Traffic Engineering: From ISP to Cloud Wide Area Networks

An annotated bibliography - preliminary draft

Rachee Singh
Microsoft

Abstract

In this work, we conduct a systematic review of two decades of work on network traffic engineering. We identify and summarize the contributions of important traffic engineering (TE) algorithms and systems developed for Internet Service Providers and cloud wide-area networks. We study the evolution of the goals of traffic engineering, the techniques and technology used in deploying these systems in large commercial networks. Finally, we outline the key challenges at the forefront of TE research and practice. To aid future work in this space, we are releasing our implementations of several recent TE algorithms. [1].

ACM Reference Format:
Rachee Singh. 2021. Traffic Engineering: From ISP to Cloud Wide Area Networks: An annotated bibliography - preliminary draft. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Network traffic engineering (TE) mechanisms route traffic to achieve desirable characteristics like high network throughput, low latency and low operational cost [7]. The exponential growth of traffic demands in commercial networks triggered an increase in the capital and operating expenses of the networks. Network operators began to deploy TE techniques to utilize the expensive network resources effectively. Moreover, traffic engineering 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.

In the early 2000s, Internet Service Providers (ISPs) like ATT developed systems to measure network topology and traffic demands [14] to inform traffic engineering decisions. In these early years of TE research and practice, network operators leveraged traditional routing protocols implemented on black-box switches. For instance, MPLS-based traffic engineering [8, 33] 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 [16, 17]. During this time, most TE innovations engineered traffic *within* the ISP network, performing *intra-domain TE.*

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

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 ISPs and other autonomous systems (ASes) to reach its destination. 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, 36]. 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 in the SDN era. 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 global wide area networks* 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 infrastructure investment in the cloud network.

Despite several parallels between the objectives of traffic engineering in ISPs and cloud WANs, cloud WANs presented some unique opportunities and challenges. First, the era of cloud traffic engineering arrived on the heels of the software-defined networking (SDN) revolution — providing cloud providers a new range of tools to implement traffic engineering 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 [23, 25, 31]. While ISP operators had to contend with uncertainty in traffic demands and plan allocations to be *oblivious* to the uncertainty [6, 32], cloud operators have a degree of control on the traffic demand matrix.

De-centralized to centralized and back. Researchers analyzed the traffic in a large cloud WAN and found that popular de-centralized TE techniques left the network susceptible to long periods of under-utilization [28]. This brought about a shift in the thinking of cloud TE architects, with both Microsoft [19] and Google [20] deploying *centralized traffic engineering* in their networks. The departure from de-centralized to centralized TE system design can be observed in most cloud TE innovations that followed [9, 27, 30]. However, over the years, practitioners realized the drawbacks of fully centralized traffic engineering *e.g.,* the centralized TE controller can become a performance bottleneck and a single point of failure. The increasing scale of cloud WAN topologies also presents a challenge to efficient centralized TE implementations [1]. Recently, Google has modified their fully centralized B4 deployment to a hierarchical-centralized architecture consisting of super nodes [34] to scale their network and achieve high reliability.

From performance to reliability. While most early traffic engineering 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 [27] and probabilistic link failures [9]. Other works have incorporated failure resilience in the TE path selection [26]. As operational TE has matured in cloud networks, the main 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.

Table 1: Categorization of traffic engineering systems.

TE Category	Type A	Type B
Network type	Cloud WAN	ISP WAN
Computation frequency	Network build-time	Network run-time
Formulation type	One-shot	Scheduling
TE control domain	Intra-domain	Inter-domain
System design	Centralized	De-centralized
Implementation mechanism	SDN	Vendor protocols

2 Design choices in wide-area traffic engineering

Traffic-engineering (TE) in wide-area networks has been well-studied. In the late-90s and early 2000s, TE was studied mainly in the context of Internet Service Provider (ISP) networks []. These works leveraged standardized protocols like MPLS-TE on black-box routers to engineer traffic. The TE implementations were distributed.

