Walking the Tightrope: Responsive Yet Stable Traffic Engineering

Srikanth Kandula
MIT CSAIL
kandula@mit.edu

Dina Katabi
MIT CSAIL
dk@mit.edu

Bruce Davie
Cisco Systems
bdavie@cisco.com

Anna Charny
Cisco Systems
acharny@cisco.com

ABSTRACT

Current intra-domain Traffic Engineering (TE) relies on *offline* methods, which use long term average traffic demands. It cannot react to realtime traffic changes caused by BGP reroutes, diurnal traffic variations, attacks, or flash crowds. Further, current TE deals with network failures by pre-computing alternative routings for a limited set of failures. It may fail to prevent congestion when unanticipated or combination failures occur, even though the network has enough capacity to handle the failure.

This paper presents TeXCP, an *online* distributed TE protocol that balances load in realtime, responding to actual traffic demands and failures. TeXCP uses multiple paths to deliver demands from an ingress to an egress router, adaptively moving traffic from over-utilized to under-utilized paths. These adaptations are carefully designed such that, though done independently by each edge router based on local information, they balance load in the whole network without oscillations. We model TeXCP, prove the stability of the model, and show that it is easy to implement. Our extensive simulations show that, for the same traffic demands, a network using TeXCP supports the same utilization and failure resilience as a network that uses traditional offline TE, but with half or third the capacity.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols; C.2.3 [**Computer Communication Networks**]: Network Operations—*Network Management*

General Terms

Algorithms, Design, Management, Reliability, Performance.

Keywords

TeXCP, Traffic Engineering, Responsive, Online, Distributed, Stable.

1. INTRODUCTION

Intra-domain Traffic Engineering (TE) is an essential part of modern ISP operations. The TE problem is typically formalized as minimizing the maximum utilization in the network [5, 6, 15, 26].

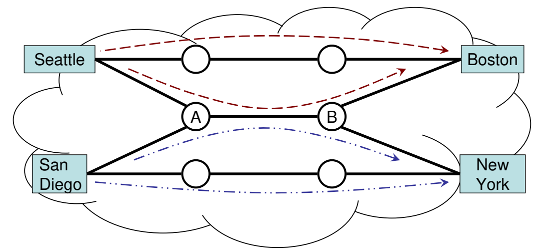


Figure 1: For each Ingress-Egress (IE) pair, there is a TeXCP agent at the ingress router, which balances the IE traffic across available paths in an online, distributed fashion.

This allows the ISP to balance the load and avoid hot spots and failures, which increases reliability and improves performance. Furthermore, ISPs upgrade their infrastructure when the maximum link utilization exceeds a particular threshold (about 40% utilization [20]). By maintaining lower network utilization for the same traffic demands, traffic engineering allows the ISP to make do with existing infrastructure for a longer time, which reduces cost.

Recent years have witnessed significant advancements in traffic engineering methods, from both the research and operational communities [6, 12, 15, 40]. TE methods like the OSPF weight optimizer (OSPF-TE) [15, 16] and the MPLS multi-commodity flow optimizer [26] have shown significant reduction in maximum utilization over pure shortest path routing. Nonetheless, because of its *offline* nature, current TE has the following intrinsic limitations:

- It might create a suboptimal or even inadequate load distribution for the realtime traffic. This is because offline TE attempts to balance load given the long term traffic demands averaged over multiple days (potentially months). But the actual traffic may differ from the long term demands due to BGP re-routes, external or internal failures, diurnal variations, flash crowds, or attacks.
- Its reaction to failures is suboptimal. Offline TE deals with network failures by pre-computing alternative routings for a limited set of failures [16]. Since the operator cannot predict which failure will occur, offline TE must find a routing that works reasonably well under a large number of potential failures. Such a routing is unlikely to be optimal for any particular failure. As a result, current TE may fail to prevent congestion when unanticipated or combination failures occur, even though the network may have enough capacity to handle the failure.

The natural next step is to use online traffic engineering, which reacts to realtime traffic demands and failures. Currently, online TE research is still in its infancy. Indeed it is challenging to build a distributed scheme that responds quickly to changes in traffic, yet does not lead to oscillations, as demonstrated by the instability of the early ARPAnet routing [23]. Prior online TE methods are either centralized [9, 10] or assume an oracle that provides global knowledge of the network [12], and most lack a stability analysis [34, 39].

Term	Definition
Network Utilization or Max-Utilization	The maximum utilization over all links in the network
Path Utilization	Maximum link utilization along a path
IE flow	The traffic flow from an ingress to an egress router along a particular path
Active path	A path on which the TeXCP agent is sending traffic (i.e., $x_{sp} > 0$)

Table 1: Definitions of terms used in the paper.

Var	Definition
R_s	Total Traffic Demand of IE pair s
P_s	Set of paths available to IE pair s
r_{sp}	Traffic of IE pair s sent on path p. i.e., $R_s = \sum r_{sp}$
x_{sp}	Fraction of IE, s, traffic sent on path p, called path weight.
u_{sp}	The utilization of path p observed by IE pair s
u_l	The utilization of link l
C_l	The capacity of link l
P_l	Set of paths that traverse link l
$\overline{u}_s$	Weighted average utilization of paths used by IE pair s

Table 2: The variables most-used in the paper. All of these variables are functions of time.

There is a need for an online TE protocol that combines practical implementation, clear performance advantage, and stable behavior. Furthermore, the community needs to quantify the performance gap between online and offline TE.

This paper presents TeXCP, a distributed responsive and stable online traffic engineering protocol. Our approach simplifies the design and analysis of online TE by splitting the problem into two components. First, a load-balancer takes as input the state of the network and shifts traffic from one path to another to minimize the utilization. Second, each path in the network has a closed-loop feedback controller that collects network feedback and ensures traffic stability on the path. Making the feedback controller work at a *faster time scale* than the load balancer achieves multiple goals: 1) The feedback controller is easy to stabilize by building on recent ideas in applying closed-loop feedback control to congestion control [21]. 2) The feedback controller stabilizes the network before the load balancer makes new decisions, giving the allusion of instantaneous feedback. 3) As a result, the load balancer is easier to design and analyze because it can ignore the complexity of the underlying network.

TeXCP works as follows. In an ISP network, like the one in Fig. 1, each ingress router may have traffic demands for a particular egress router or set of routers, e.g., traffic demands from Seattle to Boston, Boston to New York, etc. We assign to each ingress-egress (IE) pair a TeXCP agent that resides at the ingress router and uses multiple paths or tunnels to deliver traffic from the ingress to the egress. The TeXCP agent uses light-weight explicit feedback from the core routers to discover path utilization. It adaptively moves traffic from over-utilized to under-utilized paths. Traffic is split among paths at the granularity of a flow, to avoid reordering TCP packets. TeXCP's load movements are carefully designed such that, though done independently by each edge router based on local information, the system balances load throughout the network. We model TeXCP under standard assumptions and show it is stable. Further, our analysis provides a systematic approach to set the system parameters to constant values that work independent of the traffic demands and the failure scenarios.

Using simulations of multiple tier-1 topologies from Rocketfuel [30], many different traffic matrices, and changing network conditions, we evaluate TeXCP and study the performance gap between online and offline TE methods. Our results can be summarized as follows:

- For the same traffic demands, a network that uses TeXCP can support the same utilization and failure resilience as a network that uses traditional offline TE, but with a half or a third the capacity. This significantly reduces cost for the ISP.
- The network utilization under TeXCP is always within a few percent of the optimal value, independent of failures or deviations from the traffic matrix. In comparison, InvCap weight setting [11], a widely used TE method, results in an average network utilization 80% higher than optimal in the base case, and twice as high under failures. OSPF-TE, a state-of-the-art offline TE technique [15,16], achieves a network utilization that is about 20% away from optimal, in the base case, but it is highly sensitive to failures and deviations from the traffic matrix, sometimes creating a utilization that is twice or thrice the optimal.
- Compared to MATE [12], a prior online TE proposal, TeXCP converges faster, achieves a better load balance, and does not assume an oracle that provides global knowledge of the network.
- TeXCP automatically prunes additional paths whose usage does not reduce the maximum utilization in the network, and prefers shorter paths over longer paths.
- Finally, as explained in §6, TeXCP is easy to implement without any modifications to current router technology.

Our **terminology and variables** are in Tables 1 and 2.

2. PROBLEM FORMALIZATION

In an ISP network, like the one in Fig. 1, each IE pair, s, has input traffic rate R_s and multiple paths P_s that can be used to deliver the traffic from ingress to egress. A fraction of the IE traffic x_{sp} is routed along path p. The problem is: How to split the traffic of each IE pair across its available paths to minimize the maximum link utilization in the network, i.e.:

$$\min_{x_{sp}} \max_{l \in L} u_l, \tag{1}$$

subject to the constraints:

$$u_l = \sum_s \sum_{p \in P_s, p \ni l} \frac{x_{sp} \cdot R_s}{C_l}, \tag{2}$$

$$\sum_{p \in P_s} x_{sp} = 1, \quad \forall s, \tag{3}$$

$$x_{sp} \geq 0, \quad \forall p \in P_s, \forall s. \tag{4}$$

Eq. 1 states the optimization problem showing that we want to find the traffic split ratios, $\{x_{sp}\}$, that minimize the maximum utilization across all links, $l \in L$. Eqs. 2– 4 are the constraints to the optimization problem; Eq. 2 denotes that link utilization, u_l, is the total traffic on the link divided by its capacity, C_l; Eq. 3 ensures that the traffic sent by an IE pair sums up to its demands; Eq. 4 states that the traffic share on any path cannot be negative.

3. TEXCP

In a TeXCP network, edge and core routers collaborate to balance the load and route around failures. The vast majority of the new functionalities are at edge routers, and can be built in software. The ingress router in each IE pair runs a TeXCP agent. The ISP configures each agent with a set of paths it can use to deliver its IE traffic, and pins the paths using a standard protocol like RSVP-TE [7]. The TeXCP agent probes each path to discover its utilization and failure state, and splits the IE traffic across these paths to minimize the max-utilization. It adapts the split ratios in realtime to changing network conditions. The different TeXCP agents work independently, without exchanging information. Their combined effort optimizes the maximum utilization and balances the load.

