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

Shih-Hao Tseng
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, California 91125, U.S.A.
shtseng@caltech.edu

***Abstract*—Expensive optical fibers provide connectivity for wide-area networks. Nowadays, the fibers are operated in a much conservative manner. By adaptively reconfiguring the fibers to exploit its signal quality, a recent proposal demonstrates a significant increase of optical link capacity. Such a reconfiguration currently accompanies a non-ignorable delay, during which the reconfigured link is not accessible, and the mentioned approach trades off the final throughput with the induced churn during the transition. This scheme can result in high traffic disturbance during the reconfiguration.**

To overcome the drawback of the simple churn-based update, we study the rate adaptation planning (RAP) problem under reconfiguration delay. We propose a multiple step planning with perseverance constraints. This approach leads to a smoother transition, but the optimal plan is shown NP-hard without constant factor approximation (unless P=NP). Therefore, we develop an efficient LP-based heuristic algorithm. Extensive simulations show that the algorithm gives $40-50\%$ higher throughput than the no-adaptive-link case. Also, the transition is much smoother: the resulting traffic fluctuation is only $40-50\%$ of the existing churn-based approach.

I. Introduction

The core of modern wide-area networks consists of expensive optical fibers [1]–[3]. The significant cost of those fibers incentivizes the network operators to utilize them as efficiently as possible. For example, Owan [1] reconfigures optical WAN topology to improve bulk traffic transfers and [4] presents an online scheduling algorithm to minimize the aggregated flow completion time. These methods optimize network performance by manipulating fixed capacity fibers.

Besides providing a fixed capacity, the optical fibers can transmit beyond its capacity – if the signal quality allows. Recently, the optical links support transmission at 100 Gbps [3]. However, with an appropriate modulation and a suitable signal-to-noise ratio (SNR), they can potentially send at a much higher rate (up to tens of Tbps) [5]. A recent study [3] confirms that the measured SNR of the optical links exceeds the requirement for 100 Gbps by a large extent. Therefore, [3] proposes RADWAN, which reconfigures the modulations of the optical links to exploit the SNR margin and augment the link capacity. By doing so, RADWAN boosts the overall capacity by more than 100 Tbps.

The key technology that enables the modulation reconfiguration is the bandwidth variable transceivers (BVTs) [6]. State-of-the-art BVTs can switch among multiple modulation formats, but the reconfiguration takes a non-ignorable delay during which the BVTs cannot send traffic through the fibers. Therefore, RADWAN considers *churn*, which is the amount of affected traffic during the reconfiguration delay. RADWAN compensates the final throughput with the induced churn. In other words, RADWAN tries to minimize the traffic disturbance during the transition when optimizing the final performance.

A drawback of such a churn-based update is fluctuation. RADWAN performs all link reconfiguration at the same time and reroutes the affected traffic. If the network is heavily loaded, the success of the reroute may not be guaranteed. In that case, we lose the churn, and the traffic suffers throughput drop. To avoid such a disturbance, the system operator could try to shorten the reconfiguration delay or spare some bandwidth for transient reroute as suggested in [7]. Reducing the reconfiguration delay can be challenging when multiple optical links are involved as remarked in [1]. On the other hand, reserving optical bandwidth for transition hurts the performance of the optical core as the system cannot fully utilize its available link capacity.

A. Contribution and Organization

To deal with traffic fluctuation, we propose to include some intermediate steps as suggested by SWAN [7], which leverages the transitional stages to mitigate transient congestion. Also, we introduce the idea of *perseverance*. Perseverance limits the relative traffic variation between two consecutive steps. By controlling the perseverance level, we can balance the minimum traffic ratio throughout the transition and the speed of converging to the best throughput.

We formulate the multi-step perseverance-aware reconfiguration as the rate adaptation planning (RAP) problem. By depicting a translation to the mixed integer linear programming (MILP) form, we reveal a computation-demanding way to obtain the optimal solution. Although a polynomial-time algorithm would be preferable, we negate that pursuit by proving that the problem is NP-hard and there exists no constant factor approximation algorithm unless P=NP. As such, we develop linear programming (LP) based heuristic to approach RAP. Through simulations, we show that the heuristic enjoys the

978-1-7281-2700-2/19/$31.00 2019 © IEEE

benefit of adaptive links without suffering the high fluctuation as the churn-based approach.

We organize the paper as follows. In Section II, we first describe the rate-adaptive networks and the existing update metric – churn. Since churn-based update suffers traffic fluctuation, we then introduce the idea of perseverance. Section III introduces the notation to formulate our rate adaptation planning (RAP) as a discrete-time control problem. We then demonstrate how to solve RAP by considering its equivalent MILP formulation. In Section IV, we show the NP-hardness of RAP, which motivates us to develop an LP-based heuristic to approach RAP in Section V, and it is evaluated along with the optimal solution and RADWAN through extensive simulations in Section VI. Finally, we conclude the paper in Section VII.

II. Background and Motivation

We start the section with the basics and the difficulty of operating a rate-adaptive network, followed by the existing method, RADWAN [3], and its proposed metric churn. We then identify the drawbacks of a churn-based method and illustrate how a perseverance-aware reconfiguration plan could lead to a smoother update.

A. Rate-Adaptive Networks

Modern core networks consist of optical fiber links with fixed capacities. The fixed capacity is essentially a lower bound on the real capacity that the optical link can support. The real capacity is determined by the applied modulation and the experienced SNR: A modulation leading to higher capacity requires higher SNR. Although the SNR may vary, the network operator chooses the modulation based on a conservative estimation of SNR so that a fixed capacity is guaranteed.

In practice, the measured SNR is much higher than the requirement [3]. Therefore, the network operator could potentially operate the links with more aggressive modulations to commit to a larger fixed link capacity. Also, when the SNR drops, the network operator could prevent a total blackout of the link by switching to a low-rate modulation which can sustain under lower SNR. This idea leads to the design of RADWAN [3], a *rate-adaptive network*.

The main obstacle of operating such a rate-adaptive network is the *reconfiguration delay* of the links. During the reconfiguration, a link cannot carry any traffic until a new modulation is negotiated at both sides of the link. Without reconfiguration delay, a link can enjoy boosted capacity right after the update, while a non-zero reconfiguration delay prevents the traffic utilizing the link. As a result, the network operator needs to mitigate the impact of an update on the throughput.

Given a procedure to update the links in a rate-adaptive network, we have two different metrics that can evaluate its impact on the throughput: churn and perseverance. Churn quantifies the total affected amount and perseverance concerns the variation throughout the procedure. In the following subsections, we illustrate the differences between the two metrics.

