Multipath load-adaptive routing: putting the emphasis on robustness and simplicity

Amund Kvalbein
Simula, Oslo, Norway

Constantine Dovrolis and Chidambaram Muthu
Georgia Tech, Atlanta, USA

***Abstract*— We propose a routing and load-balancing approach with the primary goal of being robust to sudden topological changes and significant traffic matrix variations. The proposed method load-balances traffic over several routes in an adaptive way based on its local view of the load in the network. The focus is on robustness and simplicity, rather than optimality, and so it does not rely on a given traffic matrix, nor it is tuned to a specific topology. Instead, we aim to achieve a satisfactory routing under a wide range of traffic and topology scenarios based on each node's independent operation. The scheme avoids the instability risks of previous load-responsive routing schemes, it does not load the control plane with congestion-related signaling, and it can be implemented on top of existing routing protocols. In this paper, we present the proposed scheme, discuss how it aims to meet the objectives of robustness and load-responsiveness, and evaluate its performance under diverse traffic loads and topological changes with flow-level simulations.**

I. Introduction

New applications, such as VoIP, IPTV and distributed gaming, as well as demanding Service Level Agreements between ISPs and commercial or government enterprise networks, impose new requirements on Internet routing. Instead of just providing good performance over long timescales (say hours), it is now important that an ISP can also deal effectively with short-term overload conditions that last from few seconds to several minutes. For such timescales, it is not possible to rely on mechanisms that require human intervention. Instead, an automated load-responsive routing or online Traffic Engineering (TE) mechanism is needed [6], [12]. Further, it is important that these routing/TE mechanisms do not only avoid congestion, but also that they result in routes with low delay (an issue that is often overlooked in the traffic engineering literature).

Load-responsive routing has been the focus of intense research since the early seventies and it was even deployed in the early ARPAnet [14]. Unfortunately, many load-responsive protocols either suffer from instability [25], or they solve the stability problem through significant control overhead in terms of load state updates [6], [7], [12]. On the other hand, TE methods that require an estimate of the Traffic Matrix (TM) can provide optimal routing, in terms of congestion-related metrics [26], but they face the significant challenge that the TM elements vary significantly with time, especially when we are also interested in short timescales. Additionally, link/router failures and management operations require us to also consider robustness in terms of topological changes.

In this paper, we present *Routing Homeostasis*, or simply "Homeostasis".[1] Homeostasis is a new approach to routing and load balancing where *the primary goal is robustness to variations in traffic patterns and topology, while providing good routes in terms of delay under light traffic loads*. The main components in Homeostasis are the use of multiple non-equal cost routes to each destination, and adaptive assignment of traffic to these routes based on the local view of the traffic load.

Even though many advanced routing protocols promising improved performance have been proposed and implemented over the last decades, most networks still base their route selection on shortest-path calculations using link-state protocols like OSPF or IS-IS. The main reasons for this are the simplicity and distributed nature of these protocols. An important design goal in Homeostasis is to retain this simplicity. Homeostasis stays in the context of destination-based hop-by-hop forwarding, and does not involve any complex configuration or parameter optimizations. It is implementable on top of current routing protocols, and does not create additional control traffic. Network nodes make their forwarding decisions independently, and adaptation to changes in the traffic is done based only on local information. With the focus on low overhead and local operation, Homeostasis *prefers simplicity over optimality*.

Multipath routing, together with the adaptive load balancing scheme we describe next, are the key ingredients of robustness in Homeostasis. Instead of using a single best path, or the set of minimum equal-cost next-hops, traffic is spread on a limited set of loop-free unequal-cost next hops. Multipath routing can give rapid fault recovery upon topological changes, and it can absorb short-term overloads by using additional next-hops when needed.

As previously mentioned, load-adaptive routing schemes either suffer from instability risks, or they introduce significant signaling load in the routing plane. To avoid both problems, Homeostasis does not change the set of available routes to a destination based on the load. Routes are chosen based on propagation delays (a static metric that does not vary with load), while load balancing is performed on a local basis without any signaling between different routers. Of course, local load balancing is less effective than global load balancing in rerouting traffic upon congestion. We believe that the stability risks, or control overhead, of global load

[1] This name is inspired by biological homeostasis - the mechanism with which organisms manage to maintain robust and stable function, despite large and unpredictable changes in their environment.

balancing schemes (such as load-responsive routing or online TE) are more important issues for network operators than the advantage of such schemes in terms of capacity utilization.

Homeostasis can be implemented over existing intradomain routing protocols, such as OSPF, IS-IS or EIGRP, because it does not require additional communication between routers. The multipath routing tables can be constructed using the routing updates of any link-state or distance-vector protocol, as long as propagation delay becomes the metric of each link. The load balancing module can be implemented locally, as part of the forwarding engine's functionality, and it does not require changes in the routing information propagated in the routing protocol.

An important goal in Homeostasis is to avoid or minimize queuing delays. Instead of *reacting* to increased queuing delays or packet losses, Homeostasis attempts to proactively *avoid* congestion. To do so, the load balancing module monitors the utilization of each selected out-link that is currently used to route traffic. When that utilization exceeds a certain threshold, new flows are routed to the next available next-hop in terms of propagation delay. In this manner, higher-delay routes are used only when needed; in light-load conditions, traffic is routed through the single minimum delay path.

