Perseverance-Aware Traffic Engineering in Rate-Adaptive Networks with Reconfiguration Delay

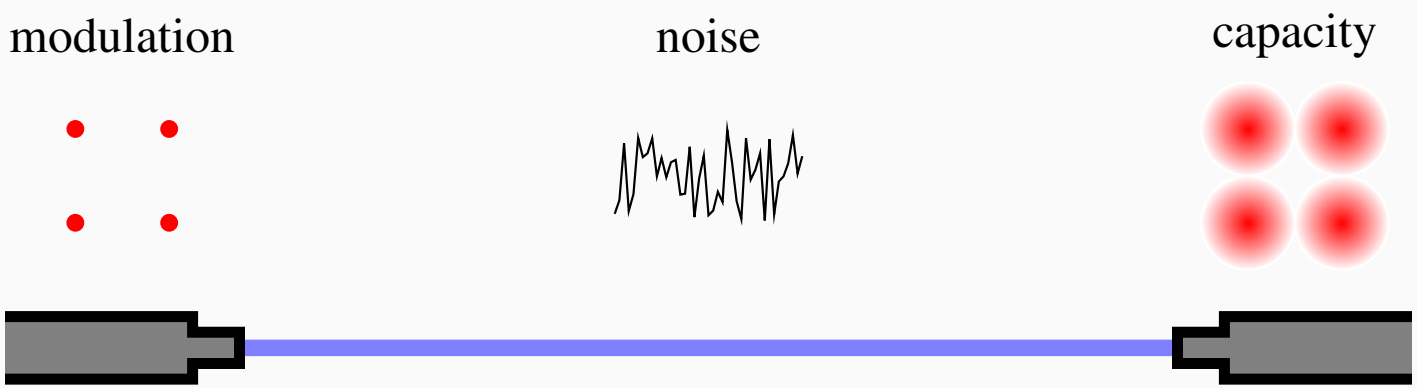
Shih-Hao Tseng, (pronounced as "She-How Zen")

October 10, 2019

Department of Computing and Mathematical Sciences,
California Institute of Technology

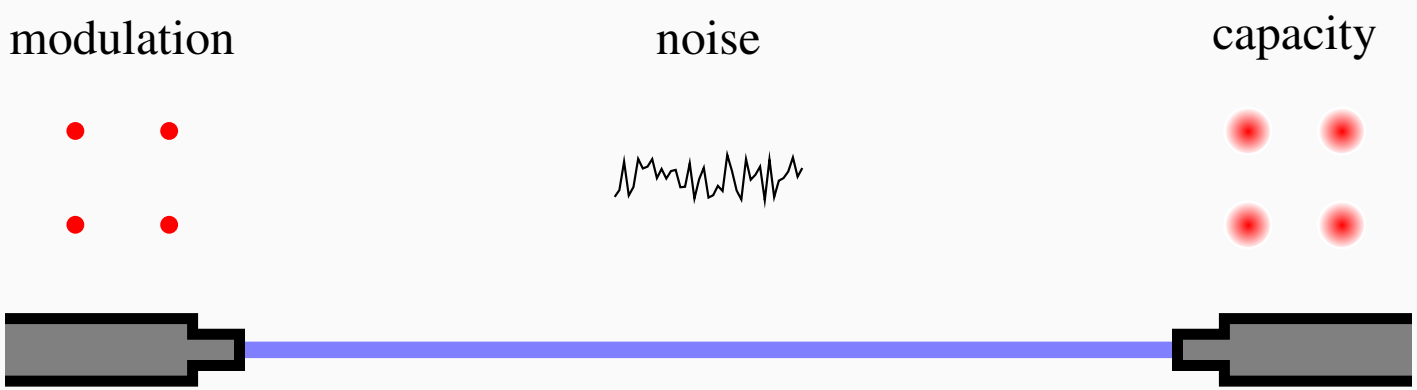
Optical Networks

- Modern wide-area networks consist of expensive optical fibers.
- The capacity of the optical fibers is determined by the signal-to-noise ratio (SNR) and the adopted modulation (such as PSK, QAM, etc.).



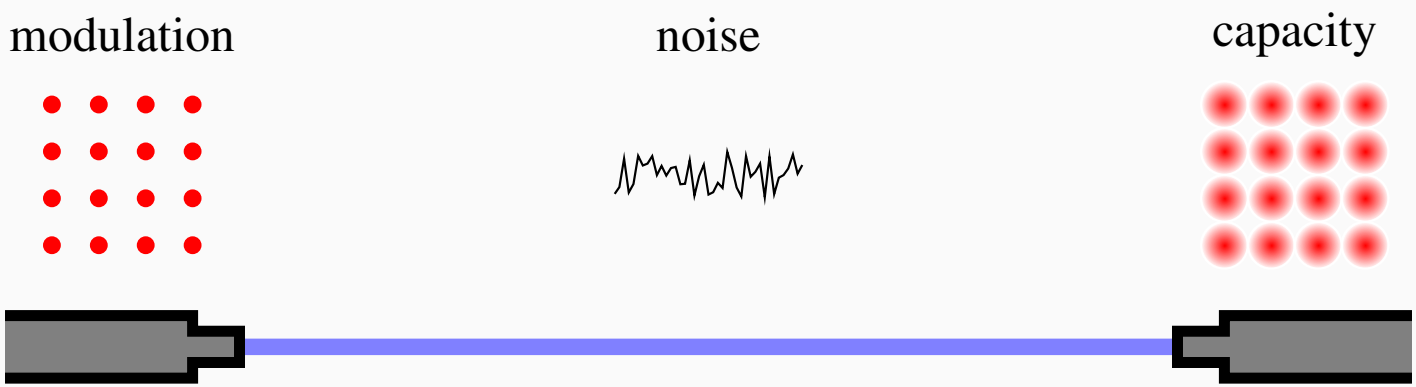
Rate-Adaptive Networks

- In practice, SNR is much better than required.
- RADWAN (Singh et al., 2018) leverages bandwidth variable transceivers (BVTs) to change the modulation and vary the capacity.



Rate-Adaptive Networks

- In practice, SNR is much better than required.
- RADWAN (Singh et al., 2018) leverages bandwidth variable transceivers (BVTs) to change the modulation and vary the capacity.



Rate-Adaptive Networks: Challenge

- Reconfiguration delay: During the change of modulation, the optical link is down for a while.

One-Shot Update and Churn

- The reconfiguration delay causes traffic disturbance, which is named *churn* in RADWAN.

