Practical Programmable Topologies in the WAN

Regular Paper

Klaus-Tycho Foerster[1], Monia Ghobadi[2], Ratul Mahajan[3], Asaf Valadarsky[4]

[1]University of Vienna, [2]Microsoft Research, [3]Intentionet, [4]Hebrew University of Jerusalem

[1]klaus-tycho.foerster@univie.ac.at , [2]mgh@microsoft.com, [3]ratul@ratul.org, [4]asaf.valadarsky@mail.huji.ac.il

Abstract

With the rapid adoption of Reconfigurable Optical Add-Drop Multiplexers (ROADMs) in wide-area networks (WANs), backbone providers have the ability to change the physical layer of their networks by programming wavelengths on each edge. Yet the physical layer of today's WANs has remained static because (i) there is no simple way to predict the throughput gains, and (ii) there is no clear path to the practical insertion of programmable topologies into the already complex traffic engineering (TE) schemes. To overcome these challenges, we first provide bounds on throughput gains of programmable WANs, propose a simple heuristic to estimate gains by considering the physical topology, and present opportunities for programmability by studying traffic traces. Second, we argue that the path to adoption should not involve rewriting cross-layer TE formulations. Instead, we propose a Reflow graph as an abstraction that permits topology programmability in the TE algorithms. A Reflow graph augments the physical graph and uses it as input to the unmodified TE algorithm, capturing a variety of TE formulations. As such, we can also directly transfer standard consistent network update schemes to optimize across layers. Moreover, we evaluate our abstraction using a testbed and simulations: our results show that Reflow has competitive computation times and is feasible in practice, allowing to quickly react to unplanned events such as fiber cuts.

1 Introduction

Today, all wide-area network (WAN) communication is carried over optical equipment. Large service providers, such as Google, Microsoft, and Facebook, own $O(100{,}000)$ miles of optical fiber cables capable of carrying $O(100)$ of Tbps of data across the globe. Optical backbones cost billions of dollars, and as online services are becoming an integral part of today's society, these backbones need to be highly efficient. Inspired by reconfigurable topologies in data center networks [10, 11, 14, 22, 33, 38, 39, 41, 42, 48, 59], recent research has shown that reconfiguring the optical topology in the WAN leads to greater efficiency and cost savings [31, 43, 46]. As a result, service providers are looking toward enabling *Topology Programming* (TP) together with *Traffic Engineering* (TE) to boost efficiency [15, 24, 45].

TP in a WAN is achieved using modern optical devices, such as Reconfigurable Optical Add-Drop Multiplexers (ROADMs) [1, 51]. Flexible-grid capable ROADMs [50] are considered one of the most significant advances in Dense Wavelength Division Multiplexing (DWDM) systems technology over the last decade [15]. They allow the switching of incoming wavelengths from an input port to any output port, enabling the network operator to program the allocation of wavelengths across fibers during deployment and maintenance. As thus, ROADMs are deployed widely in the WAN [15].

Despite much recent attention to TP in the WAN [6, 15, 24, 28, 30, 31, 60], there is no unified formulation of how much value it brings to currently deployed TE schemes. Without such understanding, providers are reluctant to adopt TP, since it requires a radical change in a network's control system. In this paper, we show that the gains of enabling TP on a graph can vary widely across topologies. To shed light on this variance, we quantify the gains of TP based on the topology and prove that the throughput gain factor is at most $2\Delta - 1$, where Δ is the maximum degree in the graph. This bound holds for any combination of topology, traffic changes, and fiber cuts (edge failures), as long as the realistic assumption holds that each node has at least as many ROADMs as incident fibers. We also provide matching lower bounds.

We then propose a simple heuristic to estimate practical throughput gains and evaluate it on two realistic topologies. Even though the topologies are different, they are structurally similar enough to lead to the same average throughput gain factor of 40%. This coincides with the findings of previous work [31, 60], but we obtained it without a TP implementation.

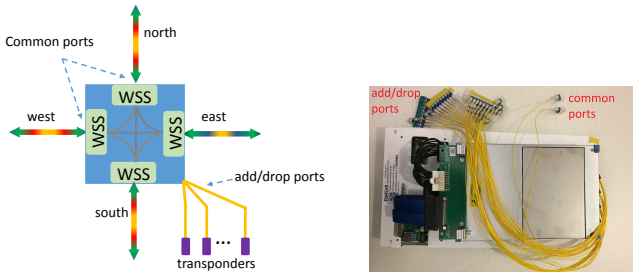
A second and equally important question is how to achieve these gains in practice. Current solutions (most notably OWAN [30, 31] and Sedona systems [6, 21]) involve the cross-layer orchestration of the IP and optical layers to simultaneously solve the TE and TP problems. However, this requires providers to throw out their existing TE formulations, write a new joint TP+TE formulation, and ensure it achieves the same goals as those intended by the original TE—a task that is so tedious that it hinders adoption.

We solve this challenge by keeping the TE and TP problems decoupled while sliding TP under the TE algorithm. More specifically, we use an abstraction to program the topology without having to modify the TE formulation. Our abstraction, the "Reflow graph," is an augmentation of the physical topology, with additional dummy nodes and edges used as the new input to the TE algorithm. In our method, we mimic a TP+TE joint optimization by solving the unmodified TE on the Reflow graph instead of the original graph. The key concept in the Reflow graph is that we represent wavelength programmability as network "flows," enabling us to use multi-commodity flow formulations to solve a cross-layer optimization. In doing so, we show that many TE methods carry over to the Reflow abstraction: solving the unmodified TE on Reflow is the equivalent of solving a cross-layer TE formulation for a good subset of TE formulation classes for all linear throughput maximization TEs, as well as for max-min fairness TEs such as SWAN [25] and B4 [29]. Furthermore, because Reflow relies on network flows, several properties of current TE practices, such as consistent updates, also carry over. Our simulation results show that Reflow also obtains competitive runtimes in comparison to an integral optimization approach. Finally, we use a testbed to benchmark key abstraction assumptions in practice and show an end-to-end working scheme.

2 Optics Background

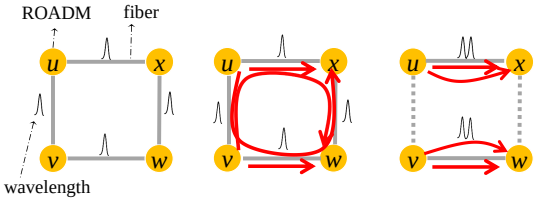
This section provides a brief overview of optical backbone networks with a focus on programmability.

ROADM. A Reconfigurable Optical Add Drop Multiplexer (ROADM) is an optical device that combines multiple wavelengths into fiber cores while adding/dropping wavelengths to/from the existing multi-wavelength signal. This is achieved through the use of Wavelength Selective Switching (WSS) modules. ROADMs allow operators to dynamically reconfigure ports to carry any combination of wavelengths while adding/dropping any wavelength at any time [1, 4]. Figure 1(a) illustrates the structure of a 4-way ROADM that is capable of steering wavelengths between its north, south, west, and east *common ports* while adding/dropping any of the wavelengths. The added/dropped wavelengths are modulated/demodulated from/to binary signals in the electrical domain. Figure 1(b) shows the evaluation board used in our testbed consisting of 40 add/drop ports and 2 common ports [2]. ROADMs open the door to research on orchestrating IP and optical layers as recent research shows ROADMs are capable of programming wavelengths in tens to hundreds of milliseconds [12, 31].



(a) Depiction of a 4-way ROADM. (b) The ROADM evaluation board used in our testbed.

Figure 1: A Reconfigurable Add/Drop Multiplexer (ROADM) uses Wavelength Selective Switches (WSS) to steer wavelengths between its ports. The common ports carry the light to/from other ROADMs. The add/drop ports are used to add/drop a wavelength to/from the common ports.



(a) Physical graph (b) TE's throughput: 1+1 (c) (TE + TP)'s throughput: 2+2

Figure 2: Enabling topology programmability (TP) on top of traffic engineering (TE) can lead to throughput gains. By reprogramming the wavelengths across the cut, the throughput can increase from two to four, a gain of 100%.

Fiber. Optical fiber cables are used as means to transmit light between the two ends of the fiber. In WAN backbones, optical fiber cables act like freeways, running for hundreds of miles to connect distant cities. Using Dense Wavelength Division Multiplexing (DWDM), O(100) of separate wavelengths can be multiplexed into a lightstream transmitted on a single optical fiber. The frequency, modulation, and number of optical wavelengths that are multiplexed onto fiber depend on the technology.

Transponder. A transponder is used to convert the signal between the optical and electrical domains of each wavelength. The data-rate carried by an optical wavelength depends on the modulation implemented on the transponder. Earlier systems used simple modulations, such as OOK (on-off keying) with 10 Gbps per wavelength. Today, 100 Gbps per wavelength is widely available through more efficient modulation, such as Polarization-Multiplexed Quadrature Phase-Shift Keying (PM-QPSK).

Physical host graph. The physical host graph is the WAN's physical backbone, i.e., its optical links and nodes. The nodes represent optical cross-connects (such as ROADMs) points-of-presence where one or more ports from IP routers connect to the optical node. The optical nodes (i.e., ROADMs), use wavelength division multiplexing to combine multiple wavelengths onto a single fiber and steer different combinations to its neighbors (such as north/south/west/east fibers in Figure 1(a)).

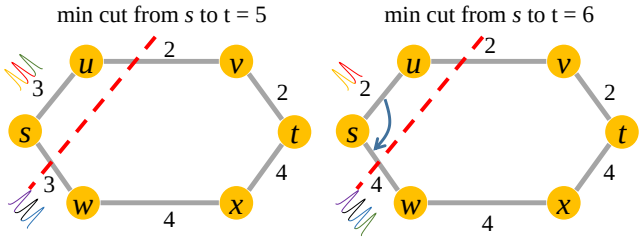
3 Motivation

In this section, to explain the need for programmable WAN topologies, we give examples when smart TE optimizations cannot beat TP. Furthermore, we illustrate the potential throughput gains of a programmable WAN topology by using a small max-flow problem. Finally, we provide evidence of the benefits of programmable topologies by studying the changes in a traffic matrix of a large scale WAN.

3.1 TE $\leq$ TE+TP

This section provides intuition on why traffic engineering alone falls short when compared with TE combined with TP. In a static topology, the maximum possible throughput, even with optimal traffic engineering, is bounded by the minimum (multi-)cut of the graph [19]. However, with topology programming, we could allocate the wavelengths according to the traffic demand and augment the cut based on the current traffic demand.

