Traffic Engineering: From ISP to Cloud Wide Area Networks

Rachee Singh
Microsoft

Nikolaj Bjørner
Microsoft

Umesh Krishnaswamy
Microsoft

Abstract

We conduct a systematic review of two decades of work on wide-area network traffic engineering (TE). We summarize the contributions of important TE algorithms and systems developed for Internet Service Provider and cloud wide-area networks. We study the evolution of the goals of TE (from performance to reliability), TE system design (from de-centralized to fully-centralized to partly-centralized) and the technology used in deploying these systems in large commercial networks (from vendor-specific protocols to software-defined implementations). We define a taxonomy of TE systems to categorize developments in TE research. We identify trends at the forefront of TE research and practice to motivate an agenda for future work in this area. Finally, to aid future work, we are releasing our summaries and implementations of several recent TE algorithms.[1].

CCS Concepts

• **Networks** → **Network algorithms**; **Traffic engineering algorithms**; **Network design and planning algorithms**;

Keywords

Wide Area Networks, Software Defined Networking, Traffic Engineering.

ACM Reference Format:
Rachee Singh, Nikolaj Bjørner, and Umesh Krishnaswamy. 2022. Traffic Engineering: From ISP to Cloud Wide Area Networks. In *The ACM SIGCOMM Symposium on SDN Research (SOSR '22), October 19–20, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3563647.3563652

[1]https://github.com/racheesingh/cloud-te-tutorial

SOSR '22, October 19–20, 2022, Virtual Event, USA

ACM ISBN 978-1-4503-9892-3/22/10…$15.00
https://doi.org/10.1145/3563647.3563652

1 Introduction

Network traffic engineering (TE) mechanisms route traffic to achieve desirable characteristics like high network throughput, low latency, fault tolerance and low operational cost [6]. The exponential growth of traffic demands in commercial networks triggered an increase in the capital and operating expenses of these networks. As a result, network operators began to deploy TE techniques to utilize the expensive network resources effectively. TE enabled network operators to absorb seasonal variations in traffic demands (*e.g.,* demands peak during work hours and weekdays) without over-provisioning the network for peak demands.

De-centralized TE in ISP networks. In the early 2000s, Internet Service Providers (ISPs) like AT&T developed systems to measure network topology and traffic demands [14] to inform traffic engineering decisions. During this time, network operators leveraged traditional routing protocols implemented on black-box switches. For instance, MPLS-based traffic engineering [7, 41] enabled networks to implement *de-centralized* routing decisions. Other researchers optimized link weights of interior gateway protocols (IGPs) like OSPF to achieve desired routing in the network [15, 16]. At this time, most innovations engineered traffic *within* the ISP network, performing *intra-domain* traffic engineering.

Inter-domain TE on the Internet. As the number of networks on the Internet grew, it became increasingly important to improve the *end-to-end* performance of traffic that traversed multiple autonomous systems (ASes) on the Internet. Researchers began to investigate *inter-domain* TE to improve end host performance [3, 4, 13]. Aside from the objective of improving end host performance, inter-domain TE was also used to reduce the cost of bandwidth exchanged between ASes on the Internet [18, 44]. The exchange of traffic between networks on the Internet is charged depending on the *business relationships* in their mutual *peering contract*. Reducing the cost of exchanging traffic, called *peering cost*, added another dimension to the goals of inter-domain TE.

From ISP to Cloud TE. Early 2010s saw the large-scale commercialization of cloud computing and the emergence of global cloud providers like Amazon, Google and Microsoft. The low-latency and high bandwidth requirements of cloud workloads led cloud providers to provision *private planet-scale wide area networks* (WANs) of their own. Private WANs

are expensive resources. Similar to ISP operators in the previous decade, cloud providers also began engineering traffic to efficiently utilize their network infrastructure.

TE in the SDN era. Despite several parallels between the objectives of TE in ISPs and cloud WANs, cloud WANs presented some unique opportunities and challenges. First, the era of cloud TE arrived on the heels of the software-defined networking (SDN) revolution — providing cloud providers a new range of tools to implement TE capabilities. Second, unlike ISP traffic, a large volume of cloud traffic is *internal* to the network. This includes periodic storage backups and replication of search indices between cloud datacenters. Such traffic is discretionary and tolerant to delays, giving cloud TE the freedom to schedule it when the network is under-utilized [25, 27, 38]. While ISP operators had to contend with uncertainty in traffic demands and plan allocations to be *oblivious* to the uncertainty [5, 40], cloud operators have a degree of control on the traffic demand matrix. Major cloud providers deployed software-defined TE in parts of their WANs responsible for carrying discretionary workloads. Thus, cloud providers operated *two wide-area networks*: one that leveraged SDN-based TE [20, 22] and the other that used switch-native bandwidth allocation and tunneling protocols.

De-centralized to centralized TE. Researchers analyzed the traffic in a large cloud WAN and found that popular decentralized TE techniques left the network susceptible to long periods of under-utilization [32]. This brought about a shift in the thinking of cloud TE architects, with both Microsoft [20] and Google [22] deploying *centralized traffic engineering* in their WANs. The departure from de-centralized to centralized TE system design can be observed in most cloud TE innovations that followed [9, 30, 37].