One-Shot Update and Churn

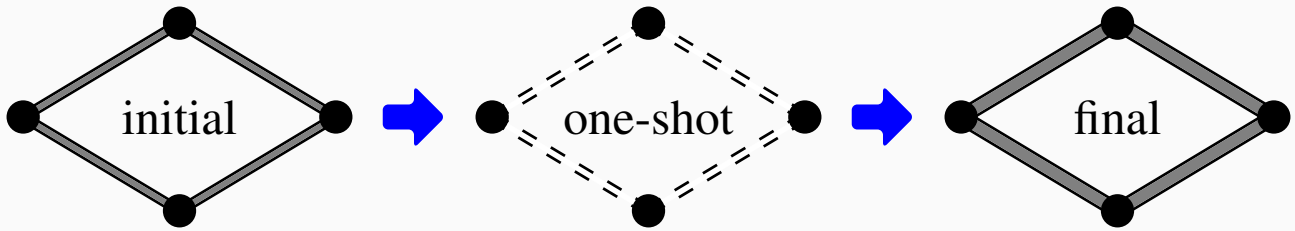
- The reconfiguration delay causes traffic disturbance, which is named *churn* in RADWAN.
- Adaptive links bring higher final throughput while causing churn. RADWAN updates the links in one-shot and addresses the trade-off by

$$\max \ \text{(final throughput)} - \epsilon \cdot \text{(churn)}$$

where ϵ is the trade-off factor.

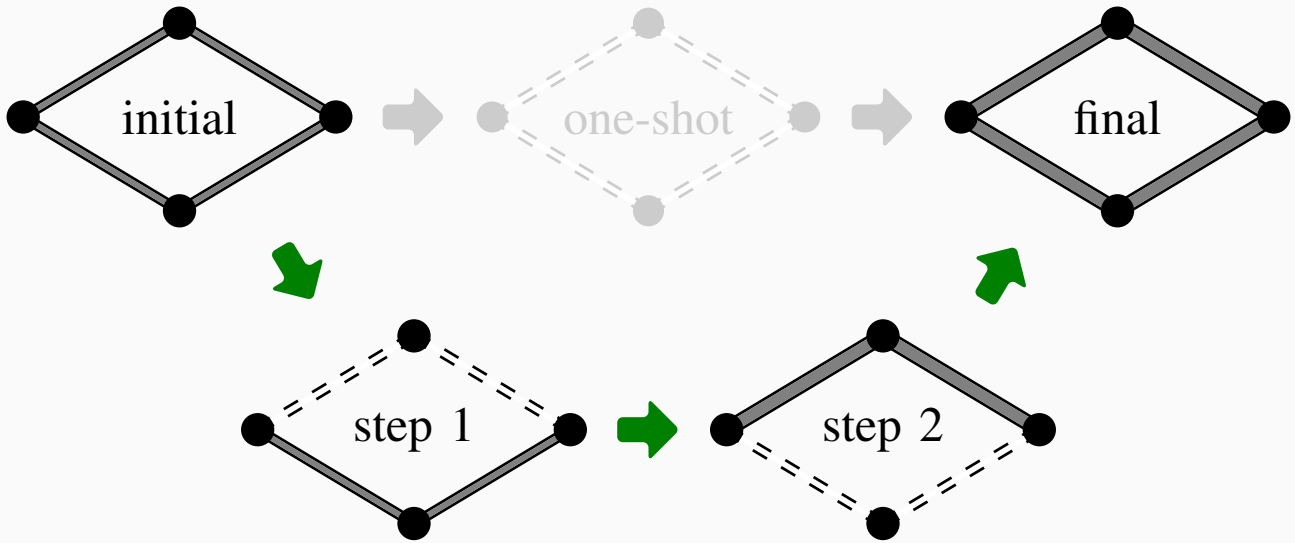
One-Shot Update

- One-shot update leads to considerable traffic fluctuation.



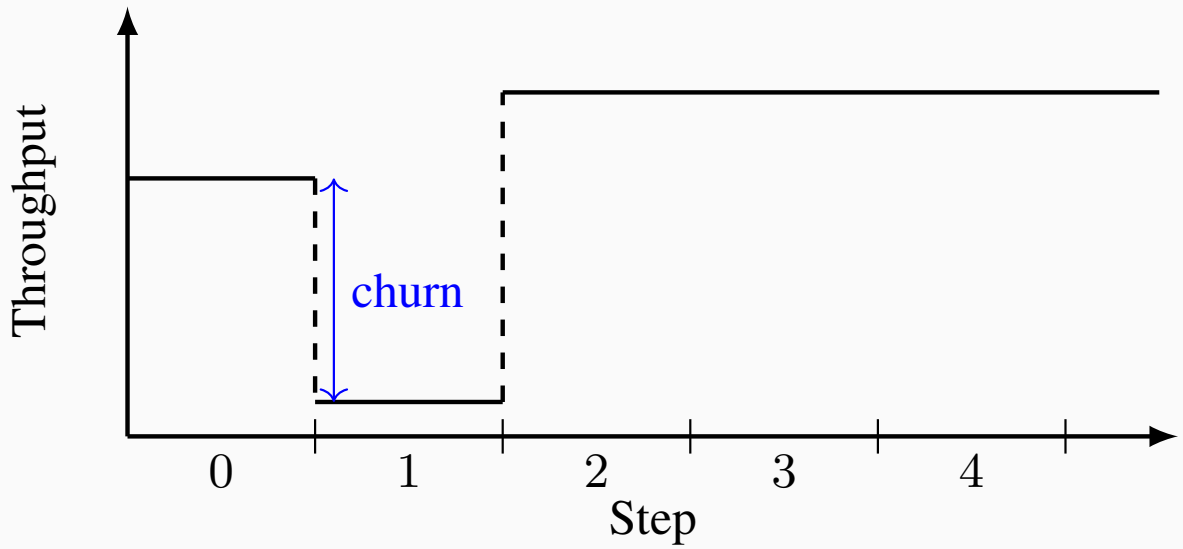
Multi-Step Reconfiguration

- One-shot update leads to considerable traffic fluctuation.
- We can update links in batches to reduce the traffic fluctuation by introducing intermediate steps similar to SWAN (Hong et al., 2013).



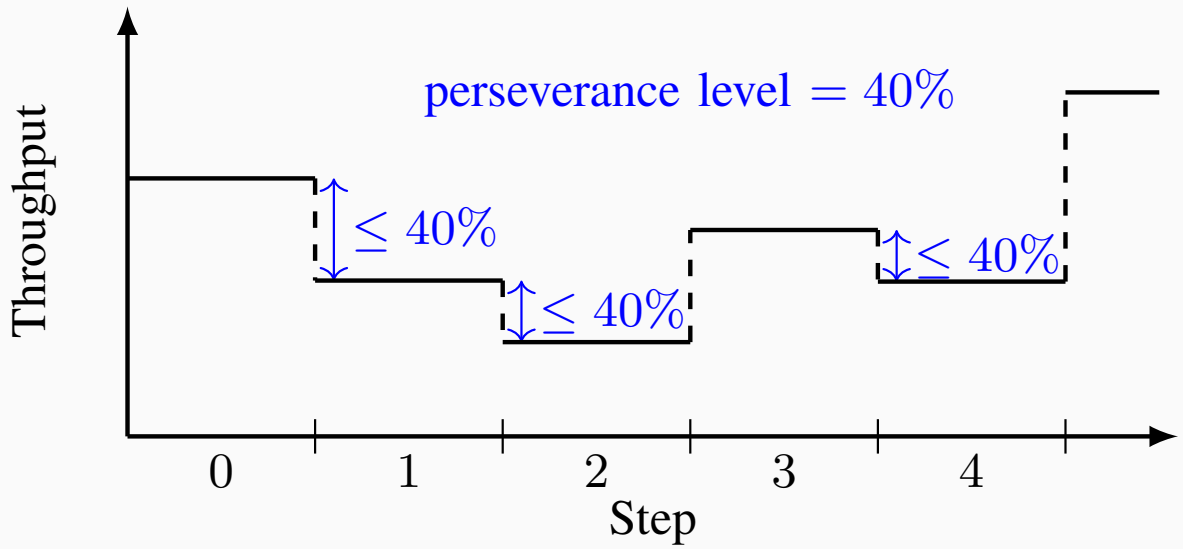
Multi-Step Reconfiguration and Perseverance

- Given a multi-step plan, we can consider not only the total impact (churn) but also the smoothness of the update.