Before delving into the details, we note that TeXCP does not care which IP prefixes are mapped to which ingress-egress pair. BGP can freely reroute an IP prefix and change its egress points. Furthermore, although our description focuses on point-to-point traffic, the architecture can easily handle point-to-multipoint traffic.[1]

3.1 Path Selection

The ISP configures each TeXCP agent with a set of paths that it can use to deliver traffic between the corresponding IE pair. By default, TeXCP picks the K-shortest paths that connect the ingress to the egress router, where a path length is set to its propagation delay. Though preferable, these paths need not be link-disjoint.

A few points are worth noting. First, the choice of the per-IE paths is based solely on the topology and is independent of the state of each path (e.g., congestion, failure). Thus, paths are computed offline and rarely re-computed. Second, a TeXCP agent uses the realtime congestion and failure state of a path to determine whether to use a path and how much traffic to send on the path. Thus, the actual number of paths used by a TeXCP agent may be much smaller than K and depends on whether increasing the number of active paths decreases the max-utilization. The default is $K = 10$ paths.

3.2 Probing Network State

To balance its IE pair traffic, a TeXCP agent needs to keep track of the utilization of each path available to it (i.e., maximum link utilization along the path). The TeXCP agent maintains a `probe timer` which fires every T_p seconds. T_p should be larger than the maximum round trip time in the network. The default value is 100ms. Smaller values of T_p make TeXCP converge faster whereas larger values decrease the overhead (§6). When the timer fires, the TeXCP agent sends a small probe on each of its paths. A router that sees a probe packet checks whether the utilization reported in the packet is smaller than the utilization of its output link, in which case it overwrites the utilization in the probe with its own. The egress node at the end of the path unicasts the contents of the probe packet to the ingress node, which delivers it to the appropriate TeXCP agent. This packet (i.e., the probe's ack) goes directly to the ingress node and is not processed by intermediate routers.

Note that probe packets, like ICMP packets, do not follow the fast data path, and hence need no modifications to router hardware. Slow path processing may add an extra 1–2ms of delay to the probe at each router [18]. We show that this is negligible for our purpose (§5). Finally, we use probes only for ease of explanation; an equivalent approach with much lower overhead is described in §6.

Probe Loss is an indication of failure (or congestion) along the path. In TeXCP, probes have sequence numbers. If a previous probe is not acknowledged by the next time the probe timer fires, the agent exponentially increases its estimate of corresponding path utilization to $\max(1, \rho u_{sp})$, where u_{sp} is the path utilization from the previous T_p, and ρ is a parameter > 1, that defaults to 1.2. As a result, failed and highly congested paths are quickly recognized, causing TeXCP to divert traffic from them to less congested paths.

3.3 The Load Balancer

Each TeXCP agent runs a load balancer that splits the traffic of an IE pair, s, among its available paths, P_s, with the objective of maintaining the max-utilization in the network as low as possible.[2]

The load balancer maintains a `decision timer`, which fires every T_d seconds. For stability reasons (see §4), the decision interval is set to 5 times the probe interval, i.e., $T_d \geq 5T_p$.

Our distributed load balancer works iteratively; every time the decision timer fires, the load balancer at TeXCP agent s, computes a change in the fraction of IE traffic sent on path p, called Δx_{sp}. This change must satisfy the following constraints:

(1) Once at equilibrium, traffic assignment does not change and $\Delta x_{sp} = 0, \forall s, \forall p \in P_s$.
(2) Conservation of traffic implies $\sum_{p \in P_s} x_{sp} = 1, \forall s$.
(3) A path whose rate is zero cannot have its rate decreased, i.e., $x_{sp} = 0 \Rightarrow \Delta x_{sp} \geq 0$.
(4) Each update should decrease the maximum utilization, i.e., if p is a path with maximum utilization at TeXCP agent s, then either $x_{sp} = 0$ or $\Delta x_{sp} < 0$.

Taking the above into account, we adopt the following load balancing algorithm. When the decision timer fires, the TeXCP agent updates the fraction of traffic on its paths as follows:

$$\Delta x_{sp} = \begin{cases} \frac{r_{sp}}{\sum_{p'} r_{sp'}}(\overline{u}_s - u_{sp}) & \forall p, u_{sp} > u_{min} \\ \frac{r_{sp}}{\sum_{p'} r_{sp'}}(\overline{u}_s - u_{sp}) + \epsilon & p, u_{sp} = u_{min}. \end{cases} \tag{5}$$

The term r_{sp} is the draining rate of path $p \in P_s$, which is computed as the number of bits sent on the path since the last time the decision timer fired divided by the decision interval T_d. The term u_{sp} is the most recent value of the path's utilization reported by the probes, u_{min} is the minimum utilization across all paths controlled by this agent, $0 < \epsilon \ll 1$ is a small positive *constant*, and $\overline{u}_s$ is the average utilization normalized by the rates, i.e.:

$$\overline{u}_s = \frac{\sum_{p \in P_s} r_{sp} \cdot u_{sp}}{\sum_{p' \in P_s} r_{sp'}}. \tag{6}$$

Before assigning traffic to paths, we normalize the traffic fractions, x_{sp}, to ensure they are positive and sum up to 1:

$$\hat{x}_{sp} = max(0, x_{sp} + \Delta x_{sp}), \tag{7}$$

$$x_{sp} = \frac{\hat{x}_{sp}}{\sum \hat{x}_{sp'}}. \tag{8}$$

Eq. 5 is fairly intuitive. In particular, setting $\Delta x_{sp} \propto (\overline{u} - u)$ means that any path whose utilization is above the average utilization should decrease its rate whereas any path whose utilization is below average should increase its rate. But this is not enough. Paths with larger minimum capacity need more traffic to achieve the same utilization as smaller capacity paths. To cope with this issue, Eq. 5 sets $\Delta x_{sp} \propto r_{sp}$. This makes the change in a path's traffic proportional to its current traffic share, which is the contribution of this TeXCP agent to the path utilization.

Additionally, Eq. 5 uses the normalized average utilization rather than the standard average to deal with inactive paths, i.e., paths for which $r_{sp} = 0$. Assume one of the paths controlled by the TeXCP agent has a very high utilization compared to the others, but the TeXCP agent is not sending traffic on that path ($r_{sp} = 0$). The TeXCP agent cannot move traffic from that path because it is not sending any traffic on it. Yet, the high utilization of this inactive path might make the unnormalized average utilization much higher than all other utilizations. This may prevent the TeXCP agent from moving traffic away from other highly utilized paths, which halts

[1] Multi-exit traffic can be dealt with by imagining that the potential exit routers are connected to a virtual egress node.

[2] For the interested reader, we note that the standard approach to solve such optimizations is to make small iterative adjustments in the direction of gradient descent of the cost function. But, this does not work here because our cost function $\max u_l$ is not differentiable everywhere. Further, even when differentiable, the derivative is zero for all paths not traversing the max-utilization link. Picking a different cost function, such as $\max u_l^j$ where $j \gg 1$, avoids this problem but makes convergence fairly slow.

the optimization. To prevent such a situation, Eq. 6 normalizes the average utilization by the rates.

The last point to note about Eq. 5 is the use of ϵ. Since TeXCP performs online load balancing, it must react when changing network conditions reduce load on a previously congested path. Without ϵ, $r_{sp} = 0 \Rightarrow \Delta x_{sp} = 0$, which means that if at some point the algorithm decided a particular path is highly utilized and should not be used, it will never use it again even if the utilization decreases later. To prevent this situation, Eq. 5 always moves a small amount of traffic to the path with the minimum utilization (unless all paths have the same utilization). ϵ is a small positive *constant* chosen according to Theorem 4.2 (Eq. 21), to ensure convergence.

Finally, for practical reasons, we would like to use a small number of paths for each IE pair, even if that makes the max-utilization slightly higher than the optimal achievable value. Thus, TeXCP stops using a path when its rate falls below a threshold (e.g. less than 10% of the total traffic of the IE pair). More formally:

$$u_{sp} > 0.9\, u_{max} \;\; and \;\; x_{sp} < \sigma \Rightarrow x_{sp} = 0, \tag{9}$$

where u_{max} is the maximum utilization over all paths controlled by agent s, and σ is a small constant set by default to 0.1.

3.4 Preventing Oscillations and Managing Congestion

On the face of it, realtime load balancing appears simple; all one has to do is to iteratively move traffic from the over-utilized paths to the under-utilized paths until the utilization is balanced. Unfortunately, this is over-simplistic. Early experiences in the ARPAnet show that adaptive routing protocols which react to congestion can lead to persistent oscillations and instability [23]. Stability remains a challenge for the success of any adaptive protocol, particularly when it has to respond quickly [36].

Fig. 1 shows an example of how oscillations might occur in a distributed adaptive load balancing protocol. There are two TeXCP agents, one controls the traffic from Seattle to Boston, and the other controls the traffic from San Diego to New York. Each agent has two paths, as shown. At first, the middle path, (link A−B), is under-utilized and both agents decide to shift traffic to it. Each agent moves traffic without knowing that the other is doing the same, and as a result the total traffic on link A−B becomes larger than anticipated by any of the TeXCP agents. This causes both agents to move traffic away from the middle path, leading to under-utilization; and the cycle repeats resulting in traffic oscillations. The actual situation is even harder as each link could easily be shared by hundreds of IE pairs. Network delays and drops complicate this situation further as the two agents use obsolete knowledge of utilization and may substantially overshoot the target utilization.

Solution Idea: TeXCP addresses this oscillation problem by borrowing ideas from recent congestion control protocols, in particular XCP [21]. There is a subtle yet important connection between congestion control and load balancing. Both deal with the send rate of flows; congestion control deals with flows from senders to receivers; load balancing deals with flows from ingress to egress routers. As described above, uncoordinated actions of multiple TeXCP agents that share a path may cause traffic oscillations. Similarly, in congestion control, if multiple sources that share a bottleneck link adjust their rates independently, they can cause the utilization of the bottleneck link to oscillate. For example, if the total increase exceeds the spare bandwidth, the bottleneck link gets congested causing all the sources to back off together, which in turn leads to under-utilization. Similarly to XCP, TeXCP solves the oscillation problem by using explicit feedback from the routers. In particular, the router at link A−B issues feedback such that the traffic increase by the two TeXCP agents never overshoots the capacity of link A−B. Below we explain the feedback computation in detail.