B. One-Shot Update and Churn

A natural way to evaluate the reconfiguration impact is to consider the total traffic on the reconfigured links, which is named *churn* in RADWAN. This quantity reflects the affected traffic that needs to be handled (or dropped) during a one-shot update. Taking into account the churn, RADWAN aims to maximize the final throughput after adjusting the link capacities while minimizing the induced churn during the update, which is written as

$$\max \ \text{(final throughput)} - \epsilon \cdot \text{(churn)} \tag{1}$$

where ϵ is the trade-off factor. The network operator can alter ϵ to prefer higher final throughput (small ϵ) or lower churn (larger ϵ).

The basic idea behind such a churn-based approach is to mitigate the impact caused by the one-shot update. However, the one-shot update leads to considerable traffic fluctuation during the reconfiguration. Also, this approach tends to allow large churn when a large final throughput is achievable. We address these issues in the following subsection.

C. Multi-Step Reconfiguration and Perseverance

Instead of updating all the links in one shot, we can update them in batches to reduce the traffic fluctuation. To do so, we introduce some intermediate steps to form a multi-step reconfiguration plan as in SWAN [7], which utilizes additional steps to prevent transient congestion.

Given a multi-step plan, we can consider not only the total impact (as churn describes) but also the smoothness of the update. To achieve a smoother transition, we introduce the idea *perseverance* to regulate the transient throughput drop. In particular, we define *perseverance level* to be the maximum allowed throughput drop between two consecutive steps. By maintaining a perseverance level throughout the update, we rule out the sudden blackout of traffic that might happen in a churn-based solution.

We use Fig. 1 to illustrate the difference between churn and perseverance. Two different procedures are applied to the flows in a network, resulting in different discrete throughput processes. Both processes start with the same initial throughput and end at the same final state. The main difference is that the above churn-based process shifts to the final state swiftly with more considerable throughput fluctuation. The below procedure, however, controls the traffic variation rate throughout the transition. More specifically, it maintains a perseverance level of 0.6 that allows throughput to drop no more than 40% for any two consecutive steps.

To incorporate perseverance into consideration, we consider an optimization problem as follows, which is different from RADWAN's churn-based proposal (1):

$$\begin{aligned} \max \ & \text{(final throughput in } T \text{ steps)} \\ \text{s.t.} \ & \text{(perseverance level} \geq \rho) \end{aligned} \tag{2}$$

One advantage of such a perseverance-aware optimization is that a higher achievable final throughput does not cause the

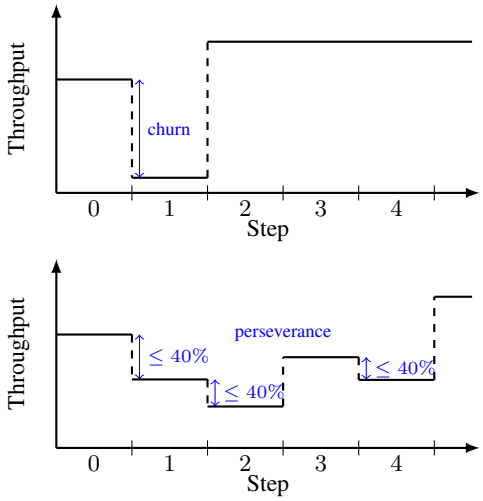


Fig. 1. The difference between churn and perseverance. Churn measures the total impact, while perseverance regulates to the relative variation between steps (the maximum allowed throughput drop at the next step). The perseverance-aware traffic engineering schedules the reconfiguration of the links so that the traffic experiences smaller fluctuation during the update comparing to the churn-based one-shot update.

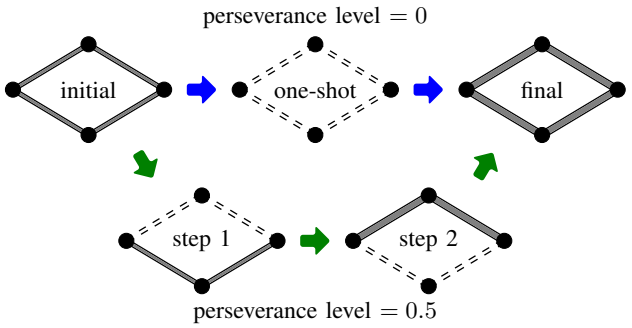


Fig. 2. We can include intermediate steps and maintain a perseverance level to mitigate the traffic disturbance during the reconfiguration. The above one-shot update augments all the link capacities from 1 to 2 at once. As a result, all traffic (gray) on the corresponding links is interrupted. Instead, by adding two intermediate steps, the below plan could manifest a perseverance level 0.5 throughout the update, which disturbs less traffic (at most half of the throughput).

degradation of the perseverance level, while in (1), a higher achievable final throughput could tolerate a larger churn under a fixed ϵ.

We demonstrate this approach by an example shown in Fig. 2, in which each link has capacity 1 and can be augmented to 2. The augmentation takes 1 unit reconfiguration time. A flow sends 2 unit traffic (marked gray) initially from left to right. If we update the links according to RADWAN (1), we will prefer a one-shot update as the above one in Fig. 2 when the trade-off factor $\epsilon \in [0, 1]$. However, if we consider a multi-step solution using (2) with $T = 3$ and $\rho = 0.5$, we will find some solution as the below one in Fig. 2. This plan prevents transient network blackout and induces less traffic disturbance.

Fig. 2 shows that a perseverance-aware reconfiguration plan can lead to a much smoother transition than a simple churn-based solution. As a result, we investigate in the following context how to update under perseverance constraints.

III. Formulation

In this section, we first formulate the perseverance-aware update problem, and then we develop methods to derive the update plan later. We start the section with our notation and model. The rate adaptation planning (RAP) problem is then formulated in Section III-C as a discrete-time control problem. In Section III-D, we show how to translate the control problem into a mixed integer linear program (MILP), which can be solved by some MILP solvers.

A. Notation

We denote by $\mathbb{Z}$ the set of integers and $\mathbb{Z}_+$ the positive integers. $[a, b]$ is an interval between a and b, while $\{a, b\}$ only consists of a and b. $[a, b]_{\mathbb{Z}}$ refers to the set of integers within $[a, b]$. For brevity, we abbreviate $i \in [1, I]_{\mathbb{Z}}$ as $i \in I$ for some integer I. Given a set S, we denote by $|S|$ the cardinality of S.

B. Model

We model the network by a graph $G = (V, L)$ where V is the set of nodes and L is the set of directed links. Each directed link $l \in L$ offers an alterable capacity $c_l \geq 0$ based on its current modulation mode $m_l \in M_l$, where M_l is the set of all available modulation modes for the link. To switch to a different modulation mode, the link undergoes a reconfiguration phase when it is down ($c_l = 0$). Every modulation mode $m \in M_l$ requires a minimum signal-noise ratio (SNR) $\text{SNR}_m^{\min}$ to provide the corresponding capacity $c_l[m_l] \geq 0$. The larger the capacity, the higher the minimum SNR is required. If current SNR of the link (SNR_l) is smaller than the modulation requirement, the link is down.

