TIE Breaking: Tunable Interdomain Egress Selection

Renata Teixeira
Univ. Pierre et Marie Curie
Paris, France
renata.teixeira@lip6.fr

Timothy G. Griffin
University of Cambridge
Cambridge, UK
Timothy.Griffin@cl.cam.ac.uk

Mauricio G. C. Resende
AT&T Labs–Research
Florham Park, NJ
mgcr@research.att.com

Jennifer Rexford
Princeton University
Princeton, NJ
jrex@cs.princeton.edu

***Abstract*— In a large backbone network, the routers often have multiple egress points they could use to direct traffic toward an external destination. Today's routers select the "closest" egress point, based on the intradomain routing configuration, in a practice known as early-exit or hot-potato routing. In this paper, we argue that hot-potato routing is restrictive, disruptive, and convoluted, and propose a flexible alternative called TIE (Tunable Interdomain Egress selection). TIE is a flexible mechanism that allows routers to select the egress point for each destination prefix based on both the intradomain topology and the goals of the network administrators. In fact, TIE is designed from the start with optimization in mind, to satisfy diverse requirements for traffic engineering and network robustness. We present two example optimization problems that use integer-programming and multicommodity-flow techniques, respectively, to tune the TIE mechanism to satisfy network-wide objectives. Experiments with traffic, topology, and routing data from two backbone networks demonstrate that our solution is both simple (for the routers) and expressive (for the network administrators).**

I. Introduction

The Internet's two-tiered routing architecture was designed to have a clean separation between the intradomain and interdomain routing protocols. For example, the interdomain routing protocol allows the border routers to learn how to reach external destinations, whereas the intradomain protocol determines how to direct traffic from one router in the Autonomous System (AS) to another. However, the appropriate roles of the two protocols becomes unclear when the AS learns routes to a destination at multiple border routers—a situation that arises quite often today. Since service providers peer at multiple locations, essentially *all* of the traffic from customers to the rest of the Internet has multiple egress routers. In addition, many customers connect to their provider in multiple locations for fault tolerance and more flexible load balancing, resulting in multiple egress routers for these destinations as well. We argue that selecting among multiple egress points is now a fundamental part of the Internet routing architecture, independent of the current set of routing protocols.

In the Internet today, border routers learn routes to destination prefixes via the Border Gateway Protocol (BGP). When multiple border routers have routes that are "equally good" in the BGP sense (e.g., local preference, AS path length, etc.), each router in the AS directs traffic to its *closest* border router, in terms of the Interior Gateway Protocol (IGP) distances. This policy of *early-exit* or *hot-potato* routing is hard-coded in the BGP decision process implemented on each router [1]. Hot-potato routing is an appealing mechanism for two main reasons. First, hot-potato routing can limit the consumption of bandwidth resources in the network by shuttling traffic to the next AS as early as possible. Second, under hot-potato routing, a router's choice of egress point is guaranteed to be consistent with the other routers along the forwarding path, because packets are forwarded to neighboring routers that have selected a BGP route with the same (closest) egress point.

Although consistent forwarding is clearly an important property for any routing system, routers now have other ways of achieving this goal. In particular, the greater availability of tunneling technology allows for more sophisticated egress-selection rules, which are not tied to the IGP metrics. Internet Service Providers (ISPs) increasingly use tunneling technologies—such as IP-in-IP encapsulation or MultiProtocol Label Switching (MPLS)—to support Virtual Private Networks (VPNs) or to avoid running BGP on their internal routers. We capitalize on tunneling techniques to revisit the hard-coded policy of selecting egress points based on IGP distances, because we believe that hot-potato routing is:

- **Too restrictive:** The underlying mechanism dictates a particular policy rather than supporting the diverse performance objectives important to network administrators.
- **Too disruptive:** Small changes in IGP distances can sometimes lead to large shifts in traffic, long convergence delays, and BGP updates to neighboring domains [2], [3].
- **Too convoluted:** Network administrators are forced to evaluate the impact of changes in the IGP metrics on BGP routing decisions, rather than viewing the two parts of the routing system separately.

Selecting the egress point and computing the forwarding path to the egress point are two very distinct functions, and we believe that they should be decoupled. Paths *inside* the network should be selected based on some meaningful performance objective, whereas *egress selection* should be flexible to support a broader set of traffic-engineering goals. These objectives vary by network and destination prefix; therefore a mechanism that imposes a single egress selection policy cannot satisfy this diverse set of requirements.

In this paper, we propose a new mechanism for each router to select an egress point for a destination, by comparing the candidate egress points based on a weighted sum of the IGP distance and a constant term. The configurable weights provide flexibility in deciding whether (and how much) to base BGP

decisions on the IGP metrics. Network-management systems can apply optimization techniques to automatically set these weights to satisfy network-level objectives, such as balancing load and minimizing propagation delays. To ensure consistent forwarding through the network, our mechanism relies on the use of tunnels to direct traffic from the ingress router to the chosen egress point. Our new mechanism is called TIE (Tunable Interdomain Egress) because it controls how routers break ties between multiple equally-good BGP routes Our solution does not introduce any new protocols or any changes to today's routing protocols, making it possible to deploy our ideas at one AS at a time and with only minimal changes to the BGP decision logic on IP routers. The paper makes the following contributions:

- **Flexible mechanism for egress-point selection:** TIE is: (i) flexible in balancing the trade-off between sensitivity to IGP changes and adaptability to network events, (ii) computationally easy for the routers to execute in real time, and (iii) easy for a management system to optimize based on diverse network objectives.
- **Optimization of network-wide objectives:** We present example problems that can be solved easily using TIE. First, we show how to minimize sensitivity to internal topology changes, subject to a bound on propagation delay, using integer programming to tune the weights in our mechanism. Second, we show how to balance load in the network without changing the IGP metrics or BGP policies, by using multicommodity-flow techniques to move some traffic to different egress points.
- **Evaluation on two backbone networks:** We evaluate the effectiveness of TIE for the two optimization problems, using traffic, topology, and routing data from two backbone networks (i.e., Abilene and a large tier-1 ISP).

In the next section, we discuss the problems caused by hot-potato routing, and describe an alternative where each router has a fixed ranking of the egress points. Then, Section III presents the TIE mechanism. Sections IV and V present the two optimization problems and evaluate our solutions on topology, traffic, and routing data from two backbone networks. In Section VI, we discuss how to limit the number of configurable parameters and how to deploy TIE without changing the existing routing protocols. After a brief overview of related work in Section VII, we conclude the paper in Section VIII. An Appendix describes how we determine the network topology, egress sets, and traffic demands from the measurement data collected from the two backbone networks.

II. The IGP/BGP Boundary

The Internet routing architecture has three main components: (i) interdomain routing, which determines the set of border (or *egress*) routers that direct traffic toward a destination, (ii) intradomain routing, which determines the path from an ingress router to an egress router, and (iii) egress-point selection, which determines which egress router is chosen by each ingress router for each destination. In this section, we first describe how tying egress selection to IGP distances leads to harmful disruptions and over-constrained traffic-engineering problems. Then we explain how the alternative of allowing each ingress router to have a fixed ranking of egress points is not flexible enough (for traffic engineering) or adaptive enough (to large changes in the network topology).

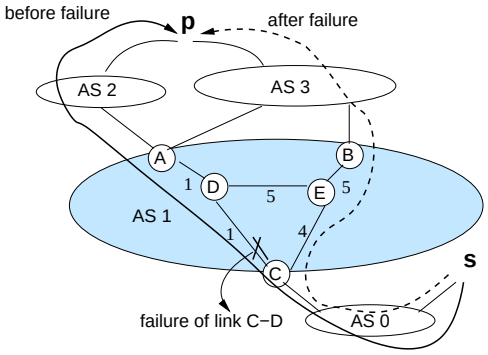


Fig. 1. Link failure causes router C to switch egress points from A to B for destination prefix p.

Our discussion of the two approaches draws on the example network in Figure 1. AS 1 has five routers (A, B, C, D, and E) and each internal link has an IGP metric. Routers A and B are both egress points for destination prefix p, because they learn routes to p via *external* BGP (eBGP). Each of them selects a best route[1], and propagates it via *internal* BGP (iBGP) to routers inside the AS. Routers A and B propagate their best route to p to router C. Under hot-potato routing, router C chooses the BGP route learned from A because the IGP distance to A is 2, which is smaller than the distance of 9 to B. However, if the C–D link fails, all traffic from C to p would shift to egress router B, with an IGP distance of 9 that is smaller than the new IGP distance of 10 to A. In this section, we argue that these kinds of routing changes are disruptive. Yet, continuing to use egress-point A might not be the right thing to do, either, depending on the propagation delay, traffic demands, and link capacities. Instead, network administrators need a mechanism that is flexible enough to support sound performance trade-offs.

A. Hot-Potato Routing

Hot-potato routing adapts automatically to topology changes that affect the relative distances to the egress points. Although hot-potato routing seems like a reasonable way to minimize resource consumption, IGP link weights do not express resource usage directly. The IGP distances do not necessarily have any relationship to hop count, propagation delay, or link capacity, and selecting the closer egress point does not necessarily improve performance. In addition, small topology changes can lead to performance disruptions:

- **Large shifts in traffic within and between ASes:** A single link failure can affect the egress-point selection for tens of thousands of destinations at the same time,

[1] A has the choice between the route through AS 2 and AS 3. In this example, we assume that the two routes are equivalent when comparing BGP attributes, so A decides which route to pick based on a tie break such as the age of the route or the router ID.

leading to large shifts in traffic [2]. In fact, hot-potato routing changes are responsible for many of the largest traffic variations in a large backbone [3].