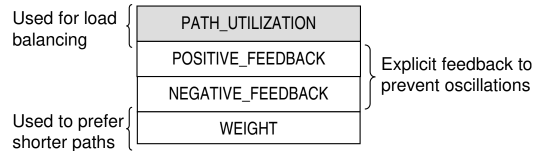


Figure 2: Probe Packet. Feedback is returned in two fields because Positive Feedback is additive while Negative Feedback is multiplicative.

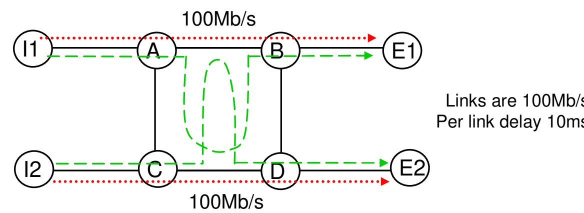


Figure 3: Simple topology with many optimal traffic splits; all except sending on the shortest paths cause suboptimal delays.

Solution Detail: We need to (a) compute the appropriate feedback that prevents overshooting,[3] (b) divide it among the IE flows traversing the link, and (c) return a per IE-flow feedback to the TeXCP agents.

(a) Computing aggregate feedback: Every T_p, core routers compute the link utilization and the explicit feedback. Link utilization, u_l, is the average amount of traffic sent down the link in the last T_p, whereas aggregate feedback, Φ, is an upper bound on how much the *aggregate* traffic on the link should increase/decrease by. The core router makes the increase proportional to the spare bandwidth, S (S = capacity − load), and the decrease proportional to the queue size, Q. Hence,

$$\Phi = \alpha \cdot T_p \cdot S - \beta \cdot Q, \tag{10}$$

where α and β are constant parameters chosen according to Theorem 4.1 to ensure stability (see §4).

(b) Computing per-IE-flow feedback: Next, we need to divide the aggregate feedback, Φ, among the IE flows traversing the link. We adopt a Max–Min allocation, i.e., when all IE pairs have enough traffic demands to consume their allocated bandwidth shares the bandwidth is divided equally, but if an IE flow does not have enough demand to consume its fair share, the extra bandwidth is divided fairly among those IE flows which do have demands, and so on. Indeed by imposing a notion of fairness among IE pairs, we prevent congested IE pairs from starving others (e.g., a huge increase in the traffic from Seattle to Boston does not starve the San Diego to Boston traffic, even on the shared links).

So how to divide the aggregate feedback, Φ, among the IE flows to achieve Max-Min fairness? The standard approach is to use Additive-Increase Multiplicative-Decrease (AIMD). The core router, based on its congestion state, computes either a positive feedback δ^+ or a negative feedback δ^- and sends it to each flow:

$$\begin{aligned} \Phi \geq 0 \quad &\Rightarrow \quad \delta^+ = \frac{\Phi}{N},\ \delta^- = 0, \\ \Phi < 0 \quad &\Rightarrow \quad \delta^+ = 0,\ \delta^- = \frac{\Phi}{\phi_l}, \end{aligned} \tag{11}$$

[3] To the interested reader, we note that our feedback is a much simplified version of XCP's feedback. Core routers distribute feedback to probe packets in contrast to XCP's use of headers in data packets. Also, core routers function with less information; they do not need flow's throughput/cwnd as probes come at a constant rate. Also, instead of per-flow RTT, routers use T_p, a constant upper bound on RTT.

where ϕ_l is the aggregate load on link l, and N is the number of IE flows. Since each IE flow sends one probe every T_p, a router can easily estimate N by counting the number of probes during T_p.[4]

(c) Sending the feedback to the TeXCP agents: Core routers communicate the feedback to the TeXCP agents by annotating their probes (§3.2). A probe contains the fields in Fig. 2, which get overwritten by each core router along the path as follows:

PATH_UTILIZATION = *max* (PATH_UTILIZATION, u_l)
POSITIVE_FEEDBACK = *min* (POSITIVE_FEEDBACK, δ^+)
NEGATIVE_FEEDBACK = *max* (NEGATIVE_FEEDBACK, δ^-)

When the probe reaches the egress router, the router unicasts these values to the corresponding TeXCP agent at the ingress (§3.2). The feedback in the probes bounds the send rate along any path. In particular, each TeXCP agent, s, maintains an additional per path variable g_{sp}, which is the allowed rate on path p. When it receives a probe ack, it updates g_{sp} as:

$$g_{sp} = g_{sp} + \delta^+ - \delta^- \times g_{sp},$$

where δ^+ and δ^- here are the positive and negative feedback from the probe ack. The actual rate on path p is the minimum of the allowed rate, g_{sp}, and the demand, $r_{sp} = x_{sp} \cdot R_s$. If the demands are larger than the allowed rate along a path, the extra traffic is queued at the edge router and dropped if the buffer overflows.

3.5 Shorter Paths First

Minimizing the maximum utilization need not have a unique solution. Fig. 3 shows two IE pairs, each with a short and a long path. The maximum utilization does not change whether both IE pairs split traffic equally between the two paths or send only on the shortest path.

We want a TeXCP agent to prefer shorter routes over longer ones as long as that choice does not increase the maximum utilization of the network. Thus, we bias the allocation of link bandwidth towards giving more share to IE flows which have this link on the shortest path. Instead of using Max–Min fairness in allocating the explicit feedback (§3.4), we use Weighted Max–Min fairness, where longer paths have smaller weights. In particular, a TeXCP agent s assigns to each path p a weight $v_{sp} = \frac{1}{d_{sp}^{\gamma}}$, where d_{sp} is the propagation delay on path p, and $\gamma > 1$ is a parameter that controls the bias against longer paths. The default is $\gamma = 3$. The TeXCP agent estimates d_{sp} by measuring the probe RTT, and communicates this to the core router through a field in the probe (Fig. 2).

To produce Weighted Max–Min Fairness instead of Max–Min Fairness, a core router replaces the sum of IE flows, N, in Eq. 11 with a weighted sum as follows:

$$\Phi \geq 0 \quad \Rightarrow \quad \delta^+ = \frac{v_{sp}\Phi}{\sum_{p' \in P_l} v_{sp'}},\ \delta^- = 0,$$
$$\Phi < 0 \quad \Rightarrow \quad \delta^+ = 0,\ \delta^- = \frac{\Phi}{\phi_l}. \qquad (12)$$

Note the negative feedback does not change as it is already a proportional factor. We use Eq. 12 in all our experiments.

4. ANALYSIS

To make the analysis tractable, we adopt a fluid model and assume queues and traffic are unbounded. We also assume that each path has a single bottleneck and that the TeXCP load balancers are synchronized. Finally, we assume that IE traffic demands change at time scales much larger than the dynamics of the TeXCP load balancer ($\gg T_d$), and thus do not affect stability. These assumptions, though simplistic, are commonly used in the literature [21, 25] to make analysis manageable. Simulations in §5 show that TeXCP works properly even when these assumptions do not hold.

ISP (AS#)	Where?	Rocketfuel routers	Rocketfuel links	PoPs cities	PoPs links
Ebone (1755)	Europe	87	322	23	38
Exodus (3967)	Europe	79	294	22	37
Abovenet (6461)	US	141	748	22	42
Genuity (1)	US	-	-	42	110
Sprint (1239)	US	315	1944	44	83
Tiscali (3257)	Europe	108	306	50	88
AT&T (7018)	US	-	-	115	296

Table 3: Rocketfuel topologies used in evaluation.

Proof Idea: Our approach is to decouple the effect of network delays from load balancing. We ensure that when a TeXCP load balancer shifts traffic from one path to another, the impact of this traffic shift on link utilizations stabilizes before any load balancer makes a new decision. If this is the case, we can analyze the stability of the load balancer ignoring network delays and assuming that traffic changes take effect instantaneously. Thus, our stability analysis has the following 3 steps.

Step 1: First, we prove that explicit feedback stabilizes the per-IE flow rates, and consequently the utilizations are stable.

THEOREM 4.1. *Let $d > 0$ be the round trip delay in the ISP network, if the parameters T_p, α, and β satisfy:*

$$T_p > d, \quad 0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad \text{and} \quad \beta = \alpha^2\sqrt{2},$$

then the aggregate per-IE flow on a link is stable independently of the delay, capacity, and number of IE flows.

PROOF: See Appendix A. ∎

Step 2: Second, we show that explicit feedback brings path utilization to within 90% of the desired value before the load balancer makes a new decision, i.e., by the next T_d. In particular, we pick $\alpha = 0.4$ and $\beta = 0.226$ in accordance with Theorem 4.1. In the worst case, the utilization has to ramp up from u_l=0 to full utilization (or the opposite). For α=0.4, starting with u_l=0, five iterations of Eq. 10 cause the utilization to be more than 90%[5]. Thus, by picking $T_d = 5T_p$, the load balancer allows the explicit feedback enough time to bring the utilization to about 90% of the value dictated by the previous decision, before making a new decision.

Step 3: Finally, using steps 1 and 2 to assume that traffic shifts take effect instantaneously, we prove that independently acting TeXCP load balancers do not cause oscillations.

THEOREM 4.2. *The TeXCP load balancer is stable and converges to a state in which every TeXCP agent sees a balanced load on all of its active paths (paths on which the agent sends traffic). Further, all inactive paths at an agent (paths on which the agent sends no traffic) have higher utilization than the active paths.*

PROOF: See Appendix B. ∎

Note that Theorem 4.2 proves the system is stable, shows no oscillations, and balances the load, but does not prove optimal max-utilization. We conjecture that giving preference to shorter paths, as we do, is important for achieving optimality. Simulations in §5 with multiple tier-1 ISP topologies, many different traffic matrices and changing network conditions, show that TeXCP is always within few percent of the optimal max-utilization, and is much closer to optimal than alternative approaches.