Homeostasis is not designed to achieve optimality under an expected traffic load, but seeks to give *satisfactory performance under a wide range of operating conditions*. This is in clear contrast to much of the TE literature [8], [9], [21], [26], [27], [28], which seeks to optimize some objective function *assuming knowledge of the traffic matrix*. For example, in the most recent work in this tradition [28], the authors show that an optimal traffic distribution can be achieved without explicit routing and in a distributed manner by a joint optimization of the link weights and (static) traffic split ratios with time complexity $O(N^4)$. The performance of this method is highly dependent on the quality of the TM estimate; when the experienced traffic deviates from the expected, or the topology changes due to failures or maintenance operations, the routing can be far from optimal. In contrast, Homeostasis makes no assumptions about the traffic input, and does not involve any optimization. Optimizing routing for a particular TM is problematic, since in general, it is difficult to make accurate predictions about the future traffic demands in a network. While historical measurements can give an indication about the expected long-term traffic pattern, it is well known that Internet traffic exhibits significant variation over a wide range of timescales. Recent developments such as overlay networks, peer-to-peer applications and Intelligent Route Control make it even harder to get accurate TM estimates. Homeostasis is a "nonparametric" routing mechanism, in the sense that it does not require a TM estimate.

In the rest of this paper, we introduce Homeostasis Routing, present its route selection and load balancing algorithms, and evaluate its robustness and performance (in terms of delay and network cost) with flow-level simulations. The main focus of this paper is on the objectives, key ideas and algorithms of Homeostasis Routing. We defer a more detailed evaluation study, with packet-level simulations or testbed experiments, TCP traffic, routing protocol transients, etc, in a next paper. Sec. II describes how multipath routes are selected. Sec. III describes the load balancing method. We evaluate the robustness and performance of Homeostasis in Sec. IV. We review the most related work in Sec. V, and we conclude in Sec. VI. The details about our simulation setup are explained in the Appendix.

II. Multipath routing

The ability to use more than one path to reach a destination is central for achieving robustness. In this section, we explain how multiple routes are selected, discuss issues that affects the number of routes achieved by Homeostasis, and evaluate the impact of multipath routing on delay.

We want to exploit the underlying path diversity beyond the Equal-Cost Multi-Path (ECMP) commonly employed today, by *not restricting traffic forwarding to only shortest paths*. Instead we install up to $\mathbf{K}$ loop-free next-hops towards a destination in the forwarding table (FIB) at each router. Achieving several next-hops to a destination is critical, since we rely on the ability to load-balance between them to absorb sudden changes in the traffic input. Keeping multiple entries per will increase the memory needed for FIB/RIB. The increase will depend on the vendor-specific implementation. In networks where large routing tables makes this a constraint, it may be necessary to limit the memory consumption by setting K to a low value.

Note that we do not propose a new routing protocol - we are agnostic to the choice of routing protocol for conveying reachability information. Homeostasis can be used on top of any existing link state protocol such as OSPF or IS-IS, or distance vector protocol such as RIP or DASM. We only require changes in the way the routing tables are calculated and populated. Properties like convergence time, inconsistencies during the convergence period etc. will be determined by the routing protocol.

A. Selecting next-hops

We base our route selection on the propagation delays through each neighbor. Note that our method can also work with other metrics for calculating the set of available routes to a destination. Propagation delay is selected here because we want to minimize the latency experienced by traffic, and because it is a stable property of the network that does not change with input load. The routes used to reach a destination are selected based on the announced propagation delays through each neighbor in the following way. Each router i advertises a minimum delay $d_i(t)$ to reach a destination t. This delay is calculated as

$$d_i(t) = \begin{cases} 0, & \text{if } i = t \\ min_{\forall j}(d_j(t) + d(i,j)), & \text{otherwise} \end{cases} \tag{1}$$

where j is a neighbor of i, and $d(i,j)$ is the propagation delay of link (i,j). Traffic can never be sent to a neighbor with a higher distance to t. We say that a neighbor j is a *feasible next-hop* for node i with respect to destination t if node i can send traffic bound for t through j without creating a loop. We denote this relationship $(i \rightarrow j)_t$. Node j is always a feasible next-hop if it has a lower distance to t: $d_j(t) < d_i(t) \Rightarrow (i \rightarrow j)_t$.

A router installs up to $\mathbf{K}$ next-hops towards each destination in its forwarding table, corresponding to the $\mathbf{K}$ feasible next-hops giving the shortest distance in terms of propagation delay. The motivation for limiting the number of next-hops to $\mathbf{K}$ is two-fold. First, we want to limit the amount of state information stored in the (expensive) memory used in FIBs. Second, we want to avoid using routes with very long delays. The strategy of communicating the minimum delay to the destination is similar to the approaches used by multipath distance vector protocols like DASM [29] and MDVA [23], but we differ by not installing all feasible next-hops in the forwarding table.

A node will not always achieve $\mathbf{K}$ feasible next-hops to all destinations, even if $\mathbf{K}$ physical paths exist. In particular, this will always be the case for the *closest* neighbor to the destination/egress node in the network. Since all other nodes (except the destination itself) have a longer distance to the destination, the closest neighbor can never have more than one feasible next-hop.