Throughput gains. Figure 2(a) provides a simple example of a square physical host graph. Each node in the graph corresponds to a ROADM, and each edge represents a fiber. Now, assume each ROADM has two programmable transponders (i.e., two wavelengths), and the provider allocates one wavelength to each fiber, as shown in the figure. This allocation is optimal if the topology is static, as the network operator has to ensure that the topology is capable of routing any traffic demand. At time t, the traffic demand is as shown in Figure 2(b), where nodes u and v have two units of traffic demand to x and w, respectively, and the arrows represent the TE routes. In this case, the maximum throughput with optimal TE is two units. For simplicity, we assume that each wavelength carries one unit of demand. Figure 2(c) shows a case where the provider reprograms the topology and assigns two wavelengths to (u,x) and (v,w) fibers. In this case, the total throughput is four units, illustrating that TE+TP can lead to a factor of two throughput gain. This example shows how programming the topology according to changes in the traffic demand can lead to greater throughput in the WAN.



(a) Static topology (b) Programmable topology

Figure 3: Conceptual illustration of using programmable topologies to increase the max-flow in a graph. (a) The max-flow from s to t is 5. (b) By borrowing one unit of capacity (i.e., a wavelength) from (s,u) and giving it to (s,w), the max-flow between s and t can be increased to 6. In contrast, any TE is limited to 5 in (a).

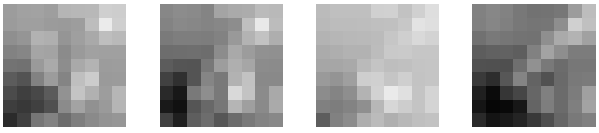
Max-flow in a Programmable Topology. As another example, we use the maximum flow problem (max-flow) [20] to illustrate the gains of having programmable topology in the WAN. Consider the graph shown in Figure 3(a) where s is the traffic source and t as the traffic sink. The number on each edge indicates the number of wavelengths (i.e., capacity) on the edge. The max-flow from s to t, which corresponds to the minimum cut in the graph (shown with dashed red line), is five in Figure 3(a). Conceptually, enabling programmability in the physical layer at node s would enable the operator to borrow one extra wavelength from edge (s,u) to (s,w) so as to increase the minimum cut between s,t, as in Figure 3(b).

The above examples intuitively show how traffic engineering alone is less efficacious than traffic engineering in programmable topologies: even optimal traffic engineering cannot increase a graph's minimum cut.

3.2 WAN Traffic

To further motivate programmable WAN topologies, we use traffic traces from distributed sites of a large scale WAN. The traffic is a mix of customer-facing and internal traffic. We collect traffic demand to/from each site on an hourly basis and aggregate the logs into a series of site-to-site traffic matrices, where each matrix represents the demand between pairs of sites. We then use IP layer's routing decisions and compute the traffic on each fiber between the sites.

Opportunity 1: Spatial traffic shift. The first opportunity to enable TP is the spatial changes in traffic pattern across WAN nodes. Figure 4 shows a heatmap of site-to-site traffic for four different hours. Rows correspond to source sites and columns to destination sites, while the color encodes the amount of traffic from the source to the destination. The darker a spot, the more traffic between its source and destination. To remove a bias for changes in the magnitude of traffic across different hours, we normalize the traffic on each heatmap. Figures 4(a) and (b) represent two hours on the same day (9 hours apart) in June; Figures 4 (c) and (d) represent the same two hours but two weeks later. As shown, the traffic pattern changes within each of the days observed, and across weeks. In fact, we observed an unpredictable change in the spatial distribution of traffic matrices within days, weeks, and months. This indicates that without TP, providers have to over-provision the WAN based on worst-case traffic predictions.



(a) (b) (c) (d)

Figure 4: Spatial changes in hourly traffic matrices: (a) and (b) are two hours on the same day in June 2017; (c) and (d) are the same two hours but two weeks later.

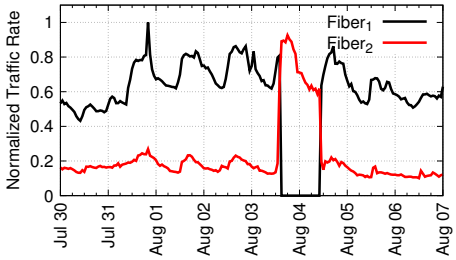


Figure 5: Traffic shift due to fiber cut.

Opportunity 2: Failure induced traffic shift. Next, we study the effect of fiber cuts on traffic pattern changes per fiber. Figure 5 shows the traffic across two fibers with the same source site. On August 3^{rd}, a fiber cut causes a sudden drop on $Fiber_1$ which, in turn, makes the traffic on $Fiber_2$ jump to four times its usual volume. This shift in traffic presents an opportunity for TP to dynamically reprogram wavelengths between the two adjacent fibers. Without such programmability, service providers would need to over-provision static wavelengths as stand-by resources to satisfy the massive shift of demand in the event of fiber cuts. In this case, even though both fibers share the same source site, $Fiber_2$ is provisioned with four times more wavelengths than is usual, such that it is ready to act as the protect path for $Fiber_1$.

4 Gains of Programmable Topologies

Can programmable topologies' opportunities be identified even before any implementation, simply by considering the physical graph G? In this section, we answer this question. We prove that for the class of TEs with linear throughput maximization as their objective,[1] the gain factor of enabling programmable topologies is between one and $2\Delta-1$, where $\Delta = max_{v \in V}\delta(v)$ is the maximum of the node degrees δ in the physical graph G. We further prove that the bounds hold for any graph, under any edge (i.e., fiber) failure scenario, and provide matching lower bound examples.

Intuitively, this result quantifies how much the multi-commodity flow (*MCF*) of the physical graph G may be improved by reprogramming the wavelengths based on worst-case traffic demand D. Clearly, the exact gains will depend on the real traffic demand, but this bound gives WAN operators an estimate of the maximum gains of programmable topologies to assess their possible deployment. In §4.2 we compute the gain factor for two real optical topologies and show that while extreme upper and lower bound cases exist, the average gain factor is around 40% for both real topologies.

4.1 Gain Bounds

Physical host graph (G). We model the physical host graph as $G=(V,E,\sigma,\mu)$, with $|V|=n$ nodes and $|E|=m$ edges. Nodes correspond to ROADMs and edges correspond to physical fibers. Each node $v \in V$ has two attributes: (i) $\delta(v)$ that is the degree of node v; (ii) $\sigma(v)$, $\sigma: V \rightarrow \mathbb{N}$ that is the total number of programmable wavelengths in v that can be allocated to v's edges (i.e., the same number as the programmable transponders in v). Depending on the modulation technology, each fiber edge has an attribute, $\mu(e)$, that corresponds to the maximum possible wavelengths on the fiber.[2]

Wavelength allocation (Λ, G_Λ). In order to route traffic on an edge $e=(v,w)$ in G, we need to assign wavelengths to e. As mentioned in §2, in optical communications, a transponder is the device that sends and receives the optical signal from a fiber. Each wavelength requires a transponder on the sending node and receiving node. Although fiber is unidirectional, today's transponders enforce bidirectionality. Hence, programming a wavelength between v and w requires the reverse path to be programmed from w to v.[3] Given this, we define a wavelength allocation Λ of a graph G by $\Lambda: V \times E \rightarrow \mathbb{N}$ representing the number of transponders allocated on each edge by a node. In finding a new wavelength allocation, nodes are limited by the pool of programmable transponders they have ($\sigma(v)$). Hence, the total number of wavelengths on each edge, $c(e)$ (the *capacity*), cannot exceed the (i) the number of channels $\mu(e)$ and (ii) the number of available transponders on its neighboring nodes; i.e., $\sum_{e=(v,w)} \Lambda(v,e) \leq \sigma(v)$, $\forall v \in V$ and $c(e) = \min(\Lambda(v,e), \Lambda(w,e), \mu(e))$. We denote G_Λ when wavelength allocation Λ is applied to G (a programmable WAN, in contrast to a static WAN).

Static WAN (G_*). The state-of-the-art in binding wavelengths to fibers is a static allocation based on the history of traffic demand, growth prediction, and failure resiliency. Once a wavelength allocation is set up, it does not change for months. We assume the static topology is an optimal traffic-oblivious wavelength allocation able to route traffic demands as best as it can. More formally, it minimizes the throughput gain obtainable by reprogramming the wavelengths under any (adversarial) traffic and failure scenarios. Without this assumption, it is easy to fabricate unreasonably large gain factors, see Appendix A.4. Furthermore, to prevent corner cases with infinite gain, we only consider $\sigma(v) \geq d(v)$ (i.e., each node has enough wavelengths to allocate at least one wavelength to each neighboring edge). In practice, this is a reasonable assumption, since, on average, the total number of available wavelengths (i.e., transponders) on each node is $O(10)$ times larger than the node's degree; i.e., $\sigma_{avg}(v) \geq O(10) \times d_{avg}(v)$.[4]

Gain factor. For any TE problem we can define the gain R_{TE} of moving from a static to programmable WAN by $R_{TE} = \frac{TE(G_\Lambda, D)}{TE(G_*, D)}$, where $TE(G_*, D)$ is the solution to the TE problem over the static topology G_* for demand matrix D, and $TE(G_\Lambda, D)$ is the solution for the TE using the programmable wavelength allocation Λ obtained by reprogramming the wavelengths with respect to the demand matrix D.[5]

As we are mostly interested with optimizing throughput based TE, we will focus on linear-throughput maximization (i.e., maximum multi-commodity flow) which we denote by LTE and it's corresponding gain by R.

Theorem 1. *Given a physical graph G with $\sigma(v) \geq d(v)$, $\forall v \in V$, we have $1 \leq R \leq 2\Delta - 1$, where Δ is the maximum degree in G. This bound holds for any graph, under any edge failure scenario, and for any linear throughput maximization objective function (e.g., multi-commodity flow optimization, max-flow [20]).*

See Appendix A.1 for the proof, also providing matching lower bounds. The intuition for the $O(\Delta)$ upper bound is as follows. From a node's perspective, borrowing wavelengths from all of its adjacent edges and lending them to

[1] $\max c_1 f_1 + c_2 f_2 + c_3 f_3 + ...$, with constant c_i and flow size f_i. [2] For example, $\mu(e) = 120$ wavelengths for QPSK modulation with 37.5 GHz spacing. [3] Recent work shows this assumption is not optimal; there are gains in building a unidirectional WAN using unidirectional transponders [57], but we leave this discussion for future work. [4] For confidentiality reasons, we cannot share the exact number. [5] Again note that G_* and G_Λ share the same physical graph G; the difference lies in the ability to reallocate wavelengths.

one edge is, at most, bounded by the node's degree. Hence, with programmability, the maximum gain is in the order of the maximum degree in the graph. Furthermore, fiber cuts can be understood as new traffic matrices: already being resilient to a single fiber cut allows reprogrammability to achieve gains of $\Omega(\Delta)$. Moreover, if every fiber has to be lit by at least one wavelength at all times, even under failures, the reconfiguration gain is bounded by Δ. We defer details to Appendix A.1. Lastly, deviating from a pure throughput perspective can increase the gains to $\omega(\Delta)$, see A.5.