[4]Indeed, the router may know the number of IE flows without any computation; if paths are pinned with MPLS, then N is just the number of transit LSPs.

[5]In 5 iterations, a spare bandwidth of 100% goes to $(1 - 0.4)^5 < 10\%$.

ISP(AS#)	# of paths used	
	avg	std
Ebone(1755)	4.275	1.717
Exodus(3967)	4.769	1.577
Abovenet(6461)	4.653	2.038
Genuity(1)	4.076	1.806
Sprint(1239)	4.175	1.935
Tiscali(3257)	4.525	1.980
AT&T(7018)	3.976	1.785

Table 4: Though a TeXCP agent is configured with a maximum of K =10 paths, it achieves near-optimal max-utilization using many fewer paths.

Technique	Description	Distributed?	Reacts to changes in traffic?	Robust to failures?
Oracle	LP based on multi-commodity	No	No	No
TeXCP	in §3.3 and §3.4	Yes	Yes	Yes
OSPF-TE$_{Base}$	Optimal Link weights for a TM [15]	No	No	No
OSPF-TE$_{Failures}$	Opt. weights for few critical failures [16]	No	No	Limited number of anticipated failures
OSPF-TE$_{Multi-TM}$	Opt. weights over multiple TMs [15]	No	Optimizes over multiple demands	No
MATE	in [12]	Sim. needs global knowledge	Yes	Yes
InvCap	Common Practice	-	No	No

Table 5: Various load balancing techniques.

5. PERFORMANCE

We evaluate TeXCP and compare it with prior work.

5.1 Topologies & Traffic Demands

ISPs regard their topologies and traffic demands as proprietary information. Thus, similar to prior work [6, 29], we use the Rocketfuel topologies in Table 3. To obtain approximate PoP to PoP topologies, we collapse the topologies so that "nodes" correspond to "cities". Rocketfuel does not provide link capacities; so we assign capacities to links as follows. There is a marked knee in the degree distribution of cities–i.e., cities are either highly connected (high-degree) or not. The high degree cities are probably Level-1 PoPs [20], with the rest being smaller PoPs. We assume that links connecting Level-1 PoPs have high capacity (10Gb/s) and that the others have smaller capacity (2.5Gb/s). This is in line with recent ISP case studies [1, 20].

Similarly to [6], we use the gravity model to compute estimated traffic matrices. This approach assumes that the incoming traffic at a PoP is proportional to the combined capacity of its outgoing links. Then it applies the gravity model [33] to extrapolate a complete TM. The TMs used in our experiments lead to max. utilizations in the range 25-75%. For lack of space, we omit similar results for bimodal TMs [6] and topologies generated using GT-ITM [19].

5.2 Metric

As in [6], we compare the performance of various load balancing techniques with respect to a particular topology and traffic matrix (TM) using the ratio of the max-utilization under the studied technique to the max-utilization obtained by an oracle, i.e.:

$$Metric = \frac{\text{max-utilization}_{\ Tech.}}{\text{max-utilization}_{\ Oracle}}.$$

5.3 Simulated TE Techniques

We compare the following techniques (see Table 5):

(a) Oracle: As the base case for all our comparisons, we use Matlab's linprog solver to compute the optimal link utilization for any topology and traffic matrix. This is the standard off-line centralized oracle which uses instantaneous traffic demands and solves the multi-commodity flow optimization problem [26].

(b) TeXCP: We have implemented TeXCP in ns2 [27]. The implementation uses Eqs. 5,12. The TeXCP probe timer is set to $T_p = 0.1$s, and thus $T_d = 0.5$s. TeXCP uses the constants $\alpha = 0.4$ and $\beta = 0.225$ as per Theorem 4.1. The processing time of a probe at a core router is uniformly distributed in [0,2]ms, consistent with Internet measurements of the delay jitter for packets processed on the slow path [18]. Packet size is 1KB, and buffers store up to 0.1s.

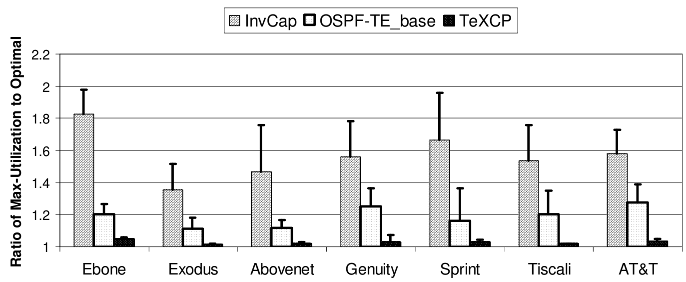


Figure 4: When traffic matches TM, TeXCP results in a max-utilization within a few percent of the optimal, and much closer to optimal than OSPF-TE or InvCap. Figure shows both average (thick bars) and maximum (thin bars) taken over 40 TMs.

(c) OSPF-TE: We implemented 3 versions of the OSPF weight optimizer. The first, which we call **OSPF-TE**$_{Base}$, is from [15]. Given a traffic matrix, it searches for link weights that result in low max-utilization.[6] The second, **OSPF-TE**$_{Failures}$, computes link weights that result in low max-utilization even when few critical failures happen [16]. The third, **OSPF-TE**$_{Multi-TM}$, simultaneously optimizes weights for multiple traffic matrices. Our implementation gives results consistent with those in [15, 16].

(d) MATE: We compare the performance of TeXCP with MATE, a prior online TE protocol [12]. MATE's simulation code is proprietary. Therefore, we compare TeXCP against MATE's published results [12], after consulting with the authors to ensure that the simulation environments are identical.

(e) InvCap: A common practice sets a link weight to the inverse of its capacity and runs OSPF [11].

5.4 Comparison With the OSPF Optimizer

We would like to understand the performance gap between online and offline traffic engineering. No prior work provides a quantitative comparison of these two approaches. Hence, in this section, we compare TeXCP with the OSPF weight optimizer (OSPF-TE), one of the more sophisticated and highly studied offline TE techniques [15, 16]. Given a topology and a traffic matrix, OSPF-TE computes a set of link weights, which when used in the OSPF intra-domain routing protocol produce a routing with low max-utilization. We also compare against InvCap, a common practice that sets link weights to the inverse of link capacity.

(a) Static Traffic: First, we investigate the simplest case in which IE traffic demands are static, i.e., the actual realtime traffic completely matches the long term demands in the TM.

[6] It minimizes the total cost in the network; where cost is assigned to each link based on a piece-wise linear function of the link utilization [15].

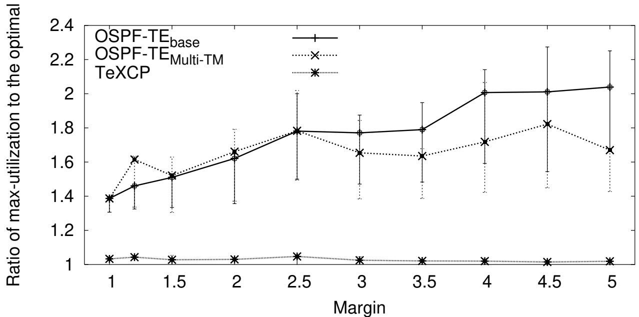


Figure 5: When traffic deviates from TM by a margin, TeXCP stays within a few percent of the optimal max-utilization; OSPF-TE$_{Base}$ and OSPF-TE$_{Multi-TM}$ lead to much larger max-utilization.

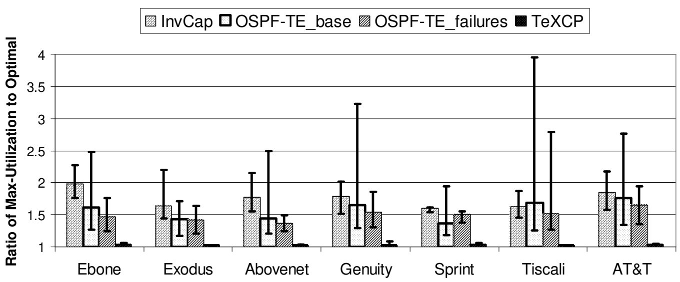


Figure 6: Under failures, TeXCP's max-utilization is within a few percent of the optimal; InvCap, OSPF-TE$_{Base}$, and OSPF-TE$_{Failures}$ become highly suboptimal. Figure shows the $90th$ percentile (thick) and maximums (thin) taken over multiple TMs.

Fig. 4 plots the ratio of the max-utilization under a particular technique to the optimal max-utilization for that topology and TM. The figure shows both the average taken over 40 different TMs and the maximum value. *The figure shows that using a TE technique such as OSPF-TE or TeXCP can substantially reduce the network utilization.* For example, for the AT&T network, using InvCap weights produces a network utilization that is on average 60% higher than the optimal. Using OSPF-TE produces an average network utilization that is 20% higher than the optimal, whereas with TeXCP the value is less than 5% higher than the optimal. These results have direct implications on the required link capacity–i.e., under static demands, a network that runs TeXCP can support the same required demands, under the same utilization, but with 55% less capacity than a network that uses InvCap weights and 15% less capacity than a network that uses OSPF-TE.

One might wonder why OSPF-TE does not achieve optimal max-utilization though the traffic demands are static. Indeed, finding optimal link weights that minimize the max-utilization is NP-hard. OSPF-TE uses a heuristic to search for a good weight setting [15], but is not guaranteed to find the optimal one. TeXCP stays within a few percent of the optimal max-utilization; the small deviations from optimal are likely due to the limited number of paths.

(b) Deviation from Long Term Demands: OSPF-TE does not rebalance the load when the realtime demands deviate from the long term averages provided in the TM. Theoretically, one can compute new weights using the new traffic demands, but in reality computing new weights takes time and accurately estimating changing traffic matrices is difficult [15]. More importantly, the new weights have to be pushed to the routers and the OSPF routing protocol has to rerun and converge to new routes. Rerunning OSPF with new weights may cause transient loops and substantial congestion [8] before routes converge, so ISPs avoid it if possible [20].