We denote by $k_i(t)$ the number of loop-free next-hops for destination t that a node i is actually able to install in its FIB under a given routing. $k_i(t)$ is limited either by $\mathbf{K}$, or by the number of feasible next-hops i can achieve for t in the topology. In order to increase robustness, we want to minimize the number of instances where $k_i(t) = 1$.

B. Increasing path diversity

One way to decrease the number of situations where $k_i(t) = 1$ is to include neighbor j in the set of feasible next-hops for node i when $d_i(t) = d_j(t)$ in some situations. To avoid loops, we then need a tie-breaker that gives a *strict total order* of the nodes with respect to t, so that $(i \rightarrow j)_t \Rightarrow (j \nrightarrow i)_t$ and $(i \rightarrow j)_t \wedge (j \rightarrow k)_t \Rightarrow (i \rightarrow k)_t$. As we shall see, the choice of tie-breaker is important for avoiding situations where $k_i(t) = 1$.

We make two observations with respect to the use of tie-breakers. First, there is a connection between the granularity with which we measure the delay of a link, and the frequency with which the tie-breaker is applied. A coarser granularity gives more situations where two neighbors have the exact same distance to a destination, and must rely on the tie-breaker to determine which node can send traffic through the other. Second, increased use of the tie-breaker normally decreases the number of situations where $k_i(t) = 1$. This is because an increasing number of ties can only be positive for nodes with $k_i(t) = 1$, since they can end up with additional feasible next-hops.

We test the performance of three different tie-breaking strategies that are used in situations where $d_i(t) = d_j(t)$. In the *node-ID* strategy, we break ties in favor of the node with the higher ID. In the *destination-based* strategy, we break ties in favor of the node with the higher node ID for some (even numbered) destinations, and the opposite for other (odd numbered) destinations. In the *degree-based* strategy, we break ties in favor of the node with the lowest node degree.[2]

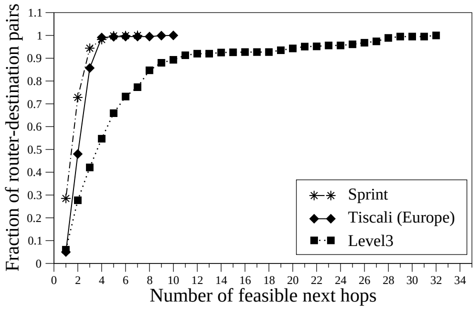

Fig. 1. Number of next hops achieved for all source-destination pairs in different networks.

The three tie-breaking strategies were tested in the POP-level Tiscali, Sprint and Level3 networks[3] with different link delay granularities. By a granularity of x seconds, we mean an approach where all links with a delay from 0 to x seconds get a weight of 1, links with delay x to $2x$ get a weight 2 etc. These tests show that the degree-based tie breaking strategy consistently gives a lower number of instances where $k_i(t) = 1$. This is not unexpected, since high-degree nodes will often have a higher number of feasible next-hops for a destination, and are not dependent on "luck" with the tie-breaker to achieve route diversity. Our tests also show that a coarser granularity gives an improved path diversity. Setting the delay granularity to 3-ms seems to be a good trade off which gives a low fraction of instances where $k_i(t) = 1$, while it is low enough to clearly prefer short links over long ones. In the rest of this paper, we use the degree-based tie-breaker, and measure link delays with a 3-ms granularity.

Figure 1 shows a CDF for the number of feasible next-hops achieved for all source-destination pairs for the POP-level Tiscali, Level3 and Sprint networks. We see that there is a clear connection between the connectivity of the networks and the path diversity achieved at each node. In the well-connected Level3 network, some nodes potentially have a large number of loop-free next-hops for some destinations. Depending on the value of the parameter $\mathbf{K}$, many of these potential next-hops will not be installed in the FIB. Next, we observe that in the Tiscali and Level3 networks, we achieve $k_i(t) > 1$ for about 95% of node pairs. In the Sprint network, which has a significantly sparser topology, we see that $k_i(t) = 1$ for a somewhat higher fraction of node pairs. This is not surprising, since the connectivity of the network topology is important for the degree of multipath routing achieved in any destination-

[2] The information needed to calculate this is readily available if a link state protocol is used to convey topology information. With a distance-vector protocol, each node must inform its direct neighbors about its node degree.

[3] More information about the network topologies is given in the appendix.

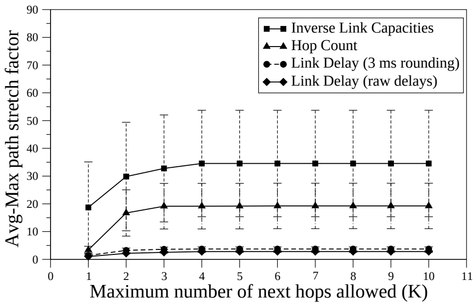

Fig. 2. Maximum possible path stretch given by the routing, averaged over all source-destination pairs, for the Sprint network.

based routing.[4]

Note that even if there are some cases where $k_i(t) = 1$, so that a node can only use a single next-hop to reach a destination t, there are usually few links for which *no* traffic can be diverted to another link in case of congestion. Consider a situation where a node can only reach a destination t through link l. However, another destination t' that is reached through l can also be reached through an alternative link l'. Hence, we are still able to avoid congestion on link l by diverting some of the traffic destined for t' to l'. The method used to assign traffic to different next-hops is the subject of the next section.