During the reconfiguration, N flows, which are defined by their source and destination, share the network. Each flow $n \in N$ has a predetermined path set P^n to send from its source to its destination. For each path $p \in P^n$, we denote by x_p^n the sending rate on the path. Through the paths, flow n sends at an aggregate rate

$$x^n = \sum_{p \in P^n} x_p^n. \tag{3}$$

We say $l \in p$ if the path p goes through the link l, and we denote the aggregate traffic on link l by

$$y_l = \sum_{n \in N} \sum_{l \in p : p \in P^n} x_p^n. \tag{4}$$

The capacity of a path is defined as its bottleneck link capacity.

C. Discrete-Time Control Formulation

Assuming each reconfiguration takes one unit time, we formulate the rate adaptation problem as a discrete-time control problem with a finite horizon T, which is determined by the network operator according to its computation and deployment ability. We associate a parenthesized time (step) $t \in T$ after each variable to refer to its value at time (step) t. Also, we

denote by $t = 0$ the initial state. For example, $x^n(T)$ is the aggregate sending rate of flow n at time T, and $y_l(t)$ is the aggregate traffic on link l at time t. The same rule applies to $x_p^n(t)$, $c_l(t)$, and $m_l(t)$. Meanwhile, the aggregate rules (3) and (4) are maintained for each time. Given that reconfiguration time is relatively short, we assume that the N flows do not change before the reconfiguration finishes.

The goal of the rate adaptation planning (RAP) problem is to find the sending rate $x_p^n(t)$ and the modulations $m_l(t)$ to maximize the overall throughput at the end of the horizon. We can write RAP as a discrete-time control problem with the objective

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

subject to the capacity constraints (5), perseverance constraints (6), initial constraints (the parameter values at $t = 0$), and some other feasibility constraints (such as $x_p^n(t) \geq 0$ and $m_l(t) \in M_l$).

The capacity constraints are as follows

$$y_l(t) \leq c_l(t) \tag{5}$$

for all $l \in L$ and $t \in T$, where

$$c_l(t) = \begin{cases} c_l[m_l(t)], & \text{if } \begin{array}{l} \text{SNR}^{\min}_{m_l(t)} \leq \text{SNR}_l, \text{ and} \\ m_l(t-1) = m_l(t); \end{array} \\ 0, & \text{otherwise.} \end{cases}$$

Here, $c_l(t)$ is non-zero only when the chosen SNR is feasible and the links l is not currently under reconfiguration (changing the modulation m_l).

The perseverance constraints regulate how much traffic we should preserve during the reconfiguration. Given a perseverance level $\rho \in [0, 1]$, the perseverance constraints require

$$\rho x^n(t-1) \leq x^n(t) \tag{6}$$

for all $n \in N$ and $t \in T$. Without the perseverance constraints (or equivalently when $\rho = 0$), the naive solution to RAP would be simply reconfiguring all the links at once to their best possible capacity according to the current SNR.

It is not trivial to see how we can approach RAP. We demonstrate in the following subsection that RAP can also be formulated as a mixed integer linear program (MILP) which allows further investigation.

D. Mixed Integer Linear Programming Formulation

RAP in the discrete-time control form is complicated, especially when we can potentially modify the modulation of the links multiple times before T. The following optimality property allows us to consider only the reconfiguration plans which change the modulation of each link at most once. As such, we only need to decide for each link when to reconfigure. This leads to a much simpler MILP formulation.

Lemma 1. *If there exists an optimal solution of RAP, there must exist another one such that each link either keeps its modulation at all times or is reconfigured at most once to the highest possible capacity.*

Proof. We prove the lemma by construction. Let $\left\{x_p^n(t), m_l(t)\right\}$ (for all $n \in N, p \in P^n, l \in L, t \in T$, omitted hereafter for brevity) be an optimal solution, we construct the other solution $\left\{\hat{x}_p^n(t), \hat{m}_l(t)\right\}$ as follows: The sending rate stays the same ($x_p^n(t) = \hat{x}_p^n(t)$). For each link l that is first reconfigured at time $t' \in T$, we define

$$\hat{m}_l(t) = \begin{cases} m_l(0), & \text{for all } t < t'; \\ m_l^{\max}, & \text{for all } t \geq t'. \end{cases}$$

where

$$m_l^{\max} = \text{argmax}\left\{m : m \in M_l, \text{SNR}_m^{\min} \leq \text{SNR}_l\right\}$$

is the modulation that gives the highest possible capacity. If the link l stays the same at all times in $m_l(t)$, so does $\hat{m}_l(t)$.

It is easy to see that $\left\{\hat{x}_p^n(t), \hat{m}_l(t)\right\}$ satisfies the capacity constraints (5) since $\hat{m}_l(t)$ provides higher (or equal) capacity than $m_l(t)$. The perseverance constraint (6) is also satisfied as $\hat{x}_p^n(t) = x_p^n(t)$. In addition, both solutions have the same objective value. Therefore, $\left\{\hat{x}_p^n(t), \hat{m}_l(t)\right\}$ is also an optimal solution, and the lemma is proven. □

Lemma 1 suggests that we can simply consider three link states only: *before*, *under*, and *after* reconfiguration. Based on the observation, we introduce the auxiliary integer variables $z_l(t)$ for each link l, which indicate whether l has been reconfigured ($z_l(t) = 1$) or not ($z_l(t) = 0$) at time t. By definition, $z_l(t)$ is non-decreasing over t and $z_l(0) = 0$.

Another implication from Lemma 1 is that we only need to consider three possible link capacities: the original capacity, the best possible capacity, or zero when the link is down. Inspired by the fact, we define the minimum possible capacity $c_l^{\min}$ and the maximum possible capacity $c_l^{\max}$ as follows. The minimum possible capacity is the original capacity if sustainable, or zero otherwise:

$$c_l^{\min} = \begin{cases} c_l[m_l(0)], & \text{if } \text{SNR}_{m_l(0)}^{\min} \leq \text{SNR}_l; \\ 0, & \text{otherwise,} \end{cases}$$

and the maximum possible capacity is given by

$$c_l^{\max} = \max\left\{c_l[m] : m \in M_l, \text{SNR}_m^{\min} \leq \text{SNR}_l\right\}.$$

Without loss of generality, we define $c_l^{\max} = 0$ if the no modulation meets the criterion. By definition, we know $c_l^{\max} \geq c_l^{\min}$.

