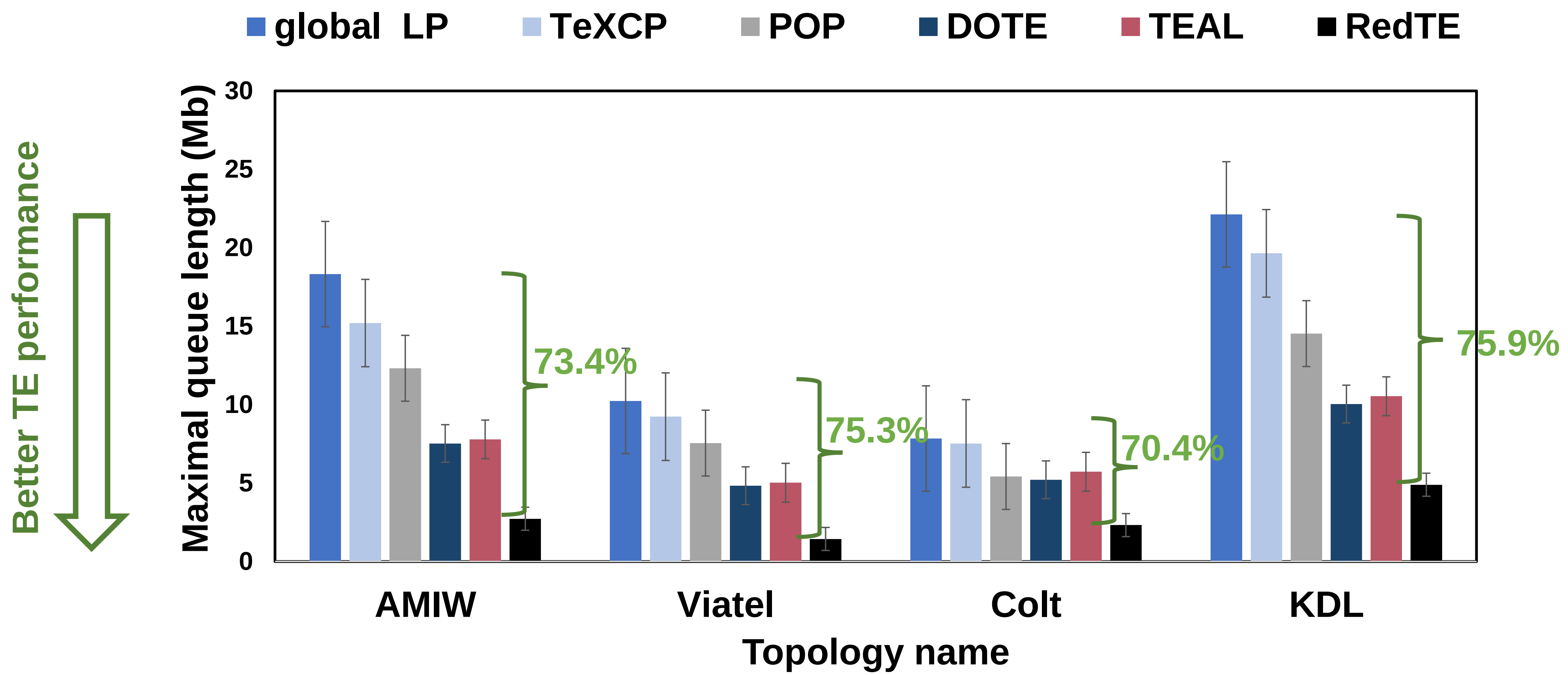
TE Performance in Simulation

RedTE reduces up to **42.2%** of average normalized MLU compared to other methods