C. Propagation delays

We look at the propagation delay when we allow routing over $\mathbf{K}$ loop-free paths as described in this section. We look at the delays when link weights are equal to the raw propagation delays, and when they are equal to propagation delays measured with a 3-ms granularity. For reference, we also include results when all links have unit weights, and when links have weights equal to the inverse of their capacity (we explain how link capacities are determined in the appendix).[5]

Figure 2 shows the upper bound on the *path stretch* in the Sprint network with the different routing schemes, as a function of $\mathbf{K}$. The path stretch is the ratio between the propagation delay of a given path, and the propagation delay of the shortest path between the two endpoints. We calculate the path stretch of the longest valid route that can be taken for each source-destination pair in the network, and show the average value over all source-destination pairs. The vertical bars indicate the 95% confidence interval for the average.

As one could expect, routing based on link delays has a significantly smaller worst-case path stretch than the hop-count or inverse capacity weight settings. This difference increases when we move from single-path ($\mathbf{K} = 1$) to multi-path ($\mathbf{K} > 1$) routing, motivating our use of delay as the main metric for calculating next-hops. We also observe that there is a large variance between different source-destination pairs, as indicated by the wide confidence intervals. The use of 3-ms delay granularities does not significantly increase the max path stretch compared to using raw delays. Note that the path stretch plotted in Fig. 2 will only be experienced if the *worst possible choice* with respect to propagation delay is made at *each single router* on the path from a source to a destination, which is unlikely to occur in practice.

III. Load adaptation

The selection of next-hops described above does not depend on the load in the network, and only changes in response to topology changes. In our approach, all load adaptation takes place by adjusting the amount of traffic a router sends to each feasible next-hop. Here, we describe how this load balancing is performed. The load balancing method is designed to minimize delays under normal load, react to congestion based on local information only, and avoid reordering of packets belonging to the same (TCP) flow.

The assignment of traffic is done solely based on a router's local view of the load situation in the network. Routers do not distribute any information about their load to other routers in the network. Hence, our approach requires no additional signaling, and can work directly with any existing routing protocol.

A. Load balancing objective

Importantly, the feasible next-hops towards a destination t are not treated equally by our load balancing mechanism. Instead, they are *ranked according to their propagation delays*. The basic idea is that we want to use the shortest path next-hop as long as the utilization of this link stays below a certain threshold, which we call the *spilloverThreshold* θ. Only when the utilization of the shortest path next-hop exceeds θ, will traffic bound for t be sent on the second shortest path. When the secondary next-hop also exceeds this threshold, we will start using the third shortest path, and so on until the utilization of all available paths reach θ. If two or more routes have the same delay, we adopt a min-max load balancing strategy to choose among them, where new flows are assigned to the out-link with the least utilization. We also use the min-max strategy if all feasible next-hops are above θ.

The reasoning behind this strategy is that up to a certain utilization threshold, the queuing delay of a link is negligible. When utilization exceeds this threshold, queues might start to build up. We aim to avoid the unpredictability of queuing and congestion, and thus we prefer congestion-free feasible next hops, even if that means increased propagation delays.

Note that θ can be set differently for each link, depending on the link capacity. The selection of θ will depend on the capacity of the link. In a low capacity link, a sudden burst of traffic can cause queuing delays even if the averaged utilization stays well below 70-80%. In a high capacity link on the other hand, statistical multiplexing allows us to run a link at a higher utilization before queuing delays start to accumulate.

[4] For the Sprint network, the high number is caused mainly by the fact that there are only two trans-atlantic links. Some nodes can reach a significant fraction of the network through only one of these.

[5] Note that we do not look at any link weight settings produced by heuristics like the one presented in [8] for two reasons - they are not designed to work with multipath routing, and they are tuned to work with a particular expected input TM. Here we focus on multipath routing, and methods where the routing is based on physical properties rather than an expected traffic pattern.

B. TCP awareness

Splitting traffic belonging to a single TCP flow over several paths can lead to packet reordering, which can trigger TCP slow-start with its adverse impact on performance. To avoid this, we need a mechanism that makes sure that packets belonging to the same TCP flow is forwarded along the same path.

The most straightforward way of preventing reordering is to hash the packet header directly to the correct out-link. This way, traffic within the same TCP flow will always select the same next-hop. Such an approach works fine as long as the splitting ratio between the available next-hops stays (relatively) constant. In our routing scheme, we want to be able to dynamically adjust the splitting ratio between the selected next-hops on short timescales, based on the current load situation. The direct hashing approach is therefore not suitable.

In our approach, traffic is assigned to a next-hop on the granularity of flows defined by the `<srcIP, destIP, srcPort, destPort, protocol>` 5-tuple in the IP header. We maintain a cache where the hash value of a packet header is mapped to the correct out-link. Once a flow is mapped to a next-hop, all packets belonging to that flow will be sent over the given link. The load balancing is performed by assigning *new* TCP flows (previously unseen hash values) to the currently preferred out-link as determined by the load balancing objective. This extra level of indirection gives us a more fine-grained control of the next-hop selection, at the cost of maintaining some soft-state in the routers. Modern routers have extensive support for performing per-flow operations, and maintaining the necessary state to maintain consistent forwarding for a flow is not prohibitive in most cases [4]. Note that since the goal is to avoid reordering, an entry in the flow cache can be deleted after a very short period of inactivity.