With the above definitions, the three link states at time $t \in T$ are corresponding to

$$\begin{aligned} &\text{before: } z_l(t-1) = 0,\ z_l(t) = 0,\ c_l(t) = c_l^{\min}; \\ &\text{under: } z_l(t-1) = 0,\ z_l(t) = 1,\ c_l(t) = 0; \\ &\text{after: } z_l(t-1) = 1,\ z_l(t) = 1,\ c_l(t) = c_l^{\max}. \end{aligned}$$

Accordingly, we can rewrite the capacity constraints (5) as

$$y_l(t) \leq c_l^{\min}(1 - z_l(t)) + c_l^{\max} z_l(t-1) \tag{7}$$

for all $l \in L$ and $t \in T$.

Using the above transformation, we then reformulate RAP as an MILP below.

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

$$\text{s.t.} \quad \text{capacity constraints (7)}$$

$$\text{perseverance constraints (6)}$$

$$\text{initial constraints}$$

$$x_p^n(t) \geq 0 \qquad \forall t \in T, n \in N, p \in P^n \qquad (8)$$

$$z_l(t-1) \leq z_l(t) \qquad \forall t \in T, l \in L \qquad (9)$$

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

The feasibility constraints (9) ensure that $z_l(t)$ is non-decreasing. Once we obtain the optimal solution to the MILP, we can derive the corresponding optimal control solution by setting the optimal sending rate and

$$m_l(t) = \begin{cases} m_l(0), & \text{if } z_l(t) = 0; \\ m_l^{\max}, & \text{if } z_l(t) = 1, \end{cases}$$

for all $t \in T$. Since $z_l(t)$ corresponds to a modulation operation in the reconfiguration plan, we call it the *configuration at time* t, or simply the *configuration* when referring to all $t \in T$.

IV. Analysis

Although MILP solvers can solve RAP in the MILP form, they may not solve it in polynomial-time. In the following analysis, we show in Proposition 1 that it is hard to solve or even closely approximate RAP in polynomial time. This result justifies the need for good heuristics.

We then ask why RAP is hard to solve. We find that if we relax the time horizon, with a mild assumption on the available paths for each flow, it is possible to update all the links in finite steps under perseverance constraints (Proposition 2 and Corollary 1). In other words, we can always reach the optimal final throughput despite that the update sequence may be extremely long. As computing and deploying a long update sequence is not trivial, these results may lead to future research on online reconfiguration algorithm. In this work, we will focus on obtaining the reconfiguration plan in a given T.

Below, we begin our analysis with Proposition 1.

Proposition 1. *RAP is NP-hard, and it cannot be approximated to a constant factor in polynomial time unless P=NP.*

Proof. We prove the proposition by providing a polynomial-time reduction from the maximum independent set (MIS) problem to RAP. Since MIS is NP-hard without constant factor approximation (unless P=NP), the proposition follows.

Given an MIS instance, e.g., the left graph in Fig. 3, we transform each node to a source-destination pair representing a flow as the right graph in RAP. The available path set P^n of each flow n consists of one direct path (dashed) and some other paths (solid). Each direct path goes through a link of capacity equal to the number of MIS adjacent nodes. For each edge in the MIS graph, we generate a pair of paths connecting

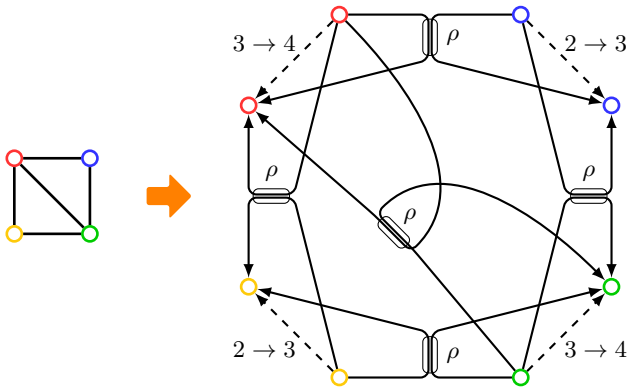


Fig. 3. We show in Proposition 1 that RAP is NP-hard by a polynomial-time reduction from the maximum independent set (MIS) problem, which is NP-hard without a constant factor approximation (unless P=NP).

the flows corresponding to the MIS endpoints and sharing one link of capacity ρ.

We consider the RAP on the transformed graph with $T = 2$ and perseverance level ρ. At $t = 0$, only the dashed paths send at the full rate. The SNRs change such that the best possible capacities of the dashed links are augmented by 1.

The optimal solution to RAP maximizes the number of reconfigured dashed links. Once a dashed link is reconfigured, its carrying traffic occupies all shared links due to the perseverance constraint. As such, the flows that share a link with the reconfigured flow cannot change their dashed link. Equivalently, once an MIS node is chosen, i.e., with its corresponding flow reconfiguring the dashed link, its neighboring MIS nodes cannot be selected. Therefore, by choosing the MIS nodes corresponding to the reconfigured RAP flows, solving RAP leads to the optimal MIS solution.

□

Unless P=NP, Proposition 1 suggests that it is impossible to design an approximation algorithm to RAP with a constant factor performance guarantee. Therefore, we shift to find good heuristics to approach RAP in Section V.

Although it is hard to determine the optimal solution in a given horizon T, it is possible to find a feasible plan if T is long enough. The following proposition and corollary reveal that we can always reconfigure the network within finite steps in the presence of alternative paths, despite the length of the resulting plan.

Proposition 2. *Suppose $\rho < 1$ and P^n contains at least 2 non-zero capacity edge-disjoint paths for all $n \in N$. Starting with a feasible initial state, there exists a finite step rate adaptation plan to reconfigure any link $l \in L$.*

Proof. We prove the proposition by the following naive strategy: We keep shrinking the traffic by

$$x_p^n(t) = \rho x_p^n(t-1)$$

until we spare enough bandwidth to move all traffic out of the link l.

Consider the non-zero capacity edge disjoint paths $p_1, p_2 \in P^n$ where $l \in p_1$ and let $c' > 0$ be the capacity of path p_2. We define the shrinking step bound j_l^n as

$$j_l^n = \operatorname{argmin} \left\{ j \in \mathbb{Z}_+ : \rho^j \leq \frac{c'}{c' + c_l} \right\}$$

which is a finite number since $\rho < 1$. j_l^n upper bounds the number of steps we need to shrink the original traffic to move flow n out of link l. To see the reason, we compute the spare bandwidth of p_2 after shrinking. Pick an arbitrary link $l' \in p_2$, we know after j_l^n shirking steps, the spare capacity of link l' is given by

$$\begin{aligned} &c_{l'} - y_{l'}(j_l^n) = c_{l'} - \rho^{j_l^n} y_{l'}(0) \\ &\geq (1 - \rho^{j_l^n}) c_{l'} \geq (1 - \frac{c'}{c' + c_l}) c' = \frac{c_l c'}{c' + c_l} \end{aligned}$$

by the definition of j_l^n and that c' is the capacity of p_2. Meanwhile, the total traffic on l is shrunk to

$$y_l(j_l^n) = \rho^{j_l^n} y_l(0) \leq \frac{c' c_l}{c' + c_l}.$$

As such, it is feasible to move all the traffic from p_1 to p_2 after j_l^n steps.