4.2 Gain of Realistic Topologies

In this section, we propose a simple heuristic to estimate throughput gains and test it on two realistic topologies. To do so, one could consider using the bisection bandwidth as a commonly used metric to compare throughput of G_* and G_Λ. However, bisection bandwidth forces partitioning the network into two parts of equal size. To go beyond this metric and cover all possible traffic patterns across a topology, we consider all $\Theta(2^n)$ partitions of different sizes. For each partition, the gain factor of programmable topologies is computed. For example in Figure 2, the partition corresponding to the traffic demand is $\{u,v\}$ in one set and $\{x,w\}$ in the other set. In 2(b), the throughput across this partition is 2, while the throughput across the same partition is 4 in 2(c), hence a gain factor of $2=\Delta$.

We compute the gain factors of programmable topologies for every partition for different topologies, obtaining polynomial runtime for every partition instance, by formulating it as an integer min-cost flow problem with a virtual super-source/destination. We use two realistic topologies: the fiber path of Internet2 topology [27] as well as a large service provider's backbone topology (we call this topology Yellow3). Figure 7 shows a CDF of gain factors for all partitions for realistic topologies. Interestingly, even though the exact topologies of Yellow3 and Internet2 are different, they are structurally similar enough to lead to the same average throughput gain factor of 1.4 (i.e., 40% gain). In Yellow3, the largest gain factor is obtained by a partition between the east and the west coast of the US. This result agrees with the findings of previous work [31, 60] but we obtained it with a standard computation without having to implement the TP itself. We are aware that our dataset of two topologies is too small to extrapolate the usefulness of our method for all realistic topologies and plan to perform further studies in future work.

5 Adopting Programmable Topologies in Practice

The previous sections suggest the advantages of a programmable graph that can adapt to changes in traffic demand (caused by failures, bulk transfers, or simple changes in the traffic matrix). Before we embrace programmable topologies, however, we need to answer another question: how can we transform productionized TE algorithms (such as B4 [29] or SWAN [25]) developed for a static WAN topology into an algorithm that can take advantage of programmability in its topology?

Programming topologies is hard. To answer this question, we first show that computing the optimal wavelength allocation for a given traffic matrix is NP-hard, even if the TE is polynomial for a static WAN. (Proof: Appendix A.2.)

Theorem 2. *Given a physical graph G with $\sigma(v) \geq d(v), \forall v \in V$, and a traffic matrix D, computing an optimal wavelength allocation Λ for a linear throughput maximization LTE(G_Λ,D) is NP-hard.*

Close approximations are tractable. Assuming P $\neq$ NP, we need to resort to non-optimal techniques for polynomial runtimes. The good news is that in linear TEs, finding constant-factor approximations is tractable. In Corollary 4, we show a simple rounding technique to obtain an $x/(x-1)$-approximation when $\sigma(v) \geq x \cdot d(v), \forall x \in \mathbb{N}_{>1}$.

Joining TE and TP. Before getting to the rounding technique, let us first iterate the adoption options. One approach is to write a cross-layer TE optimization algorithm while embedding concepts such as wavelength, transponder, and optical segments in the new TE (i.e., OWAN [30, 31]). This approach is powerful in enabling any TE to run on programmable optics. However, it requires writing a new optimization algorithm for each TE formulation. This is a tedious task because it breaks the layering between the IP and physical layers by cluttering the TE algorithm with concepts that are specific to the physical layer, making it hard to guarantee that the new TE is able to honor the same objectives as the old TE.

Migration issues. Moreover, it needs to be specified how to consistently update the network status to both the new wavelength allocation and IP routes. Just like IP route changes cannot be rolled out instantaneously, changing wavelength allocations takes time, leaving the network in a temporarily less powerful state. We provide further insights in §7.

Providing an abstraction. Next, we ask if it is possible to enable topology programmability into the static topology TE without having to rewrite the formulation itself. To answer this question, we introduce *Reflow graphs* as an abstraction that captures programmability in the physical layer for TE algorithms. With this abstraction we transform the static graph into one with a programmable topology. Once the Reflow graph is built, the unmodified TE formulation is executed on it. After this step, the solution on the Reflow graph is mapped onto a wavelength allocation on the original graph.

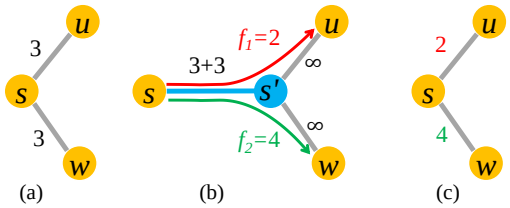


Figure 6: (a) Solving max-flow for s in the static topology of Figure 3(a) involves edges with fixed capacities of 3. (b) Adding dummy node s' and dummy edge (s,s'). Now, solving max-flow on this graph finds f_1 and f_2 as the flows over the two edges, with $f_1+f_2 \leq 3+3$. This graph represents the intuition for wavelength programmability. (c) Translating the solution on (b) back to the original graph with $f_1=2$ and $f_2=4$.

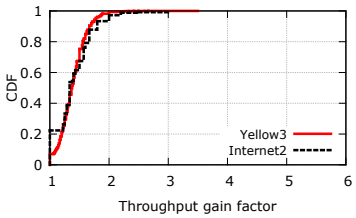


Figure 7: CDF of reconfiguration gain factors over all partitions for two real topologies: Yellow3 (renamed to preserve anonymity) and Internet2 [5]. The average gain factor for the evaluated real topologies is 1.4 (i.e., 40% more throughput), agreeing with [31, 60].

This way, the Reflow graph enables porting TE algorithms to leverage the programmability of the physical layer without having to re-implement the IP layer TE. By abstracting optical programmability into the IP layer, Reflow also enables cross-layer optimization for consistent update schemes. We explain in §6 how the Reflow graph is built, evaluating it in §7.

6 Reflow Graph Abstraction

In this section, we present an abstraction which enables the operator to take advantage of programmable topologies without having to change the TE formulation. Towards this goal, we first present an intuition for our Reflow abstraction (§6.1). Then, we outline the challenges to capture programmability in the physical layer for any TE algorithm (§6.2). Next, we explain the key idea of Reflow graph that enables us to port cross-layer optimization formulations into a standard flow problem (§6.3): we show how the unmodified TE algorithm can be executed on a Reflow graph and how the solution is mapped to a wavelength allocation on the original graph. Lastly, we demonstrate in §6.4 how Reflow enables consistent cross-layer updates. Putting it all together, Reflow graph enables porting TE algorithms to leverage the programmability of the physical layer without having to re-implement the IP layer TE.

6.1 Intuition for Reflow

The intuition for a Reflow graph is illustrated in Figure 6, where we assume infinite channels for simplicity. Finding the max-flow in Figure 6(a) involves allocating flows to each edge while respecting the capacity constraint on each edge. In a programmable WAN topology, edge capacities can be dynamically allocated as long as the total available capacity at each node (i.e., the amount of available transponders) is not violated. Figure 6(b) illustrates an augmented graph where a dummy node s' and dummy edge (s,s') are added to the original graph shown in Figure 6(a). The capacity of edge (s,s') is set to the total amount of available capacity in s, and the capacity of other edges in the network is set to ∞. Figure 6(b) solves the max-flow problem. It shows the optimal amount of flow f_1 and f_2 on edges (s',u) and (s',w), corresponding to the dynamic allocation of edge capacities for (s,u) and (s,w), respectively, as shown in the original graph in Figure 6(c).

6.2 Three challenges for Reflow

The naïve abstraction in §6.1 lacks three practical considerations, which we illustrate below and in Figure 8.

1) Edge capacities are limited. First, in practice, edge capacities ($\mu(e)$) are not ∞. Rather, they reflect the total number of wavelengths a fiber can carry depending on the modulation technology.

2) Receiving traffic is not free. Second, allocating a wavelength on an edge requires a transponder on both sending and receiving sides. Consider Figure 8(a) and assume node s has 6 transponders in total, and the number of available wavelengths on any edge, $\mu(e)$, is also 6. Now assume the solver has allocated $f_1=2$ wavelengths from s to u and $f_2=4$ wavelengths from s to w. If we don't enforce bidirectionality, it is possible for the solver to decide to allocate $f_3=4$ wavelengths from u to s and $f_4=2$ wavelengths from w to s, as shown in Figure 8(b). However, this is an infeasible wavelength allocation because when the results are translated back to the original graph, the number of wavelengths on each edge will be the maximum number of wavelengths in each direction (as shown in 8(c)). In this case, there would be 4 wavelengths on (s,u) and 4 wavelengths on (s,w), adding up to a total of 8 transponders on s, as opposed to the available number of transponders (6).

3) Wavelengths are bidirectional. Third, although fiber is unidirectional, today's transponders enforce bidirectionality: programming a wavelength between nodes s and u requires the reverse path to be programmed from u to s. To accommodate this, we introduce *dummy flows* as shown in Figure 8(d). Intuitively, these dummy flows *flip* the problem: the allocation of dummy flows *removes* capacity from the edges.

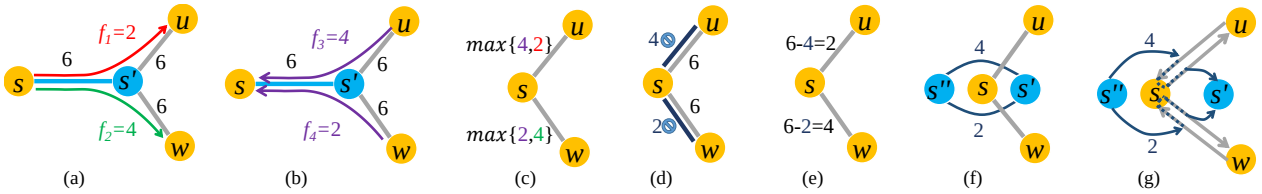


Figure 8: For 8(a), we add additional demands of $u \to s : 4$ and $w \to s : 2$ in Figure 8(b). These demands can all be satisfied as the capacity on the edges is 6. However, translating back to a wavelength allocation results in 8(c), which requires 8 transponders on s. In order to restrict s to 6 transponders, we *flip* the problem. Instead of allocating capacity, we reduce maximum edge capacity by blocking, as shown in 8(d); we add dummy flows (in blue) of combined size 6, s.t. the remaining capacity is $2+4=6 \leq \sigma(v)$ as well. Now, the transponder number $\sigma(s)=6$ is respected, resulting in the wavelength allocation in 8(e). It remains to encapsulate the dummy flows from 8(d) into the inputs of the TE, as conceptualized in 8(f). We zoom into 8(f) in 8(g), showing how a single dummy flow can block both directions of a bidirectional edge connection. A further example can be found in Figure 15 in the Appendix.

For example, to force the assignment of 2 wavelengths from u to s in 8(b), we block 4 out of the 6 possible wavelengths, resulting in 2 remaining wavelengths. Figures 8(d) and (e) show how blocking 4 wavelengths from u to s, and 2 wavelengths from w to s, results in the feasible allocation.