Towards partially de-centralized TE. However, over the years, practitioners have understood the drawbacks of fully centralized traffic engineering — the centralized TE controller can become a performance bottleneck and a single point of failure. The increasing size of cloud WAN topologies also presents a challenge to the efficiency of centralized TE implementations [2]. Recently, Google has modified their fully centralized B4 deployment to a hierarchical-centralized architecture consisting of super nodes [42] to scale their network and achieve high reliability. Moreover, Microsoft operates several autonomous TE controllers responsible for engineering traffic withinin *slices* of the global WAN [28].

From performance to reliability. While most early TE systems optimized routing along one or more axes of network performance *e.g.,* throughput, latency and congestion, recent work has focussed on baking *resilience to link failures* in TE systems. For instance, researchers have proposed TE algorithms that ensure the highest possible throughput in the face of k simultaneous link failures [10, 30] and probabilistic link failures [9]. Other work has incorporated failure resilience in the TE path selection [29]. As operational TE deployments have matured, the focus of cloud operators has shifted from squeezing the highest throughput from the network to ensuring that the network continues to perform reasonably well when inevitable link failures happen.

Unifying split-WAN architectures. Today, it has became operationally challenging for cloud operators to maintain two WANs with different protocol stacks — one that uses SDN TE and the second that relies on MPLS RSVP-TE [8]. This has led cloud providers to unify their split-WAN architecture and revisit a key question: should the unified WAN leverage software-defined TE [20, 22] or a standards-based approach (*e.g.,* RSPV-TE)? Benefitting from their experience of operating both types of WANs and recent development of new TE standards, cloud providers will navigate the trade-offs between software-defined and standards-based WAN TE in the near future.

TE as a tool for network design. Finally, recent work has used TE techniques to *design* wide-area networks by allocating capacities to network links [36], illustrating the utility of TE algorithms for achieving objectives aside from allocating traffic to links during network operation.

TE is active area of research and development in WANs, datacenters and other computer networks [34]. We define a taxonomy of TE systems to categorize related work in WAN TE (§2). To enable a systematic examination of traffic engineering techniques, we define the TE problem and formulate the TE optimization for a variety of network problems (§3). We discuss trends in TE research and practice (§4). Motivated by the trends in TE practice, we pose research questions to set an agenda for future research in traffic engineering (§5). We summarize the key takeaways of major TE advancements in last two decades in a longer version of this paper to serve as a reference for future work [1].

Table 1: Taxonomy of traffic engineering systems.

Taxonomy Category	Type A	Type B
Network type	Cloud WAN	ISP WAN
TE control domain	Intra-domain	Inter-domain
System design	Centralized	De-centralized
Implementation technology	SDN	Vendor protocols
Allocation duration	One-shot	Scheduling
Run-time frequency	Network build-time	Network run-time

2 Taxonomy of traffic engineering

We define a taxonomy of TE systems to classify recent advancements in traffic engineering. Table 1 summarizes our

Table 2: Traffic engineering systems classified based on the taxonomy of Table 1. (●) represents the settings supported by an algorithm, (◉) represents natural extensions of the presented approach and (○) shows settings that are not supported by an algorithm.

TE Systems		Network Type		Control Domain		Objective						Formulation Characteristics			
		ISP	*Cloud*	*Intra-*	*Inter-*	*T.put*	*Lat.*	*Avail.*	*Cong.*	*Fairness*	Cost	*Centralized*	*SDN*	*Tunnel*	*One-shot*
(2001)	MATE [12]	●	○	●	○	○	●	○	○	○	○	○	○	●	●
(2005)	TeXCP [26]	●	○	●	○	○	○	○	●	○	○	○	○	●	●
(2006)	COPE [40]	●	○	●	○	○	○	○	●	○	○	○	○	●	●
(2009)	Cooperative TE [11, 24]	●	○	○	●	○	○	○	●	○	○	○	○	○	●
(2013)	SWAN [20]	○	●	●	○	●	●	○	◉	●	○	●	●	●	●
(2013)	B4 [22]	○	●	●	○	●	○	○	◉	●	○	●	●	●	●
(2014)	FFC [30]	○	●	●	○	●	○	●	◉	○	○	●	●	●	●
(2014)	Tempus [27]	○	●	●	○	●	○	○	◉	○	○	●	●	●	○
(2016)	OWAN [25]	○	●	●	○	○	●	○	○	○	○	●	●	○	○
(2016)	Pretium [23]	○	●	●	○	○	○	○	○	○	●	●	●	○	○
(2017)	Espresso [43]	○	●	○	●	●	●	○	◉	●	○	○	●	○	◉
(2017)	Edge Fabric [33]	○	●	○	●	◉	◉	○	●	○	○	●	●	○	◉
(2018)	RADWAN [37]	○	●	●	○	●	◉	●	◉	○	○	●	●	●	●
(2018)	B4 and After [42]	○	●	●	○	◉	●	○	●	●	○	○	●	○	◉
(2018)	SMORE [29]	○	●	●	○	●	●	●	○	○	○	●	●	●	●
(2019)	TeaVaR [9]	○	●	●	○	●	○	●	○	○	○	●	●	●	●
(2019)	Lancet [10]	◉	◉	●	○	○	○	●	○	○	○	●	●	○	●
(2021)	Cascara [35]	○	●	○	●	◉	●	○	○	○	●	●	●	○	●
(2021)	NCFlow [2]	○	●	●	○	●	○	○	○	○	○	○	●	●	●
(2021)	CodedBulk [38]	○	●	●	○	●	○	○	○	○	○	○	○	○	●
(2021)	Shoofly [36]	○	●	●	○	●	○	○	○	○	●	●	●	●	●
(2022)	Blastshield [28]	○	●	●	○	●	●	○	◉	●	○	○	●	●	●