Consequently, to move all the traffic out of link l, we need to shrink the traffic by

$$j_l = \max \{ j_l^n : n \in N \}$$

steps, which is finite as N is finite. □

Corollary 1. *Suppose $\rho < 1$ and P^n contains at least 2 non-zero capacity edge-disjoint paths for all $n \in N$. Starting with a feasible initial state, there exists a finite step rate adaptation plan to reconfigure all links $l \in L$.*

Proof. Proposition 2 suggests that we can reconfigure an arbitrary link in j_l steps. Therefore, we can reconfigure all the links in sequence, taking at most $\sum_{l \in L} j_l$ steps. □

From the proof of Proposition 2, we can see that j_l is smaller when ρ is smaller. Correspondingly, the worst-case horizon estimation $\sum_{l \in L} j_l$ in Corollary 1 is also smaller. This relationship matches the intuition: When ρ is smaller, we are more willing to endure traffic fluctuation, and hence, a shorter plan is possible.

V. Algorithms

Since Proposition 1 suggests that we cannot approximate RAP to a constant factor, we develop an LP-relaxation based heuristic to approach RAP, as shown in Fig. 4. The main idea is to find a feasible reconfiguration plan that respects perseverance constraints and then maximize the usage of available links at each step. To do so, we first study the properties of feasible solutions to RAP to iteratively identify step by step the links to update, to which we refer as a *configuration* (Section V-A). Using the properties, we derive a two-step LP and a greedy uprounding procedure in Section V-B to compute the configuration for one step. Based on the obtained feasible configurations, we assign the sending rates of the flows to utilize the available links fully in Section V-C. As such, we can obtain a *work conserving* reconfiguration plan for RAP, and we summarize our algorithm in Section V-D.

A. Properties of Feasible Solutions

Before developing the heuristic, we first study the properties of feasible solutions to RAP.

Lemma 2. *If there exists a feasible solution to RAP, there exists a feasible solution with the same objective value such that $z_l(T) = z_l(T-1)$ for all $l \in L$.*

Proof. Let z_l be the existing optimal solution. We prove the lemma by showing that setting $z_l(T) = z_l(T-1)$ is still optimal.

If $z_l(T-1) = 1$, we know $1 \geq z_l(T) \geq z_l(T-1) = 1$, and hence $z_l(T) = z_l(T-1)$.

If $z_l(T-1) = 0$ and $z_l(T-1) \neq z_l(T) = 1$, the upper bound in (7) is 0. By setting $z_l(T) = z_l(T-1)$, we can augment the upper bound to $c_l^{\min} \geq 0$. Therefore, the original x_p^n is still feasible and the objective value remains the same, and the lemma is proven. □

Lemma 2 allows us to find only $z_l(1)$ to $z_l(T-1)$. The next lemma reveals that any two consecutive steps in a feasible solution satisfy a specific constraint set.

Lemma 3. *For any feasible solution to RAP, it satisfies the following constraint set:*

capacity constraints (7)

$$x^n(t) \geq \rho^t x^n(0), \quad x^n(t+1) \geq \rho x^n(t) \quad \forall n \in N \qquad (10)$$

initial constraints (given $z_l(t-1)$)

feasibility constraints (8) and (9)

$$0 \leq z_l(t) \leq 1, \quad 0 \leq z_l(t+1) \leq 1 \qquad \forall l \in L \qquad (11)$$

for all $t = 1, \ldots, T-1$.

Proof. For an arbitrary $t = 1, \ldots, T-1$, consider a feasible solution consisting of $x_p^n(t)$ and $z_l(t)$. We know $z_l(t), z_l(t+1) \in \{0, 1\}$, and hence (11) is satisfied directly. Also, since

$$x^n(t) \geq \rho x^n(t-1) \geq \rho^2 x^n(t-2) \geq \cdots \geq \rho^t x^n(0),$$

the feasible solution satisfies (10) as well. As the other constraints remain the same, we prove that the feasible solution satisfies the constraint set. □

B. Two Step Relaxation and Greedy Uprounding

The configurations $z_l(t)$ is the main obstacle toward obtaining a solution to the MILP. Therefore, we focus on how to determine $z_l(t)$ for each $t \in T$. Our strategy is to fix $z_l(t)$, step by step, in a greedy manner. To do so, we first consider the following two-step LP.

$$\max \quad \sum_{n \in N} x^n(t+1) \qquad \text{(2-step LP}(t)\text{)}$$
$$\text{s.t.} \quad \text{constraint set in Lemma 3}$$

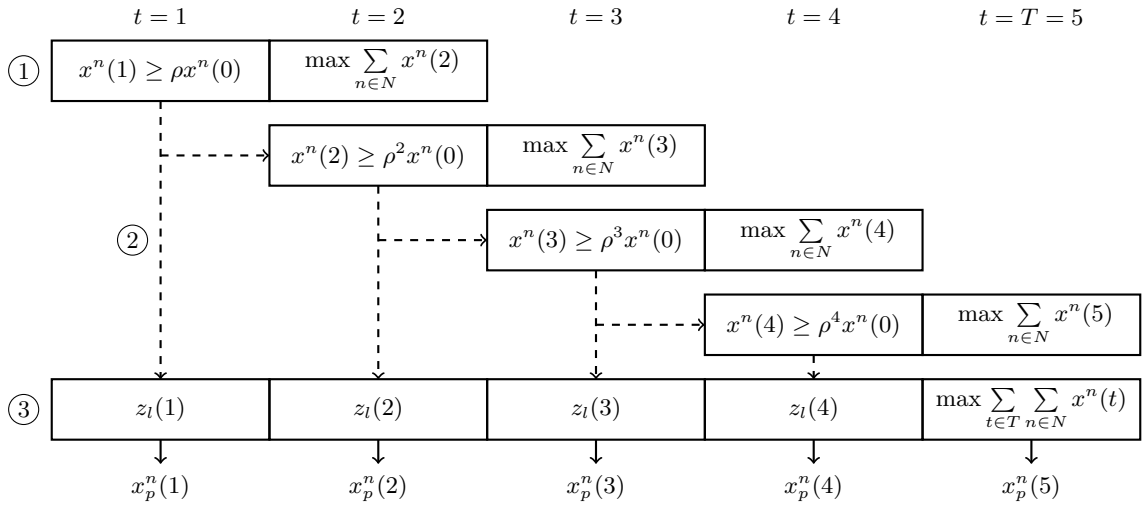


Fig. 4. Since RAP is NP-hard without constant factor approximation (unless P=NP), we propose the LP-based heuristic Algorithm 2 to approach RAP. Algorithm 2 iteratively determines the configuration at time t ($z_l(t)$) by ① solving the two step LP with the initial state $z_l(t-1)$ and then ② using Algorithm 1 to greedily reconfigure links. Once the configuration is fully determined, Algorithm 2 adopts the ③ work conservation objective to generate the sending rate on each path $x_p^n(t)$.