Multi-Step Reconfiguration and Perseverance

- Given a multi-step plan, we can consider not only the total impact (churn) but also the smoothness of the update.
- We propose *perseverance* to describe the smoothness of the transition. The *perseverance level* is defined as the maximum allowed throughput drop between two consecutive steps.



Multi-Step Reconfiguration and Perseverance

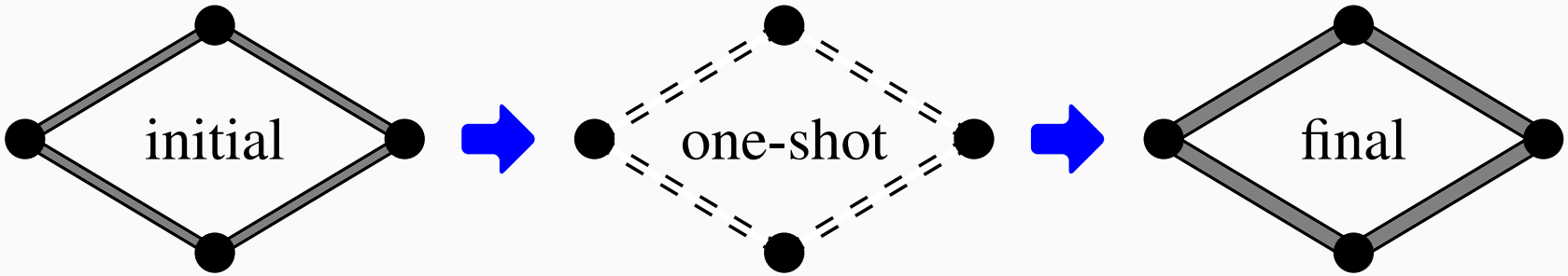
- Incorporating perseverance into consideration, we consider the optimization as follows, which is different from RADWAN's churn-based proposal:

$$\begin{aligned} \max \quad & \text{(final throughput in } T \text{ steps)} \\ \text{s.t.} \quad & \text{(perseverance level} \geq \rho) \end{aligned}$$

  where ρ is the lower bound on the perseverance level.

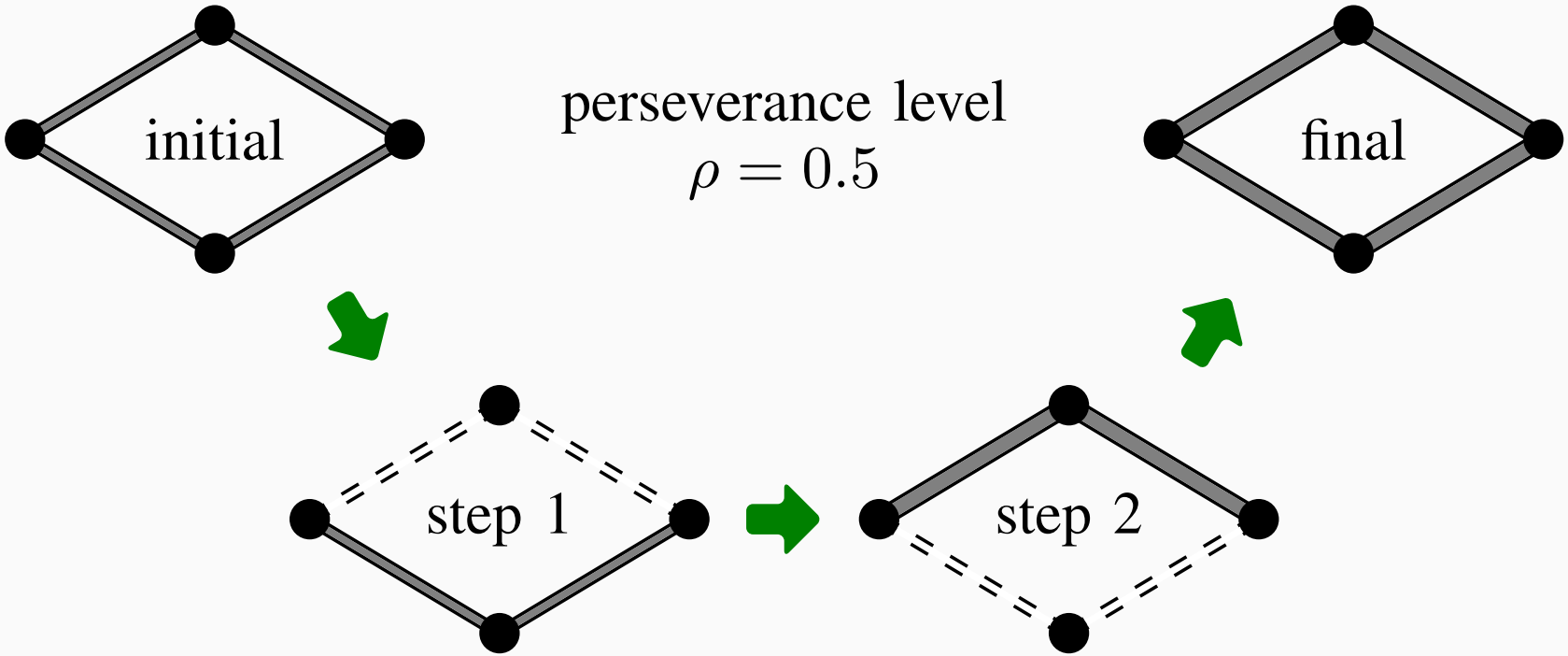
- A multi-step reconfiguration allows higher final throughput without the degradation of perseverance level.

Multi-Step Reconfiguration and Perseverance



perseverance level
$\rho = 0$

Multi-Step Reconfiguration and Perseverance



Rate Adaptation Planning (RAP) Problem

- Consider a network shared by N flows. Each sends at rate $x^n(t)$ during step t. With the horizon T, we can write down the discrete-time control formulation of the *rate adaptation planning (RAP)* problem as follows

$$\text{RAP} = \max \sum_{n \in N} x^n(T)$$

subject to capacity constraints, perseverance constraints, initial constraints, and feasibility constraints.