TE taxonomy and Table 2 summarizes recent work related to the taxonomy. It consists of the following categories:

Network type. We categorize TE systems based on the type of network they were designed for. Since this work focusses on wide-area network TE, the network type of the TE systems we classify is either an ISP or cloud network. We note that ISP WANs pre-date cloud networks and were the first to deploy TE in production settings.

TE control domain. TE systems can engineer traffic on links within the WAN *i.e.,* intra-WAN TE or links between the WAN and other networks on the Internet *i.e.,* inter-WAN TE. Thus, the control domain describes the reach of TE systems' traffic allocations and can be inter-domain or intra-domain.

System design. In the early days of TE research, ISP WAN operators used standardized protocols on black-box switches to engineer traffic. These TE implementations were distributed — individual routers were involved in the process of identifying the network path and allocating traffic to achieve TE objectives. While cloud operators began with decentralized TE implementations, they observed that MPLS-based TE led to sub-optimal utilization of links and latency inflation [32]. As a result, both Google's B4 [22] and Microsoft's SWAN [20] developed centralized TE for their WANs. In centralized TE systems, a *controller* program has a global view of the network and it computes globally optimized traffic allocations for the WAN. The implementation of the centrally computed allocations depends on the technology the operator chooses, commonly consisting of one or more tunneling protocols (*e.g.,* MPLS, IP-in-IP). In contrast from centralized, in fully de-centralized TE implementations, each router allocates traffic on links with local information only.

Implementation technology. In the early 2000s, ISPs leveraged switch-native protocols (*e.g.,* MPLS, OSPF) to engineer traffic in their networks to effectively load-balance across switches and edges between them. Since then, the goals of traffic engineering have evolved as have the tools used to implement them. Since the onset of the SDN era, WAN operators have leveraged software-defined centralized controllers to implement more efficient TE systems.

Allocation duration. TE algorithms often calculate *one-shot* traffic allocations for near real-time traffic demands. In the one-shot operation, the TE algorithm computes traffic allocations in the network without sharing any network state with previous rounds of TE computation. In contrast, some

TE algorithms [27, 38] *schedule* long-running data transfers in the network.

Run-time frequency. While traditionally, TE algorithms are used at regular and frequent intervals (*i.e.,* run-time) of time to allocate traffic in the network, they have also been used at *build-time* to design the WAN [36] or to allocate capacity to links.

Using the TE taxonomy (Table 1), we categorize advancements in TE research in the last few decades in Table 2. We sub-divide the taxonomical categories (*i.e.,* network type, control domain, TE objective and formulation characteristics) to extract important sub-features of the TE system. For instance, the objectives of TE systems have evolved to meet a variety of goals including throughput maximization (*T.put*), latency minimization (*Lat.*), congestion minimization (*Cong.*), improving availability (*Avail.*), fairness in flow completion (*Fairness*), and minimizing network cost (*Cost*). TE formulations can have different design characteristics based on the problem and network type. This includes centralized vs. de-centralized designs that leverage different implementation technologies (SDN vs. vendor protocols) to solve edge or path formulations of network design or traffic allocation problems (§3). We summarize the key findings of ISP and cloud TE systems shown in the timeline of Table 2 in the longer version of this paper [1].

3 The TE optimization problem

Network TE consists of systems and algorithms responsible for routing traffic to achieve characteristics like load-balancing, high link utilization, low end-to-end latency, fairness *etc* in the network. For a given network topology and traffic demands, TE allocates traffic on network links to achieve the goal or *objective* of the TE system. Often, the TE system is subject to *constraints* that bound traffic allocation on links by their capacity or ensure that all demands are met. TE finds traffic allocations on links subject to these constraints. While the early work on traffic engineering did not explicitly state the objectives and constraints, most modern TE systems operate within this framework (Figure 1).

Inputs. The network topology consists of nodes, links, link capacities and paths or tunnels. The network topology and demands between all pairs of nodes in the network are input to the TE controller. These inputs are subject to change over time and TE systems periodically provide a fresh set of topology and demands as inputs to the controller.