C. Stability considerations

Stability is a major issue in any load-adaptive routing protocol. However, the phrase has different meanings in different contexts. Network operators often speak of stability in the context of routing oscillations in the control plane. An instability event would typically mean a situation where a route flaps between two different links. In the TE literature, stability is often defined as a situation where the amount of traffic on each link converges asymptotically to a steady value under constant traffic input. This puts the focus on data plane stability. In the context of Intelligent Route Control, the focus has been on the "self load effect", meaning that a flow can oscillate between paths because the system does not consider the effect of that flow on the underlying path [10]. This is an example of control plane instability, coupled with data plane instability.

Our focus is on control plane stability. Different from several previous methods, we do not adapt routes based on load changes, which can result in instabilities both in the control plane and the data plane [25]. Instead, we prefer load-insensitive routing, and perform adaptation at the forwarding plane, i.e., by load-balancing between a stable set of feasible next-hops. This way we clearly avoid control plane instabilities. Additionally, the pinning of flows to fixed routes avoids flow-level oscillations.

Homeostasis does not, however, guarantee stability in the data plane. It is possible to construct traffic scenarios where cyclic dependencies between the links in the network will lead to fluctuations in link loads even with a constant input TM. Such data plane oscillations cannot be avoided with a local load balancing scheme like ours. We argue, however, that this type of instability is less important in practice, since the input traffic changes on short timescales in anyway as a result of variations in the TM, caused by variations in flow sizes, flow arrival rates, and flow throughputs.

D. Performance of the load balancing

Unlike previous proposals [6], [7], [12] our load-balancing is performed at each router based on local information only. This has the advantage of avoiding a large signaling overhead, but is limited by not knowing the load situation of the entire path to the destination. For example, a node experiencing congestion on one of its links has no way of signaling to its upstream neighbors so that the incoming traffic can be reduced. Hence, there is a risk of making decisions at one node that can have adverse consequences at a downstream node. However, in a well provisioned network where each node individually selects from a set of feasible next-hops, the chance that all or most nodes make a bad decisionat the same time is small. As we shall see in the evaluations, Homeostasis can successfully avoid congestion by spreading traffic on more paths. This emphasizes the importance of achieving several feasible next-hops at each router.

We compare the performance of our load balancing method to that of four other load balancing strategies. With all load balancing strategies, each router selects up to $\mathbf{K} = 4$ next-hops for each destination, using link delays measured with a 3-ms granularity as the weight metric as explained in Sec. II. The details of the evaluation setup is described in the appendix. The following load balancing schemes are chosen because they are *local* adaptation schemes that determine the amount of traffic to be sent through each next-hop without communicating with other nodes:

DEFT-LB. This scheme assigns flows to a next-hop with a probability that decreases exponentially with the extra length of the path compared to the shortest path, as described in [27]. Here, the relative amount of traffic sent through feasible next-hop j is e^{-x}, where $x > 0$ is the difference in distance through j and the shortest possible route. Note that we do not compare with the complete DEFT scheme, which also involves an optimized link weight setting based on a TM estimate. We only compare the Homeostasis load balancer with their local load-balancing rule.

EIGRP style. This scheme assigns flows to next-hops with a probability that is inversely proportional to the distance through each feasible next-hop, as described in the configuration documentation for the EIGRP routing protocol [5]. If

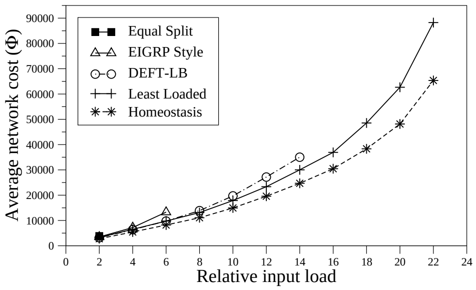

Fig. 3. Average network cost in the Tiscali network for different load balancing schemes.

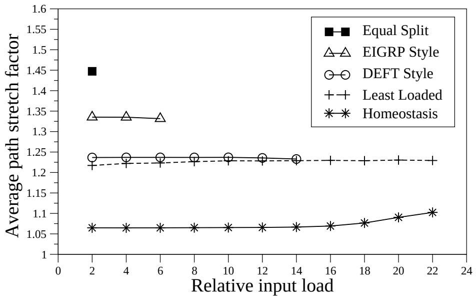

Fig. 4. Average path stretch experienced in the Tiscali network for different load balancing schemes.

$d_{i,max}(t)$ denotes the distance to t through the feasible next-hop representing the *longest* route, then each feasible next-hop gets an integer relative weight $d_{i,max}(t)/d_{i,j}(t)$.

Least Loaded. This scheme assigns a flow to the feasible next-hop link that currently has the lowest utilization.

EqualSplit. This scheme splits the flows equally between the feasible next-hops.

We use a cost function Φ to describe the cost associated with routing a certain traffic demand through the network. Φ is a convex cost function that associates a cost with each link based on its utilization, so that heavily loaded links get a much higher cost than lightly loaded links. We adopt the commonly used cost function introduced in [8].