With the rise of large commercial cloud providers in the last decade, networking researchers and practitioners have re-visited traffic engineering to achieve desired traffic properties in cloud networks. While cloud operators began with decentralized TE implementations, they observed that MPLS-based TE led to sub-optimal utilization of links and latency inflation [28].

As a result, both Google's B4 [20] and Microsoft's SWAN [19] were the first known cloud networks to develop and deploy centralized traffic engineering in their wide area network.

3 Summary of related work

3.1 Cloud traffic engineering

In the last decade, the high investment cost and importance of cloud networks in meeting latency-sensitive client demands has led to large-scale adoption of traffic engineering systems in cloud networks. This trend is pervasive across different commercial cloud providers [19, 20], traffic types [2, 29, 35] and traffic engineering objectives [2, 9, 10]. We summarize the capabilities of cloud traffic engineering systems in this section.

SWAN [19] (2013). This work identifies that distributed traffic engineering using standardized protocols like MPLS-TE lead to low utilization of inter-DC links in cloud networks. The authors argue that the lack of a global-view of demands and capacity leads to sub-optimal routing of traffic. SWAN made a strong argument for centralized traffic engineering that globally coordinates the sending rate of services and centrally allocates paths or tunnels between senders and receivers. The traffic allocations for all demand pairs are determined by a global optimization that takes demands, tunnels and network topology as inputs. The objective of the SWAN optimization is to maximize the allocated throughput while choosing paths with lowest latency in every optimization time step (one shot formulation). Additionally, SWAN ensures min-max fairness by solving one multi-commodity flow objective for every priority class of traffic in the network.

B4 [20] (2013). Similar to SWAN, B4 is Google's software defined WAN. B4 uses OpenFlow to program the data plane of routers developed in-house with merchant silicon to implement the routing decisions from a centralized traffic engineering controller. The tunnel abstraction for multipath routing in the network is implemented with IP in IP encapsulation. The goal of the TE algorithm is to ensure min-max fairness across flow groups of different priority classes. B4 supports both centralized TE-based routing and a fallback option to use standard shortest path first (SPF) routing. The two technologies co-exist by using different forwarding tables for SPF routing and TE routing on the switch hardware. While the B4 SIGCOMM 2013 paper does not explicitly state the optimization algorithm, the write-up suggests that they solve for a max-min fair traffic allocation to maximize the link utilizations or throughput.

FFC [27] (2014). Control and data plane failures were a cause for concern in cloud networks with centralized software-defined TE systems. Data plane faults (*e.g.,* link and switch failures) trigger route recomputation, resulting in switches making decisions to forward traffic, disregarding link capacities – leading to congestion. Control plane failures occur when the TE controller fails to reconfigure switches, leading to switches operating with stale forwarding state. This work proposes to proactively handle faults such that TE's allocations incorporate the possibility of up to k simultaneous faults in the network. As the resulting formulation has exponential number of constraints, the authors transform the constraints to a bounded-M sum problem and encode the transformed problem using a sorting network.

Tempus [25] (2014). Most cloud TE systems solve *one-shot* formulations where they recompute traffic allocations to flow groups in every 5-minute time interval. Instead, Tempus focusses on large and long-transfers in the WAN with pre-assigned deadlines. Thus, Tempus packs these transfer across network paths and future timesteps.

OWAN [23] (2016). In this work, authors propose to have a central controller program both the network and the optical layer to schedule bulk transfers in wide area networks. The work relies on rapid reconfiguration of underlying optical topology using deployment of in ROADMs in WANs which can be remotely reconfigured to switch wavelengths within 100s of milliseconds. This cross-layer optimization approach allows OWAN to achieve higher throughputs compared to mechanisms that schedule bulk transfer by programming the network layer alone.