Objectives. TE algorithms implicitly or explicitly encode the goals of network operators. Common TE objectives load-balance traffic across links in the network to prevent over-utilizing specific links or maximize the overall throughput of the network. While there are many different ways to route traffic in the network, TE algorithms arrive at specific allocations that satisfy the objective function.

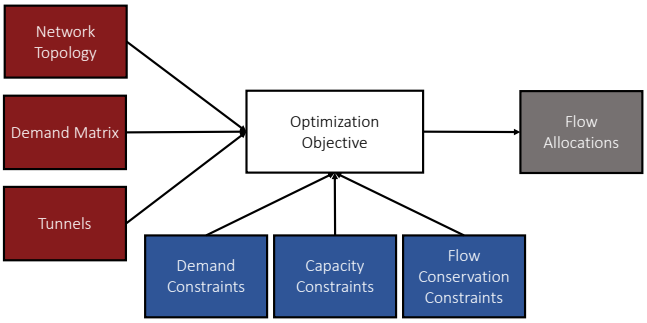

Figure 1: Overview of traffic engineering algorithms. The inputs to TE algorithms (in red) consist of the network topology, demand matrix and other topological features like network tunnels between nodes. The algorithm is subject to constraints (in blue) to ensure that all demands get met, link capacities are respected and network flow is conserved. The objective TE algorithms can be to minimize latency, maximize throughput subject to the constraints. TE algorithms output flow allocations along links in the network (in grey).

Constraints. TE algorithms are subject to certain constraints. For example, total allocation on any link is bounded by the capacity of the link. Similarly, some TE algorithms enforce that the traffic allocations meet the traffic demands input to the algorithm.

Outputs. The output of the TE algorithm is traffic allocations on all links in the network. The TE controller uses router-specific programs, commonly called as router agents [20] to translate the traffic allocations to router protocol configuration in the network.

Variables of TE optimization. TE is commonly formulated as a multi-commodity flow optimization problem. We denote the network as $G : \langle V, E \rangle$ with vertices V and edges E. c_e represents the capacity of an edge e. D is the set of traffic demands between source and destination vertices where $demand(d)$ is the demand between src_d and dst_d for $d \in D$.

We summarize common TE formulations from previous work to illustrate typical uses of TE in wide-area networks. We first describe path formulation of the TE problem which improves the scalability of TE in practice (§3.1). Next, we formulate a commonly used WAN TE objective of maximizing network throughput (§3.2). We then show that TE objectives can capture multiple goals like high throughput and low latency (§3.3). We show how previous work has developed TE algorithms that allocate demands to be maximally resilient to link failures in the network (§3.4). Finally, we show TE can be a versatile tool for designing WANs (§3.5). We are also releasing Python implementations of these TE formulations [1].

3.1 Edge vs. path formulations of TE

Early implementations of the TE problem computed traffic allocations for every edge in the network. This is called the *edge formulation* of TE. The edge formulation scales poorly with increase in network size. As a result, researchers developed the *path formulation* of TE that allocates demands on k paths between a source and destination node [20]. Since demands are allocated to a small set of paths as opposed to the large number edges in the network, the path formulation is significantly faster to solve. In networking technology, MPLS tunnels are a way of implementing paths between nodes in the network [41]. For instance, $T(d)$ denotes the set of tunnels for a demand d. Each tunnel is a path from src_d to dst_d for some demand d. In this case, the TE optimization solves for $f(t)$ which is the flow through tunnel t.

3.2 Maximize network flow

Commonly, TE algorithms aim to maximize the network's throughput using the flow allocations. This objective is subject to constraints that the total traffic allocation on an edge should not exceed its capacity and the total allocation across all tunnels of a demand should not exceed the demand itself. We show the objective and constraints of this formulation in the following:

$$\text{Maximize} \sum f(t)$$

Subject to:

$$c_e \geq \sum_{t \ni e} f(t) \text{ for each edge } e$$

$$demand(d) \geq \sum_{t \in T(d)} f(t) \text{ for each demand } d$$

We note that the objective and the constraints in the maximum throughput path formulation of TE are linear. Linear Programming solvers, both commercial [19] and open-source [17, 21] can be used to solve these TE formulations.

3.3 High network flow with low latency

Operators are also known to implement TE formulations with multiple objectives like maximizing network throughput while achieving low latency. We encode the latency of a link into its weight. The weight of a tunnel, $weight(t)$, is the sum of weights of links in the tunnel $\sum_{e \in t} weight(e)$. ϵ represents the relative importance of reducing latency over increasing network flow through the TE optimization. ϵ is a constant that is configured by the operator. The maximum throughput with low latency objective is shown below. This objective is subject to the same constraints as before.

$$\text{Maximize} \sum f(t) \cdot (1 - \epsilon \cdot weight(t))$$

3.4 Failure resilience through TE

Researchers have developed forward fault collection [30] to allocate flow in the network in a manner that is maximally resilient to a set of failure scenarios.