Figure 3 shows the average network cost Φ in the Tiscali network over two hours of simulated time for different load balancing methods. For each routing scheme, we keep increasing the input load as long as the input traffic can be successfully routed, meaning that less than 1% of the flows are dropped because of congestion (see appendix). We see that the Homeostasis and Least Loaded schemes are able to route a much higher traffic demand than the other load balancing schemes. However, Homeostasis routes the same amount of traffic at a lower cost, since it seeks to minimize delays and hence normally uses fewer links. These two schemes also performed the best in similar experiments in the Sprint and Level3 networks. This shows that exploiting local information about the current load situation when assigning flows to a feasible next-hop can significantly increase the capacity of a network with multi-path routing. The EqualSplit scheme gives very poor performance, since it unnecessarily sends traffic over potentially long paths even when these are congested. The same is to a lesser extent true for the *EIGRP-like* scheme, which gives lower performance than *DEFT-LB*.

Figure 4 shows the average path stretch experienced by the flows in the same network. The Homeostasis scheme gives a significantly smaller path stretch than the other schemes, due to its ability to use the shortest path as long as this is lightly loaded. This difference is consistent over all our three networks, and becomes even more significant if we consider the *maximum* path stretch experienced by any flow in our simulations (not shown). The path stretch with the Homeostasis scheme increases slightly for high input loads, when more traffic is sent over non-shortest paths.

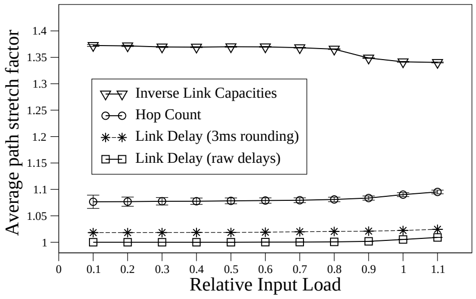

Fig. 5. Average path stretch for all simulated flows in the Sprint network.

We also look at the impact of routing (i.e., the link weight setting) on the experienced path stretch. Figure 5 shows the average path stretch experienced by all simulated flows in the Sprint network for different link weight settings, using $\mathbf{K} = 4$. For all weight settings, we use the Homeostasis load balancing scheme, and we keep increasing the input load until 1% of the flows cannot be routed. Note that link capacities are set so that all weight setting strategies are able to route approximately the same amount of traffic.

We observe that the actual path stretch in the network is well below the upper bounds reported in Fig. 2. Using delay as a link weight metric results in much lower delays than the Inverse Capacity link weight setting, which focuses mainly on minimizing link utilization.[6]

IV. Robustness evaluation

In this section, we evaluate the robustness of Homeostasis, focusing mainly on the performance of the load balancing method. We develop a novel way of evaluating robustness, by calculating the fraction of TMs of different classes that can be successfully routed in a network, and by showing results for both network cost and path stretch.

[6] For the Inverse Capacity link weight setting, we see a decrease in path stretch at high input load. This happens when the high-capacity links exceed θ, and traffic is diverted to (shorter) low-capacity routes.

A. Deviations from expected operating conditions

We look at four different challenges to network performance. Three of them are related to variations in the input traffic, while the fourth relates to topological variations. First, we look at *Gaussian variations* in the TM. In these scenarios, we draw each element in the TM from a normal distribution $N(\mu, \sigma^2)$, where μ is the corresponding value from the base TM, and the variance $\sigma^2 = \alpha\mu$. α is known as the *peakedness* of the traffic, and is set to 1 in our experiments [20]. Second, we look at *hot-source* scenarios, where all TM elements from a single source s are doubled, while all other elements are left unchanged. Such situations could for example be caused by a failure in a neighboring AS, which causes traffic to be rerouted and enter the network at an unusual ingress node, or by a sudden increase in the popularity of some content. Third, we look at corresponding *hot-sink* scenarios, where all TM entries with a given destination t are doubled. This situation might also be triggered by external failures, or by a sudden increase in demand at a certain location. Finally, we look at performance (after rerouting) in single link failure scenarios. In each class of deviations, we measure the fraction of scenarios that can be successfully routed, and the cost Φ and the average path stretch for each of these scenarios.

B. Operating range

We first look at the diversity in the traffic input that each load balancing method can successfully route. Table I shows the fraction of TMs where more than 99% of flows could be routed for the Tiscali (T), Sprint (S) and Level3 (L) networks and 4 different classes of TM variations. We also indicate the 95% confidence intervals for the fractions. We look at 50 random TMs with Gaussian variations. For the other classes of variations, we run one simulation for each possible hot-source, hot-sink and single-link failure scenario. The input load with the base TM is the same for all load balancing schemes, and is set high enough that the differences become clear. At this input load, only the Homeostasis (H) and the Least Loaded (LL) load balancing strategies are able to route any of the input TMs, with the Homeostasis scheme having the highest success rate in two of the three networks. These results emphasize the benefits of adaptive load balancing for handling a much wider range of traffic inputs.

Figure 6 shows the fraction of routable TMs with Gaussian variations in the Sprint network, as a function of increasing input load. We observe that the fraction of routable TMs drops rapidly from 100% to 0 when the load exceeds a certain critical point. That critical load is signifcantly higher with Homeostasis (and Least Loaded) than with the other schemes.