- **Changes in the downstream path:** When the egress point changes, the traffic moves to a different downstream forwarding path that may have a different round-trip time or available bandwidth, which may disrupt the communicating applications. In addition, the abrupt increase in traffic entering the neighboring AS may cause congestion.
- **BGP update messages for neighboring domains:** A change in egress point may also change the AS path. If A selects the route via AS 2 in Figure 1, the failure of the C–D link causes router C to switch from a path through AS 2 to one through AS 3, forcing C to send a BGP update message to AS 0. Global BGP convergence may take several minutes [4]. If AS 0 switches to a BGP route announced by another provider, the traffic entering AS 1 at router C would change.

Even if the hot-potato routing change does not lead to new BGP update messages, long convergence delays can occur inside the AS depending on how the router implements the BGP decision process. An earlier measurement study [2] discovered long convergence delays because the underlying routers in the network only revisited the influence of IGP distances on BGP decisions once per minute; during the convergence period, data packets may be lost, delayed, or delivered out of order. This particular problem, while serious, can be addressed by having routers use an event-driven implementation that immediately revisits the BGP routing decisions after a change in the intradomain topology. In contrast, the three problems listed above are fundamental.

In a large network, IGP changes that affect multiple destination prefixes happen several times a day, sometimes leading to very large shifts in traffic [3]. Not all of these events are caused by unexpected equipment failures—a large fraction of them are caused by planned events, such as routine maintenance[2]. A recent study of the Sprint backbone showed that almost half of IGP events happened during the maintenance window [5]. Often, shifts in egress points are not necessary. The new intradomain path to the old egress point, although a little longer IGP-wise, may offer comparable (or even better) performance than the path to the new egress point. Following the failure of the C–D link in Figure 1, the path C, E, D, A might be less congested or have lower propagation delay than the path C, E, B. Moreover, many internal network changes are short-lived; a study of the Sprint backbone showed that 96% of failures were repaired in less than 15 minutes [5]. Maintenance activities are often done in periods of lower traffic demands. During these periods the network would comfortably have extra capacity to tolerate the temporary use of non-closest egress points, which would avoid disrupting the non-negligible amount of data traffic traversing the network during the maintenance period.

[2]Maintenance activities happen very frequently to upgrade the operating system on the routers, replace line cards, or repair optical amplifiers. In addition, construction activities may require moving fibers or temporarily disabling certain links.

Besides being disruptive, the tight coupling between egress selection and IGP metrics makes traffic engineering and maintenance planning extremely difficult. Network administrators *indirectly* control the flow of traffic by tuning the IGP metrics [6], [7] and BGP policies [8], [9]. Finding good settings that result in the desired behavior is computationally challenging [6], due to the large search space and the need to model the effects on egress-point selection. Imposing even more constraints, such as minimizing egress-point changes across all routers and destination prefixes, makes the problem increasingly untenable. In addition, once the local search identifies a better setting of the IGP metrics or BGP policies, changing these parameters in the routers requires the network to go through routing-protocol convergence, leading to transient performance disruptions.

B. Fixed Ranking of Egresses at Each Ingress

A natural alternative would be to configure each router with a fixed ranking of the egress points, where the router would select the highest-ranked element in the set of egress routers for each destination. This solution can be realized using today's technology by establishing a tunnel from each ingress router to each egress router, and assigning an IGP metric to the tunnel[3]. The data packets would follow the shortest underlying IGP path from the ingress router to the chosen egress router. The hot-potato mechanism would still dictate the selection of egress points, but the metric associated with each tunnel would be defined statically at configuration time rather than automatically computed by the IGP. Thus, this technique allows network administrators to rank the egress points from each router's perspective. Each ingress router selects the highest-ranked egress point independent of internal network events, short of the extreme cases where the egress point becomes unreachable or the BGP route is withdrawn and the router is forced to switch to the egress point with the next highest rank.

For the example in Figure 1, router C could be configured to prefer egress A over B. Then, when the C–D link fails, C would continue to direct traffic toward router A, though now using the path C, E, D, A. This would avoid triggering the traffic shift to B, changes in the downstream forwarding path, and BGP updates to neighboring domains. However, although the fixed ranking is quite robust to internal changes, sometimes switching to a different egress point *is* a good idea. For example, the path C, E, D, A may have limited bandwidth or a long propagation delay, making it more attractive to switch to egress-point B, even at the expense of a transient disruption. In the long term, network administrators could conceivably change the configuration of the ranking to force the traffic to move to a new egress point, but the reaction would not be immediate. Similarly, the administrators could reconfigure the

[3]For example, network administrators can use MPLS [10], [11] to create label-switched paths (LSPs) between all ingress-egress pairs, which creates a full-mesh of virtual links in addition to IP links. Configuring each LSP as an IGP *virtual link* ensures that each tunnel appears in the intradomain routing protocol. The metric assigned to the tunnel would then drive the hot-potato routing decision hard-coded in the routers. These metrics have to be carefully chosen to be smaller than IGP link weights

IGP metrics or BGP policies to redistribute the traffic load, at the expense of searching for a suitable solution, reconfiguring the routers, and waiting for the routing protocol to converge. All these approaches react too slowly to network changes.

The mechanisms available today for selecting egress points represent two extremes in the trade-off between robustness and automatic adaptation. Hot-potato routing adapts immediately to internal routing changes (however small), leading to frequent disruptions. Imposing a fixed ranking of egress points, while robust to topology changes, cannot adapt in real time to critical events. Neither approach offers enough control for network administrators to engineer the flow of traffic and plan for maintenance. We ask a natural question: *Is there a mechanism for egress-point selection that is flexible enough to control the flow of traffic in steady state, while responding automatically to network events that would degrade performance?*

III. Tunable Interdomain Egress Selection

In this section, we propose a mechanism for selecting an egress point for each ingress router and destination prefix in a network. Ideally, an optimization routine could compute the egress points directly based on the current topology, egress sets, and traffic, subject to a network-wide performance objective. However, the routers must adapt in real time to events such as changes in the underlying topology and egress sets, leading us to design a simple mechanism that allows a separation of timescales—enabling both rapid adaptation to unforeseen events and longer-term optimization of network-wide objectives. We first describe our simple mechanism and then our proposal for using it in operational networks.

A. TIE Ranking Metric

Our mechanism allows each router to have a ranking of the egress points for each destination prefix. That is, router i has a metric $m(i, p, e)$, across all prefixes p and egress points e. For each prefix, the router considers the set of possible egress points and selects the one with the smallest rank, and then forwards packets over a tunnel that follows the shortest path through the network to that egress point. Although we propose using tunnels between every pair of routers to guarantee consistent forwarding, our approach differs from the scheme in Section II-B in several key ways. First, our ranking metric has *finer granularity*, in that we allow an ingress router to have a different ranking for different destination prefixes. Second, our ranking metric is computed rather than statically configured, allowing the ranking to *adapt to changes* in the network topology and egress set. Third, our metric is *not tied directly to the underlying tunnel* that directs traffic from an ingress point to the chosen egress point, allowing us to achieve the finer granularity of control without increasing the number of tunnels. Our approach is also more flexible than tuning BGP routing policies, in that one router can start using a new egress point while other routers continue to use the old one.

To support flexible policy while adapting automatically to network changes, the metric $m(i, p, e)$ must include both configurable parameters and values computed directly from a real-time view of the topology. We represent intradomain routing topology as an undirected weighted graph $G = (N, L)$, where N is the set of nodes and L is the set of IP links, as summarized in Table I. Based on the link weights, each router $i \in N$ can compute the IGP distance $d(G, i, e)$ to every other router $e \in N$. The egress set $E(p) \subseteq N$ consists of the edge nodes that have equally-good BGP routes for prefix p, i.e., all edge routers that learn routes to p with the same local preference, AS path length, origin type, and Multiple-Exit Discriminator (if routes have the same next-hop AS) from external neighbors. In Figure 1, the egress set $E(p) = \{A, B\}$. Note that routers A and B do not necessarily use the same next-hop AS to reach p. The definition of egress set implies that all nodes in $E(p)$ select their external route to reach p. A node $i \notin E(p)$ selects the egress point $argmin_e\{m(i, p, e) \,|\, e \in E(p)\}$. The metric is computed as a weighted sum of the IGP distance and a constant term:

Undirected graph	$G = (N, L)$, nodes N and links L
Ingress and egress nodes	$i \in N$ and $e \in N$
IGP distance on graph	$d(G, i, e)$, $i, e \in N$
Destination prefix	$p \in P$
Egress set	$E(p) \subseteq N$
Ranking metric	$m(i, p, e)$, $i, e \in N$, $p \in P$
Tunable parameters	$\alpha(i, p, e)$ and $\beta(i, p, e)$

TABLE I
Summary of notation.

$$m(i, p, e) = \alpha(i, p, e) \cdot d(G, i, e) + \beta(i, p, e),$$

where α and β are configurable values. The first component of the equation supports automatic adaptation to topology changes, whereas the second represents a static ranking of routes for that prefix. Together, these two parameters can balance the trade-off between adaptability and robustness. This simple metric satisfies our three main goals:

- **Flexible policies:** By tuning the values of α and β, network administrators can cover the entire spectrum of egress-selection policies from hot-potato routing to static rankings of egress points. Hot-potato routing can be implemented by setting $\alpha = 1$ and $\beta = 0$ for all nodes and prefixes. A static ranking can be represented by setting $\alpha = 0$ and, for each node i, $\beta(i, p, e)$ to a constant value for all values of p. Our mechanism can also realize a diverse set of policies in between.
- **Simple computation:** The metric is computationally simple—one multiplication and one addition—based on information readily available to the routers (i.e., the IGP distances and the α and β values). This allows routers to compute the appropriate egress point for all destination prefixes immediately after a change in the network topology or egress set.
- **Ease of optimization:** The mechanism offers two knobs (α and β) that can be easily optimized by a management system based on diverse network objectives. In Section IV and V, we explore the power of this mechanism to express two different policies, and we demonstrate that it is easy

to optimize by showing that the optimization problems we define are tractable.