Flows enable Reflow. The core idea of Reflow abstraction is encapsulating these dummy flows as additional inputs to the TE. To this end, for each node s in the physical graph, we add 2 dummy nodes, s'' and s', as shown in Figure 8(f): here, s'' has to distribute 6 dummy flows, added to the demand matrix over its two edges. To idea how to ensure blocking both edge directions of an edge is shown in Figure 8(g), where we zoom into Figure 8(f): a dummy flow has to traverse both directions of an edge, ensuring that the remaining wavelength capacity is identical for both directions.

6.3 Building the Reflow abstraction

Using the ideas of § 6.2, we show that the network operator does not need to modify the TE formulation at all, but can still fully benefit from reprogrammability by just augmenting the TE inputs: the physical graph and the demand matrix.

Theorem 3. *Let G be a physical graph with demand matrix D. Let TE be either a fractional or integral LTE. There is a physical graph $G_{\approx}$ with demand matrix $D_{\approx}$ such that the solution of $TE(G_{\approx}, D_{\approx})$ can be translated in polynomial time to the optimal fractional/integral wavelength allocation which maximizes $TE(G,D)$.*

We note that the construction described in §6.2 and Figure 8 does not list all details of the Reflow graph, but rather gives a high level intuition of the construction. We defer the specific proof details to Appendix A.3 due to space constraints.

Obtaining integral wavelengths. The Reflow abstraction with integral flows provides an optimal solution to the NP-hard problem (Theorem 2) of cross-layer optimization. The key enabler are integral dummy flows, as their optimization presents an NP-hard problem as well [18]. As such, the network operator can obtain optimal results by trading in additional computation time. While fractional dummy flow sizes can be computed in polynomial time, the resulting wavelength allocations will be fractional as well. In other words, the wavelengths cannot be exactly allocated. Reflow can tackle this issue by holding back transponders to *round up* fractional wavelength allocations. Recall from §4 that nodes in practice have around 10 times as many transponders as neighbors. When the optimal integral solution is not possible, this provisioning enables bounding the throughput loss by $10/(10-1)$, i.e., at most by 11%. This throughput loss will diminish when the number of transponders increases. As such, the Reflow abstraction also enables the network operator to perform cross-layer optimization in polynomial time, at a small throughput loss.

Corollary 4. *Let $x \in \mathbb{N}_{>1}$ s.t. $\forall v \in V : d(v) \geq x \cdot \sigma(v)$. Using a polynomial-time rounding scheme to obtain integral wavelengths in Theorem 3 yields an approximation ratio for the objective function of $x/(x-1)$.*

6.4 Consistent Updates in Reflow

After the new wavelength allocations have been computed, both the wavelengths and the IP traffic must be migrated to the new network state. Even though we show in §7 that wavelength allocations can be changed in less than 400ms, packets sent across those temporarily unavailable wavelengths will be dropped, with the additional challenge of synchronizing optical and IP layer changes. The toolkit of consistent network updates [17], designed to deal with such transient congestion and packet drops, cannot yet handle a cross-layer migration of wavelengths and traffic flows. The current state-of-the-art method is to carefully extend consistency schemes such as Dionysus [32] to the specified cross-layer scheme, as in OWAN [25].

Reflow enables consistent update schemes in optics. The advantage of Reflow is that existing consistent update schemes do *not* need to be modified. A key enabler is that Reflow expresses transponder assignments as flow allocations, i.e., removing any optical component. As such, consistent network updates are performed on the Reflow graph itself, where a migration of dummy flows corresponds to shifting transponder-to-edge mappings. The old and new network states are specified as $G^{\text{old}}_{\approx}$ and $G^{\text{new}}_{\approx}$ with both dummy flow and IP traffic allocations. When both states are provided as an input to consistent flow migration schemes, intermediate network states are computed [16, 58], corresponding to consistent cross-layer network updates.

Both wavelength allocations and IP traffic route changes are covered, where $1)$ changing the path of a dummy flow maps directly onto changing a transponder assignment and changing $2)$ the path of IP traffic in Reflow maps onto route changes in the IP layer. In other words, a single network update can contain both optical and IP layer changes, enabling Reflow to provide an abstraction for optimal (e.g., minimum schedule length) cross-layer migration.

6.5 Coverage of Reflow

So far, we have showed how Reflow can express linear throughput maximization objectives, e.g., max-flow or MCF. However, the quality of an abstraction is also measured in its expressiveness. Reflow can also be used in the context of concurrent sharing objectives, flow priorities, hierarchical bandwidths, and path length properties, with partial support of hop-count optimization, k-shortest path routing, and max-min fairness. We provide further details in Appendix A.6.

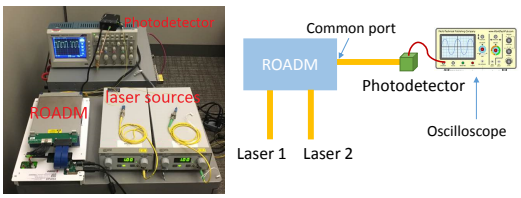
7 Evaluations

As the practical throughput benefits of programmable topologies, both in the data center and the WAN, have been vastly analyzed recently [6, 10, 15, 24, 28, 30, 31, 42, 43, 60], with theoretical bounds provided in §4, we focus our attention to the practical considerations of using the Reflow abstraction - what are the impacts of reprogrammability on the IP layer (§7.1), and finally what are the runtime implications of using Reflow (§7.2).

7.1 Reflow Abstraction Assumptions

Abstraction-based topology programming depends on two fundamental assumptions: (i) the time it takes to reprogram the topology is short, allowing the TE engine to use consistent updates to reroute traffic while wavelengths are being programmed; (ii) reprogramming wavelengths in the topology is not destructive to the IP traffic of other wavelengths in the topology. Together, these assumptions enable the TE engine to handle wavelength reprogramming similar to the IP layer's flow reprogramming.

In this section, we use a testbed to verify the above assumptions. Our testbed consists of four Finisar evaluation boards, each with 40 add/drop ports supporting data rates of 10, 40, 100, and 400 Gbps [2]. To generate traffic with different wavelengths, we use 10 Gbps tunable DWDM transceivers from Finisar [7] plugged into a 7050s Arista switch. We discovered that Finisar's tunable transceivers do not work out of the box when plugged into Arista or Cisco switches because of a mismatch between the two vendors on where to read the description of the physical device. This problem is visible when running the command `show idprom transceiver ethernetX` on the switch. Arista requires byte 0 on page A0 in the transceivers to be 0x0b, whereas Finisar writes 0x03 on that location. We manually modified these bytes to make the switch recognize the transceivers. We also found that the tunable DWDM transceivers manufactured by FS.COM [3] are already compatible with Arista.



(a) Physical setup (b) Logical setup

Figure 9: Programming latency experiment.

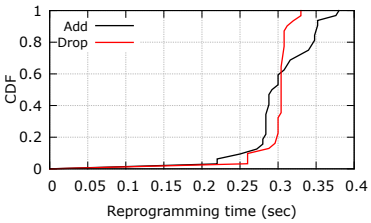
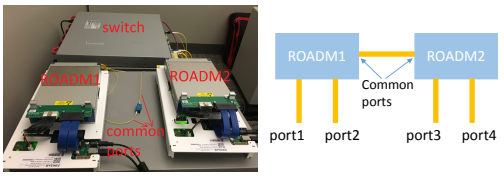


Figure 10: CDF of the time it takes to reprogram the ROADM to add/drop wavelengths.

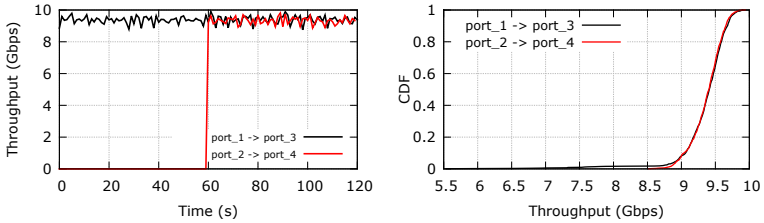
Reprogramming latency. To measure the reprogramming latency of the ROADMs, we use the setup shown in Figure 9; two laser sources are connected to the ROADM where they are multiplexed into the common port. We then use a python-based controller to reprogram the ROADM by adding and dropping the light from Laser 1 and 2 to/from the common port. To accurately measure programming latency, we connect a fiber-coupled photodetector to the common port and measure instantaneous light intensity at the photodetector using an oscilloscope. We repeat this experiment 100 times and plot the CDF of the time it takes to program the ROADM to add and drop wavelengths; this is depicted in

Figure 10. Our results show that the programming time is less than 400 milliseconds. This experiment micro-benchmarks the ROADM's programming time, but two components might require additional time. First, commodity transceivers take a few seconds to lock to the signal and start decoding bits. This additional time is not fundamental, however, and prior work has been able to mitigate it [47]. Second, amplifiers along the path might require extra time to adjust their amplification factors. We admit that we have not fully explored this limitation. Our conversations with field experts suggest that we can use machine learning and statistical data to infer the required settings on the amps prior to reprogramming. We leave exploring this line of thought to future work.



(a) Physical setup (b) Logical setup

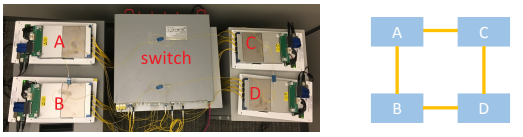
Figure 11: Throughput experiment.



(a) Adding a new wavelength. (b) CDF of throughput.

Figure 12: Adding/dropping wavelengths does not impact the throughput of other wavelengths sharing the same fiber.

Impact of reprogramming on IP traffic. Next, we verify that adding and dropping wavelengths to/from the common port does not negatively impact the throughput of ongoing flows. We use the setup shown in Figure 11 where two ROADMs are connected through their common ports. We program $port_1$ to send 10 Gbps of traffic to $port_3$. After 60 seconds, we program $port_2$ to send 10 Gbps of traffic to $port_4$ using another wavelength while sharing the common port with the first flow. Figure 12(a) shows that both flows are capable of achieving 10 Gbps and adding a new wavelength does not negatively impact the throughput of the on-going flow between $port_1$ and $port_3$. We repeat this experiment 1000 times and measure the throughput of both flows (during the time they are both active). Figure 12(b) shows the CDF of throughput for both flows is overlapping, suggesting that programming wavelengths does not have a destructive effect.



(a) Physical setup (b) Logical setup

Figure 13: Fiber cut experiment.

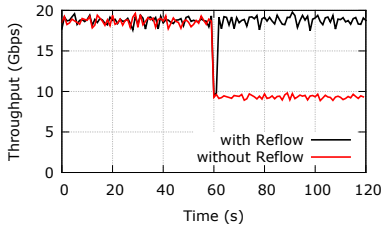


Figure 14: Enabling Reflow graph can mitigate the impact of fiber cut by programming idle wavelengths based on the current traffic demand.