Pretium [21] (2016). This work proposes Pretium, a framework for dynamically pricing and scheduling bandwidth transfers within

Table 2: Cloud traffic engineering systems and their capabilities. (●) represents supported settings of the algorithm, (◍) represents natural extensions of the presented approach and settings not supported are shown with (○).

TE Systems		WAN Traffic Type		Objective						Formulation Type			
		Intra-	*Inter-*	*T.put*	*Lat.*	*Avail.*	*Cong.*	*Fairness*	Cost	*Centralized*	*SDN*	*Tunnel*	*One-shot*
(2013)	SWAN [19]	●	○	●	●	○	◍	●	○	●	●	●	●
(2013)	B4 [20]	●	○	●	○	○	◍	●	○	●	●	●	●
(2014)	FFC [27]	●	○	●	○	●	◍	○	○	●	●	●	●
(2014)	Tempus [25]	●	○	●	○	○	◍	○	○	●	●	●	○
(2016)	OWAN [23]	●	○	○	●	○	○	○	○	●	●	○	○
(2016)	Pretium [21]	●	○	○	○	○	○	○	●	●	●	○	○
(2017)	Espresso [35]	○	●	●	●	○	◍	●	○	○	●	○	◍
(2017)	Edge Fabric [29]	○	●	◍	◍	○	●	○	○	●	●	○	◍
(2018)	RADWAN [30]	●	○	●	◍	●	◍	○	○	●	●	●	●
(2018)	B4 and After [34]	●	○	◍	●	○	●	●	○	○	●	○	◍
(2018)	SMORE [26]	●	○	●	●	●	○	○	○	●	●	●	●
(2019)	TeaVaR [9]	●	○	●	○	●	○	○	○	●	●	●	●
(2021)	Cascara [2]	○	●	◍	●	○	○	○	●	●	●	○	●
(2021)	NCFlow [1]	●	○	●	○	○	○	○	○	○	●	●	●

Table 3: ISP traffic engineering systems and their capabilities. (●) represents supported settings of the algorithm, (◍) represents natural extensions of the presented approach and settings not supported are shown with (○).

TE Systems		WAN Traffic Type		Objective				Formulation Type			
		Intra-	*Inter-*	*T.put*	*Lat.*	*Avail.*	*Cong.*	*Centralized*	*SDN*	*Tunnel*	*One-shot*
(2001)	MATE [12]	●	○	○	●	○	○	○	○	●	●
(2005)	TeXCP [24]	●	○	○	○	○	●	○	○	●	●

the cloud. The price of the transfer is a function of its deadline and priority. The authors model the internal costs of cloud networks and align the bandwidth pricing with the cost structure of the WAN.

Edge Fabric [29] (2017). Edge Fabric is Facebook's software-defined edge to the Internet. This work describes how the Edge Fabric controller reacts to congestion at the edge of the network to compute appropriate BGP route overrides and installs them in edge routers. The overall goal of this work is to overcome BGP's performance-agnostic routing behavior. In addition to the route override functionality, Edge Fabric performs measurements through alternate paths from the edge to inform performance-aware routing decisions.

Espresso [35] (2017). Espresso is Google's software-defined edge designed to overcome the limitations of Internet edge routers. Due to the high costs and lack of flexibility of Internet-scale routers, Google implemented a new edge architecture where packet processing, routing are moved to software packet processors running on servers at the edge. With the routing and protocol complexity taken out, Espresso can leverage simple MPLS switches in place of Internet-scale routers. For edge traffic engineering, Espresso uses an algorithm to greedily optimize the client performance.

RADWAN [30] (2018). RADWAN proposes to adapt the capacity of physical links in the WAN in response to changing signal quality. However, capacity adaptation of optical links comes at the cost of link downtime of 1 minute or so. RADWAN balances the tradeoff between capacity gain from capacity adaptation and capacity loss from link downtime from by incorporating the decision of adapting link capacities in the traffic engineering algorithm. The formulation maximizes bandwidth allocations in the network while minimizing the network churn resulting from a capacity reconfiguration event.