In addition, when the network-management system changes the α and β values, the affected routers can move traffic from one path to another without incurring any convergence delays. This fast convergence is possible because the network already has tunnels between each pair of routers. Changing the α and β values merely changes which paths carry the traffic.

B. Using TIE

We do not envision that network administrators will configure all values of α and β by hand. Instead, we propose an architecture as presented in Figure 2. The upper box represents the tasks of a management system that configures the routers, and the lower box captures the tasks running on each router in the network. Network administrators define the high-level goal of the egress-selection policy for the network or for a set of destination prefixes (such as minimizing sensitivity to failures, minimizing delay, or balancing link load). The management system takes as input the current network design and the administrator's specifications, runs an optimization routine to find the appropriate values for the parameters α and β, and configures the routers accordingly. Once the management system configures the TIE parameters, the routers apply the BGP decision process as usual, except for using the metric m to select between multiple equally-good BGP routes.

With TIE the egress-point selection can change for two reasons: high-level policy changes (expressed by changes in α and β) or routing changes. Policy changes happen because of changes in network objectives or the network design. Routing changes—changes in the IGP distances or egress sets—happen in response to network events such as link failures or BGP updates from neighboring domains. Reaction to routing changes must be done in real time to avoid bad network performance, whereas policy changes happen less often and can be implemented slowly. Our architecture benefits from this separation of timescales. Policy changes require running an optimization routine, which is executed completely off line by the management system running on a separate machine. Under routing or policy changes, routers only need to perform one addition and one multiplication to recompute m. This simple on-line computation also happens under BGP updates. Routers can be pre-configured with default values of α and β for newly announced prefixes. The management system will revisit these values at the time of the next optimization.

IV. Minimizing Sensitivity

In this section, we present a prototype of a management system to select values of α and β to minimize the sensitivity of egress-point selection to equipment failures, subject to restrictions on increasing the propagation delay. After presenting a precise formulation of the problem, we present a solution that has two phases—simulating the effects of equipment failures to determine the constraints on the α and β values and applying integer-programming techniques to identify optimal settings. Then, we evaluate the resulting solution using topology and routing data from two backbone networks.

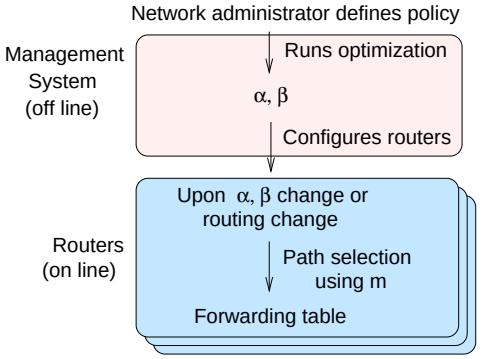


Fig. 2. A management system optimizes α and β for a high-level policy and configure routers. Routing adapts the egress-point selection at real time in reaction to network events.

A. Problem Definition

Consider a well-provisioned backbone network that supports interactive applications, such as voice-over-IP and online gaming. The network administrators want to avoid the transient disruptions that would arise when an internal failure causes a change in the egress point for reaching a destination, as long as continuing to use the old egress point would not incur large delays. By setting the IGP link weights according to geographic distance, the shortest IGP path between two nodes would correspond to the smallest delay and the closest egress point would be the best choice. Hence, for this problem, the best egress point $b(G, i, p)$ for node i and prefix p is the node $e \in E(p)$ with the smallest IGP distance $d(G, i, e)$. If an internal failure occurs, the administrators want node i to continue directing traffic to $b(G, i, p)$ unless the delay to this egress point exceeds $T \cdot d(G, i, b(G, i, p))$ for some threshold $T > 1$. If the delay to reach the egress point exceeds the threshold, the administrators want node i to switch to using the (new) closest egress point to minimize the propagation delay. Table II summarizes the notation.

Threshold for tolerable delay ratio	T
Set of topology changes	ΔG
Topology change	$\delta \in \Delta G$
Network topology after change	$\delta(G)$
Best egress point for (i, p) on G	$b(G, i, p)$

TABLE II
NOTATION FOR THE PROBLEM OF MINIMIZING SENSITIVITY TO TOPOLOGY CHANGES WITH BOUNDED DELAY.

In an ideal world, the routers could be programmed to implement this policy directly. For example, upon each IGP topology change δ, each node i could revisit its egress selection for each prefix by performing a simple test for the new topology $\delta(G)$:

if $(d(\delta(G), i, b(G, i, p)) \leq T \cdot d(G, i, b(G, i, p)))$,
then $b(\delta(G), i, p) = b(G, i, p)$
else $b(\delta(G), i, p) = argmin_e\{d(\delta(G), i, e) \mid e \in E(p)\}$.

Modifying every router in the network to implement this

egress-selection policy would guarantee that the network always behaves according to the specified goal. However, supporting a wide variety of decision rules directly in the routers would be extremely complicated, and ultimately network administrators would want to apply a policy that is not supported in the routers. In the next subsection, we show that TIE is expressive enough to implement this policy. Instead of having the routers apply the test in real time, the network-management system configures the TIE parameters at design time based on the policy, and the routers adapt automatically when internal changes occur.

B. Solving the Sensitivity Problem with TIE

Solving the problem with our mechanism requires us to find values of $\alpha(i,p,e)$ and $\beta(i,p,e)$, for each $i,e \in N$ and $p \in P$, that lead to the desired egress-point selections over all topology changes ΔG. Our solution has two main steps. First, a *simulation phase* determines the desired egress selection both at design time (under graph G) and after each topology change (under graph $\delta(G)$). The output of this phase is a set of constraints on the α and β values for each (i,p) pair. Then, an *optimization phase* determines the values of α and β that satisfy these constraints. For this problem, the egress-point selection for each (i,p) pair can be made independently.

1) Simulation Phase: To illustrate how we construct the constraints on α and β for the initial topology G and each topology change δ, consider the example in Figure 3(a). In the initial topology, node A would select node B as the egress point because B is closer than C. We can express this by $m(A,p,B) < m(A,p,C)$ for topology G, as shown by the first constraint in Figure 3(b). Then, we consider each topology change δ and determine the preferred egress selection with the policy in mind, where $T = 2$ and δ_1 is the failure of the link with cost 4 and δ_2 is the failure of the links with costs 4 and 6. In the new graph $\delta_1(G)$, A is closer to C (with a distance $d(\delta_1(G),A,C)$ of 5) than to B (with a distance $d(\delta_1(G),A,B)$ of 6). However, since $d(\delta_1(G),A,B) < 2 \cdot d(G,A,B)$, A should continue to select egress-point B. This decision is expressed by the second equation in Figure 3(b). We use the same methodology to evaluate the best egress selection after δ_2. In this case, the distance from A to B is above the threshold, so A should switch to using egress-point C, as expressed by the third equation.

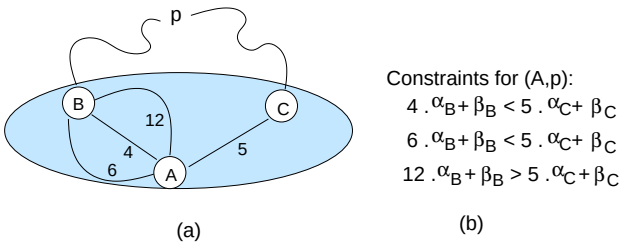


Fig. 3. Example illustrating constraints on values of α and β.

More generally, our algorithm consists of two main steps. First, we compute the distances $d(\cdot,i,e)$ for the original graph G and all topology changes $\delta \in \Delta G$ using an all-pairs shortest path algorithm. (For simple topology changes, such as all single-link failures, an incremental Dijkstra algorithm can reduce the overhead of computing the $|\Delta G|+1$ instances of the all-pairs shortest paths.) Then, we generate the constraints for each (i,p) pair as presented in Figure 4.