TE Performance in Simulation

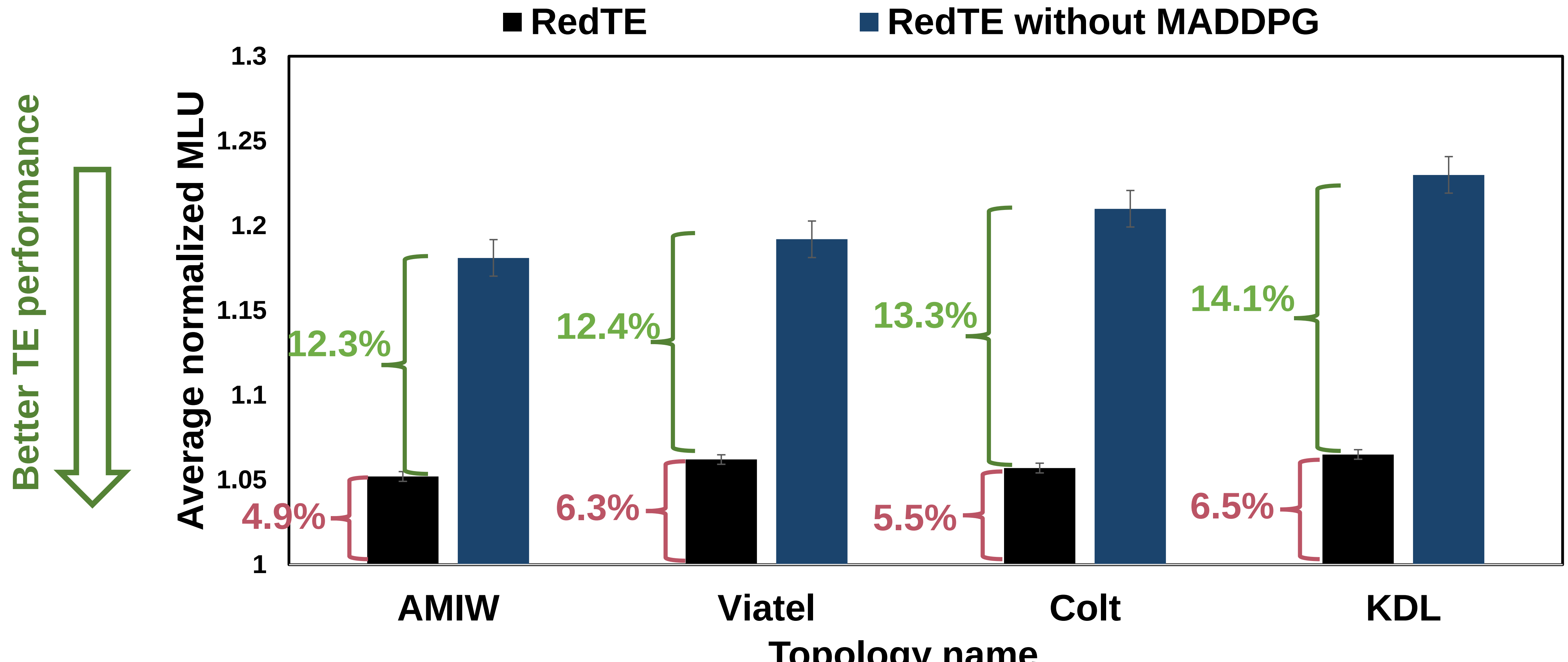
RedTE reduces up to **75.9%** of maximal queue length compared to other TE methods



Microbenchmark (Design #1)