B4 and After [34] (2018). This work describes the evolution of B4 from best-effort content copy-workloads to carrier-grade availability and 100X more traffic. As the scale and reliability demands expanded, the B4 architecture evolved into *hierarchical-centralized* with each B4 site consisting of fully meshed super nodes. This mesh of super nodes at each site allows for failures of inter-site links or intra-site super nodes. However, hierarchical TE introduces implementation challenges – requiring 8X more forwarding rules in switching chips than supported by current switches. The authors propose mechanisms to handle such scaling challenges in this work.

SMORE [26] (2018). The key insight of this work is that the tunnels for traffic engineering impact important characteristics like load balancing and reaction to link failures. The authors propose to use oblivious routing to compute the set of tunnels between source and demand node pairs. These tunnels are then used as input to the TE linear program.

TeaVaR [9] (2019). While previous work had devised techniques to engineer traffic in the presence of k simultaneous link failures, this work notes that not all failures are equally likely. Thus, preparing for unlikely failures can lead to network operators over-provisioning capacity. To prevent this, this work proposes engineering traffic in the presence of probabilistic link failures, taking into account the likelihood of link failures in bandwidth allocation decisions.

Cascara [2] (2021). Cascara is a cloud edge TE system that aims to reduce the cost of inter-domain bandwidth incurred by cloud providers. Cascara leverages *latency equivalent* peer links from the cloud to reach clients such that the percentile billing costs are minimized without impacting latency significantly.

NCFlow [1] (2021). Develops a solution for decentralized traffic engineering.

3.2 ISP traffic engineering

In late 1990s, traffic on the Internet was growing rapidly and ISPs were devising techniques to meet these demands. It was soon realized that rapid capacity provisioning alone would not be enough to meet the growing demands and meeting performance goals. As a result, ISPs invested in measuring and modeling their backbones to deploy mechanisms for allocating demands on desired paths through the network. The overall goal of *engineering traffic* within the ISP network was to efficiently utilize the expensive network resources to achieve performance goals (*e.g.*, minimal latency, maximal throughput, fairness between customer allocations).

Over time, reliable delivery of customer traffic became an important concern for networks. As a result, networks began to have multiple upstream providers, commonly called *multi-homing*. Multihoming enabled redundancy in upstream providers, reducing reliance on a single service provider. Due to the existence of multiple inter-domain links to reach the same client, ISPs also began to engineer *edge traffic* for reliability, performance and peering costs. In the following sections, we discuss intra-domain (§3.2.1) and inter-domain (§3.2.2) traffic engineering systems and engineering efforts for ISP networks.

3.2.1 Intra-domain traffic engineering We summarize representative intra-domain TE literature.

TE with MPLS [7, 8, 33] (1999 – 2002). In this research direction, authors describe mechanisms of implementing traffic engineering using Multi-protocol Label Switching (MPLS), constraint-based routing and interior gateway protocols (IGPs).

ATT NetScope [14] (2000). NetScope generates global views of the network from configuration, topology and traffic demand measurements. It leverages the global view to enable operators to answer "what-if" questions about various traffic engineering scenarios.

TE with intra-domain protocols [15–17] (2000 – 2002). In these works, authors design mechanisms to leverage traditional intra-domain routing protocols like OSPF and IS-IS to implement traffic engineering. The primary mechanism of influencing the operation of shortest-path based routing protocols like OSPF is modifying the edge weights to achieve desirable traffic allocations in the network. Fortz and Thorup [17] design algorithms to achieve traffic engineering goals by modifying link weights appropriately and minimally to reduce convergence time.

TE with robustness to traffic changes [5, 6] (2003, 2004). This direction of work aims to understand the relationship between the accuracy of predicted traffic matrices and the efficiency of TE algorithms that rely on the predictions. The authors introduce the idea of *oblivious performance ratio* of traffic engineering schemes under possible traffic matrices. The oblivious ratio is the worst performance ratio a TE algorithm can get with respect to all possible traffic matrices. Often the oblivious performance ratio can be as high as 2, implying that to be robust to changes in the traffic matrix, the TE schemes achieve link utilizations that are 2X worse than the optimal.