1) Identify the closest egress point in the original graph: $b = argmin_e\{d(G,i,e) \,|\, e \in E(p)\}$,
2) For each $e \in E(p) \setminus \{b\}$, generate the constraint "$\alpha(i,p,b)\cdot d(G,i,b)+\beta(i,p,b) < \alpha(i,p,e)\cdot d(G,i,e)+\beta(i,p,e)$"
3) For each $\delta \in \Delta G$
   a) Identify the preferred egress point b': If $d(\delta(G),i,b) \le T \cdot d(G,i,b)$, then $b' = b$. Else, $b' = argmin_e\{d(\delta(G),i,e) \,|\, e \in E(p)\}$.
   b) For each $e \in E(p) \setminus \{b'\}$, generate the constraint "$\alpha(i,p,b')\cdot d(\delta(G),i,b')+\beta(i,p,b') < \alpha(i,p,e)\cdot d(\delta(G),i,e)+\beta(i,p,e)$"

Fig. 4. Algorithm of the simulation phase.

Step 2 runs once (on the original graph) and step $3(b)$ runs $|\Delta G|$ times (on each topology change), generating a constraint for each alternative to the desired egress point for that configuration. As a result, the algorithm produces $(|\Delta G|+1)\cdot(|E(p)|-1)$ constraints for each pair (i,p). The size of $E(p)$ is limited by the number of edge nodes that have best BGP routes for a prefix; in practice, the size is usually one, two, or three, or at most ten. Fortunately, any prefixes that have the same egress set produce the same constraints, and the same values of α and β. The number of unique egress sets is typically orders of magnitude less than the number of prefixes, which substantially reduces the running time of the algorithm. In order to reduce the complexity and number of configurable parameters, we group all routers in the same PoP into a single node; these routers typically make the same BGP routing decisions anyway, since they essentially act as one larger router. Ultimately, the running time of the algorithm is dominated by the number of topology changes in ΔG. In practice, the set of topology changes is restricted to failures of a single piece of equipment at a time such as IP links, IP routers, or link-layer equipment (such as an optical amplifier, which may impact multiple IP links at the same time).

2) Optimization Phase: In the optimization phase, we compute α and β values that satisfy the constraints for each pair (i,p). In theory, *any* settings that satisfy the constraints would achieve our optimization goal. However, several practical issues drive how we set up the optimization problem:

- **Finite-precision parameter values:** The α and β values should have finite precision to be configured and stored on the routers. Since the parameter values only have meaning relative to each other, we can limit ourselves to considering integer solutions. This leads us to apply *integer* programming to solve the problem.
- **Robustness to unplanned events:** Although we optimize the parameters based on the topology changes in ΔG, the real network might experience events outside of our

model. If optimizing based on ΔG results in solutions with $\alpha = 0$ for an (i, p) pair, then router i would never adapt to a change in IGP distance, however large. To increase the robustness to unplanned events, we add an extra constraint that $\alpha(i, p, e) \geq 1$ for all i, p, and e.

- **Limiting the number of unique parameter values:** To reduce the overhead of configuring and storing the α and β parameters, we prefer solutions that reduce the number of unique values. As such, we attempt to minimize an objective function that is the sum across all of the α and β values, which favors solutions with $\alpha = 1$ and $\beta = 0$, selecting different values only when necessary to satisfy the constraints.

For each (i, p) pair, the simulation phase generates a set of linear inequalities and a linear objective function. Since we want our variables (α and β) to have integer values, we need to solve an integer-programming problem. We use the CPLEX [12] solver with the AMPL interpreter to find the α and β values for each (i, p) pair. Although integer-programming problems are sometimes difficult to solve, our constraints are typically easy to satisfy because many constraints are identical or are subsumed by other constraints. For instance, the second constraint in Figure 3(b) is stricter than the first constraint (i.e., because $4\alpha_B < 6\alpha_B$). In fact, for most of the (i, p) pairs, CPLEX computes the values of α and β during a pre-processing phase that analyzes the constraints. Very few (i, p) pairs required more than three simplex iterations in the root node of the branch-and-bound tree to identify parameters that satisfy the constraints and minimize the objective function.

C. Evaluation

We evaluate the effectiveness of TIE for achieving our goal of minimizing sensitivity to equipment failures on the Abilene network and a tier-1 ISP backbone. We obtain the network topology G and the egress sets $\{E(p)\}$ as described in the Appendix. For this problem, we set the IGP link weights to the geographic distance between the PoPs to approximate the propagation delay. We optimize TIE for two sets of topology changes ΔG (single link failures and single node failures) and three different delay thresholds T (1.5, 2, and 3).

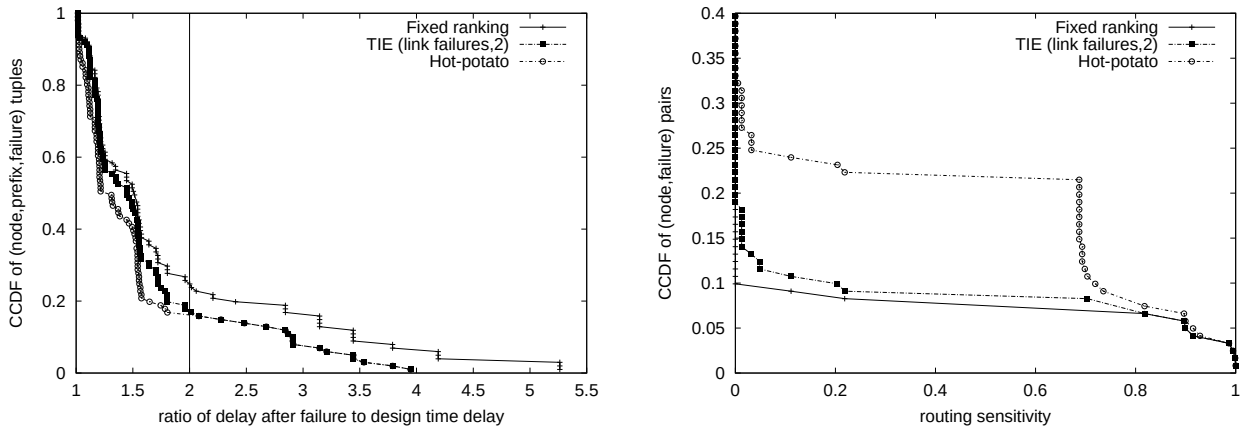
We ran the simulation and the optimization phases on different machines because the raw measurement data could only be stored on one machine, and the CPLEX license resides on another. The simulation phase ran on a 900MHz Ultrasparc-III Copper processor of a Sun Fire 15000. This phase consumed 3.2 MB of RAM and took 0.5 and 31.1 seconds to build the constraints for all pairs (i, p) for the Abilene and ISP networks, respectively. The optimization phase ran on a 196 MHz MIPS R10000 processor on an SGI Challenge. This phase consumed just under 4 MB of RAM and took 37 seconds and 12 minutes to run for the Abilene and ISP networks, respectively. The management system selects new α and β parameters very infrequently, and this selection does not delay the routers from picking routes. Thus, 12 minutes of running time is perfectly reasonable. In addition, we expect that the optimization phase would complete much faster if we invoke the CPLEX library directly from a C program rather than the AMPL interpreter and if we run it on more powerful machine.

In the resulting configuration for the Abilene network, α was equal to 1 for 93% of the (i, p, e) tuples and had only four distinct values ($\alpha \in [1, 4]$); β was zero for 90% of the (i, p, e) tuples and had only three distinct values ($\beta \in \{0, 1, 3251\}$). The ISP network has a much larger number of destination prefixes and distinct egress sets, which resulted in a broader range of values for the parameters ($\alpha \in [1, 19]$ and $\beta \in \{0, 1, 3411, 4960, 5185, 5009\}$). However, the vast majority of α values (88%) were equal to one, and 69% of β values were zero. The small number of distinct values for the parameters, and the large number of $\alpha(i, p, e) = 1$ and $\beta(i, p, e) = 0$, help reduce the overhead of configuring and storing the parameters, as discussed in more detail in Section VI. The fact that most (i, p) pairs have $\alpha(i, p, e) = 1$ and $\beta(i, p, e) = 0$ reveals that there are just a few points in the network that need some hysteresis to keep them from over-reacting to small IGP changes. TIE provides enough flexibility for the management system to identify the specific places where this hysteresis is needed to achieve the network-wide goals.

After generating the values of $\alpha(i, p, e)$ and $\beta(i, p, e)$ for each one of these scenarios, we simulate the behavior of each network with this configuration. For comparison, we also simulate the behavior of the network using hot-potato routing (by setting $\alpha(i, p, e) = 1$ and $\beta(i, p, e) = 0$ for all (i, p, e)), and the fixed ranking egress selection (by setting $\alpha(i, p, e) = 0$ for all (i, p, e), and $\beta(i, p, e) = d(G, i, e)$). We simulate the behavior of these egress-selection policies under the set of all single-link failures and the set of all single-node failures. For conciseness, we only present the results for single-node failures, the results for the other instances lead to the same conclusions. We compare the three mechanisms using two metrics:

- **Delay ratio:** For each (i, p, δ) we compute the delay for i to reach the best egress point for p after the topology change δ ($d(\delta(G), i, b(\delta(G), i, p))$), and divide it by the delay to reach the best egress in the original topology ($d(G, i, b(G, i, p))$).
- **Routing sensitivity:** For each (i, δ) the routing sensitivity represents the fraction of prefixes at i that change egress point after a topology change δ. This metric is the routing-shift function ($\mathcal{H}^{RM}$) defined in [13] and represents the fraction of a router's BGP table that changes egress points after an intradomain routing change.