By using the MADDPG algorithm in model training

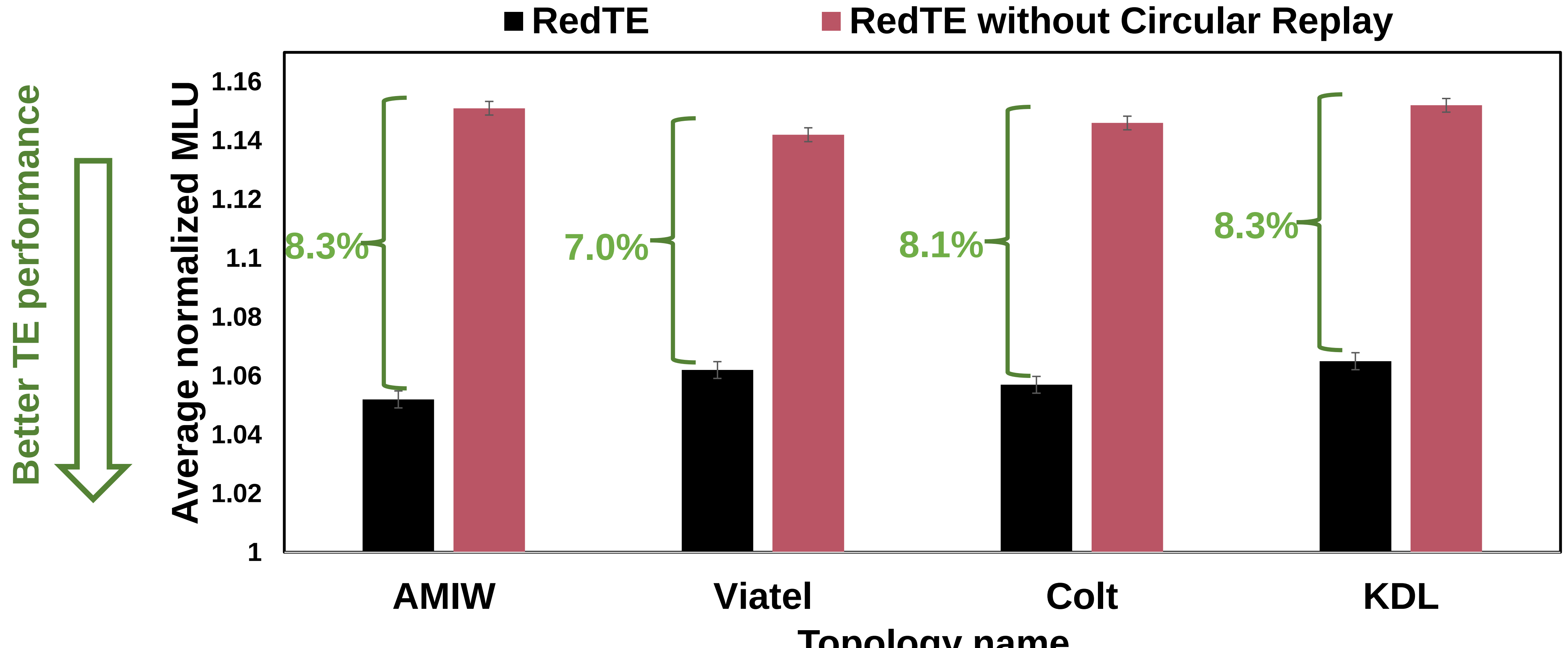
- RedTE reduces the average normalized MLU by up to **14.1%**



Microbenchmark (Design #2)