Failure scenarios. Forward fault correction encodes each failure scenario using a *failure group*. Every failure group contains a set of edges that are disabled when the failure occurs. We represent a failure group with g. F is the set of failure groups. Often production WANs aim to be resilient in the face of two simultaneous link failures. This means that in the event of two links failing at the same time, the network can continue to meet all demands. So, we also consider the scenario of pairwise failures.

For a network with set of edges E, the failure groups F are all pairs of edges from E:

$$F = \{\{e, e'\} \mid e, e' \in E\}$$

Characteristic functions. A failure group g determines a characteristic function α defined on edges and tunnels. We define $\alpha(e)$ for an edge and $\alpha(t)$ for a tunnel in the network:

$$\alpha(e) = 0 \text{ if } e \in G, \text{ else } = 1$$

$$\alpha(t) = \alpha(e_1) \cdots \alpha(e_k) \text{ where } t = e_1, \ldots, e_k$$

We note that the characteristic functions are integer variables with only possible values 0 or 1.

Worst case flow. FFC defines b_d as the worst case bandwidth allocation for a demand d computed by the TE optimization due to all possible failure scenarios considered by the operator. The objective of FFC is to minimize the accumulated gap between b_d and the demand d.

Explosion of constraints. The following constraints solve for a flow $f(t)$, such that edge capacities are not exceeded and in the worst case failure scenario, bandwidth allocation for a demand d is b_d.

$$\text{Maximize} \sum_d b_d$$

Subject to:

$$\sum_{t \ni e} f(t) \leq c_e \text{ for each edge } e$$

$$b_d \leq \sum_{t \in d} \alpha(t) \cdot f(t) \text{ for each } d, \text{ each failure scenario } \alpha \in F$$

$$b_d \leq demand(d) \text{ for each demand } d$$

$$0 \leq f(t) \text{ each tunnel } t$$

Dual formulation to deal with scale. Solving the naive FFC formulation is not scalable since the set of pairwise failure scenarios grows quadratically with the size of the network. Moreover, enumerating other combinations of link failures (*e.g.,* triplets, quadruplets) makes this problem worse.

Luckily we can draw on fundamentals of linear programming to develop a formulation that only grows modestly with the network size. This observation was made in a technical note by some of the authors of FFC [30]. More recently, Lancet [10] develops a general setting that exploits LP duality to tackle the scale problem of such formulations. Finally, we re-formulate the dual of the FFC linear program and are releasing a Python implementation of the dual [1].

3.5 Network design through TE

We will now consider an instance of network provisioning or design problem that leverages TE optimization. In contrast to online TE, provisioning is performed offline using long-term forecasts of traffic demand patterns. The goals of network provisioning include allocating hardware resources like fiber links and router ports. Recently, Shoofly [36] has focussed on the allocation of router ports by identifying network hops that can be *bypassed* by optical wavelengths on the fiber of a optical WAN. Keeping signals in the optical domain longer using these bypasses, allows network operators to *release* ports on electrical packet switches and routers for future use. Moreover, it reduces the volume of traffic that routers need to handle on their ports and fabric.

Shoofly solves the provisioning problem using a TE formulation. In usual cases, TE allocates traffic to meet as much demand as is possible. However, in Shoofly's provisioning problem, TE must meet all the demands while trying to allocate traffic to certain special edges. These special edges are possible new edges that the provisioned network will have once optical bypasses are implemented.

Shoofly assumes we are given, as input, a network with a set of possible bypasses or shortcuts and an annotation of the number of optical wavelengths allocated on each edge. u_e represents the bandwidth of one wavelength on edge e. s is a network shortcut due to an optical bypass and u_s is the capacity of one wavelength of shortcut s. We introduce auxiliary and output variables for the optimization. w_s is the number of wavelengths assigned to shortcut s. We note that w_s is an integer, making Shoofly require Mixed Integer Programming. x_e^t is the flow allocated to tunnel t that traverses edge e. y_s^t is the number of wavelengths allocated to traverse the shortcut s as part of tunnel t. Shoofly then solves an optimization problem to maximize the saving of router ports:

$$\text{Maximize} \sum_s |s| \cdot w_s$$

The optimization problem is subject to allocating flows to tunnels and shortcuts such that all long-term forecasted demands are met. Shoofly has the following constraints:

- Flows can be allocated to meet all demands:
$$demand(d) \leq \sum_{t \in d} f(t) \text{ for each demand } d$$
- Allocated flows don't exceed the edge capacities:
$$\sum_{t \ni e} x_e^t + e_u \cdot \sum_{s \ni e} w_s \leq e_c \text{ for each edge } e$$
- Flow passing through each shortcut is bounded by wavelengths assigned to s:
$$\sum_{t ins} y_s^t \leq u_s \cdot w_s \text{ for each shortcut } s$$
- Shortcuts are assigned on duplex links so as to have the same capacity in both link directions:
$$w_s = w_{s'} \text{ for each } s, s' \text{ such that } s' = reverse(s)$$
- The flow in a tunnel t is capped at the contribution of each edge to t:
$$f(t) \leq x_e^t + \sum_{s \ni e} y_s^t \text{ for every tunnel } t, \text{ and edge } e \in t$$

Shoofly's optimization is solving for traffic allocations to meet demands, in the same way as traffic engineering does. However, it incentivizes the optimization to allocate traffic on the bypasses to identify which bypasses are feasible while continuing to meet traffic demands. Network operators analyze the outputs of Shoofly to find feasible optical bypasses in the network. This process is infrequent compared to TE which is solved every 5 minutes in WANs. Shoofly's use of TE highlights the utility of TE as a general tool for network design and provisioning.