To cope with deviations from the TM, OSPF-TE uses a special tweak which we call OSPF-TE$_{Multi-TM}$. This technique optimizes the link weights for multiple traffic matrices at once (e.g., the peak hour TM, the off-peak TM, . . .).

Fig. 5 compares the max-utilization under OSPF-TE$_{Base}$, OSPF-TE$_{Multi-TM}$, and TeXCP as the actual traffic demands deviate from the long term average demands expressed in the traffic matrix. The figure is for the AT&T topology. The x-axis is the deviation margin; e.g., a margin of 1.5 means that the actual demands are randomly chosen to be at most 50% away from the long term demands [6]. The graph shows the average and standard deviation, over 40 TMs with that margin. *The figure shows that as the actual traffic deviates from the traffic matrix (long term demands), the performance of OSFP-TE$_{Base}$ degrades substantially. This is expected as OSFP-TE is an offline protocol that cannot react to changing demands. In contrast, TeXCP reacts in realtime to the actual demands and its performance is always near optimal independent of the margin. OSPF-TE$_{Multi-TM}$ is only marginally better than OSPF-TE$_{Base}$.*

(c) Failures: ISPs provision their network with enough capacity to support the demands if a failure occurs (i.e., max-utilization stays $\ll$ 100%). We investigate the amount of over-provisioning required, under various TE techniques.

Although the OSPF intra-domain routing protocol will re-route around an unavailable link, it does not rebalance the load after a failure. Thus, after a failure, the max-utilization may become very suboptimal. OSPF-TE$_{Failures}$ [16] addresses this issue by optimizing the weights over a set of critical single link failures, those that cause the largest increase in max-utilization upon failure.[7]

We compare the performance of TeXCP, OSPF-TE$_{Failures}$, and OSPF-TE$_{Base}$, under failures. Fig. 6 plots the ratio of the max-utilization under a particular TE technique to the optimal max-utilization, for single link failures. The figure plots the $90th$ percentile and the maximum value taken over all possible single link failures in the given topology. *The figure shows that the 90th percentile of max-utilization under a single link failure is much higher with OSFP-TE than TeXCP.*[8] *These results have interesting implications for capacity provisioning. The figure reveals that for the same level of failure provisioning, an ISP that uses OSPF-TE needs to buy double or triple the capacity needed under TeXCP.* Simulations of multiple links failures (i.e., SRLG failures) show similar results, which we do not report for lack of space.

In summary:

- For the same traffic demands, a network that uses TeXCP can support the same utilization and failure resilience as a network that uses traditional offline TE, but with a half or a third the capacity. This creates a major cost reduction for the ISP.
- The max-utilization under TeXCP is always within a few percent of optimal, independent of failures or deviations from the TM. In comparison, InvCap results in an average max-utilization 60% higher than optimal in the base case, and twice as high under failures. OSPF-TE, achieves a max-utilization that is about 20% away from optimal in the base case, but is highly sensitive to failures and deviations from the TM, sometimes creating a utilization that is twice or thrice the optimal.

5.5 Comparison With MATE

We also compare TeXCP with MATE, a prior online TE proposal [12]. TeXCP borrows from MATE, and prior work on MPLS-TE [26, 39], the idea of building multiple ingress-to-egress tunnels and splitting traffic among them. TeXCP, however, differs from

[7] Optimizing over five most critical failures, as recommended by [16], takes about one hour on a 2GHz, 4GB RAM P4 machine.

[8] The averages over all link failures, represented by the lower end of the thin error bars, also show the same trends.

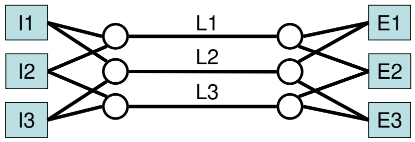


Figure 7: Topology for comparing TeXCP against MATE.

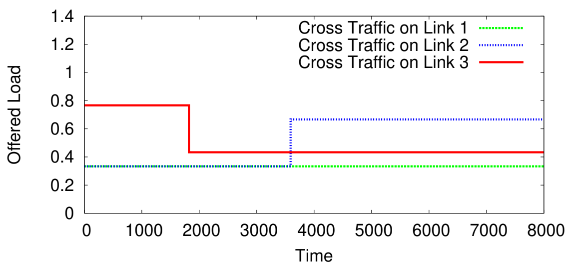
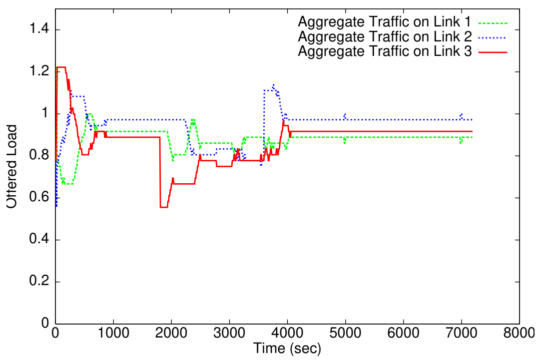
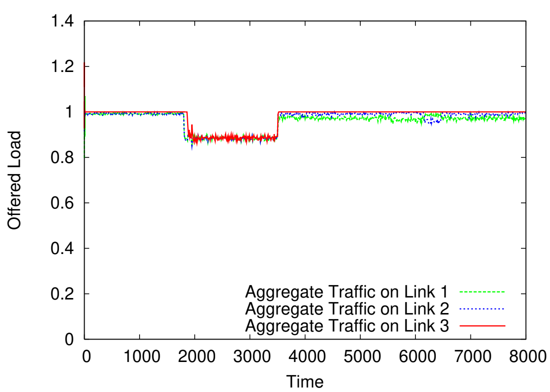


Figure 8: Changes in cross traffic during the MATE simulation.



a) MATE, reproduced from [12]



b) TeXCP

Figure 9: Comparison between the performance of TeXCP and MATE over the topology in Fig. 7 and for the cross traffic in Fig 8. TeXCP converges faster to the optimal balance and exhibits a smoother curve.

MATE in a few important aspects. First, the TeXCP load balancer (Eq. 5) minimizes max-utilization while MATE minimizes the sum of the delays in the network. Unless the network is congested, the delay on a path is constant and equal to propagation delay. Second, TeXCP is fully distributed whereas MATE's simulations assume that ingress nodes have instantaneous knowledge of the whole network state. On the other hand, MATE does not need the core routers to report link utilization to ingresses.

We compare the performance of TeXCP with MATE. MATE's simulation code is proprietary. So, we compare TeXCP against MATE's published results [12], which we reproduce here for convenience. We run TeXCP in the simulation environment reported in [12]. In particular, we use the same topology, simulate traffic demands similarly using Poisson sources, use the same cross-traffic, and split traffic using the hashing method described in [12]. The topology, shown in Fig. 7, consists of 3 ingress-egress pairs sharing 3 bottleneck links. Each bottleneck carries cross traffic uncontrolled by TeXCP. Fig. 8 shows changes in cross traffic during the simulation. All simulation parameters, such as capacities, delays, queue size, number of bins, etc. have been confirmed by one of the authors of MATE. *Our results, in Fig. 9, show that TeXCP is more effective at balancing the load than MATE and converges faster.*

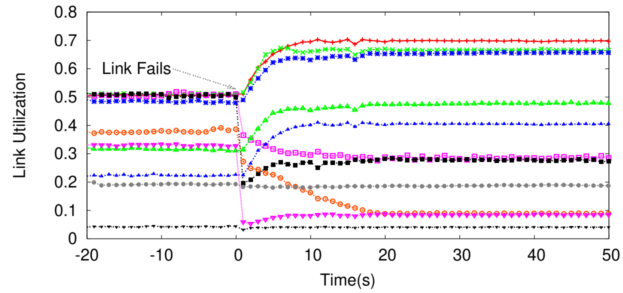


Figure 10: Changes in link utilization during TeXCP convergence for a representative sample of links in the Sprint topology. The individual link utilizations converge smoothly without oscillations.

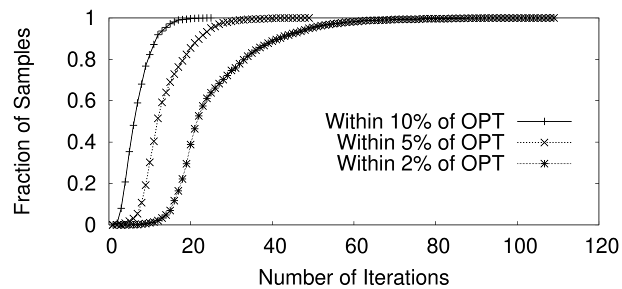


Figure 11: The CDF of TeXCP convergence time for Sprint. TeXCP quickly converges to within 5% to 10% of the optimal.

5.6 TeXCP Convergence Time

As with any online protocol, an interesting question is how long it takes TeXCP to rebalance the load after a failure or a traffic disturbance. The convergence time depends on many factors such as the topology, the maximum number of paths per IE pair, how far the initial traffic splits are from the balanced ones, etc. Fig. 11 shows the CDF of the convergence time for the Sprint topology, where each IE pair is allowed a maximum of $K = 10$ paths. Time is measured as the number of iterations–i.e., number of T_d intervals. Various samples are generated using 200 different TMs, starting at 20 different random initial traffic split ratios. *The figure shows that TeXCP takes only 10-15 iterations to converge to within 10% of the optimal max-utilization, and a few dozens of iterations to converge to 5% of the optimal.* Other topologies show similar trends.

Furthermore, TeXCP converges smoothly without oscillations as predicted by the analysis in §4. Fig. 10 shows the link utilizations for a representative subset of the links in the Sprint topology when an unforeseen event (link failure) happens. It shows that the individual link utilizations steadily converge without oscillations. TeXCP experiments in §5.4 for other topologies and traffic demands show similar trends.