Putting it all together. Finally, we show the power of topology programmability in practice by connecting four ROADMs to create a rectangular topology as shown in Figure 13. The four edges in 13(b) represent the common ports; for clarity of presentation, we have omitted the depiction of add/drop ports. We start the experiment by generating 20 Gbps of traffic between nodes A and C using two wavelengths. One wavelength takes the $A \rightarrow C$ direct edge and another takes the $A \rightarrow B \rightarrow D \rightarrow C$ path. This way, the topology can support the entire 20 Gbps demand, as shown in Figure 14. Next, we simulate a fiber cut by manually disconnecting the fiber between A and B. This will cause a loss of capacity and throughput is dropped to 10 Gbps. However, our controller detects this and drops the wavelength on the $A \rightarrow B$ edge and adds it to the $A \rightarrow C$ edge; this allows the throughput to be restored to 20 Gbps (black curve in Figure 14). Without such programmability, the throughput would remain at 10 Gbps (red curve in Figure 14).

7.2 Proof of Concept: Runtime simulations for Reflow

Lastly, we perform a small runtime evaluation of the Reflow abstraction. We use the Yellow3 topology and generate traffic matrices (TMs) with different levels of sparseness (10 random TMs were generated for each level), and use the LTE optimization goal and measure (1) the maximum throughput achievable, and (2) the runtime it takes to solve the LP. We compare Reflow to three baselines: *LP* which represents a cross-layer, fractional, solution to the optimization problem, and *MIP* representing an integral (and hence wavelength translatable) solution to the optimization problem.

We found that we do not need to resort to the fractional approximation mentioned in §6.3 for our simulation size and hence use the direct integral wavelength assignment. For completeness we also evaluate another baseline, *Static*, which involves no topology adaptation. We observe that Reflow (Max $2.89s$, Avg $1.61s$) incurs little computational overhead over the *Static* (Max $0.87s$, Avg $0.42s$) baseline and runs slightly faster than the cross-layer *MIP* solution (Max $4.64s$, Avg $2.33s$). This change in running time can be attributed to the little overhead Reflow adds to the optimized network graph and commodities, whereas a standard formulation needs to jointly optimize the physical layer alongside the TE. Moreover, in our simulations Reflow (and *MIP*) obtain $> 99.9\%$ of the throughput of (the not wavelength-translatable) *LP* (Max $0.87s$, Avg $0.34s$). We are aware of the limitations of our small simulation: more complex traffic patterns and specifically optimized formulations could easily change the relative outcome of our tests, which we leave to future work. Notwithstanding, our results show that Reflow is easily usable and correct in practice, with competitive computation times and realistic deployment properties, as evaluated in §7.1.

8 Related Work

Our work builds on several lines of related research. The work most related to ours is by Jin et al. [31] on cross-layer optimization between IP and optical layers which shows optical reconfiguration can provide latency gains for deadline driven bulk transfers. However, they use heuristic algorithms for scheduling and reconfiguration of the optical devices, without providing mathematical analysis or theoretical guarantees. In this work, we provide theoretical bounds and also propose a simple method to estimate throughput gains. A follow-up work studies the makespan of scheduling algorithms in programmable topologies, where all transfers are restricted to be across single-hop paths [30]. The authors consider both online and offline variants, analyzing competitive ratios of scheduling problems. Both works optimize the IP and optical layers together with a cross-layer algorithm whereas in this paper, we provide a mechanism to use the current TEs without having to rewrite a cross-layer TE. Furthermore, we provide polynomial approximation guarantees.

We are inspired by the large body of work on reconfigurable topologies in data center networks as well. Prior work shows the benefits of reconfigurable topologies in data center networks by adding wireless/optical edges to the electrical topology [14, 33, 38, 39, 48, 59], or by creating all optical data center interconnects [10, 11, 22, 41, 42]. The main difference between the WAN and data centers is the geographical scale of the WAN. While the underlying assumption in a reconfigurable data center is that optical links are either connected via a single optical switch or have line-of-sight, the sheer scale of the WAN creates physical limitations on the fiber paths and the physical topology itself making it impossible to directly adopt the data center proposals.

Similarly, Expander-based [34, 53, 54, 56] topologies are becoming popular, but realizing such topologies is extremely challenging in the WAN. Creating an expander-based topology requires all nodes to have equal degree, an assumption that does not hold in the WAN. In fact, a recent study of the US long-haul fiber-optic infrastructure shows nodes' degrees vary from 1 to 5 [13] and Internet2's physical infrastructure shows node degrees between 2 and 5 [27]. Another limiting factor for these approaches is the total number of nodes in the graph: a WAN topology has an order of 10s of nodes, whereas expander-based proposals show gains for networks with large number of nodes. Even if we build a physical topology with fixed node degrees, we are limited by the placement and cost of the optical fibers – cabling suggestions which leverage the close proximity of the racks and bundle the cables together, like those suggested in [54, 56], are not achievable in the WAN. A generalization of expanders, called conductors [55], may be used to construct such mixed-degree static topologies, but it remains to be seen if there would be any significant gains for such small networks or if such topologies could be built in a WAN-cabling friendly way, we leave such investigation for future work.

9 Conclusion

In this paper, we take the first steps to a practical deployment of a programmable physical layer in the WAN. Towards this goal, we make two new contributions. First, we provide bounds on throughput gains of topology programmability for any graph under any traffic change or fiber cut scenario, propose a simple heuristic to estimate practical throughput gains, and present opportunities for programmability based on traffic traces. Second, we present the Reflow abstraction to enable IP-layer TE algorithms to perform cross-layer optimization. We show the expressiveness of Reflow captures a wide spectrum of TE objectives. In addition, the Reflow abstraction matches the throughput of a cross-layer TE optimization, both in theory and practice. We evaluate our abstraction using a testbed and simulations. The testbed results show that programming wavelengths do not impact the throughput of other wavelengths and they enable a quick recovery of fiber cuts. Our simulation results show that Reflow achieves competitive runtimes to cross-layer TE.

We believe that Reflow is a first step towards a greater objective, namely breaking out of the black box modeling of optical networks: powerful optical network components deserve to benefit from well-developed optimization toolkits, especially since the current WAN deployments are already all optical.

References

[1] ADVA ROADMs. http://www.advaoptical.com/en/products/technology/roadm.aspx.

[2] Dual wavelength selective switch (wss). https://www.finisar.com/roadms-wavelength-management/fws0120bscfal.

[3] Dwdm sfp+. http://www.fs.com/c/dwdm-sfp-plus-66?dwdm-tunable=20807.

[4] Infinera introduces flexible grid 500g super-channel roadm. http://www.gazettabyte.com/home/2014/3/14/infinera-introduces-flexible-grid-500g-super-channel-roadm.html.

[5] Internet2. http://www.internet2.edu.

[6] Sedona systems. http://sedonasys.com/.

[7] Tunable dwdm transceivers. https://www.finisar.com/optical-transceivers/ftlx6872mcc.

[8] C. Avin, K. Mondal, and S. Schmid. Demand-aware network designs of bounded degree. In *DISC*, 2017.

[9] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls, RFC:2702, 1999.

[10] N. H. Azimi, Z. A. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer. Firefly: a reconfigurable wireless data center fabric using free-space optics. In *SIGCOMM*, pages 319–330. ACM, 2014.

[11] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong. Enabling wide-spread communications on optical fabric with megaswitch. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 577–593, Boston, MA, 2017. USENIX Association.

[12] A. L. Chiu, G. Choudhury, G. Clapp, R. Doverspike, M. Feuer, J. W. Gannett, J. Jackel, G. T. Kim, J. G. Klincewicz, T. J. Kwon, G. Li, P. Magill, J. M. Simmons, R. A. Skoog, J. Strand, A. V. Lehmen, B. J. Wilson, S. L. Woodward, and D. Xu. Architectures and protocols for capacity efficient, highly dynamic and highly resilient core networks (invited). *IEEE/OSA Journal of Optical Communications and Networking*, 4(1):1–14, January 2012.

[13] R. Durairajan, P. Barford, J. Sommers, and W. Willinger. Intertubes: A study of the US long-haul fiber-optic infrastructure. In *SIGCOMM*, pages 565–578. ACM, 2015.

[14] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*, pages 339–350. ACM, 2010.

[15] M. Filer, J. Gaudette, M. Ghobadi, R. Mahajan, T. Issenhuth, B. Klinkers, and J. Cox. Elastic optical networking in the microsoft cloud (invited). *J. Opt. Commun. Netw.*, 8(7):A45–A54, Jul 2016.

[16] K.-T. Foerster. On the consistent migration of unsplittable flows: Upper and lower complexity bounds. In *IEEE NCA*, 2017.

[17] K.-T. Foerster, S. Schmid, and S. Vissicchio. Survey of consistent network updates. *CoRR*, abs/1609.02305v2, 2018.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996.

[20] S. Gass and A. Assad. *An Annotated Timeline of Operations Research: An Informal History*. An Annotated Timeline of Operations Research: An Informal History. Springer, 2005.

[21] O. Gerstel. Control architectures for multi-layer networking: Distributed, centralized, or something in between? In *2015 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–16, March 2015.

[22] M. Ghobadi, R. Mahajan, A. Phanishayee, N. R. Devanur, J. Kulkarni, G. Ranade, P. Blanche, H. Rastegarfar, M. Glick, and D. C. Kilper. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *SIGCOMM*, 2016.

[23] J. He and J. Rexford. Toward internet-wide multipath routing. *IEEE Network*, 22(2):16–21, 2008.

[24] T. Hofmeister, V. Vusirikala, and B. Koley. How can flexibility on the line side best be exploited on the client side? In *Optical Fiber Communication Conference*, page W4G.4. Optical Society of America, 2016.

[25] C. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *SIGCOMM*, pages 15–26. ACM, 2013.

[26] T. Ibaraki and S. Poljak. Weak three-linking in eulerian digraphs. *SIAM J. Discrete Math.*, 4(1):84–98, 1991.

[27] Internet2. Internet2's network infrastructure topology. https://www.internet2.edu/media/medialibrary/2017/05/17/I2-Network-Infrastructure-Topology-Layer_2logos-201705_2brmRD6.pdf.

[28] D. J. Ives, P. Bayvel, and S. J. Savory. Routing, modulation, spectrum and launch power assignment to maximize the traffic throughput of a nonlinear optical mesh network. *Photonic Netw. Commun.*, 29(3):244–256, June 2015.

[29] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: experience with a globally-deployed software defined wan. In *SIGCOMM*, pages 3–14. ACM, 2013.

[30] S. Jia, X. Jin, G. Ghasemiesfeh, J. Ding, and J. Gao. Competitive analysis for online scheduling in software-defined optical wan. In *INFOCOM*, 2017.

[31] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford. Optimizing bulk transfers with software-defined optical WAN. In *SIGCOMM*, 2016.

[32] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer. Dynamic scheduling

of network updates. In *SIGCOMM*, pages 539–550. ACM, 2014.

[33] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. In *HotNets*. ACM SIGCOMM, 2009.