Rate Adaptation Planning (RAP) Problem: Constraints

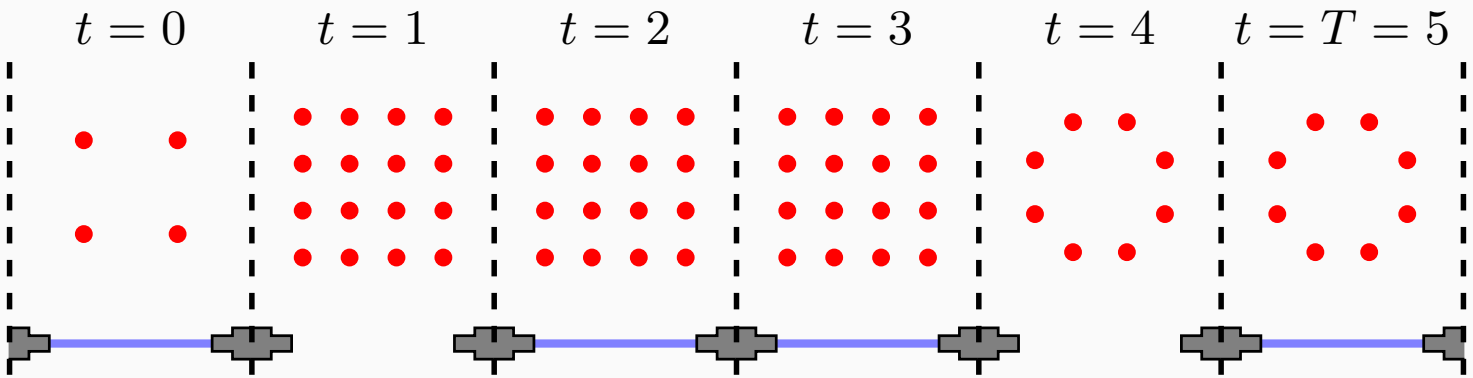
- **Perseverance constraints:** Given a perseverance level ρ, the perseverance constraint can be written as

$$\rho x^n(t-1) \leq x^n(t)$$

  for all $t = 1, 2, \ldots, T$.
- **Initial constraints:** $x^n(0)$ is given for all n. Each link has an initial capacity.
- **Feasibility constraints:** Each flow n has a predetermined path set to send its traffic. $x^n(t)$ is the sum of the traffic along all the paths.

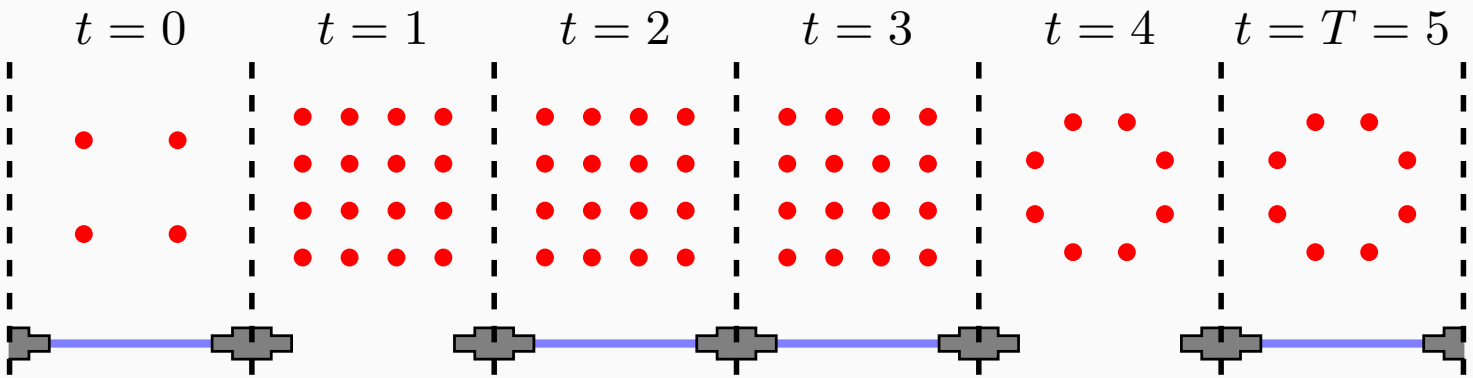
Rate Adaptation Planning (RAP) Problem: Constraints

- **Capacity constraints:** The capacity of a link c_l is determined by the adopted modulations (and the underlying SNR). Once the modulation is changed, the link is down for one step.



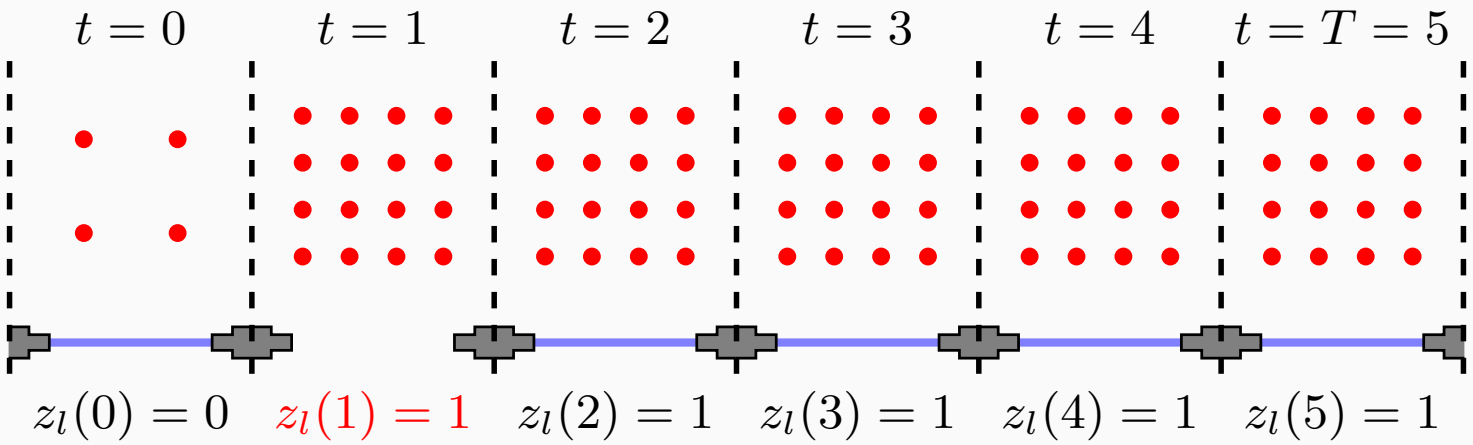
Mixed Integer Linear Programming Formulation

- Under a fixed SNR, we can show that an optimal update plan can be achieved by changing the modulation on each link l at most once – to one providing the highest capacity.