Figure 5(a) presents the complementary cumulative distribution function (CCDF) of the delay ratio for the Abilene network. A delay ratio equal to one means that the delay after the failure is the same as the delay in the original network. Many of the node failures do not affect the path between an ingress node and a best egress node for a prefix. Therefore, we omit all values that had a delay ratio of one. Given that the link weights are set according to geographic distance, the delay ratio achieved by hot-potato routing represents the smallest feasible delay ratio. Fixed ranking represents the delay to reach the old egress point after the failure. In this plot, we present the results for TIE optimized for single-link failures and $T = 2$,



(a) Propagation delay ratio (CCDF) (b) Routing sensitivity (CCDF)

Fig. 5. Comparison of egress-selection schemes on the Abilene network under single-node failures with TIE optimized for single-link failures and $T = 2$.

and evaluate the schemes against single-node failures. The results of TIE optimized for single-node failures were very similar (in fact most of the values of α and β were the same).

Despite being optimized for a different set of topology changes, TIE still behaves according to the original goal. TIE exceeds the delay threshold of 2 for only 20% of the (i, p, δ), and hot-potato routing also exceeds the threshold in each of these cases. Fixing the ranking of egress points leads to delays that are higher than the delay achieved by TIE in the majority of instances. Whenever the fixed-ranking scheme lies below the threshold of 2, TIE is below it as well. When the fixed-ranking scheme exceeds the threshold, TIE shifts to an egress point that is at or below the threshold. This is the reason why the TIE curve lies *below* the fixed-ranking curve for delay ratios under 2.

Below the threshold of 2, TIE has higher delay than hot-potato routing in exchange for lower sensitivity values as shown in Figure 5(b). This graph plots the CCDF of routing sensitivity for all (i, δ) pairs. Fixing the ranking of egress points has the lowest sensitivity. In fact, the fixed-ranking scheme has a non-zero sensitivity only when the best egress point fails, forcing even this scheme to change to the second-ranked egress point (i.e., the one that was second-closest in the initial topology). The TIE curve follows the fixed ranking for most points. TIE only experiences egress changes when they are unavoidable. The gap between the hot-potato and the TIE curve—around 15% of the (i, δ) pairs—represents the scenarios for which egress-selection disruptions could be avoided without violating the delay threshold.

Although we observe similar behavior in the results for the large ISP network (presented in Figures 6(a) and 6(b)), the gap between the curves is not as large as for the Abilene network. In this case, we optimize TIE for single-link failures with a delay threshold $T = 3$. The ISP network has many more choices of egress points per prefixes than the Abilene network. Therefore, the delay to reach the closest egress point in the original topology is likely to be very small, and setting the threshold to three times this delay still gives reasonably short delays. This network also has more path diversity than the Abilene network. In a more diverse graph, it is more likely that there is still a low-delay path to the initial egress point, even after the failure. Contrasting the delay ratio and routing sensitivity of the two networks illustrates that there is not a single policy that fits all networks. Compared to the Abilene network, the ISP network could safely put more emphasis on setting the β values, because its rich connectivity makes it unlikely that equipment failures would lead to significant changes in the IGP distance between a pair of routers. The TIE mechanism is flexible enough to accommodate both of these networks.

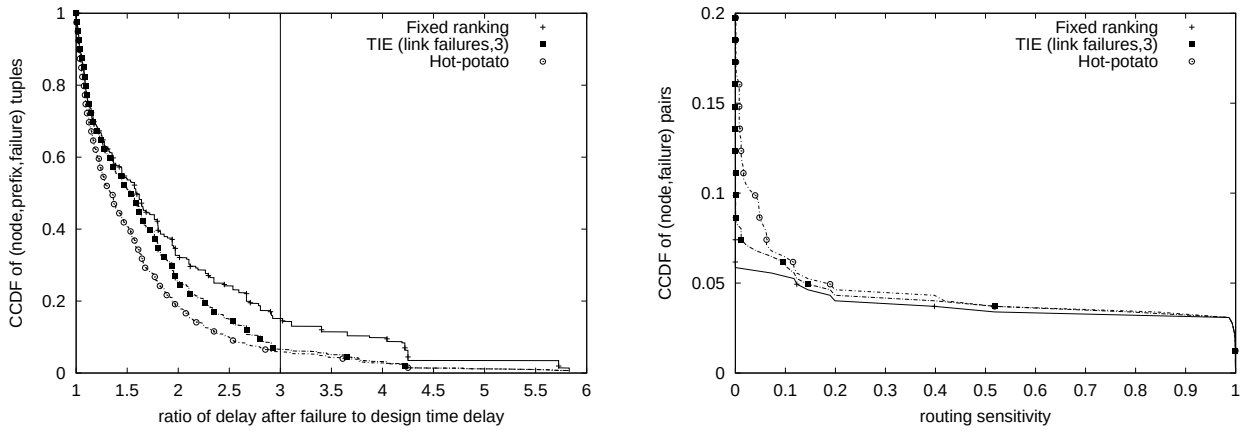
In this section, we assume that the egress set for each destination prefix is stable when determining the values of α and β. Our evaluation shows that even when an egress node is removed from the egress set (which can represent either a node failure or a BGP route withdrawal), TIE behaves as expected. We can extend the formulation of this problem to find solutions that are robust to egress-set changes. For instance, we can configure TIE to react slowly to the announcement of new routes (i.e., additions to the egress set) by setting the values of $\alpha(\cdot, p, e)$ and $\beta(\cdot, p, e)$ to be very high for all $e \notin E(p)$. We can also model BGP dynamics by extending our notion of topology change δ to include changes to the egress sets.

V. Traffic Engineering

This section demonstrates the expressiveness of TIE for doing traffic engineering. We propose an optimization problem that balances link utilization on the network only by selecting the appropriate egress point for each pair (i, p) (i.e., by setting the values of $\beta(i, p, e)$). This is in contrast with the common practice of optimizing link utilization by either tweaking IGP link weights or BGP policies. After defining the optimization problem and presenting our solution, we evaluate our solution by comparing the link utilizations achieved using TIE to that using the current network configuration.

A. Problem Definition: Balancing Link Utilization

Traffic engineering—adapting the flow of traffic to the prevailing network conditions—is a common task that can be performed in several ways. Traffic engineering considers



(a) Propagation delay ratio (CCDF) (b) Routing sensitivity (CCDF)

Fig. 6. Comparison of egress-selection schemes on the ISP network under single-node failures for TIE optimized for single-link failures and $T = 3$.

a network topology (G) with the capacity of each link ($c(\ell)$), and the traffic demands $v(i,p)$ (i.e., the volume of traffic to destination prefix p that enters the network at ingress router i), as summarized in Table III. The effects of the IGP weights on the intradomain paths can be represented by the routing matrix $R(i,e,\ell)$, which captures the fraction of traffic from router i to router e that traverses link ℓ. If the network has one shortest path between i and e, $R(i,e,\ell)$ is one for any link ℓ on that path, or zero otherwise; if multiple shortest paths exist, $R(i,e,\ell)$ may be fractional. The flow of traffic also depends on the egress set $E(p)$ and the egress point $b(i,p)$ that router i uses to reach prefix p.

Link capacity	$c(\ell)$, for $\ell \in L$
Traffic demand	$v(i,p)$ for $i \in N$, $p \in P$
Routing matrix	$R(i,e,\ell)$, for $i,e \in N$, $\ell \in L$
Egress selection	$b(i,p) \in E(p)$ for $i \in N$, $p \in P$
Link traffic load	$t(\ell)$ for $\ell \in L$
Link utilization	$u(\ell) = t(\ell)/c(\ell)$, $\ell \in L$
Multicommodity flow path	$\tau(i,e,p) \subset G$
Decision variable	$x(i,e,p) \in \{0,1\}$
Link congestion penalty	$\phi(u(\ell))$, $\ell \in L$
Objective function	$\Phi = \sum_{\ell \in L} \phi(u(\ell))$

TABLE III
NOTATION FOR THE TRAFFIC-ENGINEERING PROBLEM

Traffic engineering involves tuning the network configuration to minimize some function of the load on the links. The load $t(\ell)$ on link ℓ can be determined as follows:

$$t(\ell) = \sum_{i \in N} \sum_{\substack{p \in P, \\ b(i,p) = e, \\ e \in E(p)}} v(i,p) \cdot R(i,e,\ell)$$

and the resulting link utilization is $u(\ell) = t(\ell)/c(\ell)$. The common approach to traffic engineering is to formulate an optimization problem that minimizes an objective function that penalizes solutions in terms of the load they place on each link. In our work, we consider a function $\phi(u(\ell))$ that increasingly penalizes loads as they near or pass the link's capacity. This piecewise-linear function can be expressed by the equation

$$\phi(u(\ell)) = \begin{cases} u(\ell), & u(\ell) \in [0, 1/3) \\ 3 \cdot u(\ell) - 2/3, & u(\ell) \in [1/3, 2/3), \\ 10 \cdot u(\ell) - 16/3, & u(\ell) \in [2/3, 9/10), \\ 70 \cdot u(\ell) - 178/3, & u(\ell) \in [9/10, 1), \\ 500 \cdot u(\ell) - 1468/3, & u(\ell) \in [1, 11/10), \\ 5000 \cdot u(\ell) - 16318/3, & u(\ell) \in [11/10, \infty) \end{cases} \quad (1)$$

that was introduced in [14] and used in several other traffic-engineering studies. The network-wide objective function Φ is the sum of the link penalties—i.e., $\Phi = \sum_{\ell \in L} \phi(u(\ell))$.