C. Robust performance

Next, we assess the quality of routing for TMs that can be successfully routed, using network cost and propagation delay as metrics. Figure 7 shows the performance of different load balancing schemes with Gaussian variations in the TM (top) and single-link failures (bottom). Results are shown for the Tiscali, Sprint and Level3 networks. Each mark represents one TM (top) or topology (bottom). The input load is set to the highest value that all load balancing schemes could successfully route with the default TM. Marks in the lower left corner indicate good performance (low delay and low cost), while marks in the upper right corner indicate poor performance (high delay and high cost).

		Gaussian	Hot-src	Hot-sink	Link fail
Tiscali	H	27%±12%	67%±17%	60%±18%	75%±10%
	LL	73%±12%	83%±14%	70%±17%	83%±9%
	D	0%	0%	0%	0%
	EI	0%	0%	0%	0%
	ES	0%	0%	0%	0%
Sprint	H	100%	97%±6%	97%±6%	91%±7%
	LL	96%±5%	88%±11%	79%±14%	66%±12%
	D	0%	0%	0%	0%
	EI	0%	0%	0%	0%
	ES	0%	0%	0%	0%
Level3	H	100%	100%	98%±4%	100%
	LL	63%±13%	83%±11%	64%±14%	77%±5%
	D	0%	0%	0%	0%
	EI	0%	0%	0%	0%
	ES	0%	0%	0%	0%

TABLE I

FRACTION OF ROUTABLE TMS

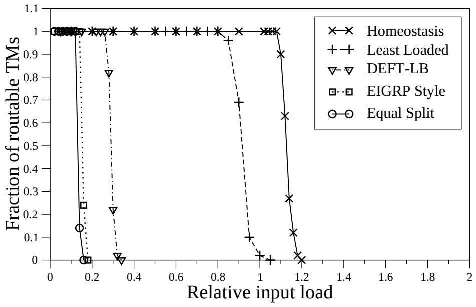

Fig. 6. Fraction of routable TMs with Gaussian variations in the Sprint network for increasing input load.

We observe that Homeostasis consistently outperforms other load balancing strategies. It has the lowest delay and the lowest cost in all networks and for all types of TM variations. The relative differences in cost and delay vary between the three networks we look at. The difference between the load balancing schemes is smaller in the sparse Sprint topology, where the number of available paths is smaller. This indicates that the advantage of the Homeostasis load balancing approach is stronger when routing and the underlying topology allow more path diversity.

The trends seen here are consistent also for the hot-source and hot-sink TM variations. We have also run simulations at the higher input load used in Tab. I, where only the Homeostasis and Least Loaded schemes can successfully route any TMs. Those simulations show similar differences in the performance of these two schemes.

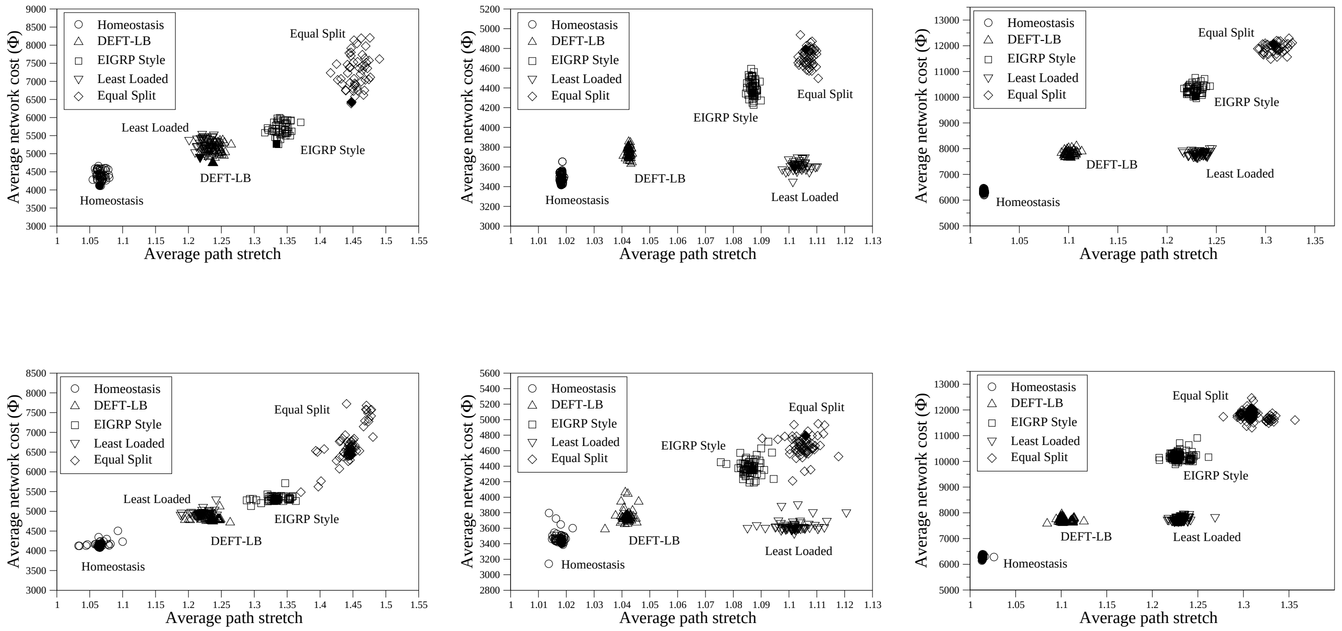

Fig. 7. Cost vs path stretch with Gaussian variations in the TM (top) and single-link failures (bottom). Tiscali (left), Sprint (middle) and Level3 (right)

V. Related work

The related work in routing is vast, and we cannot give an extensive overview here. We refer the reader to [3], which gives a good overview of the TE and load-adaptive routing efforts up to 2001, and [15] which gives an overview of the entire routing “landscape”.