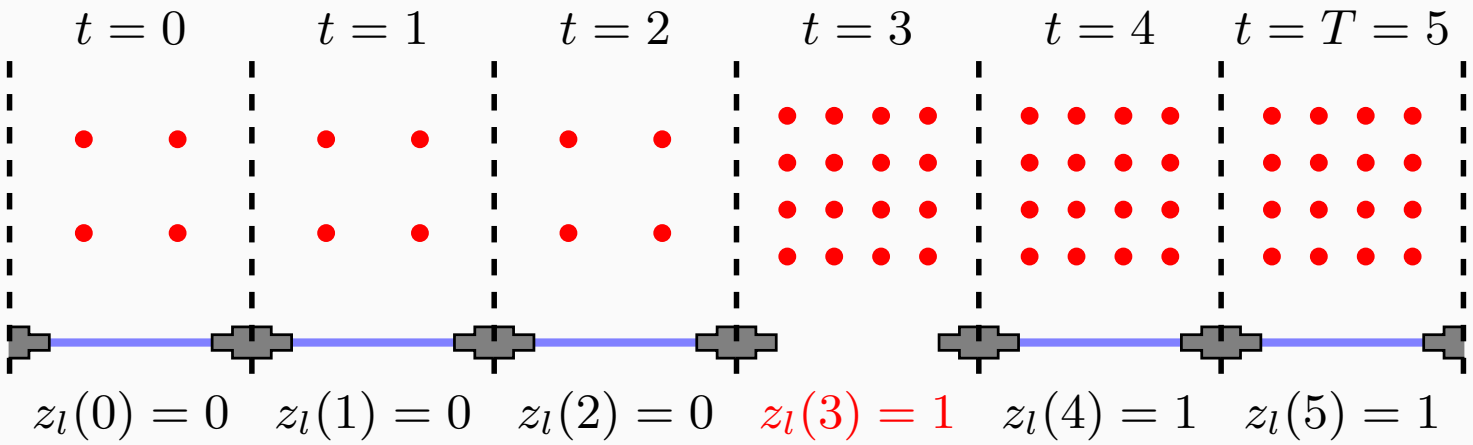
Mixed Integer Linear Programming Formulation

- Under a fixed SNR, we can show that an optimal update plan can be achieved by changing the modulation on each link l at most once – to one providing the highest capacity.
- As such, we introduce the auxiliary integer variable $z_l(t)$ for each link l to indicate whether the modulation of l has been changed at step t.



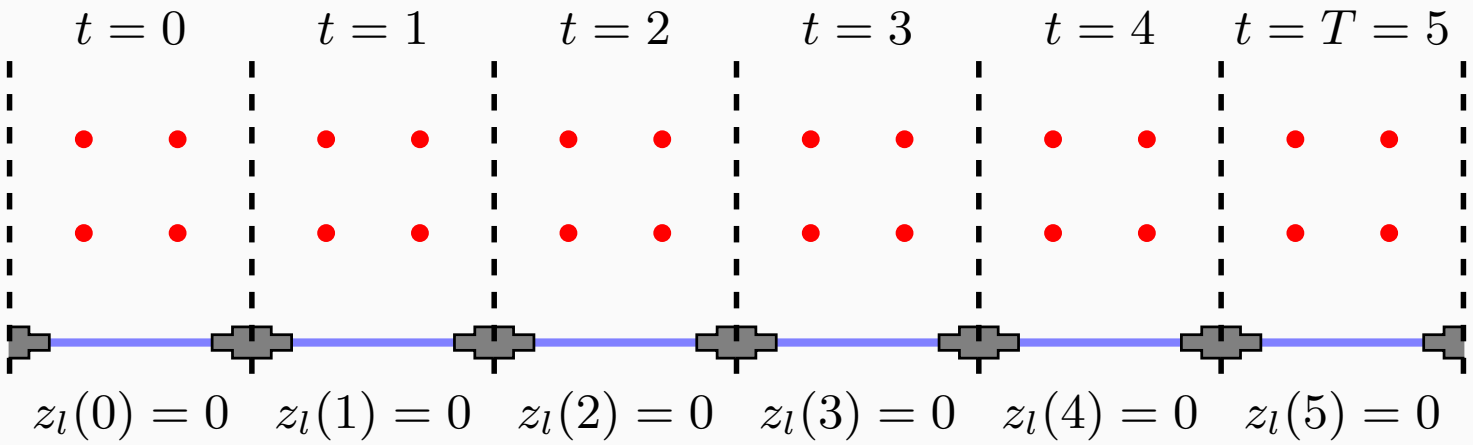
Mixed Integer Linear Programming Formulation

- Under a fixed SNR, we can show that an optimal update plan can be achieved by changing the modulation on each link l at most once – to one providing the highest capacity.
- As such, we introduce the auxiliary integer variable $z_l(t)$ for each link l to indicate whether the modulation of l has been changed at step t.



Mixed Integer Linear Programming Formulation

- Under a fixed SNR, we can show that an optimal update plan can be achieved by changing the modulation on each link l at most once – to one providing the highest capacity.
- As such, we introduce the auxiliary integer variable $z_l(t)$ for each link l to indicate whether the modulation of l has been changed at step t.

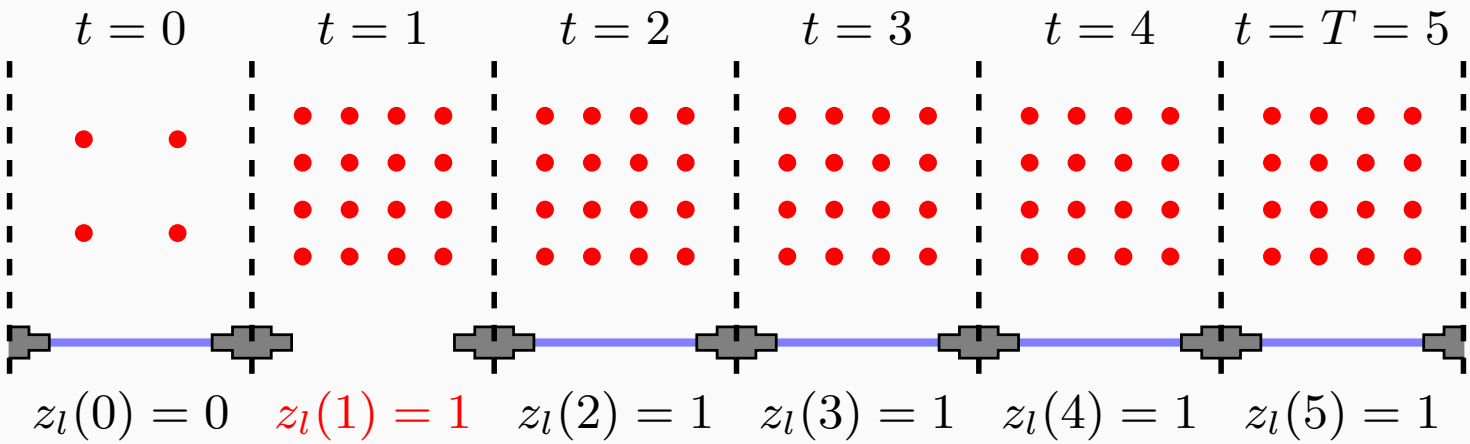


Mixed Integer Linear Programming Formulation

- **Transformed capacity constraints:** Using $z_l(t)$, we can write the capacity as

$$c_l(t) = c_l^{\min}(1 - z_l(t)) + c_l^{\max} z_l(t-1)$$

where $c_l^{\min}$ and $c_l^{\max}$ are the minimum and the maximum achievable capacity of the link under the SNR.