Lemma 3 puts that any feasible MILP solution to RAP will also satisfy the constraints in 2-step LP(t). Therefore, our strategy is to construct a feasible MILP solution by iterating 2-step LP(t) over time.

Since 2-step LP(t) does not necessarily give integer $z_l(t)$, we use Algorithm 1 to upround $z_l(t)$. The idea is to rank the links based on their relaxed $z_l(t)$ solution, or $\hat{z}_l$ in the algorithm. The closer $z_l(t)$ is to 1, the earlier we consider to upround it. Starting with the initial configuration $z_l(t-1)$, we try to upround the links with $\hat{z}_l > 0$ one at a time according to the above strategy.

Algorithm 1: Greedy Upround

1: $\hat{z}_l \leftarrow z_l(t)$ for all $l \in L$.
2: $z_l(t) \leftarrow z_l(t-1)$ for all $l \in L$.
3: **for** from large to small $\hat{z}_l > 0$ **do**
4: Upround the corresponding $z_l(t)$ to 1.
5: If 2-step LP(t) is infeasible, downround $z_l(t)$ to 0.
6: **end for**

C. Work Conserving Reconfiguration Plan

Given a configuration, there can be multiple plans with different transient sending rates $x_p^n(t)$ that all maximize the final aggregate sending rate $\sum_{n\in N} x^n(T)$. Throughout those plans, we prefer the one that is *work conserving*, i.e., sending as much as possible. To find a work conserving reconfiguration plan, we can fix the configuration and try to maximize the total sending rate at each step under the perseverance constraints. This process can be done by optimizing the following objective

$$\sum_{t\in T}\sum_{n\in N} x^n(t), \tag{12}$$

which is the sum of the total sending rate at each step, with the given configuration under the same RAP constraint set. Notice that this optimization also leads to the maximum final throughput $\sum_{n\in N} x^n(T)$ under the given configuration.

A similar concern applies to the optimal RAP, which can give the optimal final throughput with low-performance transient steps. For example, in a one-shot reconfigurable scenario, updating at the beginning yields the same final performance as doing so at the end, although the former plan leads to higher throughput in between. Ideally, a work conserving plan should not procrastinate, and to avoid such procrastination, we can optimize RAP twice: We first solve for the optimal final throughput and fix it under the objective (12) to get the work conserving optimal reconfiguration plan. This time we only pin down the final throughput without fixing the configuration and rely on the optimization solver to find the swiftest update plan.

D. Linear Programming Based Heuristic

We develop our LP-based heuristic Algorithm 2 using the components described in the previous sections. Starting with $z_l(0) = 0$, Algorithm 2 iterates the two step LP through all time $t = 1, \ldots, T-1$ to get the uprounded $z_l(t)$. Using the configuration formed by the uprounded $z_l(t)$, we solve (12) for the work conserving reconfiguration plan. The whole procedure is summarized in Fig. 4 and Algorithm 2, where line 8 is inspired by Lemma 2.

From the structure of Algorithm 2, one may wonder if we can derive an online version by iterating two step LP with both $z_l(t)$ and $x^n(t)$ as the states, i.e., we replace $x^n(t) \geq \rho^t x^n(0)$ by $x^n(t) \geq \rho x^n(t-1)$ and feed the results to the next iteration. In this way, the online algorithm does not need to know T a priori, and we don't need the full configuration to generate

Algorithm 2: LP-Relaxation Based Heuristic

1: $z_l(0) \leftarrow 0$ for all $l \in L$.
2: **if** $T \geq 2$ **then**
3: **for** $t = 1, \ldots, T-1$ **do**
4: Solve 2-step LP(t).
5: Uproud $z_l(t)$ by Algorithm 1.
6: **end for**
7: **end if**
8: $z_l(T) \leftarrow z_l(T-1)$ for all $l \in L$.
9: Solve (12) based on the resulting configuration to find work conserving reconfiguration plan.

$x_p^n(t)$. It turns out that such a design gets stuck if the two-step solution chooses to stay, even though it is possible to augment the link capacities after reducing the sending rates for a few steps. Also, when T is no longer a fixed (and tentatively small) number, the assumption that N would remain a constant may not be valid anymore, and the online algorithm would need to deal with variable $N(t)$. In sum, an online algorithm would be useful, and it requires a more sophisticated design than a straightforward generalization of Algorithm 2.

VI. Simulation

We compare our perseverance-aware Algorithm 2 with the state-of-the-art churn-based RADWAN [3] through simulations. In particular, we are interested in the following four issues: the advantage of rate-adaptive links (Section VI-B), the convergence pace under different perseverance levels (Section VI-C), the mitigation of transition fluctuation (Section VI-D), and the comparison of computation overhead (Section VI-E).

A. Simulation Setup

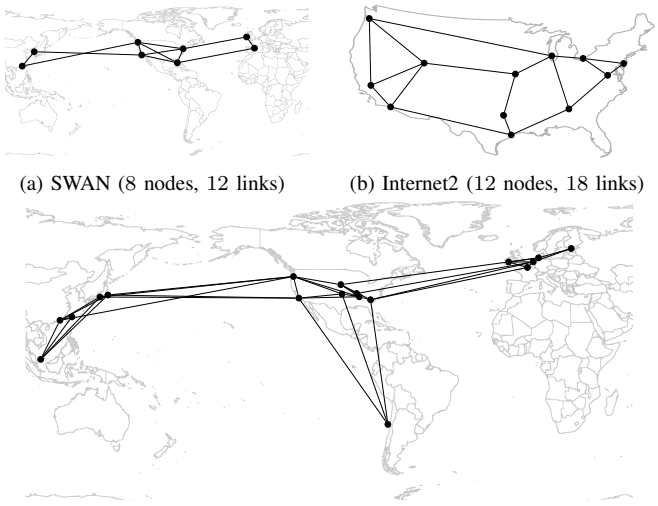
In the simulations, we consider the horizon $T = 5$ and the perseverance level $\rho = 0.5$ as our baseline.[1] Three methods are simulated: the work conserving optimal solution (OPT), Algorithm 2, and RADWAN, with COIN-OR CBC [8] as the LP and MILP solver. RADWAN updates the network in one shot by reconfiguring the links that will carry traffic exceeding their original bandwidth. We set RADWAN's churn trade-off parameter $\epsilon = 0.1$, which regulates the importance of churn in optimization.[2]

We base our simulations on three different existing WAN topologies, as shown in Fig. 5: Microsoft's SWAN [7] (8 nodes, 12 links), Internet2 [9] (12 nodes, 18 links), and Google's B4 [10], [11] (18 nodes, 39 links). Each link has 100 Gbps initially, and according to the data in [3, Figure 1], we assume that the maximum achievable capacity of a link l is

$$c_l^{\max} = 25 \left\lfloor \frac{15(\mathrm{SNR}_l - 3) + 50}{25} \right\rfloor,$$

[1]We will justify the choice of the baseline parameters in Section VI-B.