MATE [12] (2001) and TeXCP [24] (2005). MATE and TeXCP are *online* traffic engineering systems that react to instantaneous changes in traffic. MATE leverages MPLS tunnels for TE to minimize network delays. TeXCP implements online traffic-engineering decisions through distributed router-level agents in ISP networks. In contrast with previous approaches, TeXCP actively probes TE tunnels for availability and utilization to inform traffic allocations made by distributed router agents. Additionally, TeXCP has mechanisms enabling routers to give feedback to senders to prevent oscillatory behavior resulting from adaptive traffic allocation decisions.

COPE [32] (2006). This work argues to strike a balance between TE schemes that rely on traffic matrix predictions and those that are oblivious to traffic matrices [5, 6]. This class of TE algorithms, called common-case optimization with penalty envelope (COPE), optimize routing for predicted demands to achieve high efficiency under normal network conditions. They also bound the worst-case performance penalty to ensure acceptable performance under unpredictable changes in the traffic matrix.

Cooperative TE [11, 22] (2009). This work proposes various degrees of co-operation between ISPs and content providers on the Internet to achieve better traffic engineering for the ISP and better server selection (SS) for the content provider. The authors propose distributed traffic engineering and server selection algorithms with information sharing between the ISP and content provider to achieve the Nash bargaining solution of the combined TE-SS problem.

3.2.2 Inter-domain traffic engineering We summarize selected inter-domain TE literature.

A Measurement-Based Analysis of Multihoming [3] (2003).

Guidelines for inter-domain TE [13] (2003).

References

[1] 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. https://www.usenix.org/conference/nsdi21/presentation/abuzaid

[2] 2021. Cost-effective Cloud Edge Traffic Engineering with Cascara. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association. https://www.usenix.org/conference/nsdi21/presentation/singh

[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 (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, Lee Breslau, and Edith Cohen. 2004. Coping with Network Failures: Routing Strategies for Optimal Demand Oblivious Restoration. *SIGMETRICS Perform. Eval. Rev.* 32, 1 (June 2004), 270–281. https://doi.org/10.1145/1012888.1005719

[6] 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 (SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 313–324. https://doi.org/10.1145/863955.863991

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

[8] 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

[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] A. Destounis, G. Paschos, S. Paris, J. Leguay, L. Gkatzikis, S. Vassilaras, M. Leconte, and P. Medagliani. 2018. Slice-based column generation for network slicing. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1–2. https://doi.org/10.1109/INFCOMW.2018.8406982

[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] A. Feldmann and J. Rexford. 2001. IP network configuration for intradomain traffic engineering. *IEEE Network* 15, 5 (2001), 46–57. https://doi.org/10.1109/65.953233

[16] 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

[17] 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

[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] 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.

[20] 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.

[21] 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 (SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 73–86. https://doi.org/10.1145/2934872.2934893

[22] 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

[23] 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 (SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 87–100. https://doi.org/10.1145/2934872.2934904

[24] 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 (SIGCOMM '05)*. Association for Computing Machinery, New York, NY, USA, 253–264. https://doi.org/10.1145/1080091.1080122

[25] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. [n. d.]. Calendaring for Wide Area Networks. In *SIGCOMM'14*.

[26] 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

[27] 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

[28] 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 (IMC '11)*. Association for Computing Machinery, New York, NY, USA, 463–472. https://doi.org/10.1145/2068816.2068859

[29] 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.

[30] 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 (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 547–560. https://doi.org/10.1145/3230543.3230570

[31] 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

[32] 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

[33] 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

[34] 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

[35] 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.

[36] 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 (NSDI'10)*. USENIX Association, USA, 3.