By employing circular traffic replay

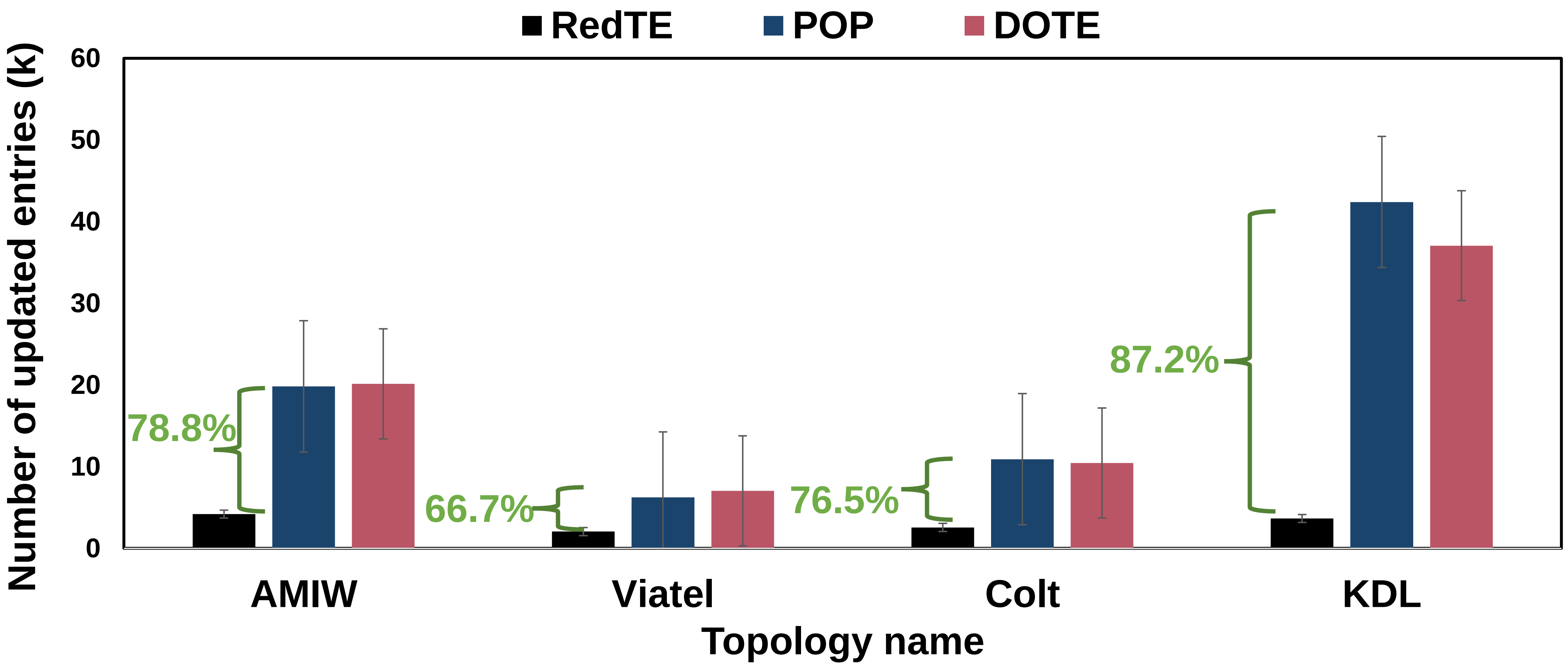
- RedTE reduces the convergence time of model training by up to **61.2%**
- RedTE reduces the average normalized MLU by up to **8.3%**



Microbenchmark (Design #3)

By using the new reward function that includes a penalty term

- RedTE reduces the number of updated entries by up to **87.2%**



Control Loop Latency

- RedTE achieves **<100ms** control loop latency in both our WAN testbed and large-scale simulations
- RedTE speeds up the control loop by **2.7X - 341.1X**, compared with other TE systems

Data collection time / Computation time / Rule table updating time

topology (#nodes, #edge)	Colt (153, 354)	AMIW (291, 2248)	KDL (754, 1790)
global LP	– / 2120.75 / 120.70	– / 4803.46 / 200.17	– / 32022.00 / 519.30
POP	– / 68.98 / 113.00	– / 228.00 / 193.05	– / 1427.03 / 452.10
DOTE	– / 50.50 / 105.85	– / 150.15 / 198.10	– / 563.40 / 504.30
TEAL	– / 24.95 / 123.27	– / 69.42 / 223.56	– / 476.73 / 563.38
RedTE	**3.45 / 5.26 / 29.60**	**5.19 / 7.69 / 47.10**	**11.09 / 12.57 / 71.90**

Conclusion

1 Key Finding:

TE can mitigate bursts if its control loop latency can be smaller than the burst time-scale