[2]RADWAN [3] sets ϵ to 0.001, which emphasizes on the final throughput. As we will evaluate the update disturbance in the following experiments, we set $\epsilon = 0.1$ to add more weight to churn reduction for a fair comparison.



(a) SWAN (8 nodes, 12 links) (b) Internet2 (12 nodes, 18 links)

(c) B4 (18 nodes, 39 links)

Fig. 5. The simulated network topologies. Each link has initial capacity 100 Gbps.

which maps 3 dB to 50 Gbps and 13 dB to 200 Gbps.

Each simulation consists of the following steps. We first generate flows by selecting each source-destination pair with probability 0.1. Under the initial link capacity 100 Gbps, we compute the initial state to maximize the aggregate throughput. We then randomly vary the SNR of each link to be uniformly distributed within $[7, 15]$ dB (and hence $c_l^{\max} \in [100, 225]$ Gbps). With the new SNR, we run the algorithms to compute the reconfiguration plans. In each experiment, we repeat the simulation 1000 times to collect the statistics.

B. Advantage of Rate-Adaptive Links

The first question we would ask is whether having adaptive links is beneficial in the presence of link reconfiguration delay. Without reconfiguration delay, adaptive links allow the system operator to augment the link rate, and hence the network can transmit more data. However, link reconfiguration delay leads to temporary bandwidth reduction, and it is less clear if we can enjoy the bandwidth augmentation with controlled disruption (specified by the perseverance level ρ).

To find out, we compare the methods with the most conservative choice – without adaptive links. This scenario is similar to requiring no traffic disruption ($\rho \approx 1$). In contrast, RADWAN is much more aggressive as it augments the links in one shot that might cause a significant impact on the traffic ($\rho \approx 0$). Our methods adopt a mild perseverance constraints $\rho = 0.5$. Despite having the perseverance requirements that ensure steadier reconfiguration than perseverance-agnostic approaches, Table I shows that our methods perform comparably to RADWAN and improve the overall throughput by 40% to 50% for the no-adaptive-link cases.

C. Convergence under Different Perseverance Levels

The choice of horizon T determines the performance: The RAP with a longer T searches a broader feasible region and

TABLE I
AVERAGE THROUGHPUT (Gbps) UNDER DIFFERENT WANS. OUR METHODS (OPT AND Algorithm 2) 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	Algorithm 2	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 $\epsilon = 0.1$.

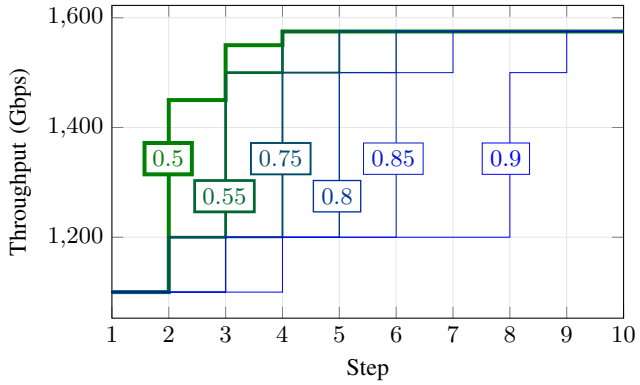


Fig. 6. Larger perseverance levels ρ (boxed values) prevent aggressive update with large disturbance, and hence, it takes more steps for Algorithm 2 to converge to the maximum throughput.

may land on the solutions with better throughput. However, a longer T also burdens the solver with a more massive load, which slows down the reconfiguration plan generation. Therefore, we should choose an appropriate-sized T given the perseverance level ρ.

Intuitively, a larger perseverance level leaves a smaller room for traffic maneuver and hence leads to a more conservative reconfiguration plan. As a result, it takes more steps to converge to the best possible throughput. This phenomenon is confirmed in Fig. 6, in which we apply Algorithm 2 for the SWAN network with $T = 10$ and various perseverance levels ρ.

We repeat the simulation in Fig. 6 1000 times and plot the statistics in Fig. 7. If a case cannot converge to the maximum throughput, we set its convergence step as 10. It turns out, for $\rho = 0.5$, Algorithm 2 converges in 5 steps in 99% of the cases, which inspires us to set the parameters accordingly.

Fig. 7 also reveals that as the perseverance level increases, the minimum convergence steps grows super-linearly. It matches the intuition: A higher ρ requires exponentially more steps to move the traffic out of a link.

D. Mitigation of Transition Fluctuation

One major benefit of introducing the idea of perseverance is to reconfigure in a "smoother" manner. We demonstrate

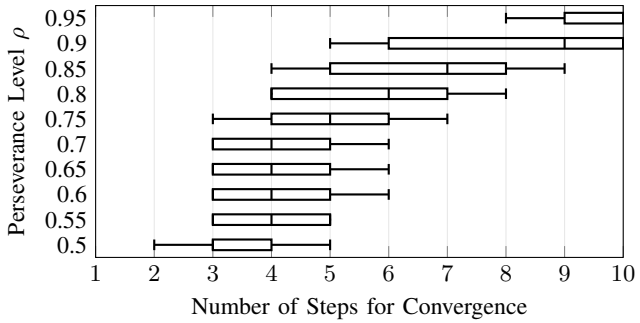


Fig. 7. The 1^{st}-5^{th}-50^{th}-95^{th}-99^{th} percentiles of the minimum number of steps needed for throughput convergence. When $\rho = 0.5$, Algorithm 2 converges in 5 steps in 99% of the 1000 random cases.

the concept by the metric *maximum throughput deviation*. The maximum throughput deviation is defined as

$$\max\left\{\left|\sum_{n\in N} x^n(t) - \sum_{n\in N} x^n(t-1)\right| : t \in T\right\},$$

which captures the maximum throughput fluctuation during the reconfiguration.