Mixed Integer Linear Programming Formulation

$$\max \quad \sum_{n \in N} x^n(T) \qquad \text{(RAP)}$$

$$\text{s.t.} \quad \text{capacity constraints}$$

$$\text{perseverance constraints}$$

$$\text{initial constraints}$$

$$\text{feasibility constraints}$$

$$z_l(t-1) \leq z_l(t) \qquad \forall t \in T, l \in L$$

$$z_l(t) \in \{0,1\} \qquad \forall t \in T, l \in L$$

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.
- Can we approximate RAP within a constant factor?

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.
- Can we approximate RAP within a constant factor?
  $\rightarrow$ No, unless P=NP.

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.
- Can we approximate RAP within a constant factor?
  $\rightarrow$ No, unless P=NP.
- Why is RAP so hard?

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.
- Can we approximate RAP within a constant factor?
  $\rightarrow$ No, unless P=NP.
- Why is RAP so hard?
  $\rightarrow$ Under some mild assumptions, we can always reach the optimal final throughput, despite that the update sequence may be extremely long.

Analysis of Rate Adaptation Planning (RAP) Problem

- Can we solve RAP in polynomial time?
  $\rightarrow$ Unlikely, RAP is NP-hard.
- Can we approximate RAP within a constant factor?
  $\rightarrow$ No, unless P=NP.
- Why is RAP so hard?
  $\rightarrow$ Under some mild assumptions, we can always reach the optimal final throughput, despite that the update sequence may be extremely long.
  $\rightarrow$ We would prefer to finish the update in a bounded number of steps. Therefore, we need some good heuristics for RAP.

Algorithm Design Ideas

- Find a feasible reconfiguration plan.
- Fix the *configuration* (i.e., when we should change the modulations of which links) and maximize the usage of available links.

Algorithm Design Ideas

- Find a feasible reconfiguration plan.
- Fix the *configuration* (i.e., when we should change the modulations of which links) and maximize the usage of available links.

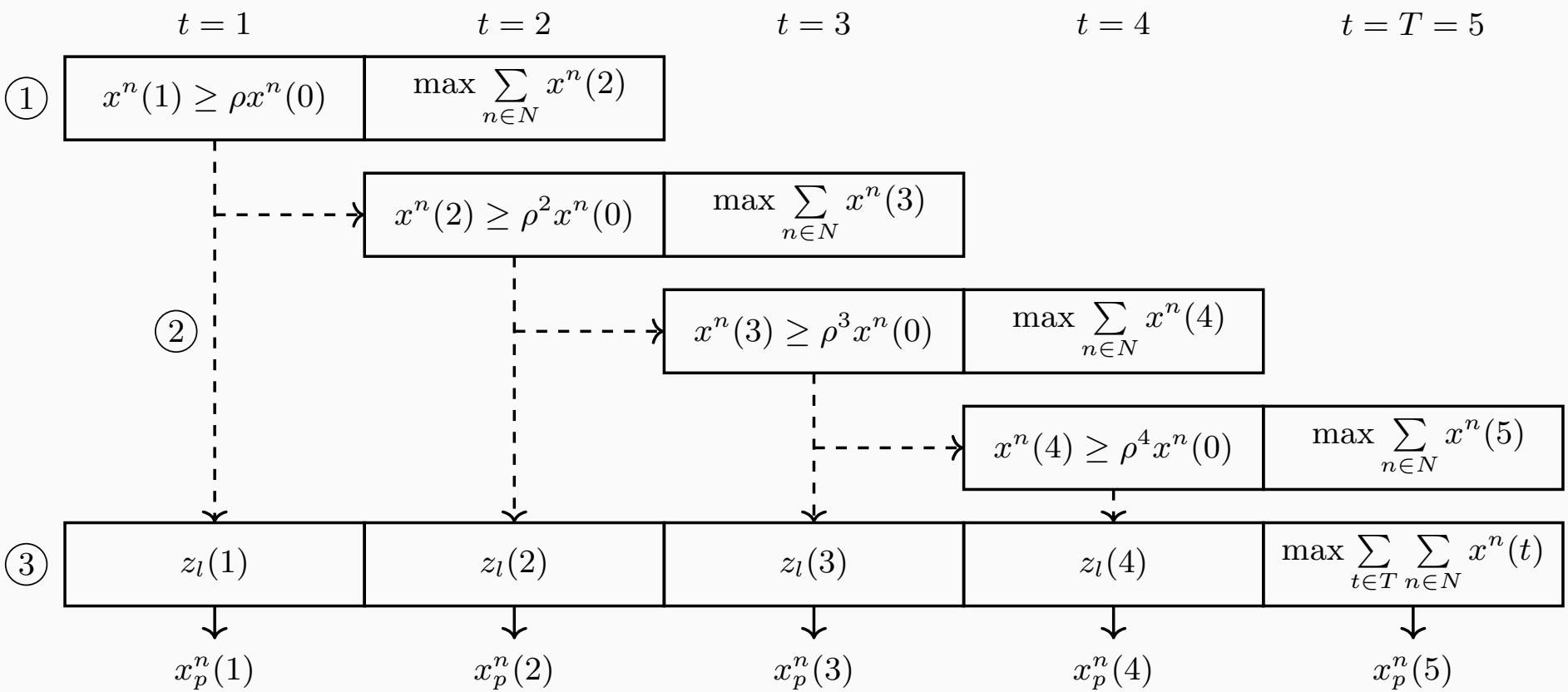
In sum, we design our algorithm (ALG) to

① solve 2-step LP relaxation at time t for relaxed $z_l(t) \in (0, 1)$;

② upround $z_l(t)$ to integers to form the configuration at t;

③ iterate through $t = 1, \ldots, T-1$ to obtain the configurations and find the *work conserving* reconfiguration plan.

Proposed Algorithm (ALG)



Simulations: Questions of Interest

- Is it still beneficial to have rate-adaptive links under the reconfiguration delay and perseverance constraints?
- What is a reasonable perseverance level?
- How well does perseverance smoothen the process?
- How hard is it to find a perseverance-aware solution?

Simulation Setup

- We simulate RADWAN, the optimal solution to RAP (OPT), and the proposed algorithm (ALG) based on the the existing WAN topologies: SWAN, Internet2, and B4.
- The baseline case: $T = 5$ and $\rho = 0.5$.

(a) SWAN (8 nodes, 12 links)

(b) B4 (18 nodes, 39 links)

Advantage of Rate Adaptive Links

- Is it still beneficial to have rate-adaptive links under the reconfiguration delay and perseverance constraints?
- What is a reasonable perseverance level?
- How well does perseverance smoothen the process?
- How hard is it to find a perseverance-aware solution?

Advantage of Rate Adaptive Links

Table 1: Average throughput (Gbps) under different WANs. Our methods OPT and ALG boost the throughput by 40% to 50% while ensuring a more steady reconfiguration plan.

topology	$\rho \approx 1$	$\rho = 0.5$		$\rho \approx 0$
	w/o adaptive links	OPT	ALG	RADWAN†
SWAN	681.85	998.623	998.611	998.625
Internet2	1071.15	1510.13	1509.89	1510.15
B4	2621.12	3919.81	3919.14	3919.87

†We also simulate RADWAN with $\epsilon = 0.001$, and the resulting average throughput remains the same as $\epsilon = 0.1$.