Network administrators can minimize the objective function by changing the intradomain paths ($R(i,e,\ell)$), interdomain routes ($E(p)$), or the egress-point selection ($b(i,p)$). Tuning the IGP link weights (to influence the intradomain paths) and the BGP policies (to influence the interdomain routes) lead to NP-complete optimization problems [6], [7]. The computational intractability of these problems forces the use of local-search techniques that repeatedly evaluate parameter settings in the hope of finding a good solution. Although local-search heuristics often produce good parameter values [6], [7], the solutions are not optimal and are not guaranteed to have performance that is close to optimal. In addition, the solutions require changing the IGP weights or BGP policies, which triggers routing-protocol convergence and leads to transient disruptions. In contrast, using TIE to control the egress-point selections $b(i,p)$ leads to a simpler optimization problem that does not require changes to the routing-protocol configuration. Since we are simply selecting among existing paths and not changing the configuration of routing protocols, our approach does not trigger routing convergence.

B. Solving the Traffic-Engineering Problem with TIE

Traffic engineering with TIE involves assigning each (i,p) pair to an egress point $b(i,p) \in E(p)$ in a way that minimizes the objective function Φ. A solution can be realized by setting $\beta(i,p,b(i,p))$ to a low value, while setting $\beta(i,p,e)$ to a high value for all $e \neq b(i,p)$, and all α values to zero. In contrast to the fixed-ranking scheme in Section II-B, we allow

a router's ranking of egress points to differ across the prefixes. In practice, we envision solving richer optimization problems that consider robustness to changes in the network topology G, the egress sets $E(p)$, and the traffic demands $v(i,p)$, which would lead to solutions that assign values to both α and β. In this paper, we focus on fixed topology, egress sets, and traffic demands, to illustrate how TIE provides the flexibility needed to balance load across the links.

We formulate the egress-selection problem as a *path-based* multicommodity-flow problem that accounts for the constraints that the routing matrix $R(i,e,\ell)$ imposes on the flow of traffic. For a router i and prefix p, we consider the topology $\tau(i,e,p)$ induced by the links $\ell \in L$ for which $R(i,e,\ell) > 0$. All links in the graph $\tau(i,e,p)$ can be used to route traffic from i to p through the egress point $e \in E(p)$. We call τ a path in the multicommodity-flow formulation. We represent the actual routing of the traffic from i to p by a $(0,1)$-decision variable $x(i,e,p)$, such that $x(i,e,p) = 1$ if and only if the path $\tau(i,e,p)$ is selected to send traffic from i to p. The choice of a path τ determines the egress point $e \in E(p)$ selected. For all pairs (i,p), the egress-selection problem requires that a single egress point $e \in E(p)$ be chosen (i.e., no more than one $x(i,e,p)$ of 1 for each (i,p) pair). We express this requirement by the following equation:

$$\sum_{e \in E(p)} x(i,e,p) = 1.$$

The contribution of the traffic going from i to p to the load on link ℓ is the product of the traffic demand $v(i,p)$, the routing-matrix element $R(i,e,\ell)$, and the decision variable $x(i,e,p)$. The total load on a link is the sum of all the contributions, i.e.

$$t(\ell) = \sum_{i \in N} \sum_{p \in P} \sum_{e \in E(p)} v(i,p) \cdot R(i,e,\ell) \cdot x(i,e,p).$$

A *piecewise-linear* integer-programming formulation for the single egress-selection problem is to minimize the objective function $\Phi = \sum_{\ell \in L} \phi(u(\ell))$ such that the $(0,1)$-decision variables $x(i,e,p)$ sum to 1 for each (i,p) pair. Defining $\phi(u(\ell))$ to be a linear variable and applying a standard transformation results in the *linear* integer-programming formulation:

$$\begin{aligned}
&\min \sum_{\ell \in L} \phi(u(\ell)) \\
&\text{s.t.} \\
&\quad u(\ell) = \Big(\sum_{i \in N} \sum_{p \in P} \sum_{e \in E(p)} v(i,p) \cdot R(i,e,l) \cdot x(i,e,p) \Big) / c(\ell), \ \forall l \in L, \\
&\quad \sum_{e \in E(p)} x(i,e,p) = 1, \ \forall i \in N, p \in P, \\
&\quad \phi(u(\ell)) \geq u(\ell), \ \forall l \in L, \\
&\quad \phi(u(\ell)) \geq 3 \cdot u(\ell) - 2/3, \ \forall l \in L, \\
&\quad \phi(u(\ell)) \geq 10 \cdot u(\ell) - 16/3, \ \forall l \in L, \\
&\quad \phi(u(\ell)) \geq 70 \cdot u(\ell) - 178/3, \ \forall l \in L, \\
&\quad \phi(u(\ell)) \geq 500 \cdot u(\ell) - 1468/3, \ \forall l \in L, \\
&\quad \phi(u(\ell)) \geq 5000 \cdot u(\ell) - 16318/3, \ \forall l \in L, \\
&\quad x(i,e,p) \in \{0,1\}, \ \forall i \in N, p \in P, e \in E(p), \\
&\quad \phi(u(\ell)) \geq 0, \ \forall l \in L.
\end{aligned}$$

However, in general, this integer multicommodity-flow problem is intractable. Instead, we consider its linear-programming relaxation obtained by relaxing the integrality constraints $x(i,e,p) \in \{0,1\}$ to simply $x(i,e,p) \geq 0$. For both networks we consider, the CPLEX solver produced solutions with only integer values of $x(i,e,p)$, thus solving the integer programming problem and allowing us to configure the $\beta(i,p,e)$ values to pick the single egress point $b(i,p)$ for each (i,p) pair. We set α to zero for all (i,p,e) tuples. For the egress e for which $x(i,e,p) = 1$, we set $\beta(i,p,e) = 0$. For every other egress e', we set $\beta(i,p',e') = d(i,e')$. For situations where the solution of the linear-programming relaxation is fractional, applying a simple heuristic based on randomized rounding can produce a valid egress selection. For each pair (i,p) with fractional $x(i,e,p)$ values, egress point $e \in E(p)$ is selected with probability $x(i,e,p)$. Randomized rounding is repeatedly applied and the best solution found is output by the algorithm.

C. Evaluation

We evaluate the link utilization achieved by TIE on both the Abilene and ISP networks. We obtained the network topology G, the egress sets $\{E(p)\}$, and the traffic demands $v(i,p)$, as explained in the Appendix. We aggregate all traffic from an ingress i to all destination prefixes p that share the same egress set $E(p)$ to build the ingress to egress set traffic demand $v(i,E)$ for each unique egress set E. For this problem, we use the IGP link weights as configured in each network. The CPLEX solver took 0.1 and 1.5 seconds to run on the 196 MHz MIPS R10000 processor for the Abilene and ISP networks, respectively. The current network IGP configuration is set to achieve good link utilization assuming that the egress-selection mechanism is hot-potato routing. Therefore, we compare the utilization achieved using TIE with that achieved by hot-potato routing.

Table IV presents the value of the objective function Φ for both topologies under both egress-selection policies. TIE's flexibility in balancing load allows us to find an optimal solution for both networks using the linear-programming relaxation. The solution using hot-potato routing is 40% worse than that found using TIE for the ISP network. Hot-potato routing has a congestion function close to TIE for the Abilene network, because the Abilene network is significantly under-utilized. Still, TIE does offer some (admittedly modest) improvements to the objective function.

	Abilene Network	ISP Network
Hot-potato routing	0.4513510071	8.990353677
TIE	0.4425879808	5.557480707

TABLE IV
COMPARISON OF THE NETWORK CONGESTION FUNCTION Φ BETWEEN HOT-POTATO ROUTING AND TIE.

We studied the ratio of link utilization between hot-potato routing and TIE for the ten most heavily-loaded links under hot-potato routing. The TIE solution reduces the utilization of the most utilized link by 40.9%. Although TIE increases

the load on some links, our solution reduces the utilization of two-thirds of the links, and the most utilized link in the TIE solution has 26.3% less utilization than the most utilized link under hot-potato routing.

D. Extensions

In this section, we assume that each router i can select any $e \in E(p)$ for each destination prefix p. However, this could conceivably lead to long propagation delays if i selects a far-away egress point, or to unnecessary BGP update messages to neighboring domains. We can address these concerns simply by removing certain egress points from consideration if they have high propagation delay or a BGP route with a different AS path. For instance, egresses where $d(G, i, e)$ exceeds a threshold could be removed from consideration for router i, or we could consider only the egress points that have BGP routes with the same AS path. Our solution can also treat destination prefixes for sensitive applications (such as VoIP) separately. For instance, we can set the values α and β for each VoIP prefix to minimize sensitivity and delay as discussed in Section IV. Then, we can optimize the egress selection for the rest of the prefixes for traffic engineering. The load to the VoIP prefixes is considered as background load, which cannot be changed by the traffic engineering optimization.