4 Trends in WAN traffic engineering

We highlight the recent trends in wide-area TE research.

Evolving objectives of traffic engineering. TE has been used to achieve a variety of goals in the network, most centered around improving the performance of the network. However, operating these large-scale systems is challenging especially in the face of software and hardware faults, cascading component failures and misconfigurations. Since the adoption of SDN-based TE systems in cloud WANs, there has been a shift in the goal of TE systems towards sustaining reliable inter-datacenter communication. Starting with forward fault correction (FFC [30]) that computes optimal traffic allocations despite k simultaneous link failures in the network, cloud TE has incorporated probabilistic link failures into traffic engineering objectives [9]. These TE algorithms take into account the likelihood of each link failure to ensure that the network throughput service-level agreement (SLA) continues to be met with the computed traffic allocations.

Cross-layer traffic engineering. In the past few years, researchers have devised traffic engineering algorithms that

cut across several layers of the networking stack. While traditional TE controllers interact with routing (Layer 3) and forwarding (Layer 2) protocols of network routers, cross-layer TE controllers both incorporate inputs from and influence the state of the physical layer of the network. For instance, RADWAN [37] incorporates signal quality on the WAN optical fiber to dynamically adapt the capacity of physical links in the network to meet network demand spikes. Shoofly devises a cross-layer TE algorithm to design cost-effective optical WAN topologies [36]. Thus, we observe that the TE framework (Figure 1) is versatile and can be used for designing WAN topologies [36], managing dynamic capacity networks [37] and allocating additional capacity to existing networks. We expect additional network build planning tasks can benefit from existing TE frameworks.

Learning to route. Researchers are starting to apply machine learning techniques to solve traffic engineering objectives [39] and have found promising results. Fundamentally, intra-cloud WAN traffic matrices have been found to be *predictable*, enabling ML techniques to perform well. The predictability of cloud traffic matrices over ISP traffic matrices stems from types of workloads supported by the cloud like long running cloud backups at regular intervals. Aside from the predictability of the traffic matrices, the TE problem can benefit from learning-based approaches by rapidly adapting to infrequent traffic bursts and changes to the network topology. Moreover, supervised learning enables operators to train their ML pipeline for a variety of training TE objectives without having to re-implement the TE controller logic.

Dealing with the increasing scale of WAN TE. While SDN-based WANs were small in size at their inception, they are now rapidly growing in size. This rapid growth is caused by two trends: (1) deployment of new datacenters by cloud providers and (2) increase in the purview of TE due to the unification of the split-WAN architecture. This increase in WAN size poses a challenge for existing TE deployments. Recent work has tackled reducing the TE optimization solver time [2] by computing allocations on smaller clusters in the network and combining the results for the entire network.

5 An agenda for TE research

In this section, we project an agenda for the upcoming years of research in traffic engineering. We highlight potential research directions for the community motivated by the technical trends in TE (§4).

Striking a balance between centralization and decentralization. For a decade, cloud providers relied on centralized TE due to the higher network throughput made possible by a global view of the network state and demands. However, maintaining these large-scale systems has led to the realization that bugs in the functionality of centralized TE controllers can have a global *blast radius* [28]. As a result, cloud providers are transitioning to partly-decentralized architectures where the fault domains of individual TE controllers are limited. However, it is important to determine the optimal operating point on the performance-reliability trade-off while transitioning to decentralized TE implementations. Balancing the reliability benefits of decentralization with throughput efficiency of centralization is an important challenge for future work in TE.

RSVP-TE vs. SDN. The unification of split-WAN architectures in cloud WANs begs the question: which of the two WANs is a better choice in the dynamic and evolving networks of today? As cloud providers unify their two WANs, it is important to identify which implementation will the unified WAN have: SDN-based TE or vendor-specific MPLS RSVP-TE? With the hindsight of decade-long operation of both types of WANs, we believe network operators can provide insights into the pros and cons of SDN-based TE operation over the traditional techniques.

Exploiting the characteristics of the problem. Experience of TE practitioners [28] has shown that simpler approximations and heuristic-based strategies can be preferable to more complex clean-slate designs in practice. For instance, Blastshield [28] chains multiple TE solvers in place of solving a complex multi-objective TE problem. Similarly, production TE systems have dealt with the scaling challenges of TE by reducing the size of the traffic matrix using the empirical insight that a small fraction of network flows constitute majority of the traffic volume in WANs. Such empirical insights can lead to simpler TE algorithms but run the risk of being far from optimal in corner cases. It is important for future work to quantify the fragility of TE system performance that rely on such heuristics and approximations.