3 Core Designs:

Enabling Cooperation by Introducing MADDPG

Training with Circular TM Replay

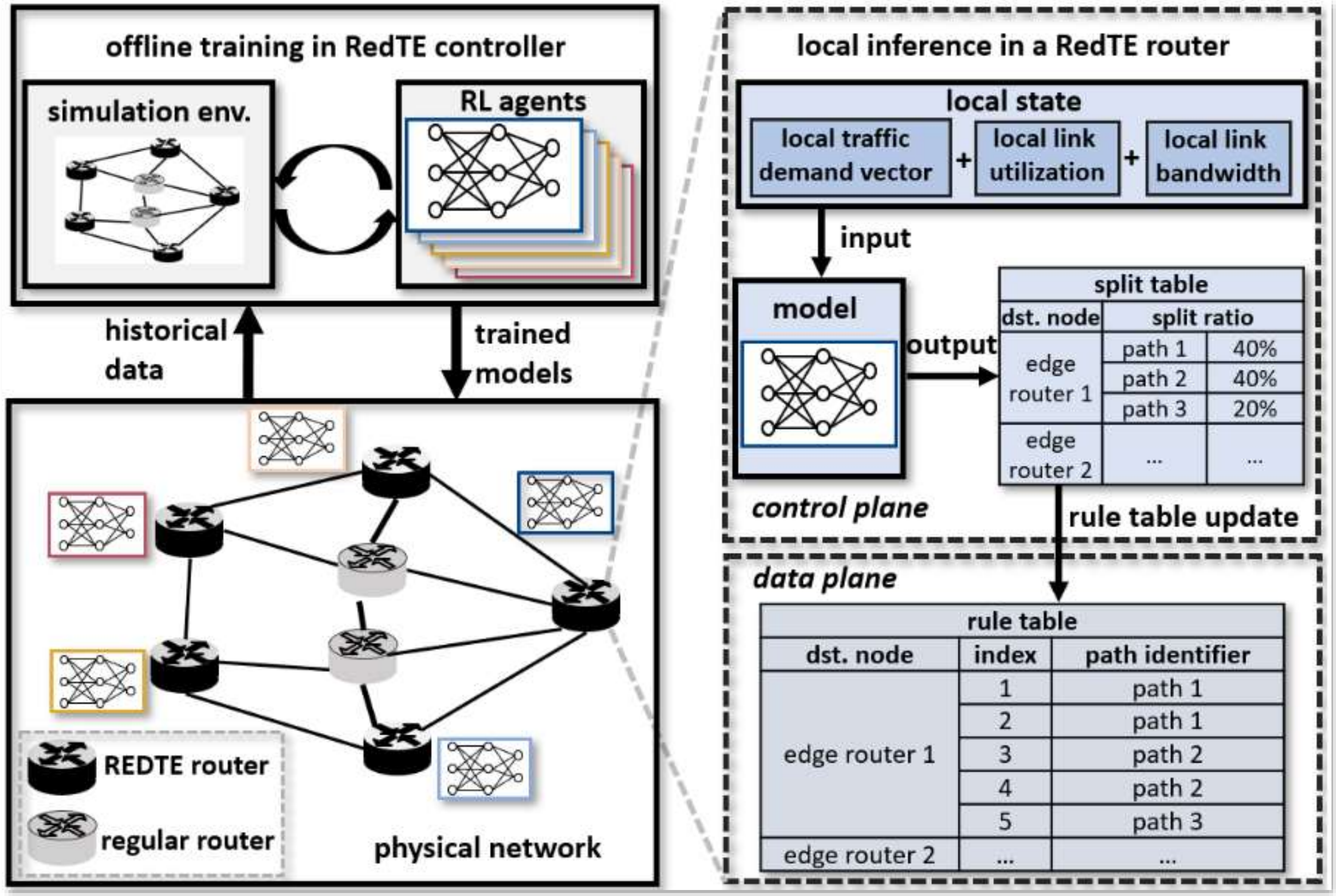
Deployment-aware Reward Function

Real Deployments and Experiments:

Improving 30% TE performance

< 100ms Control Loop Latency

RedTE Architecture



Q&A