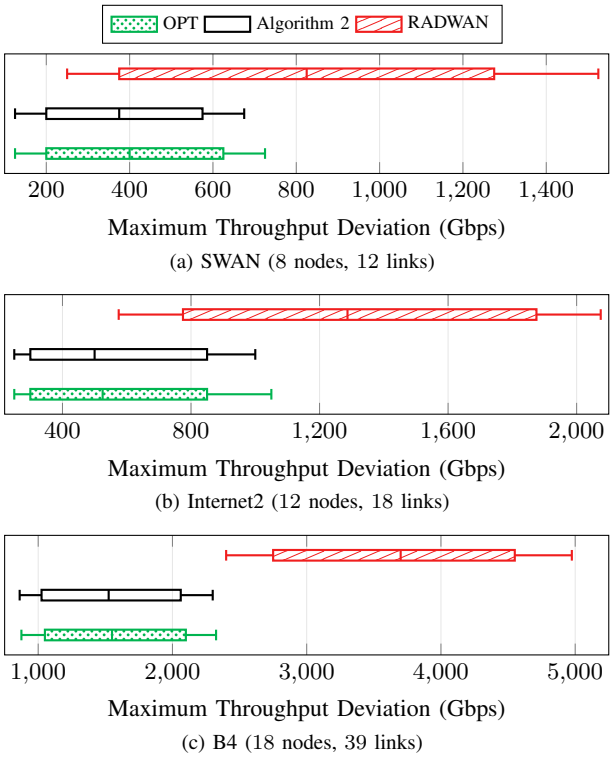
We plot in Fig. 8 the percentiles of maximum throughput deviation under SWAN, Internet2, and B4 (each formed by 1000 random cases). As expected, RADWAN suffers huge throughput deviation during the one-shot reconfiguration. The optimal solution and Algorithm 2 have much smaller deviations, only about 40% to 50%, than RADWAN. Algorithm 2 performs slightly better than the optimal solution. It is because the optimal solution always converges with fewer steps than Algorithm 2, and hence Algorithm 2 reconfigures slower with smaller fluctuation.

Another noticeable trend is that the gap between RADWAN and the other two methods becomes wider when the network scales. It is because that a larger network with more flows leads to more path diversity for moving traffic around, and hence the perseverance-aware methods can potentially perform better.

E. Comparison of Computation Overhead

Although the performance of the optimal solution and Algorithm 2 are comparable in throughput and maximum throughput deviation, the required computation overheads differ a lot. Table II summarizes the average CPU computation time to compute a reconfiguration plan in Fig. 8. Algorithm 2 uses only 22.2%, 8.0%, and 0.03% of time than OPT while reaching similar performance under the three network topologies. Consequently, we claim Algorithm 2 provides a better performance/computation-overhead trade-off than aiming for the optimal solution directly.

We also argue that the computation overhead is acceptable in practice. As reported in [3, Section 7.1], the current average reconfiguration downtime is 68 seconds. Given the significant mitigation of transition fluctuation as shown in Section VI-D, it is worth spending some milliseconds to compute a perseverance-aware plan using Algorithm 2. On the other hand, [3] also mentions that the reconfiguration time could



(a) SWAN (8 nodes, 12 links)

(b) Internet2 (12 nodes, 18 links)

(c) B4 (18 nodes, 39 links)

Fig. 8. The 1^{st}-5^{th}-50^{th}-95^{th}-99^{th} percentiles of maximum throughput deviation under different networks. The gap between RADWAN and the other two methods widens when the network has more nodes (and with more flows).

TABLE II
AVERAGE CPU COMPUTATION TIME (MS) AND FRACTION. ALGORITHM 2 USES MUCH LESS TIME AND SCALES BETTER THAN OPT.

topology	OPT	Algorithm 2	fraction $\left(\frac{\text{Algorithm2}}{\text{OPT}}\right)$
SWAN	67.7	15.0	22.2%
Internet2	497.1	39.6	8.0%
B4	1.2×10^6	332.0	0.03%

potentially be reduced to 35 ms if not turning off the laser. Once some new technology shortens the reconfiguration delay, we might want to strive for even lower computation overhead and new algorithm design.

VII. CONCLUSION

We investigate the rate adaptation planning (RAP) problem under reconfiguration delay and introduce the perseverance constraints to regulate the temporary disruption. With the perseverance constraints, we formulate RAP in both discrete control and MILP form. RAP is then shown NP-hard without constant factor approximation (unless P=NP), which motivates us to develop an LP-based heuristic algorithm. The simulations show that the proposed algorithm converges quickly under mild choices of perseverance level. It improves the overall throughput by 40% to 50% comparing to the no-adaptive-link case. Meanwhile, it gives a much smoother transition, with only 40% to 50% maximum throughput deviation, than RADWAN.

Besides perseverance, there are also some other metrics the network operator might consider during the reconfiguration. For instance, it might be preferable to sustain some throughput level throughout the update. This objective is related to the multi-step routing reconfiguration problems previously studied in [7], [12], and it is possible to extend the existing methods to generate a link update plan with throughput level guarantee.

ACKNOWLEDGEMENTS

We thank our shepherd Marco Chiesa and ICNP reviewers whose comments helped us improve the paper.

REFERENCES

[1] X. Jin *et al.*, "Optimizing bulk transfers with software-defined optical WAN," in *Proc. ACM SIGCOMM*, 2016, pp. 87–100.
[2] M. Ghobadi and R. Mahajan, "Optical layer failures in a large backbone," in *Proc. ACM IMC*, 2016, pp. 461–467.
[3] R. Singh *et al.*, "RADWAN: Rate adaptive wide area network," in *Proc. ACM SIGCOMM*, 2018, pp. 547–560.
[4] S. Jia *et al.*, "Competitive analysis for online scheduling in software-defined optical WAN," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
[5] R.-J. Essiambre *et al.*, "Capacity limits of optical fiber networks," *Journal of Lightwave Technology*, vol. 28, no. 4, pp. 662–701, 2010.
[6] J. K. Fischer *et al.*, "Bandwidth-variable transceivers based on four-dimensional modulation formats," *Journal of Lightwave Technology*, vol. 32, no. 16, pp. 2886–2895, 2014.
[7] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 15–26, 2013.
[8] CBC (COIN-OR branch and cut). [Online]. Available: https://projects.coin-or.org/Cbc
[9] The Internet2 network. [Online]. Available: http://www.internet2.edu/
[10] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 3–14, 2013.
[11] C.-Y. Hong *et al.*, "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. ACM SIGCOMM*, 2018.
[12] S.-H. Tseng *et al.*, "Time-aware congestion-free routing reconfiguration," in *Proc. IFIP Networking*, may 2016, pp. 55–63.