TE for optimal communication in server interconnects. Given the general applicability of the concepts of traffic engineering, future research can benefit from applying TE techniques in other types of networks. For instance, heterogenous multi-GPU networks like Nvidia's DGX series connect GPUs with high-speed interconnects and can benefit from TE techniques to optimize inter-GPU communication [34].

Impact of increased collaboration between ISPs and cloud providers on TE. ISPs are entering into partnerships with commercial cloud providers to operate their radio access networks (RANs) and packet cores on the cloud [31]. The migration of cellular network workloads to the cloud enforces closer collaboration between ISPs and cloud providers, increasing the willingness to exchange information about their network infrastructure and performance metrics. Inter-domain traffic engineering systems that leverage the shared information can improve end-to-end client performance in future work [11, 24].

References

[1] (Accessed on 2022-09-10). Cloud WAN Traffic Engineering Tutorial. https://github.com/racheesingh/cloud-te-tutorial.

[2] Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. 2021. Contracting Wide-area Network Topologies to Solve Flow Problems Quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 175–200. https://www.usenix.org/conference/nsdi21/presentation/abuzaid

[3] Aditya Akella, Bruce Maggs, Srinivasan Seshan, Anees Shaikh, and Ramesh Sitaraman. 2003. A Measurement-Based Analysis of Multihoming. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Karlsruhe, Germany) *(SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 353–364. https://doi.org/10.1145/863955.863995

[4] Aditya Akella, Jeffrey Pang, Bruce Maggs, Srinivasan Seshan, and Anees Shaikh. 2004. A Comparison of Overlay Routing and Multihoming Route Control. *SIGCOMM Comput. Commun. Rev.* 34, 4 (Aug. 2004), 93–106. https://doi.org/10.1145/1030194.1015479

[5] David Applegate and Edith Cohen. 2003. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Karlsruhe, Germany) *(SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 313–324. https://doi.org/10.1145/863955.863991

[6] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. 2002. RFC3272: Overview and Principles of Internet Traffic Engineering.

[7] D. O. Awduche. 1999. MPLS and traffic engineering in IP networks. *IEEE Communications Magazine* 37, 12 (1999), 42–47. https://doi.org/10.1109/35.809383

[8] Daniel O. Awduche, Lou Berger, Der-Hwa Gan, Tony Li, Dr. Vijay Srinivasan, and George Swallow. 2001. RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209. https://doi.org/10.17487/RFC3209

[9] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. 2019. TEAVAR: striking the right utilization-availability balance in WAN traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*. ACM. https://doi.org/10.1145/3341302.3342069

[10] Yiyang Chang, Chuan Jiang, Ashish Chandra, Sanjay Rao, and Mohit Tawarmalani. 2019. Lancet: Better Network Resilience by Designing for Pruned Failure Sets. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 3, Article 49 (Dec. 2019), 26 pages. https://doi.org/10.1145/3366697

[11] D. DiPalantino and R. Johari. 2009. Traffic Engineering vs. Content Distribution: A Game Theoretic Perspective. In *IEEE INFOCOM 2009*. 540–548. https://doi.org/10.1109/INFCOM.2009.5061960

[12] A. Elwalid, C. Jin, S. Low, and I. Widjaja. 2001. MATE: MPLS adaptive traffic engineering. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Vol. 3. 1300–1309 vol.3. https://doi.org/10.1109/INFCOM.2001.916625

[13] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. 2003. Guidelines for Interdomain Traffic Engineering. *SIGCOMM Comput. Commun. Rev.* 33, 5 (Oct. 2003), 19–30. https://doi.org/10.1145/963985.963988

[14] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. 2000. NetScope: traffic engineering for IP networks. *IEEE Network* 14, 2 (2000), 11–19. https://doi.org/10.1109/65.826367

[15] B. Fortz, J. Rexford, and M. Thorup. 2002. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine* 40, 10 (2002), 118–124. https://doi.org/10.1109/MCOM.2002.1039866

[16] B. Fortz and M. Thorup. 2000. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, Vol. 2. 519–528 vol.2. https://doi.org/10.1109/INFCOM.2000.832225

[17] GNU. (Accessed on 2019-10-02). GNU Linear Programming Kit. https://www.gnu.org/software/glpk/.

[18] David K. Goldenberg, Lili Qiuy, Haiyong Xie, Yang Richard Yang, and Yin Zhang. 2004. Optimizing Cost and Performance for Multihoming. *SIGCOMM Comput. Commun. Rev.* 34, 4 (Aug. 2004), 79–92. https://doi.org/10.1145/1030194.1015478

[19] Gurobi. (Accessed on 2019-10-02). GUROBI Optimization. https://www.gurobi.com/.

[20] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving High Utilization with Software-driven WAN. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 15–26.

[21] IBM. (Accessed on 2019-10-02). CPLEX Optimizer. https://www.ibm.com/analytics/cplex-optimizer.

[22] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a Globally-deployed Software Defined Wan. *SIGCOMM Comput. Commun. Rev.* 43, 4 (Aug. 2013), 3–14.