Finally, unlike OSPF convergence which might cause transient loops, the convergence time of TeXCP only affects how far we are

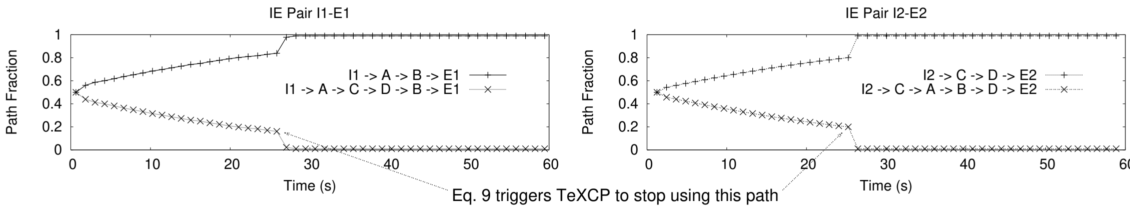


Figure 12: As long as optimality is preserved, TeXCP stops using the longer paths and shifts all traffic to the shorter paths.

from optimal utilization; no loops occur during that interval.

5.7 Number of Active Paths

Although we configure each TeXCP agent with the 10 shortest paths, TeXCP does not use all of these paths. It automatically prunes paths that do not help in reducing max-utilization. Table 4 shows the average number of paths used by each IE pair for the simulations in §5.4. *In our simulations, on average, TeXCP uses only 4 of the 10 per-IE paths it is configured with.*

5.8 Automatic Selection of Shorter Paths

We show an example simulation of how TeXCP automatically prunes longer paths, if they do not reduce the max-utilization. We simulate TeXCP on the simple network in Fig. 3. We have two IE pairs (I1, E1) and (I2, E2), each with unit traffic demands. There are two TeXCP agents, one at each ingress. Each TeXCP has one short path and one long path. Clearly, the max-utilization is the same whether the agents send all of their traffic on shorter paths or split it evenly between the two paths. Fig. 12 shows the fraction of traffic each agent sends on the two paths. Despite being initialized to send equal traffic on the long and short paths, *both TeXCP agents quickly prune the longer paths, routing all traffic on to shorter paths.*

6. IMPLEMENTATION & DEPLOYMENT

This section examines practical considerations for an ISP to deploy TeXCP using current routers. For this section, we assume IE paths are pinned using MPLS Label Switched Paths (LSPs) [32].

(a) Number of LSPs: TeXCP creates K LSPs for each IE pair ($K = 10$ in our experiments). It is common for large ISP backbones to have 200-300 egress points, resulting in 2K-3K LSP *heads* at an ingress router. This is somewhat higher than typically supported in today's backbone routers.[9] We observe, however, that ISPs have many fewer PoPs, 20-100 (Table 3). Since traffic on the ISP backbone essentially moves from one PoP to another, TeXCP tunnels need only be created between PoPs rather than between ingress-egress routers. This commonly used technique in MPLS networks [20] brings the number of LSP heads into line with current technology. We also note that a router can typically support an order of magnitude more *transit* MPLS LSPs than LSP *heads* because transit LSPs need smaller state and processing overhead.

(b) Traffic splitting: TeXCP requires edge routers to split traffic among the LSPs connecting an IE pair. Current ISP-class routers can split traffic to a destination between as many as 16 LSPs, at as fine a resolution as desired [28].

(c) Utilization estimates: TeXCP needs routers to estimate link utilization every T_p seconds. Current edge and core routers maintain counters of the amount of traffic (bytes and packets) sent on each interface. These counters can be read every T_p to get a utilization estimate. T_p=100ms allows counters enough time to stabilize.

[9] Official numbers of supported tunnels are hard to find in vendors' public documentation and tend to increase over time. One Cisco router used in ISP networks supports at least 600 tunnel heads at the time of writing.

(d) Estimating feedback: Estimating the feedback is as easy as estimating the utilization. Eq. 11 shows that all IE flows have the same feedback, be it positive or negative. A router needs to compute this one value for each outgoing interface, once every T_p. Also, the core router can offload the computation entirely, by pushing the variables (S, Q, N, ϕ_l) to the edges.

(e) Communication from core to edge routers: On the first cut, each TeXCP agent sends one *probe*/T_p down all its LSPs. For small networks $n \in [30, 100]$ and low probe rates $T_p \in [.5, 1]s$, this results in a core router processing $1.2 - 8K$ probes/sec/interface, and may be feasible. For larger networks, this is just too many.

Instead, for large networks we suggest a cleaner approach. Each core router generates one *report* packet/T_p, for each outgoing interface, which contains the link utilization and the feedback. Recall from Eq. 11, that all IE flows traversing the link receive the same feedback. Thus, unicasting the report packet to every edge router is equivalent to updating probes of every LSP. Note, a core router only generates n reports/T_p, where n is the number of edge routers, for each outgoing interface. As in §6(a), we note that using PoP to PoP LSPs substantially reduces the amount of traffic. Also note that the edge routers need to do a little more work to consolidate per-link reports into per-LSP state, but that seems a good tradeoff.

(f) Router Modification: TeXCP needs one new functionality; the load balancer (Eq. 5), which is easily built in software in existing edge routers. All other functionality either already exists in current routers or is very similar to existing functionality.

7. RELATED WORK

Several offline TE methods, like the OSPF weight optimizer [15, 16] and the multi-commodity flow formulation [26] were described in §5. Here we mention other relevant efforts.

Optimal Routing: Gallager's seminal work on minimum delay routing [17] began the field of optimal and constraint-based routing [4, 9, 10, 12, 26, 38, 41]. This work studies routing as an optimization problem and usually relies on centralized solutions, global knowledge of the network state, or synchronized nodes.

Offline Oblivious Optimizers: Instead of estimating traffic matrices accurately [13], the MPLS Oblivious optimizer [6] finds a routing that balances load independent of the traffic matrix, i.e. minimizes the worst case across all possible traffic matrices. Similarly, two new schemes [24, 42] balance load in a traffic-insensitive manner by re-routing traffic through pre-determined intermediate nodes; an idea used in processor interconnection networks.

Online Load Balancers: There are relatively few proposals for realtime adaptive multipath routing. We describe MATE [12] and compare it with TeXCP in §5.5. OSPF-OMP [40], an Internet draft, describes another technique that floods the realtime link load information and adaptively routes traffic across multiple paths.

8. FURTHER DISCUSSION

Q: Online multipath routing may reorder TCP packets hurting TCP congestion control. How does TeXCP deal with that?
A: Yes and the easy answer is either keep per-flow state at the edge routers or hash TCP flows into many buckets and assign buckets to the various paths. Neither of these methods is very satisfying. A clean and complete online TE solution has to solve two problems. The first is the intrinsic problem of being responsive yet stable, which we address in this paper. The second is to dynamically split flows without reordering TCP packets, which we address in our Flare paper [35]. We show that splitting TCP flows at the granularity of flowlets–bursts of packets within a TCP flow–balances load accurately without reordering packets and without per-flow state.

Q: Your comparison with OSPF-TE does not agree with previous studies. Can you explain, please? **A:** Actually it does. Papers [14–16] report max-utilization under OSPF-TE and the optimal (e.g., $u_{OSPF-TE} = 0.59$ and $u_{OPT} = 0.48$), whereas this paper reports the max-utilization under the studied scheme relative to the optimal. When expressed using the same metric, the results in [14–16] are consistent with ours. The metric we picked, i.e., comparing schemes in terms of their performance relative to the best solution, is a standard metric for comparative analysis widely-used both in the field of traffic engineering [5, 6] and elsewhere.

Q: Your scheme minimizes the max-utilization, but I am worried that TeXCP will end up sending traffic on very long paths. Don't you think this is an issue?
A: Minimizing the maximum utilization in the network is a widely-used TE metric in research and practice [5, 6, 26]. It removes hot-spots and reduces congestion risks by spreading the load over available paths. Also, it allows a network to support greater demands. Of course, minimizing max-utilization may sometimes increase the delay. Unlike contemporary TE [6, 15], TeXCP allows an ISP to bound this increase in delay. When the ISP configures the Boston-LA TeXCP agent with some paths, it is declaring that all of these paths have acceptable delays. Clearly the ISP should not pick a path that goes to LA via Europe; and it should not need to, given the diversity in ISP topologies [37]. Using the Rocketfuel delay estimates, we have computed the average delay difference between the TeXCP paths and the shortest path weighted by the fraction of traffic on each path. For the experiments in §5.4, this number is [3.2, 10.6] ms. Also, we re-ran the experiments in Fig. 4 by restricting TeXCP to paths that are less than $20ms$ longer than the shortest. In these runs, the weighted average delay difference goes down to [1.2, 2.7] ms. The max-utilization stays similar to Fig. 4, except for the AT&T topology where it increases by a few percent.

Q: What happens if every ISP uses TeXCP?
A: Nothing bad. The same traffic volume still enters and exits an ISP at the same points. TeXCP only balances traffic *within* an ISP and has no globally-visible changes.

Q: Links within an AS go down all the time. Are 10 paths per IE enough for failure recovery?
A: TeXCP does not replace failure discovery and link restoration techniques, such as SONET rings, DWDM optical protection and MPLS fast-reroute [20]. Actually, it complements these techniques: it rebalances the load after link restoration and helps in recovering from unanticipated or combination failures for which the domain doesn't have a pre-computed backup path. For example, about half the ISPs run MPLS in their core. When a link fails, MPLS fast-reroute quickly patches the failed MPLS tunnel with an alternative segment, shifting the traffic to different physical links. This may unbalance the traffic and create hot spots. TeXCP rebalances the traffic allowing the recovery process to go smoothly.

9. CONCLUDING REMARKS

This paper advocates online traffic engineering, which dynamically adapts the routing when traffic changes or links fail, rather than optimizing the routing for long term average demands and a pre-selected set of failures. We present TeXCP, an online distributed TE protocol, show it is stable and that it balances the load and keeps network utilization within a few percent of the optimal value. We also show that TeXCP outperforms traditional offline TE, particularly when the realtime demands deviate from the traffic matrix or unanticipated failures happen.