[34] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla. Beyond fat-trees without antennae, mirrors, and disco-balls. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, pages 64–70, 2016.

[35] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 1–14, New York, NY, USA, 2015. ACM.

[36] A. Kvalbein, C. Dovrolis, and C. Muthu. Multipath load-adaptive routing: putting the emphasis on robustness and simplicity. In *ICNP*, pages 203–212. IEEE Computer Society, 2009.

[37] Y. Lee, Y. Seok, Y. Choi, and C. Kim. A constrained multipath traffic engineering scheme for MPLS networks. In *ICC*, pages 2431–2436. IEEE, 2002.

[38] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter. Circuit switching under the radar with REACToR. In *NSDI*, pages 1–15, 2014.

[39] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky, G. Porter, and A. C. Snoeren. Scheduling techniques for hybrid circuit/packet networks. In *CoNEXT*, pages 41:1–41:13. ACM, 2015.

[40] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter. Traffic engineering with forward fault correction. In *SIGCOMM*, pages 527–538. ACM, 2014.

[41] Y. J. Liu, P. X. Gao, B. Wong, and S. Keshav. Quartz: a new design element for low-latency dcns. In *SIGCOMM*, pages 283–294. ACM, 2014.

[42] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter. RotorNet: A Scalable, Low-complexity, Optical Datacenter Network. In *SIGCOMM*, 2017.

[43] A. Morea and A. Paparella. Cost and algorithm complexity of elastic optical networks. In *Optical Fiber Communication Conference*, page M2K.4. Optical Society of America, 2016.

[44] D. Nace and M. Pióro. Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial. *IEEE Communications Surveys and Tutorials*, 10(1-4):5–17, 2008.

[45] S. Oda, M. Miyabe, S. Yoshida, T. Katagiri, Y. Aoki, J. C. Rasmussen, M. Birk, and K. Tse. Demonstration of an autonomous software controlled living optical network that eliminates the need for pre-planning. In *Optical Fiber Communication Conference*, page W2A.44. Optical Society of America, 2016.

[46] P. Papanikolaou, K. Christodoulopoulos, and E. M. Varvarigos. Joint multilayer planning of survivable elastic optical networks. In *Optical Fiber Communication Conference*, page M2K.3. Optical Society of America, 2016.

[47] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat. Integrating microsecond circuit switching into the data center. pages 447–458, 2013.

[48] G. Porter, R. D. Strong, N. Farrington, A. Forencich, P. Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat. Integrating microsecond circuit switching into the data center. In *SIGCOMM*, pages 447–458. ACM, 2013.

[49] H. Räcke. Survey on oblivious routing strategies. In *CiE*, volume 5635 of *LNCS*, pages 419–429. Springer, 2009.

[50] P. Roorda and B. Collings. Evolution to colorless and directionless roadm architectures. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference*, page NWE2. Optical Society of America, 2008.

[51] A. Sahara, Y. Tsukishima, T. Takahashi, Y. Okubo, K. Yamada, K. Matsuda, and A. Takada. Demonstration of colorless and directed/directionless roadms in router network. In *Optical Fiber Communication Conference and National Fiber Optic Engineers Conference*, page NMD2. Optical Society of America, 2009.

[52] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, Apr. 1990.

[53] A. Singla. Fat-free topologies. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, pages 64–70, 2016.

[54] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *NSDI*, 2012.

[55] H. Sun. Expander graphs in computer science. `https://resources.mpi-inf.mpg.de/departments/d1/teaching/ws10/EG/script.pdf`.

[56] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira. Xpander: Towards optimal-performance datacenters. In *CoNEXT*, 2016.

[57] W. Zhang and B. G. Bathula. Breaking the bidirectional link paradigm. In *Optical Fiber Communication Conference*, page Th1E.3. Optical Society of America, 2016.

[58] J. Zheng, H. Xu, G. Chen, and H. Dai. Minimizing transient congestion during network update in data centers. In *ICNP*, pages 1–10. IEEE Computer Society, 2015.

[59] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. Mirror mirror on the ceiling: flexible wireless links for data centers. In *SIGCOMM*, pages 443–454. ACM, 2012.

[60] P. Zhu, J. Li, D. Wu, Z. Wu, Y. Tian, Y. Chen, D. Ge, X. Chen, Z. Chen, and Y. He. Demonstration of elastic optical network node with defragmentation functionality and sdn control. In *Optical Fiber Communication Conference*, page Th3I.3. Optical Society of America, 2016.

A Appendix

A.1 Gain is limited by maximum degree against an oblivious static configuration

Recall that a optimal traffic-oblivious wavelength allocation is defined by minimizing the gain factor of reconfiguration. As such, to prove Theorem 1, we have to implicitly deal with an adversary that may fix an optimal wavelength allocation – which is still static and may not be reprogrammed. However, the trick is that we do not even need to know the wavelength allocation that we compete against.

*Proof of **Theorem 1**.* Consider a wavelength allocation Λ_A, where each node v distributes its $\sigma(v)$ transponders evenly over all neighbors, possibly wasting up to $\sigma(v)-1 \leq \Delta-1$ transponders at each node to obtain identical numbers for all edges. Furthermore, even more transponders can be wasted due to μ restricting the number of wavelengths possible. Λ_A cannot be better than an optimal traffic-oblivious wavelength allocation for any linear objective function, as Λ_A is feasible, i.e., we have a lower bound on static performance

Next, consider a wavelength allocation Λ_B, which we obtain as follows: We begin with Λ_A, but multiply every transponder assignment of a node to a neighbor by $2\Delta-1$. Observe that Λ_B does not have to be feasible, but it clearly can satisfy any flows feasible in Λ_A. Furthermore, assume there is a feasible Λ_C which results in a better output for the objective function than on Λ_B: this leads to a contradiction as any flows feasible in Λ_C are also feasible in Λ_B.

Assume Λ_B has a gain of $X > 2\Delta-1$ compared to Λ_A for some TM and linear objective function. Then, we take all flows in Λ_B and divide their size by $2\Delta-1$, i.e., they are feasible in Λ_A, but due to $X > 2\Delta-1$ we obtain a contradiction.

The above arguments also holds under edge failures (fiber cuts), with the same wavelength allocations Λ_A and Λ_B: no matter the failure scenario, no reconfiguration can beat (the possibly infeasible) Λ_B, finishing the proof. □

The above proof argument holds analogously when wavelengths have different capacities. Furthermore, when every fiber has to have at least one wavelength allocated at all times (even under (temporary) fiber cuts), a reconfiguration cannot move all $\sigma(v)$ transponders of v to a single link, but at most $\sigma(v)-(\delta(v)-1)$. Hence, the maxiumum reconfiguration gain is then at most Δ. We provide matching lower bounds next.

Matching gain factor examples of $2\Delta-1$ resp. Δ. Consider a star physical graph with n nodes, where every node has $n-1$ available transponders. A static oblivious allocation (G_*) has to keep the network connected; hence, the center node in the star, v, needs to assign one wavelength to its adjacent edges. This allocation is optimal when the traffic matrix is all-to-all. Now assume the traffic matrix suddenly becomes concentrated between v and one of its neighbors w. By reprogramming the wavelengths on idle edges, we can assign $n-1$ wavelengths to edge (v,w) enabling a throughput of $n-1$, whereas G_* only enables a throughput of 1. By raising the individual transponder numbers to $(n-1)+(n-2)$, there will be at least one leaf node which is assigned at most one transponder by the center node v, w.l.o.g. w. Hence, a reconfiguration gain of $2n-3=2\Delta-1$ can be obtained, i.e., tight bounds for any $\Delta \in \mathbb{N}_{>1}$. Lastly, if all fibers have to stay lit with at least one wavelength in the previous example, even under failures, the gain is $1+(n-2)=\Delta$.

Resiliency against failures. Furthermore, a gain of $\Omega(\Delta)$, for any $\Delta \in \mathbb{N}$, can also be obtained without changing the traffic matrix, by just considering one edge failure and the assumption that the traffic in the static allocation has to be resilient against one edge failure, for example handled by FFC [40]. We start with a star graph again, but now replace every leaf with a triangle graph, where one of the three nodes is the center node of the star, and set $\sigma(v)=\Delta$ for all leaf nodes except the center with 2Δ transponders. We fix a TM that has Δ demands between each pair of neighbors, but the center node has no traffic demands. To visualize this TM, imagine all base edges of the triangles have Δ traffic demands. Using reprogrammability, we can satisfy all traffic demands, even when a single edge fails via rerouting along the center. A static allocation has to spread out the "protection" of the center node: in other words, to be safe against a single failure, there is a triangle base which can only route $O(1)$ units of traffic. Hence, the gain factor of reconfiguration is $\Omega(\Delta)$.

Every non-clique has a gain of ≥ 2. In data center networks, the underlying topology of nodes and cables are central to throughput considerations. To illustrate that the transponder assignment σ plays a large role in WANs, we now prove the following theorem showing that nearly every topology $G=(V,E)$ can benefit from reconfiguration:

Theorem 5. *For any graph with diameter at least two, there exists a pair of (1) two TMs & (2) transponder distribution σ s.t. the throughput of a fully programmable topology is twice of the static one* even *with optimal TE.*

Proof. Pick a node v with radius two, where $N_2(v)$ are the neighbors of v in distance two, and $N_1'(v)$ are the neighbors of v which also are adjacent to at least one node in $N_2(v)$. Assign an arbitrarily high number of transponders to v and all nodes in $N_2(v)$. For each node v in $N_1'(v)$, assign as many transponders as v has neighbors in $N_2(v) \cup \{v\}$. We now create two TMs, one just between v and all nodes of $N_1'(v)$, and one between all nodes $N_1'(v)$ and all nodes of $N_2(v)$,

Input:
Physical graph $G=(V,E,\sigma)$, demand matrix D
Output:
Reflow graph $G_{\approx}=(V_{\approx},E_{\approx})$, demand matrix $D_{\approx}$

1. $V_{\approx}=V$, $E_{\approx}=E$, $D_{\approx}=D$
2. $\forall v\in V$: $V_{\approx}=V_{\approx}\cup\{v'',v'\}$
3. $\forall e=(u,v)\in E$: $E_{\approx}=E_{\approx}\cup\{dummy\_path(v'',v')\}$
4. $\forall v\in V$: $D_{\approx}=D_{\approx}\cup\{dummy\_flow(v'',v')\}$

Algorithm 1: Reflow graph construction intuition.