The traffic-engineering optimization problem as defined in this section only considers the utilization of internal links. A natural extension is to use TIE to *balance outbound load on the edge links*. We can formulate this problem by adding an artificial node per peering link and another for each destination prefix p, with each node representing a peering link connecting to the nodes for the prefixes learned in that peering session. We can solve this problem using the same methodology presented here. In addition, our traffic-engineering optimization problem currently does not set the values of α. This prevents the egress selection to automatically adapt to changes in the network topology. TIE can also be configured before *planned maintenance activities* to ensure low link utilizations during the event. In this case, the topology change δ is known in advance, so the network administrators can compute the optimal egress selection in the modified topology $\delta(G)$ and adjust α and β to achieve the desired traffic-engineering goal.

We can combine our methodology for solving the problem presented in Section IV with the one presented here to find a solution to the *robust traffic-engineering* problem. In steps 1 and 3(a) in Figure 4, instead of identifying the best egress point according to the shortest distance, we can achieve robust traffic engineering by selecting the best egress according to the solution of the path-based multicommodity-flow problem specified in Section V-B. We can also consider a delay bound by removing from the egress set any egress point that is over a threshold from the ingress. This multi-objective problem could conceivably encounter a scenario where no parameter setting would satisfy every constraint. A scenario like this, should it arise, could be handled by an extension to the integer program to minimize the *number* of constraints that are violated. This could be achieved by including an extra error term in each constraint and selecting an objective function that minimizes the total error.

VI. Implementation Issues

An AS can deploy the TIE mechanism without changing the intradomain or interdomain routing protocols, and without the cooperation of other domains. In this section, we first describe how to ensure that each router can apply TIE independently of other routers in the AS. Next we discuss how to configure the α and β parameters and how a router applies the TIE mechanism to select a BGP route for each destination prefix. Then, we discuss how moving the responsibility for BGP path selection from the routers to separate servers [15], [16] would make it possible to implement our TIE scheme without *any* modification to the decision logic running on the routers.

A. Independent Decisions at Each Node

Throughout the paper, we have assumed that each node applies the TIE mechanism to select a single best route from the set of equally-good BGP routes chosen by the border routers. In a network with a "full mesh" internal BGP (iBGP) configuration, each router learns these routes directly from the border routers. However, large networks typically employ route reflectors to overcome the scaling problems of having an iBGP session for each pair of routers. A route reflector runs the BGP decision process and propagates a single best route to its clients; as a result, the clients may choose a different best route than they would with all of the options at their disposal. Consider the common scenario with a full mesh of top-level route reflectors, with one or more route reflectors in each PoP. In this scenario, we recommend applying the TIE mechanism only on the route reflectors to allow decisions based on a complete view of the BGP routes. The client routers (i.e., other routers in the same PoP) would inherit the choice made by their common route reflector. This has the added advantage that only the route reflectors would need to be upgraded to implement the TIE mechanism.

The TIE mechanism also relies on the underlying network to forward data packets from the ingress router to the chosen egress point. However, the routers along the forwarding path do not necessarily select the same egress point, depending on how their α and β parameters are configured. This problem does not arise in hot-potato routing because each router selects the closest egress point, which ensures that the routers along the shortest path have chosen the same egress point. Rather than constraining the way α and β are set on different routers, we advocate that the network employ some form of lightweight tunneling to direct traffic over the shortest IGP path(s) from the ingress point to the egress point. For example, the ingress router could encapsulate each data packet in an IP packet where the destination corresponds to the IP address of the chosen egress router. Alternatively, the network may employ MPLS [10], [11] to create label-switched paths (LSPs) between all ingress-egress pairs, as discussed in Section II-B. Tunneling IP packets over the underlying IGP paths is a common usage of MPLS since it obviates the need for interior routers to speak BGP or have a large forwarding table, while also allowing the network to forward VPN and non-IP traffic.

B. Configuring and Applying TIE in Routers

Using the TIE mechanism requires configuring the routers with the values of α and β selected by the optimization routine. As discussed in Section III-B, rather than configuring these values by hand, we envision that a network-management system would have an automated procedure to connect to each router to set or modify the parameters. Still, configuring a large number of values may introduce significant overhead and delay. In the worst case, each router would need to be configured with two integer values for every destination prefix and edge router. For a network with 500 edge routers and 150,000 destination prefixes, this would require configuring 75 billion parameters (i.e., $500 \cdot 500 \cdot 2 \cdot 150,000$), which is clearly excessive. Fortunately, a router often has the same values of α and β across many destination prefixes and egress points. To capitalize on this observation, the TIE mechanism could have default values of $\alpha = 1$ and $\beta = 0$ (corresponding to hot-potato routing) for each prefix, allowing the management system to specify only the parameters that differ from these values. For example, in Section IV only 10% of the β values were non-zero for the tier-1 ISP backbone, which would reduce the configuration overhead by an order of magnitude.

Another way to reduce the overhead is to assign α and β at a coarser granularity than individual routers and destination prefixes. For example, the parameters could be defined for PoPs, rather than routers, particularly if TIE is implemented only at the route reflector(s) in each PoP. If the 500-router network has (say) 25 PoPs, the number of parameters would drop by a factor of 400 (i.e., 25 PoPs would be configured with two parameters per prefix for 25 egress PoPs). In addition, the parameters could be based on the destination AS (i.e., the origin AS that initially announced the BGP route), rather than the destination prefix. If the Internet has (say) 20,000 ASes and $150,000$ prefixes, this would reduce the number of parameters by an additional factor of 7.5. Together, these two optimizations would reduce the number of parameters by a factor of 3000, from 75 billion down to 25 million across all the routers in the network, which seems acceptable particularly if the management system need only specify exceptions to the default α and β values. Further reductions can be achieved by associating α and β values with the next-hop AS or other route attributes.

When α and β are not associated directly with particular prefixes and egress routers, the ingress router needs some way to know which parameters to use in selecting a BGP route for a prefix. The BGP *community* attribute [17] provides an effective way to communicate which parameters should be used. For example, the border routers could be configured to tag each BGP advertisement with a unique community value that identifies the PoP. Another community could be used to identify the origin AS or next-hop AS associated with the advertisement. Upon receiving these tagged routes via internal BGP (iBGP), a router can use these community values to index into a table that stores the α and β values.

Once the router knows which α and β values to use, the router can compute the metric m based on these parameters and the IGP distance to the egress router. Rather than applying the traditional IGP tie-breaking step, the router can implement a modified BGP decision process that uses the m metric to select the route with the most-preferred egress point. Ultimately, the TIE mechanism requires only a change in one step of the BGP decision process implemented on the routers, rather than any protocol modifications. We note that router vendors already provide features that allow network administrators to modify the operation of the BGP decision process [18], which significantly reduces the barrier to deploying TIE.

C. TIE in a Separate Path-Selection Platform

Rather than modifying the BGP decision process implemented on the routers, an AS could move the entire responsibility for BGP path selection to a separate software platform, as proposed in [15], [16]. In this setting, dedicated servers receive the eBGP advertisements and run decision logic to select BGP routes on behalf of the routers in the AS. The servers use iBGP sessions to send each router a customized routing decision for each prefix, essentially overriding the influence of the BGP decision process running on the routers.

These servers could implement the TIE mechanism for selecting the routes in real time, and might also run the offline optimization routines that set the α and β parameters; this would allow the parameters to exist only on the servers, rather than in the routers or other management systems. Even though the servers could conceivably implement any decision logic, in practice they need some separation of functionality between the real-time adaptation to network events and the longer-term optimization of the path-selection process based on network-wide goals. TIE provides a way to achieve that separation.

VII. Related Work

Our work relates to several ongoing threads of research in Internet routing:

Hot-potato disruptions: Measurement studies have shown that hot-potato routing changes can lead to long convergence delays, large shifts in traffic, and external BGP routing changes [2], [3]. Subsequent work proposed metrics of network sensitivity to internal changes to assist network administrators in minimizing hot-potato disruptions [13]. Rather than trying to control disruptions using routing protocols as they are defined today, we redesign the boundary between the two tiers of the routing system to achieve a broader set of traffic-engineering goals (including minimizing disruptions).

Traffic engineering: Controlling the flow of traffic with TIE gives more flexibility for solving the traffic engineering problem. TIE represents one more control knob beyond the conventional approach of tuning the IGP link weights [6], [7] and BGP policies [8], [9]. Whereas TIE can set α and β independently for each (i, p) pair, tuning an IGP weight can affect the IGP distances between multiple pairs of routers and affect the egress-point selection for many prefixes. Similarly, tuning a BGP policy often impacts the route preferences for many routers at once. IGP and BGP changes also lead to routing-protocol messages and convergence delays. TIE also provides an alternative to deploying a load-sensitive routing protocol,

such as the traffic-engineering extensions to OSPF and IS-IS [19], [20], [21]. Load-sensitive routing leads to higher protocol overhead and can sometimes introduce instability. More recent work [22] solves this instability problem by balancing load over a set of pre-defined paths between ingress and egress. However, none of these proposals explicitly addresses the problem of egress-point selection, making it appealing to implement TIE even in networks that already support load-sensitive routing. In our future work, we plan to compare the benefits of TIE with these alternative approaches [21].