Although we focus on traffic engineering, our work is part of a general trend away from traditional single-path congestion-insensitive Internet routing towards adaptive routing. This includes overlays [31], adaptive multi-homing [2], and traffic engineering. Recent studies show that overlay routing increases traffic variability making it much harder to estimate the TM needed for offline TE optimization [22, 29]. In that sense, our work complements recent work on overlay routing because online TE does not need to estimate TMs. Ideally, one would like online TE to balance the load within ISP networks, exploiting intra-AS path diversity [37] to avoid intra-AS bottlenecks (which, according to [3], account for 40% of the bottlenecks). Overlays, on the other hand, can adapt end-to-end routes to avoid congested peering links. Careful design, however, is needed to achieve this ideal situation, because overlay adaptations and online TE can easily destabilize each other. Our future work focuses on how to make these two adaptive systems work harmoniously to deliver the best routing to the end user.

10. ACKNOWLEDGMENTS

We thank Asfandyar Qureshi and Shan Sinha for contributing to an early version of TeXCP. We also acknowledge the support of Cisco Systems and the National Science Foundation under NSF Career Award CNS-0448287. The opinions and findings in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation or Cisco Systems.

11. REFERENCES

[1] Abilene. http://abilene.internet2.edu.
[2] A. Akella, B. Maggs, S. Seshan, A. Shaikh, , and R. Sitaraman. A Measurement-Based Analysis of Multihoming. In *SIGCOMM*, 2003.
[3] A. Akella, S. Seshan, , and A. Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *IMC*, 2003.
[4] E. J. Anderson and T. E. Anderson. On the Stability of Adaptive Routing in the Presence of Congestion Control. In *INFOCOM*, 2003.
[5] D. Applegate, L. Breslau, and E. Cohen. Coping with Network Failures: Routing Strategies for Optimal Demand Oblivious Restoration. In *SIGMETRICS*, 2004.
[6] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands. In *SIGCOMM*, 2003.
[7] D. Awduche et al. RSVP-TE: Extensions to RSVP for LSP Tunnels, 2001. IETF RFC 3209.
[8] A. Basu and J. G. Reicke. Stability Issues in OSPF Routing. In *SIGCOMM*, 2001.
[9] D. Bertsekas and R. Gallager. *Data Networks*. Englewood Cliffs, 1992.
[10] J. E. Burns, T. J. Ott, A. E. Krzesinski, and K. E. Muller. Path Selection and Bandwidth Allocation in MPLS Networks. *Perform. Eval*, 2003.
[11] Cisco. Configuring OSPF. http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/.
[12] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *INFOCOM*, 2001.
[13] A. Feldmann et al. Deriving Traffic Demands from Operational IP Networks: Methodology and Experience. *IEEE/ACM ToN*, 2001.
[14] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights in a Changing World. In *INFOCOM*, 2000.

[15] B. Fortz and M. Thorup. Optimizing OSPF Weights in a Changing World. In *IEEE JSAC*, 2002.
[16] B. Fortz and M. Thorup. Robust Optimization of OSPF/IS-IS Weights. In *INOC*, 2003.
[17] R. Gallager. A Minimum Delay Routing Algorithm using Distributed Computation. *IEEE Transactions on Computers*, 1977.
[18] R. Govindan and V. Paxson. Estimating Router ICMP Generation Times. In *PAM*, Munich, 2002.
[19] GT-ITM: Georgia Tech. Internetwork Topology Models.
[20] J. Guichard, F. le Faucheur, and J. P. Vasseur. *Definitive MPLS Network Designs*. Cisco Press, 2005.
[21] D. Katabi, M. Handley, and C. Rohrs. Internet Congestion Control for High Bandwidth-Delay Product Networks. In *SIGCOMM*, 2002.
[22] R. Keralapura, N. Taft, C.-N. Chuah, and G. Iannaccone. Can ISPs take the Heat from Overlay Networks? In *HOTNETS*, 2004.
[23] A. Khanna and J. Zinky. The Revised ARPANET Routing Metric. In *SIGCOMM*, 1989.
[24] M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and Robust Routing of Highly Variable Traffic. In *HOTNETS*, 2004.
[25] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue. In *SIGCOMM*, 2001.
[26] D. Mitra and K. G. Ramakrishna. A Case Study of Multiservice Multipriority Traffic Engineering Design. In *GLOBECOM*, 1999.
[27] The Network Simulator. http://www.isi.edu/nsnam/ns/.
[28] E. Osborne and A. Simha. *Traffic Engineering with MPLS*. Cisco Press, 2002.
[29] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On Selfish Routing in Internet-Like Environments. In *SIGCOMM*, 2003.
[30] Rocketfuel. www.cs.washington.edu/research/networking/rocketfuel.
[31] Resilient Overlay Network. htpp://nms.lcs.mit.edu/ron.
[32] E. Rosen, A. Viswanathan, and R. Callon. Multi-protocol Label Switching Architecture. RFC 3031.
[33] M. Roughan et al. Experience in Measuring Backbone Traffic Variability: Models, Metrics, Measurements and Meaning. In *ACM IMW*, 2002.
[34] A. Shaikh, J. Rexford, and K. Shin. Load-Sensitive Routing of Long-lived IP Flows. In *SIGCOMM*, 1999.
[35] S. Sinha, S. Kandula, and D. Katabi. Harnessing TCPs Burstiness using Flowlet Switching. In *HOTNETS*, 2004.
[36] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons, 1996.
[37] R. Teixeira, K. Marzullo, S. Savage, and G. Voelker. In Search of Path Diversity in ISP Networks. In *IMC*, 2003.
[38] J. Tsitsiklis and D. Bertsekas. Distributed Asynchronous Optimal Routing in Data Networks. *IEEE Trans. on Automatic Control*, 1986.
[39] C. Villamazir. Mpls optimized multipath (mpls-omp), 1999.
[40] C. Villamazir. Ospf optimized multipath (ospf-omp), 1999.
[41] S. Vutukury and J. J. Garcia-Luna-Aceves. A Simple Approximation to Minimum-Delay Routing. In *SIGCOMM*, 1999.
[42] R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone Network. In *HOTNETS*, 2004.

APPENDIX

A. PROOF OF THEOREM 4.1

This proof is similar to that in [21], and we claim no credit for it.

Model & Assumptions: We assume a fluid model of traffic, a single bottleneck along each path, the RTT is a constant d, and the IE flows have infinite demands. We also ignore boundary conditions.

Consider a bottleneck of capacity c traversed by N IE flows. Let $r_i(t)$ be the sending rate of IE flow i at time t. The aggregate traffic rate on the link is $\phi(t) = \sum r_i(t)$. The router sends some aggregate feedback every T_p. The feedback reaches the TeXCP agents after a round trip time, d. Assuming the IE flows have enough demands, the change in their aggregate rate per second is equal to the feedback divided by T_p.

$$\frac{d\phi(t)}{dt} = \sum \frac{dr_i(t)}{dt} = \frac{1}{T_p}\left(-\alpha \cdot (\phi(t-d) - c) - \beta \cdot \frac{q(t-d)}{T_p}\right).$$

The whole system can be expressed using the following delay differential equations.

$$\dot{q}(t) = \phi(t) - c \tag{13}$$

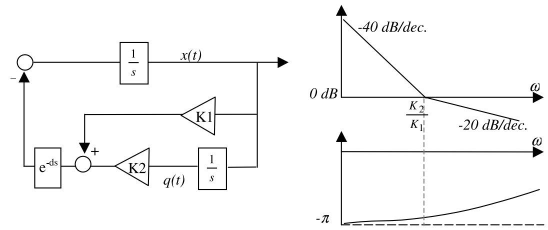


Figure 13: The feedback loop and the Bode plot of its open loop transfer function.

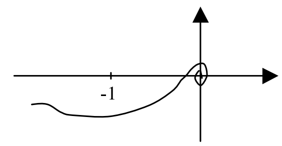


Figure 14: The Nyquist plot of the open-loop transfer function with a very small delay.

$$\dot{\phi}(t) = -\frac{\alpha}{T_p}(\phi(t-d) - c) - \frac{\beta}{T_p^2} q(t-d) \tag{14}$$

Proof Idea: This is a linear feedback system with delay. The stability of such systems may be studied by plotting their open-loop transfer function in a Nyquist plot. We prove that the system satisfies the Nyquist stability criterion. Further, the gain margin is greater than one and the phase margin is positive independently of delay, capacity, and number of IE flows.[10]

PROOF. Let us change variable to $x(t) = \phi(t) - c$.

$$\dot{q}(t) = x(t)$$

$$\dot{x}(t) = -K_1 x(t-d) - K_2 q(t-d)$$

$$K_1 = \frac{\alpha}{T_p} \quad \text{and} \quad K_2 = \frac{\beta}{T_p^2},$$

The system can be expressed using a delayed feedback (see Figure 13). The open loop transfer function is:

$$G(s) = \frac{K_1 \cdot s + K_2}{s^2} e^{-ds}$$

For very small $d > 0$, the closed-loop system is stable. The shape of its Nyquist plot, which is given in Figure 14, does not encircle -1.

Next, we prove that the phase margin remains positive independent of the delay. The magnitude and angle of the open-loop transfer function are:

$$|G| = \frac{\sqrt{K_1^2 \cdot w^2 + K_2^2}}{w^2},$$

$$\angle G = -\pi + \arctan \frac{wK_1}{K_2} - w \cdot d.$$

The break frequency of the zero occurs at $w_z = \frac{K_2}{K_1}$.

To simplify the system, we choose α and β such that the break frequency of the zero w_z is the same as the crossover frequency w_c (frequency for which $|G(w_c)| = 1$). Substituting $w_c = w_z = \frac{K_2}{K_1}$ in $|G(w_c)| = 1$ leads to $\beta = \alpha^2\sqrt{2}$.

To maintain stability for any delay, we need to make sure that the phase margin is independent of delay and always remains positive. This means that we need $\angle G(w_c) = -\pi + \frac{\pi}{4} - \frac{\beta}{\alpha}\frac{d}{T_p} > -\pi$. Since $T_p > d$ by design, we need $\frac{\beta}{\alpha} < \frac{\pi}{4}$. Substituting β from the previous paragraph, we find that we need $0 < \alpha < \frac{\pi}{4\sqrt{2}}$. In this case, the gain margin is larger than one and the phase margin is always positive (see the Bode plot in Figure 13). This is true for any constant delay, capacity, and number of IE flows. $\square$

[10] The gain margin is the magnitude of the transfer function at the frequency $-\pi$. The phase margin is the frequency at which the magnitude of the transfer function becomes 1. They are used to prove robust stability.