Input:
Physical graph $G=(V,E,\sigma)$, demand matrix D
Output:
Flow $\mathcal{F}$, wavelength allocation $\Lambda: c(e)\ \forall e\in E$

1. Compute Reflow graph $G_{\approx}=(V_{\approx},E_{\approx})$ and $D_{\approx}$ using Algorithm 1
2. $\mathcal{F}_{\approx}\leftarrow$ solve TE on $G_{\approx}$
3. $\mathcal{F}\leftarrow\mathcal{F}_{\approx}$ minus dummy flows
4. Compute transponder allocation Λ for all $(u,v)=e\in E$:
   (a) Set $c(e)$ as maximum number of possible wavelengths on e minus the dummy flows on e

Algorithm 2: Reflow graph abstraction intuition.

both with arbitrarily high demands. Maximum throughput is obtained when just considering 1-hop flows. However, no matter what static topology is chosen, each transponder of $N_1'(v)$ has to be oriented either towards v or $N_2(v)$. Hence, using the pigeonhole principle, programmability can increase the throughput by a factor of at least two. □

A.2 Maximizing throughput is hard

*Proof of **Theorem** 2.* We will perform a reduction from the NP-hard problem 3-SAT [18], in the variant of every clause having three literals. For now, we do not assume $\sigma(v)\geq d(v)$, and consider directed integral transponders (receiving is always possible) on directed edges with unit wavelength capacity and splittable flows. For our reduction, we create nodes for all clauses $c_1,...,c_k$, each being a source with $\sigma=3$, connecting them to their corresponding literal nodes, where each literal node has $\sigma=1$ and is connected to a destination t. If each source wants to send a "small" flow to t, the optimal solution is straightforward: All clauses point their transponders to their literals, all literals to t. To force the literals to make a decision regarding true or false, we add a new source ($\sigma=1$) and destination ($\sigma=0$) with unit demand for each variable, connecting the source to both literals, and both literals to the destination. Now, solving maximum throughput under demand constraints is equivalent to solving 3-SAT. To make the edges bidirectional, we employ another standard technique, used in, e.g., [26, p.98]: If an edge (u,w) is just directed, we add a source-destination pair w,u with unit demand, and turn the edge bidirectional. The same technique can be used to enforce $\sigma(v)\geq\delta(v)$, by connecting a new destination w to each node v with transponder deficiency of $y\in\mathbb{N}$, generating a corresponding source on v with demand of $y+1$, with (v,w) having a (channel) capacity of $y+1$, analogously for bidirectional transponders. □

We note that for single-source or single-destination instances, where all wavelengths have identical capacity, computing an optimal wavelength allocation for max-flow is in P, e.g., by computation with an augmenting flow algorithm.

A.3 The Reflow graph abstraction

Before providing the proof of Theorem 3, we first consider the following high-level intuition of the Reflow abstraction, which consists of two parts, the graph augmentation and the wavelength computation.

Augmenting the physical graph. Algorithm 1 provides an intuition on the steps required to create a Reflow graph for a given physical graph G. In step 1, the Reflow graph is initialized with all the vertices and edges in G. In step 2, for every vertex v in G, two dummy vertices, v'' and v', are added to the Reflow graph. Then, for every edge $e=(u,v)$ in G, a dummy path $dummy\_path(v'',v')$, is created between v'' and v'. This dummy path crosses over e as shown in Figure 8(f). Finally, in step 4, a set of dummy flows, $dummy\_flow$, is created between the dummy nodes. The size of $dummy\_flow(v'',v')$ is chosen such that the non-blocked capacity for v is $\sigma(v)$. These dummy flows are added to the original demand matrix, with two constraints to make the Reflow construction more accessible: (1) dummy flows must take dummy paths, and (2) dummy flows must be satisfied.[6] After this step, the Reflow graph and new demand matrix can be used as the input to the optimization function.

Computing wavelength allocations. Algorithm 2 provides an intuition on how the Reflow graph abstraction is used to compute a wavelength allocation Λ on G. A flow assignment $\mathcal{F}$ is obtained by solving the TE optimization function on $G_{\approx}$ (i.e., the Reflow graph). Line 3 indicates that the flow assignment on G is simply $\mathcal{F}$ minus the flows between dummy nodes. Finally, in line 4, the number of wavelengths on each edge e is determined based on the capacity of e which is not blocked by dummy flows.

[6] Both constraints can be expressed as part of the Reflow graph and its demands. The technical details are explained in the proof of Theorem 3.

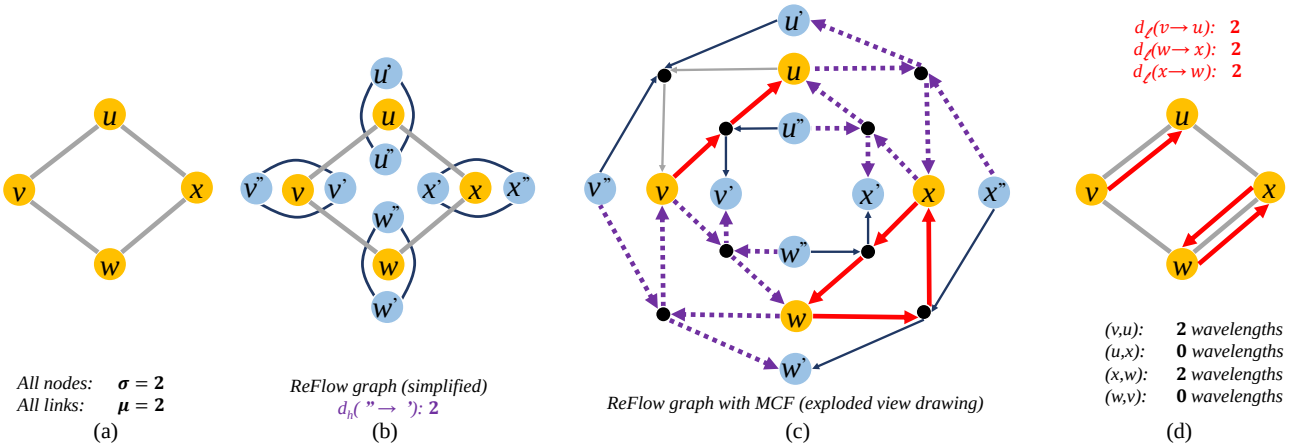


Figure 15: Illustration of a Reflow example, with path constraints still in effect for less clutter in the drawing (see Fig. 16). The physical graph G is depicted in 15(a), with its corresponding Reflow graph in 15(b). As every node has two transponders and four incident possible wavelengths, the dummy flow demands are $-2+4=2$ each. The exploded view drawing of the solution of the real demands d_ℓ (from 15(d)) in Reflow is shown in 15(c), which allocates all the dummy flows (in purple) and all the real traffic (in red). The solution in 15(c) is then translated to a wavelength allocation in 15(d), along with the real traffic.

Proof of ***Theorem*** *3.* Our proof will first assume flow priority classes and path constraints being available, expanding on our previous intuition provided, both assumptions will be lifted later during the proof construction.

Observe that when $\mu(e)$ is the number of wavelengths allocatable to an edge, after assigning a demand of $\max\left\{0,-\sigma(v)+\sum_{(v,\ldots)=e\in E}\mu(e)\right\}$ to the dummy flows originating in v'', the remaining bidirectional capacity for v is exactly $\sigma(v)$ once the flows are allocated, with each remaining capacity being integral, which is ensured due to them being in the highest priority class d_h. Path constraints ensure that the dummy flows cannot deviate from their paths. From this insight we obtain that a multi-commodity flow $\mathcal{F}'$ which is feasible for some wavelength allocation Λ_A of the physical graph G, is also feasible in the corresponding reflow graph G' along with meeting the demands of the dummy flows, combined denoted as $\mathcal{F}$, and vice versa. Hence, the optimal objective values both in the programmable physical graph and the corresponding Reflow graph, minus the values for the dummy flows, are identical. A small complete example is provided for better illustration in Figure 15 (the proof arguments are analogous for fractional transponders).

It remains to lift priorities and path constraints. We begin with the path constraints. First, observe that besides just going through the node v of the original physical graph, a dummy flow could also traverse the whole network. However, these extra paths form a loop, and can henceforth be canceled out for throughput maximization. It remains to fix the problem that a unit of dummy flow blocks one direction of an edge $e=(v,u)$, but not the other direction of an edge $e'=(v,w)$. To this end, we add another medium priority class d_m for now, to be removed later, which ranks above all real traffic d_ℓ, but below the high priority dummy flows. For said medium priority class, we will now introduce the concept of enforcer flows, illustrated in Figure 16. More precisely, for each dummy flow path from v'' via $e=(u,v)$, we introduce a demand of $\mu(e)$ for each enforcer flow source-destination pair. Then, in order to admit as many medium priority enforcer flows as possible, the high priority transponder flows will not mix their paths (further deviations can be canceled out). A detailled description in pseudocode is presented in Algorithms 3 and 4.

As the last part of the proof, we need to remove extra priorities used by the dummy and enforcer flows, i.e., d_h and d_m. For a standard max-flow or MCF problem, we handle both flows as standard, i.e., their allocated value is added to the output of the objective function. Extending it to LTEs, we make the assumption that flows can be assigned to the classes defined by the constant multipliers assigned to each flow value. Should we be allowed to pick our own constants, with the highest constant in the objective function w.l.o.g. normalized to one, we assign multipliers of 7 to the dummy flow values and of 4 to the enforcer flow values. Observe that a unit of dummy flow is worth more than two units of real traffic and a unit of enforcer flow ($7>2+4$), but that two units of enforcer flow outrank a unit of dummy flow ($4+4>7$). Hence, all dummy flows of a node v will be allocated, but in a way that x dummy flow units only prevent x enforcer flow units (belonging to v) from existing. As thus, the real flows only influence which edges are blocked, but not the allocation of dummy and enforcer flow demands. It remains to cover the case where we cannot assign multipliers to the flow values, which we cover with the key idea of "buy one, get many for free": when a unit of dummy flow of v blocks the edge (v,u), we can create more crossing/blocking options for the dummy flows, which run in parallel. Then, if one of these parallel paths is used, the others will be free for use as well. As such, we multiply the crossings for dummy flows by 7 and for enforcer flows by 4, increasing the demands analogously by 7 and 4, respectively. □

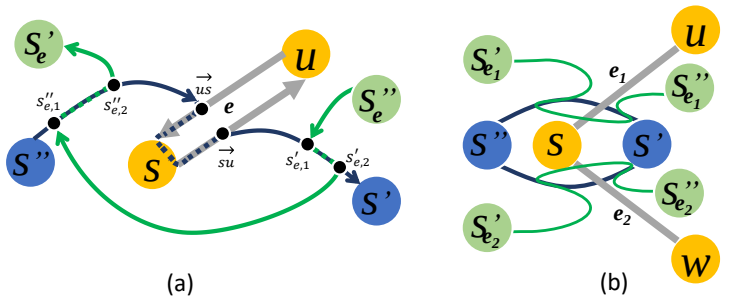


Figure 16: Enforcement of dummy flow paths: the enforcer flow from s''_e to s'_e is confined to the green path, while the transponder flow can have many options how to traverse from s'' to s'. If the blue dummy flow mixes its path, it will prevent multiple green enforcer flows from existing, unless canceled out by other blue dummy flows.