Optimizing egress-point selection: Previous research considered an optimization problem similar to our ongoing work discussed in Section V. The work in [23] focused on selecting egress points such that traffic loads do not exceed the egress-point capacities, with the secondary objective of minimizing the total distance traveled by the traffic. In contrast, we formulate an optimization problem that minimizes congestion over the links in the network, using the objective function used in earlier traffic-engineering studies [6].

Multi-homing: In recent years, an increasing number of stub ASes, such as large enterprise and campus networks, connect to multiple upstream providers for improved reliability and flexibility. In response, several research studies have considered how these networks should balance load over the multiple access links [24], [25]. However, our problem is different because we focus on networks where each destination prefix has a (possibly different) set of egress points, and the choice of egress point affects the load on links *inside* the AS.

Inter-AS negotiation: Other research has considered how a pair of neighboring ASes could coordinate to select egress points in a mutually advantageous manner [26], [27]. Where these papers focus on the negotiation process, and on the important question of what information the ASes should exchange, we propose a tunable mechanism for selecting the egress points and a way for each AS to determine its preferred egress points based on network-wide objectives.

VIII. Conclusion

IP networks are under increasing pressure to provide predictable communication performance for applications such as voice over IP, interactive gaming, and commercial transactions. These applications are sensitive to both transient disruptions (i.e., during routing changes) and persistent congestion (i.e., when the routing does not match the prevailing traffic). In this paper, we propose a new mechanism for selecting egress points that satisfies both requirements. TIE avoids the disruptions caused by hot-potato routing changes while supporting diverse network-wide objectives such as traffic engineering and maintenance planning.

TIE is simple enough for routers to adapt in real time to network events, and yet is much more amenable to optimization than today's routing protocols. In addition, TIE can be deployed in an AS without changing the intradomain or interdomain routing protocols, and without the cooperation of other domains. Our experiment for two network-management problems, using data from two backbone networks, demonstrates the effectiveness of our new mechanism and the ease of applying conventional optimization techniques to determine the best settings for the tunable parameters.

Appendix

In Section IV and V, we evaluate TIE on data from two operational networks. In this appendix, we present our methodology for obtaining the input data—the internal topology and the egress sets—from passive measurements. Since routers in the same Point-of-Presence (PoP) essentially act as one larger node, we model the topology of both networks at the PoP level.

Abilene Network. Abilene is the backbone for U.S. research network [28]. The network has 11 PoPs with one router each. The vast majority of the links are OC192, with only one OC48. For our study, we used data from April 2003. We obtained the topology G (both with designed weights and geographic distance) and link capacities $c(l)$ from the publicly-available map of the network. This map has the location of each router, as well as the link capacities and IGP weights. Each BGP speaker has around $7,500$ prefixes in its routing table. We obtained the egress set $E(p)$ for each prefix from a dump of the BGP table for a monitor that peers with every router. The network had only 23 distinct egress sets. We extracted the traffic demands from sampled Netflow data. Every router in the network has Netflow enabled with a sampling rate of $1/100$. For each router i and destination prefix p we have set $v(i,p)$ to the average traffic volume for one hour of Netflow data collected on a weekday afternoon.

Tier-1 ISP Network. We also used data collected from a tier-1 service-provider backbone on January 10, 2005. We extracted the router-level topology and IGP link weights from the link-state advertisements logged by a routing monitor. We used router configuration data to map each router to a PoP and determine the link capacities. The resulting topology has a few dozen nodes. For simplicity, we combine parallel links between a pair of PoPs into one link with the aggregate capacity. We used the PoP locations to determine the geographic distance traversed by each inter-PoP link. The network learns BGP routes for approximately $150,000$ prefixes. We build the egress set $E(p)$ for each prefix from the BGP table dumps from all top-level route reflectors in the network. The network has a few hundred distinct egress sets. We use sampled Netflow data collected around the entire periphery of the network. We aggregate all traffic entering at the same PoP i and destined to the same prefix p into a single traffic demand $v(i,p)$. Each traffic demand represents the average traffic rate over the course of the day.

Acknowledgments

We would like to thank Abilene for making their measurement data publicly available to the research community. We are grateful to Anukool Lakhina for his help in working with the Abilene data. Thanks also to Geoff Voelker and the anonymous reviewers for their comments. Renata was supported in part by the Cooperative Association for Internet Data Analysis (CAIDA).

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)." RFC 4271, January 2006.

[2] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of Hot-Potato Routing in IP Networks," in *Proc. ACM SIGMETRICS*, June 2004.

[3] R. Teixeira, N. Duffield, J. Rexford, and M. Roughan, "Traffic matrix reloaded: Impact of routing changes," in *Proc. Passive and Active Measurement Workshop*, March/April 2005.

[4] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," *IEEE/ACM Trans. Networking*, vol. 9, pp. 293–306, June 2001.

[5] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP Restoration in a Tier-1 Backbone," *IEEE Network Magazine*, March 2004.

[6] B. Fortz, J. Rexford, and M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communication Magazine*, October 2002.

[7] J. Rexford, "Route optimization in IP networks," in *Handbook of Optimization in Telecommunications* (P. Pardalos and M. Resende, eds.), Springer Science + Business Media, February 2006.

[8] N. Feamster, J. Winick, and J. Rexford, "A Model of BGP Routing for Network Engineering," in *Proc. ACM SIGMETRICS*, June 2004.

[9] S. Uhlig, "A multiple-objectives evolutionary perspective to interdomain traffic engineering in the Internet," in *Workshop on Nature Inspired Approaches to Networks and Telecommunications*, September 2004.

[10] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture." RFC 3031, January 2001.

[11] B. S. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann, May 2000.

[12] Ilog S.A., *"Ilog Cplex 9.0 User's Manual"*, October 2003.

[13] R. Teixeira, T. Griffin, A. Shaikh, and G. Voelker, "Network sensitivity to hot-potato disruptions," in *Proc. ACM SIGCOMM*, September 2004.

[14] B. Fortz and M. Thorup, "Increasing internet capacity using local search," *Computational Optimization and Applications*, vol. 29, no. 1, pp. 13–48, 2004.

[15] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The Case for Separating Routing from Routers," in *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2004.

[16] O. Bonaventure, S. Uhlig, and B. Quoitin, "The Case for More Versatile BGP Route Reflectors." Expired Internet Draft draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.

[17] R. Chandra, P. Traina, and T. Li, "BGP communities attribute." RFC 1997, August 1996.

[18] "BGP cost community." `http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801a7f74.html`.

[19] D. Katz, K. Kompela, and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2." RFC 3630, September 2003.

[20] H. Smit, "Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)." RFC 3784, June 2004.

[21] D. Awduche, "MPLS and Traffic Engineering in IP Networks," *IEEE Communication Magazine*, December 1999.

[22] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive and Yet Stable Traffic Engineering," in *Proc. ACM SIGCOMM*, August 2005.

[23] T. Bressoud, R. Rastogi, and M. Smith, "Optimal configuration of BGP route selection," in *Proc. IEEE INFOCOM*, 2003.

[24] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multi-homing," in *Proc. ACM SIGCOMM*, August 2003.

[25] D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing cost and performance for multihoming," in *Proc. ACM SIGCOMM*, September 2004.

[26] R. Mahajan, D. Wetherall, and T. Anderson, "Negotiation-based routing between neighboring ISPs," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, May 2005.

[27] J. Winick, S. Jamin, and J. Rexford, "Traffic engineering between neighboring domains." `http://www.cs.princeton.edu/~jrex/papers/interAS.pdf`, July 2002.

[28] "Abilene Backbone Network." `http://abilene.internet2.edu/`.

PLACE PHOTO HERE

Renata Teixeira is a post-doc at the Laboratoire d'Informatique de Paris 6. She has recently finished her Ph.D. in Computer Science at the University of California, San Diego. During her Ph.D., Renata worked at the AT&T Labs in Florham Park. She received her B.Sc. in Computer Science and M.Sc. in Electrical Engineering from Universidade Federal do Rio de Janeiro, Brazil in 1997 and 1999, respectively.

PLACE PHOTO HERE

Timothy G. Griffin recieved a BS in Mathematics from the University of Wisconsin and a PhD in Computer Science from Cornell. He has worked at Bell Laboratories, AT&T Research and Intel Research. He is curreently on the faculty of the Computer Laboratory at the University of Cambridge.

PLACE PHOTO HERE

Mauricio G. C. Resende is a research scientist at the Algorithms and Optimization Research Department at AT&T Labs Research. His undergraduate studies were in electrical engineering (systems concentration) at the Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Brazil (1978) and he earned a M.Sc. in operations research at the Georgia Institute of Technology (1979). He has been at AT&T Bell Labs and AT&T Labs since earning his Ph.D. in operations research at the University of California, Berkeley, in 1987.

PLACE PHOTO HERE

Jennifer Rexford is a Professor in the Computer Science department at Princeton University. From 1996-2004, she was a member of the Network Management and Performance department at AT&T Labs–Research. She received her BSE degree in electrical engineering from Princeton University in 1991, and her MSE and PhD degrees in computer science and electrical engineering from the University of Michigan in 1993 and 1996, respectively.