Convergence under Different Perseverance Levels

- Is it still beneficial to have rate-adaptive links under the reconfiguration delay and perseverance constraints?
- What is a reasonable perseverance level?
  $\rightarrow$ How does a perseverance level slow down the convergence to the final throughput?
- How well does perseverance smoothen the process?
- How hard is it to find a perseverance-aware solution?

Convergence under Different Perseverance Levels

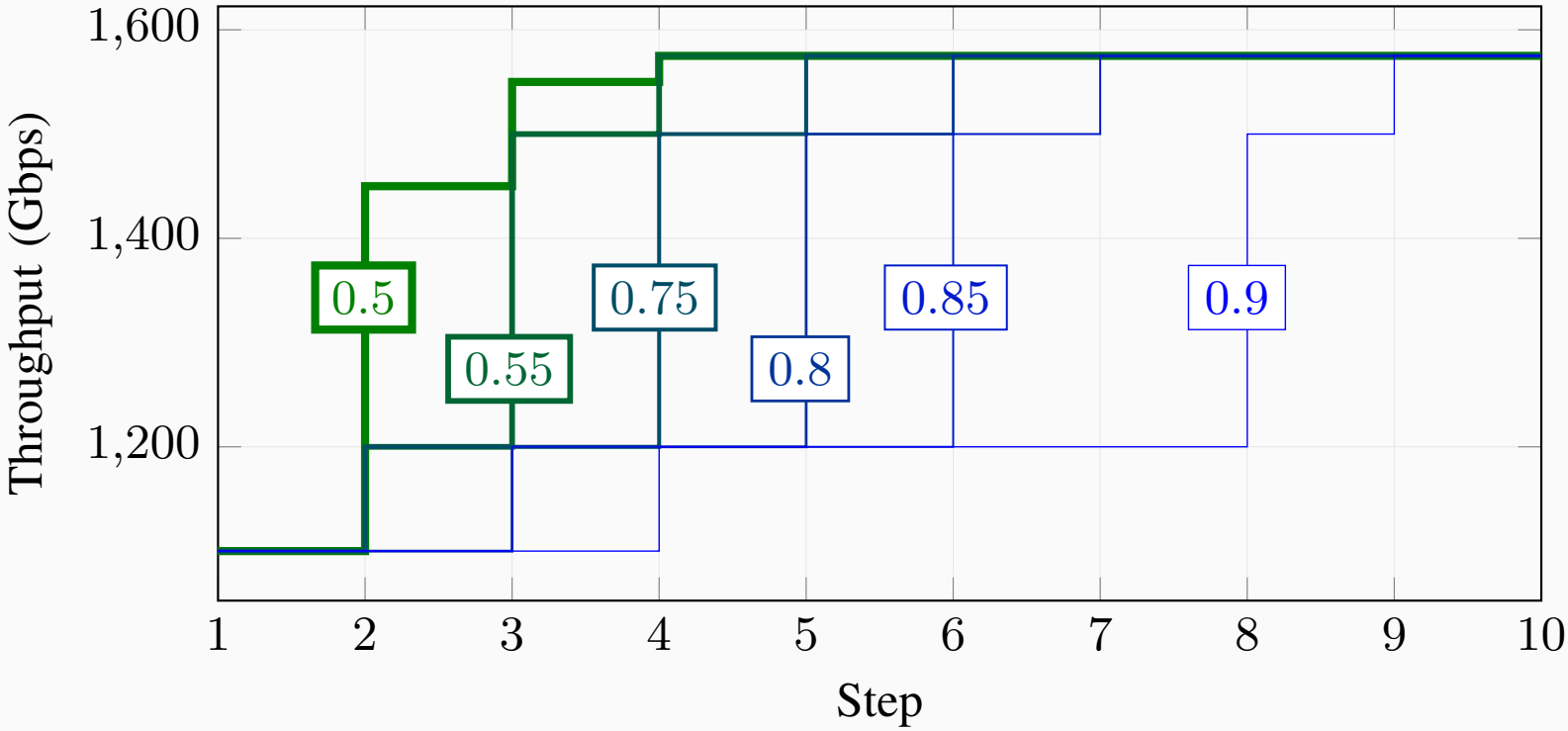


Figure 2: Larger perseverance levels ρ (boxed values) prevent aggressive update with large disturbance, and hence, it takes more steps for ALG to converge to the maximum throughput.