Input:
Physical graph $G=(V,E,\sigma)$ with $\mu(e)\ \forall e\in E$
Output:
Reflow graph $G_{\approx}=(V_{\approx},E_{\approx})$ with demands d_h,d_m

---

1. $V'=V$, $E'=\emptyset$
2. $\forall v\in V$: $V_{\approx}=V'\cup\{v'',v'\}$ \\`dummy nodes`
3. $\forall e=(u,v)\in E$: \\`construction from Fig. 16`
   (a) $V_{\approx}=V_{\approx}\cup\{\vec{uv},\vec{vu}\}\cup\left\{v''_{b,e},v'_{b,e},u''_{b,e},u'_{b,e}\right\}$
   (b) $V_{\approx}=V_{\approx}\cup\left\{u''_{e,1},u''_{e,2},v''_{e,1},v''_{e,2},u'_{e,1},u'_{e,2},v'_{e,1},v'_{e,2}\right\}$
   (c) $E_{\approx}=E_{\approx}\cup\{(u,\vec{uv}),(\vec{uv},v),(v,\vec{vu}),(\vec{vu},u)\}$
   (d) $E_{\approx}=E_{\approx}\cup$ **links_transp_enf**$(\vec{vu},\vec{uv},u'',u',u''_e,u'_e)\cup$**links_transp_enf**$(\vec{uv},\vec{vu},v'',v',v''_e,v'_e)$, w. capac. $\mu(e)$
4. $\forall v\in V$: $d_h(v''\to v')=\max\left\{0,-\lambda(v)+\sum_{(v,\dots)=e\in E}\mu(e)\right\}$
5. $\forall(u,v)=e\in E$: $d_m\left(u''_{b,e}\to u'_{b,e}\right)=d_m\left(v''_{b,e}\to v'_{b,e}\right)=\mu(e)$

links_transp_enf$(\vec{vu},\vec{uv},u'',u',u''_e,u'_e)\{$
return$(\{(u'',u''_{e,1}),(u''_{e,1},u''_{e,2}),(u''_{e,2},\vec{vu}),(\vec{uv},u'_{e,1}),(u'_{e,1},u'_{e,2}),(u'_{e,2},u'),(u''_{b,e},u'_{e,1}),(u''_{e,2},u'_{b,e})\})\}$

Algorithm 3: Reflow graph construction with flow priority classes.

Input:
Graph $G=(V,E)$, TM with flow demands d_ℓ
Reflow graph $G_{\approx}=(V_{\approx},E_{\approx})$ with integral demands d_h,d_m obtained from Alg. 3
Output
Flow $\mathcal{F}$ for d_ℓ and wavelength assignment Λ for G

---

1. Compute max-flow $\mathcal{F}_{\approx}$ on $G_{\approx}$ with priority classes d_h,d_m,d_ℓ
2. Set $\mathcal{F}$ as $\mathcal{F}_{\approx}$ restricted to low priority flows
3. Compute Λ as follows: $\forall(u,v)=e\in E$:
   (a) Set $c(e)$ as the free capacity on the path $u,\vec{uv},v$ in $G_{\approx}$, restricting $\mathcal{F}_{\approx}$ to only high and medium priority flows

---

Algorithm 4: Reflow abstraction with flow priority classes.

For purposes of expressiveness, crossing an edge (v,u) via 1 hop in the physical graph now consists of 15 hops in the Reflow graph ($7+1+7=15$). For path length considerations, all these crossing/blocking edges can be assigned a path length of zero, i.e., the weighted distance between v und u remains identical in both G and Reflow.

Further properties of Reflow graph. Reflow can also be extended to guarantee that every edge has always at least one wavelength allocated: to this end, we create a Reflow graph where every node in the physical graph G has $d(v)$ less transponders, and then add an additional edge in Reflow between every two neighbors in G, which is not subject to blocking by dummy flows. Analogously, Reflow can be used for partial deployments, where only parts of the network

are programmable. Furthermore, when every edge has to has have at least one wavelength allocated at all times, we can easily obtain a 2-approximation of the throughput by rounding down the output of a fractional transponder assignment: the resulting wavelength allocation will still be able to carry every flow at at least half its size. Note that unlike in Corollary 4, we just need the standard assumption of $\sigma(v) \geq d(c)$ for all nodes v in the physical graph. Moreover, for wavelengths of different capacities, Theorem 3 can be adapted with analogous techniques as mentioned in its proof, where a set of flows is either jointly allocated or not at all. We note that this technique requires a common integral multiple of the wavelength capacities. Lastly, Corollary 4 still holds as well, as the "lost" wavelengths will be those of minimum throughput.

A.4 Gain beyond $O(\Delta)$ when $\sigma(v) < \delta(v)$

We required in §4 that $\sigma(v) \geq \delta(v)$ holds for all nodes. Observe that when this requirement is violated, it is easy to obtain an infinite gain factor: consider a path of three nodes v_1,v_2,v_3, each with one transponder. As a static allocation can only assign one wavelength, w.l.o.g. on (v_1,v_2), reprogramming the wavelength to be on (v_2,v_3) produces an infinite gain factor for demands between v_2 and v_3 (zero versus one). As thus, we fixed the realistic assumption that $\sigma(v) \geq d(v)$ for all nodes. Furthermore, else, even approximating the minimum number of connected components otherwise is an NP-hard problem, by direct reduction from the Hamiltonian path problem [18].

A.5 Gain of $\omega(\Delta)$ under max-min fairness and $\sigma(v) \geq \delta(v)$

Interestingly, for TEs where the objective function is more complicated than a simple linear throughput maximization, the gain factors may be higher, even under the assumption $\sigma(v) \geq \delta(v)$. For example, in the case of max-min fairness TEs (such as SWAN [25] and B4 [29]), the gain factor can be $\Omega(n)$ where n is the total number of nodes even when Δ is a small constant. Note that these extreme gains require adversarial traffic matrices. We now provide a class of graphs where the maximum degree is $\Delta = 4$, but the gain factor for the throughput is $\Omega(n)$, i.e., $\Omega(n\Delta)$ due to $\Delta \in O(1)$. Our construction idea is as follows: Create a path of $2x$ nodes, with x being even, divided into a "left" and "right" part of x nodes each, with the right nodes denoted (from left to right) v_1,v_2,...,v_x In the right part, attach 2 children to every node, one additional child t to the rightmost node, resulting in a maximum degree of 4 and $4x+1$ nodes in total. Set $\sigma(v) = \delta(v)$ for every node. An optimal traffic-oblivious wavelength allocation must assign one wavelength to every edge, to prevent infinite gain factors. To obtain a gain factor of $\Omega(x)$, consider the following demands: every node in the left part wants to send $1/x$ to t, which so far, can be perfectly satisfied under max-min fairness, resulting in a throughput of exactly 1. However, for the nodes in the original right part, we now create demands $v_1 \to v_2$, $v_3 \to v_4$, ..., $v_{x-1} \to v_x$, each of size 1. Due to max-min fairness, the static wavelength allocation can only obtain a total throughput of at most 2. By using reconfiguration, we can increase the wavelength allocation in the right part to 2 for $v_1 \to v_2$, $v_3 \to v_4$, ..., $v_{x-1} \to v_x$, which allows us to satisfy all demands, with a throughput of $1+x/2$. In other words, the gain factor of reconfiguration is linear in the number of nodes, even though the maximum degree is only 4.

A.6 Coverage of Reflow

In this section of the appendix, we discuss the extension of Reflow to further TE objectives.

Path properties. Reflow can optimize the (fiber) path length [25] for latency. This is done by setting the weight of dummy edges to zero and adjusting the construction to retain original distances between real nodes.. In the case of hop-count objectives [37], Reflow can obtain an identical stretch factor of a small constant for all paths, but not the original hop count. Other path constraints such as tunnel assignments can be directly mapped onto Reflow.

k-shortest paths. [23]. As WAN topologies have mostly a low degree, we can ensure that the dummy flows of the Reflow graph won't be affected by setting $k \geq \Delta$. For example, SWAN [25] uses $k = 15$, while B4 [29] deploys $k = 4$. but values higher than $k = 4$ are positively evaluated in [29] as well. However the dummy flows cannot be used for their blocking property in shortest uniquely chosen path routing ($k = 1$), as they are then confined to a single shortest path.

Flow priorities and hierarchical bandwidths. The Reflow abstraction can capture flow priority classes [9] by assigning dummy flows to the highest priority class, to enforce that all dummy flows are allocated (in standard LTE/max-flow, there is only one priority class). Hierarchical bandwidth allocations [35] allow us to assign different priority weights to different demand levels of the same flow. Reflow expresses bandwidth hierarchies by assigning dummy flows wholly to the highest priority weight.

Forward fault correction. Systems such as FFC [40] protect against failures by providing ample capacity to re-route without congestion. When the fault cases are part of the input, we have to specify that dummy flow paths cannot fail (e.g., in [40, §4.3]), but this only protects real network hardware and flows. When all edges are to be protected, the dummy flows require alternative paths, preventing them from fulfilling their blocking role.

Fairness considerations. Max-min fairness [44] is captured by splitting dummy flows into multiple commodities, each originating from its own source. Reflow can obtain max-min fairness when the largest dummy flow has less demand than the smallest non-dummy flow allocation. To capture concurrent flow objectives [52] in Reflow, i.e., to consider the fairness of the fraction of demand allocated instead of absolutes, we route dummy flows through bottleneck edges, whose capacity corresponds to transponder numbers. By setting the dummy flow demands to sufficiently high values, the fairness objective will enforce all dummy flows to be fully allocated.

Min-max load. Min-max load objectives [36] aim at minimizing the maximum relative edge load. As the primary role of dummy flows is to increase the load across some edges (in order to block), Reflow currently only supports min-max load considerations if those edges can be removed from the min-max load objective function.

Oblivious routing. Reflow requires the routing of dummy flows to provide a wavelength allocation for given traffic demands. As oblivious routing provides routing paths independent of the network's traffic [49], there is no direct way to express obliviousness in Reflow. We believe that fundamentally different methods, possibly from network design [8], will be needed to capture oblivious routing in Reflow; we defer this to future work.

Combining objectives. Its expressiveness allows Reflow to satisfy combinations of the above mentioned TE objectives. For example, SWAN [25] requires max-min fairness, flow priorities, path properties, and k-shortest paths. Reflow can capture these objectives, with partial support for max-min fairness and k-shortest paths.

However, the partial support constraints are more on the theoretical side: for example, consider a traffic demand of 1 bit/s – Reflow cannot support max-min fairness for such small demands, unless dummy flows are also split up into small flows, using rounding to obtain feasible wavelength allocations at throughput cost. Notwithstanding, in practice, due to flow aggregation, the interactive traffic hovers around 10% of the demand per DC-pair, with elastic ranging from 5% to 40% [25, Fig. 9]. Furthermore, SWAN uses $k = 15$, which greatly surpasses the maximum degree Δ of common WAN topologies. Hence, Reflow can sufficiently express the TE objectives of SWAN in practice.