[23] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. 2016. Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 73–86. https://doi.org/10.1145/2934872.2934893

[24] Wenjie Jiang, Rui Zhang-Shen, Jennifer Rexford, and Mung Chiang. 2009. Cooperative Content Distribution and Traffic Engineering in an ISP Network. *SIGMETRICS Perform. Eval. Rev.* 37, 1 (June 2009), 239–250. https://doi.org/10.1145/2492101.1555377

[25] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. 2016. Optimizing Bulk Transfers with Software-Defined Optical WAN. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 87–100. https://doi.org/10.1145/2934872.2934904

[26] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. 2005. Walking the Tightrope: Responsive yet Stable Traffic Engineering. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Philadelphia, Pennsylvania, USA) *(SIGCOMM '05)*. Association for Computing Machinery, New York, NY, USA, 253–264. https://doi.org/10.1145/1080091.1080122

[27] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. 2014. Calendaring for Wide Area Networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 515–526. https://doi.org/10.1145/2619239.2626336

[28] Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. 2022. Decentralized cloud wide-area network traffic engineering with BLASTSHIELD. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 325–338. https://www.usenix.org/conference/nsdi22/presentation/krishnaswamy

[29] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. 2018. Semi-Oblivious Traffic Engineering: The Road Not Taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 157–170. https://www.usenix.org/conference/nsdi18/presentation/kumar

[30] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. 2014. Traffic engineering with forward fault correction. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy (Eds.). ACM, 527–538. https://doi.org/10.1145/2619239.2626314

[31] Dan Meyer. 2021. MWC Barcelona Shows the Cloud Has Eaten Telecom. https://www.sdxcentral.com/articles/news/op-ed-mwc-barcelona-shows-the-cloud-has-eaten-telecom/2021/07/. [Online; accessed 12-July-2021].

[32] Abhinav Pathak, Ming Zhang, Y. Charlie Hu, Ratul Mahajan, and Dave Maltz. 2011. Latency Inflation with MPLS-Based Traffic Engineering. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (Berlin, Germany) *(IMC '11)*. Association for Computing Machinery, New York, NY, USA, 463–472. https://doi.org/10.1145/2068816.2068859

[33] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering egress with Edge Fabric: Steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 418–431.

[34] Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi, and Rachee Singh. 2021. Synthesizing Collective Communication Algorithms for Heterogeneous Networks with TACCL. *CoRR* abs/2111.04867 (2021). arXiv:2111.04867 https://arxiv.org/abs/2111.04867

[35] Rachee Singh, Sharad Agarwal, Matt Calder, and Paramvir Bahl. 2021. Cost-effective Cloud Edge Traffic Engineering with Cascara. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 201–216. https://www.usenix.org/conference/nsdi21/presentation/singh

[36] Rachee Singh, Nikolaj Bjorner, Sharon Shoham, Yawei Yin, John Arnold, and Jamie Gaudette. 2021. Cost-Effective Capacity Provisioning in Wide Area Networks with Shoofly. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 534–546. https://doi.org/10.1145/3452296.3472895

[37] Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: Rate Adaptive Wide Area Network. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 547–560. https://doi.org/10.1145/3230543.3230570

[38] Shih-Hao Tseng, Saksham Agarwal, Rachit Agarwal, Hitesh Ballani, and Ao Tang. 2021. CodedBulk: Inter-Datacenter Bulk Transfers using Network Coding. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, 15–28. https://www.usenix.org/conference/nsdi21/presentation/tseng

[39] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. 2017. Learning to Route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (Palo Alto, CA, USA) *(HotNets-XVI)*. Association for Computing Machinery, New York, NY, USA, 185–191. https://doi.org/10.1145/3152434.3152441

[40] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. 2006. COPE: Traffic Engineering in Dynamic Networks. *SIGCOMM Comput. Commun. Rev.* 36, 4 (Aug. 2006), 99–110. https://doi.org/10.1145/1151659.1159926

[41] Xipeng Xiao, A. Hannan, B. Bailey, and L. M. Ni. 2000. Traffic Engineering with MPLS in the Internet. *Netwrk. Mag. of Global Internetwkg.* 14, 2 (March 2000), 28–33. https://doi.org/10.1109/65.826369

[42] Chi yao Hong, Subhasree Mandal, Mohammad A. Alfares, Min Zhu, Rich Alimi, Kondapa Naidu Bollineni, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Jeffrey Liang, Kirill Mendelev, Steve Padgett, Faro Thomas Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jon Zolla, Joon Ong, and Amin Vahdat. 2018. B4 and After: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *SIGCOMM'18*. https://conferences.sigcomm.org/sigcomm/2018/program_tuesday.html

[43] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 432–445.

[44] Zheng Zhang, Ming Zhang, Albert Greenberg, Y. Charlie Hu, Ratul Mahajan, and Blaine Christian. 2010. Optimizing Cost and Performance in Online Service Provider Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation* (San Jose, California) *(NSDI'10)*. USENIX Association, USA, 3.