B. PROOF OF THEOREM 4.2

Preliminaries: Before delving into the proof, we manipulate the TeXCP equations to obtain an expression of the change in link utilization. The proof uses this expression to show that the max-utilization decreases monotonically and stabilizes at a balanced load. Our variable definitions are in Table 2 and our assumptions are in §4.

Recall Eq. 5 in a simplified form:

$$\Delta x_{sp} = \begin{cases} x_{sp}(n)(\overline{u}_s(n) - u_{sp}(n)), & \forall p, u_{sp} > u_{min}, \\ \epsilon + x_{sp}(n)(\overline{u}_s(n) - u_{sp}(n)), & p, u_{sp} = u_{min}. \end{cases} \quad (15)$$

Note that perturbing by $\epsilon > 0$ only increases the value of Δx, which would otherwise sum to zero, and the re-normalization in Eq. 8, causes the new traffic fraction to be no larger than $\hat{x}$. Hence, it is easy to verify that:

$$x_{sp}(n+1) \leq \hat{x}_{sp} = x_{sp}(n) + \Delta x_{sp}(n). \quad (16)$$

Our proof goes in discrete steps. In each step, each TeXCP agent s applies Eq. 15, to adjust the amount of traffic r_{sp} sent along each path. Note:

$$\Delta r_{sp} = R_s(x_{sp}(n+1) - x_{sp}(n)) \leq R_s \Delta x_{sp}(n). \quad (17)$$

The last part is from using Eq. 16.

The change in the utilization of link l, due to this new traffic assignment, is the sum of the changes over all the paths traversing the link.

$$\begin{aligned} u_l(n+1) &\leq u_l(n) + \sum_s \sum_{p \in P_s, p \ni l} \frac{R_s}{C_l} \Delta x_{sp}(n) \\ &\leq u_l(n) + \sum_s \sum_{p \in P_s, p \ni l} \frac{R_s}{C_l} (\epsilon + x_{sp}(n)(\overline{u}_s(n) - u_{sp}(n))). \end{aligned}$$

The first part is from Eq. 17 and the second is by substituting with Eq. 15. By definition, path utilization u_{sp} is the maximum link utilization in that path, i.e. $\forall l \in p$, $u_{sp}(n) \geq u_l(n)$. Replacing u_{sp} by u_l in the last equation, and noting that $\sum_s \sum_{p \in P_s, p \ni l} \frac{R_s x_{sp}(n)}{C_l} = u_l(n)$, we get:

$$u_l(n+1) \leq u_l(n)(1 - u_l(n)) + \sum_s \sum_{p \in P_s, p \ni l} \frac{R_s}{C_l} (\overline{u}_s(n) x_{sp}(n) + \epsilon). \quad (18)$$

PROOF. We will now prove that in every step, the maximum link utilization in the network always decreases without oscillations, i.e.

$$\max_l u_l(n+1) < \max_l u_l(n). \quad (19)$$

Define $\overline{u}^M(n)$ as the maximum average utilization $\max_s \overline{u}_s(n)$. Assume link i has the maximum utilization after step n and link j has the maximum utilization after step $n+1$. **Thus, we need to prove that**

$$u_j(n+1) < u_i(n).$$

The proof easily extends to the case when multiple links might have the maximum utilization. We identify three cases.

Case 1: $u_j(n) > \overline{u}^M(n)$ **[Highly Utilized in Past]**
In this case, at step n, u_j was larger than the largest $\overline{u}_s$. Every TeXCP agent sending traffic on a path containing link j will decrease its traffic share x_{sp} according to equation 15. Thus, the utilization of link j decreases $u_j(n+1) < u_j(n)$. But, $u_j(n) \leq u_i(n)$ because i is the max-utilization link at step n. Thus, $u_j(n+1) < u_i(n)$.

Case 2: $u_j(n) < \overline{u}^M(n)$ **[Low Utilization in Past]**
We want to bound the increase in traffic on link j. Substituting $\overline{u}_s(n) \leq \overline{u}^M(n)\ \forall s$ in Eq. 18, and using $\sum_s \sum_{p \in P_s, p \ni j} \frac{R_s x_{sp}(n)}{C_j} = u_j(n)$, we get:

$$u_j(n+1) \leq u_j(n)(1 - u_j(n) + \overline{u}^M(n)) + \sum_s \sum_{p \in P_s, p \ni j} \frac{R_s \epsilon}{C_j} \quad (20)$$

Each TeXCP agent independently picks an ϵ as follows, where N is the total number of IE pairs, P is the maximum number of paths that an IE pair can use and C_{min}, the minimum capacity of all links in the network:

$$\epsilon_s = .99 * \frac{C_{min}}{NP} \frac{(\overline{u}_s - u_{min})(1 - u_{min})}{R_s}. \quad (21)$$

Substituting this value of ϵ, and using $C_{min} \leq C_j$, $u_{min} \geq u_j$, $\overline{u}_s(n) \leq \overline{u}^M(n)$, we can bound the total increase due to perturbations:

$$\begin{aligned} \sum_s \sum_{p \in P_s, p \ni j} \frac{R_s \epsilon}{C_j} &< (\overline{u}^M - u_j)(1 - u_j) \sum_s \sum_{p \in P_s, p \ni j} \frac{1}{NP}, \\ &< (\overline{u}^M - u_j)(1 - u_j). \end{aligned}$$

Plugging this in Eq. 20, and canceling common terms, we have

$$u_j(n+1) < \overline{u}^M(n).$$

Thus, the maximum utilization at step $n+1$ is smaller than $\overline{u}^M(n)$. But, $\overline{u}^M(n) \leq u_i(n)$ because $\overline{u}^M(n)$ is a linear combination of link utilizations at step n, whereas $u_i(n)$ is the maximum utilization at step n. Thus, $u_j(n+1) < \overline{u}^M(n) \leq u_i(n)$.

Case 3: $u_j(n) = \overline{u}^M(n)$
We will first show that no TeXCP agent can increase its share of traffic on any path traversing link j. Second, as shown above, $\overline{u}^M(n) \leq u_i(n)$. So, if traffic on link j reduces at this step, then $u_j(n+1) < u_j(n) \leq u_i(n)$ and we are done. Finally, we show that when the utilization of link j does not decrease, then either the maximum utilization in the network decreases or the network has reached the stable state described in Theorem 4.2.
No increase in traffic on paths traversing link j: Note that for any path q traversing link j, $u_{sq}(n) \geq u_j(n) = \overline{u}^M(n) \geq \overline{u}_s(n)$, so from Eq. 15 path q can get a larger share of traffic only if it is the minimum utilized path at some source. But, note that if this path were the minimum, then the above inequality leads to $\overline{u}_s = u_{sq}$, resulting in $\epsilon = 0$ (Eq. 21).
No reduction in traffic on link j: No reduction in traffic on link j means $u_j(n+1) = u_j(n)$. Now, if link j was not the maximum utilized link at step n, i.e. $u_j(n) < u_i(n)$, then the maximum utilization on the network has still decreased to $u_j(n+1)$ from $u_i(n)$; and we are done. We now prove the final **sub-case 3.1**.
No reduction in traffic on link j and **No Decrease in Max. Utilization in the network,** , $\overline{u}^M = u_j(n+1) = u_j(n) = u_i(n) \Rightarrow$ **stable state**.

Given the system is in sub-case 3.1, all TeXCP agents can be classified as follows: Set Z consists of IE pairs s that have $\overline{u}_s = \overline{u}^M$, and Set $\overline{Z}$ consists of the remaining. Now, we make two assertions.

First, no agent $s \in Z$ can know a path p that has a smaller utilization than $\overline{u}^M$. If it did, then it can move some fraction of its traffic onto this path using Eq. 15 and reduce the maximum utilization–a contradiction. Further, $\overline{u}_s = \overline{u}^M = max_l u_l(n)$ implies that all the paths used by agent s traverse maximum utilization links, because a linear combination $\overline{u}_s$ can be equal to the maximum only if all the values are individually equal to the maximum.

Second, no agent $s \in \overline{Z}$ can be sending traffic on a path p that traverses a maximum utilized link. If it did, $u_{sp} = \overline{u}^M > \overline{u}_s$, and agent s would move traffic away from this path reducing the maximum utilization–a contradiction. Hence, these agents do not use any of the paths traversing maximum utilization links.

These assertions lead to an interesting conclusion. Since every path used by an agent in Z traverses a max-utilization link and no agent in $\overline{Z}$ uses a path containing a max-utilization link, the subsets of paths used by the agents in these two sets is disjoint. The agents in these two sets will not interact any more. Further, agents in Z have a balanced load on all of their paths and will not move traffic anymore, in accordance with Eq. 15. Thus, $u_j(n+1)$ will never decrease and this is the final max-utilization of the system. Also, agents in Z satisfy the stable state conditions in Theorem 4.2.

We now focus on the agents in $\overline{Z}$. Since the link bandwidth consumed by agents in Z stays static after step n, we can ignore agents in Z after adjusting the link capacities accordingly. The new system has reduced capacity links, contains only the agents in set $\overline{Z}$ and has a max-utilization smaller than $u_j(n)$. We can analyze this reduced system as we did the original system. The max-utilization of the reduced system will keep strictly decreasing until we are again in sub-case 3.1. We are bound to reach sub-case 3.1 again as it is impossible for the max-utilization, a positive value, to keep decreasing indefinitely. At sub-case 3.1, non-zero number of TeXCP agents (those in the new set Z) will satisfy the stability conditions in Theorem 4.2 and can be ignored henceforth. We repeat this argument iteratively until there are no TeXCP agents of type $\overline{Z}$ left.

To summarize, eventually all TeXCP agents see equal utilization along all of the paths they use and do not know any paths with smaller utilization. In this state, no TeXCP agent can move traffic from one path to another, and the system has stabilized. $\square$