Convergence under Different Perseverance Levels

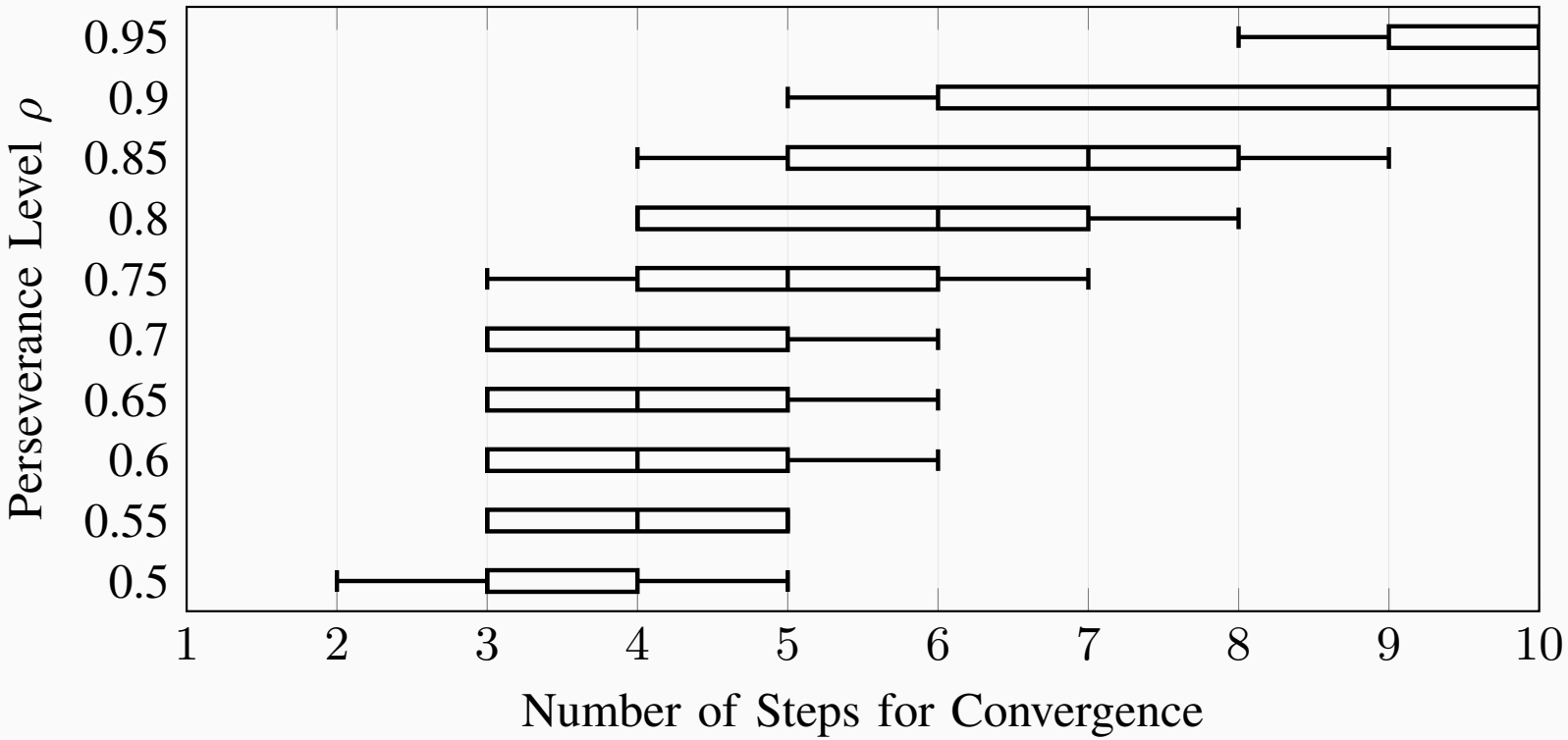
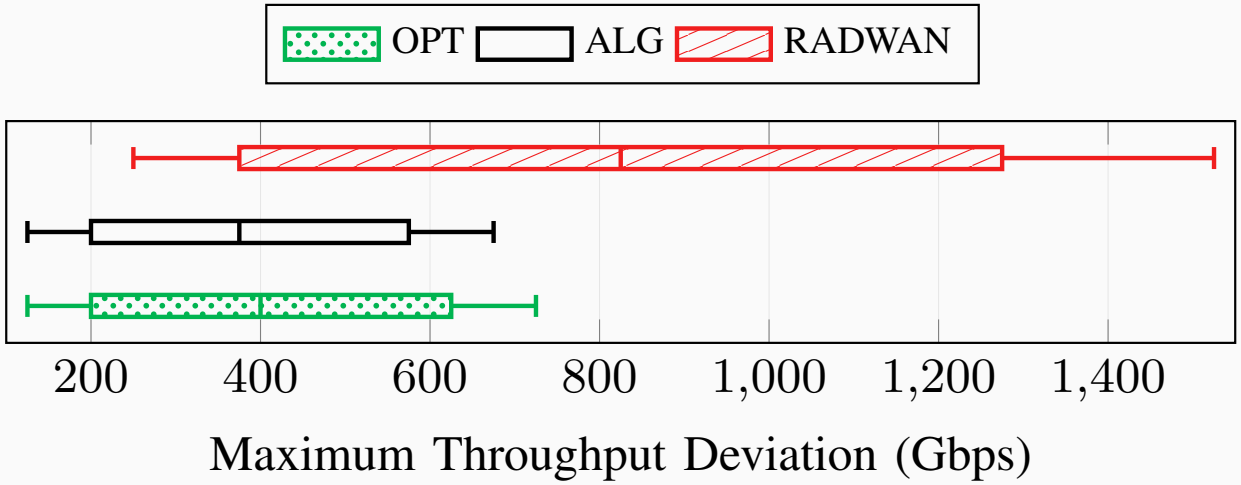


Figure 3: The 1^{st}-5^{th}-50^{th}-95^{th}-99^{th} percentiles of the minimum number of steps needed for throughput convergence. When $\rho = 0.5$, ALG converges in 5 steps in 99% of the 1000 random cases.

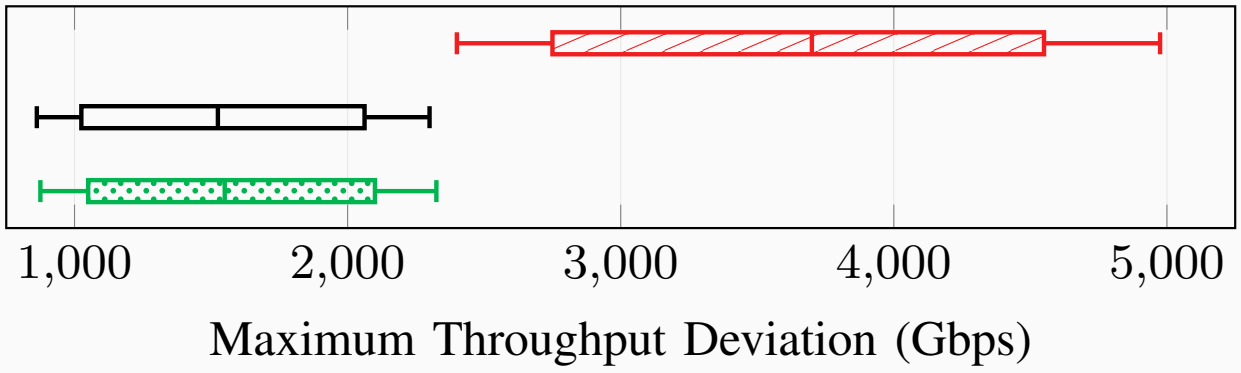
Mitigation of Transition Fluctuation

- Is it still beneficial to have rate-adaptive links under the reconfiguration delay and perseverance constraints?
- What is a reasonable perseverance level?
  $\rightarrow$ How does a perseverance level slow down the convergence to the final throughput?
- How well does perseverance smoothen the process?
- How hard is it to find a perseverance-aware solution?

Mitigation of Transition Fluctuation



(a) SWAN (8 nodes, 12 links)



(b) B4 (18 nodes, 39 links)

Comparison of Computation Overhead

- Is it still beneficial to have rate-adaptive links under the reconfiguration delay and perseverance constraints?
- What is a reasonable perseverance level?
  $\rightarrow$ How does a perseverance level slow down the convergence to the final throughput?
- How well does perseverance smoothen the process?
- How hard is it to find a perseverance-aware solution?
  $\rightarrow$ What are the computation overheads?

Comparison of Computation Overhead

Table 2: Average CPU computation time (ms) and fraction. ALG uses much less time and scales better than OPT (current reconfiguration downtime is 68 s, which can be potentially reduced to 35 ms).

topology	OPT	ALG	fraction $\left(\frac{\mathrm{ALG}}{\mathrm{OPT}}\right)$
SWAN	67.7	15.0	22.2%
Internet2	497.1	39.6	8.0%
B4	1.2×10^6	332.0	0.03%

Conclusion

- Instead of one-shot update and churn, we introduce the idea perseverance for a multi-step reconfiguration.
- We propose an efficient algorithm (ALG) to approach RAP. The proposed algorithm improves the overall throughput and smoothens the transition with small computation overhead.
- Besides perseverance, the network operators might maintain some other properties (such as throughput level) during a multi-step reconfiguration. It is possible to extend the proposed multi-step update framework and examine some other transition metrics.

Questions & Answers

References

- R. Singh, M. Ghobadi, K.-T. Foerster, M. Filer, and P. Gill, "RADWAN: Rate adaptive wide area network," in *Proc. ACM SIGCOMM*. ACM, 2018, pp. 547–560.
- C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 15–26, 2013.