Several existing and proposed routing protocols are designed to use unequal-cost paths to a destination. An approximation to minimum-delay routing is presented in [22]. As in our method, DASM [29] and MDVA [23] determine the set of feasible next-hops based on the minimum distance through each neighbor. They do not use a value **K** to restrict the number of next-hops, but instead install *all* neighbors announcing a lower distance to the destination in the forwarding table. EIGRP [1] selects as next-hops the neighbors with a distance to the destination that is no longer than the shortest path times a variance factor. None of these protocols consider the issue of increased delays with multipath routing, and they do not address how load should be split over the available next-hops.

Early routing protocols [13], [14] adapted to load variations by dynamically changing the link weights used to compute shortest paths. This approach was later shown to be unstable and to give poor performance [25], and was mostly abandoned in later protocols. In this work, we do not alter link weights or select new paths, but adapt to traffic variations by assigning new flows to the next-hop selected by our load-balancing algorithm.

Much work has been done in the area of traffic engineering, where the goal is to optimize the amount of traffic that can be routed with the available resources. Popular approaches include the use of MPLS tunnels [26], and various methods for OSPF/IS-IS link weight tuning; [8], [9], [21], [27], [28] and many others. Common for these methods is that they seek to optimize performance based on a given expected traffic matrix. They are static in their nature, and cannot deal with congestion events that take place in a timescale from few seconds to few hours.

The challenges in using estimated TMs for traffic engineering was discussed in [20]. Because of such challenges, work has been done to develop *oblivious* routing schemes that perform well under any traffic input [2], or routing that balances worst case and average performance for a set of TMs [24], [30]. Similar to us, their goal is to be robust to variations in the input traffic. However, these methods are quite different in using optimization to find the best (static) routes and traffic split ratios. These optimizations involve solving LP formulations with a large number of constraints and variables, and must be redone when the topology changes. The focus on robustness to traffic variations involve a tradeoff where performance (in terms of delay) under light traffic loads is sacrificed. Also, these methods require explicit routing (e.g., MPLS) to give good performance, where a central entity decides the split ratios in each router. In this work, we do not base our routing on an estimate of the TM, and we adapt to short-term traffic dynamics by adjusting the load balancing between the available paths.

Similar to our approach, so-called *on-line traffic engineering* methods [6], [7], [12] aim to dynamically split traffic between several available paths. These methods are also dependent on explicit path routing in order to set up unequal-cost paths to each destination. While our method adjusts the load-balancing based only on a local view of the load situation, these methods use probes or feedback from routers to assess the quality of the end-to-end path, incurring significant signalling overhead.

The strategy of load-balancing based on local load informationhas been studied (and implemented) in circuit-switched networks [11], [16]. Both the circuit-switched context and the selection of alternate routes makes this method different from ours.

VI. Conclusions

We have presented Homeostasis Routing as a method for robust, multi-path, load-responsive routing. The goal of Homeostasis Routing is to provide low-delay, congestion-free, load-responsive routes even under sudden topological changes and significant traffic matrix variations. This is achieved by installing multiple routes to each destination in the forwarding tables, and intelligently assigning flows to these based on the local view of the load situation. Routes with a shorter propagation delay are preferred as long as the network is lightly loaded, and more routes are gradually phased in as the load increases.

The proposed method has been evaluated using flow-level simulations in three POP-level ISP topologies. The evaluations demonstrate the potential benefits of our method with respect to handling a wider range of traffic and topology changes. Homeostasis, as any other routing or TE scheme, cannot always avoid congestion because it cannot ”create capacity” and it cannot do admission control. We have illustrated however that Homeostasis can route larger traffic volumes before it reaches the saturation point in which congestion appears in some links.

Important steps in our future work will be to further evaluate Homeostasis Routing using packet-level simulations and a prototype implementation that can capture the effects of transport layer congestion control. Plans for future work also include methods that increase the number of routes available at each router.

Appendix

In this appendix, we describe the evaluation setup used in our simulations. The simulations are on the flow level. This allows us to perform a large number of simulations in realistic networks with many possible TMs and topological changes. It does not, however, allow us to investigate the effects of transport layer congestion control or different queuing methods. Packet-level simulations and a prototype implementation are important next steps that we plan to pursue in future work.

A. Topologies and traffic matrix

We perform our tests on three selected topologies based on existing (inferred) ISP networks. These are the POP-level Tiscali Europe, Sprint and Level3 networks as of 2001, taken from the Rocketfuel project website [18]. This source gives us the connectivity of the topology, and an estimate of the propagation delays of the links. For our purposes, we are only interested in bi-connected network graphs, since we expect that all ISP networks are bi-connected for reliability reasons. The Rocketfuel topologies also contain some single-connected nodes, probably due to difficulties in detecting backup links. In our work, we prune all nodes with degree one from the topologies. The resulting topologies are listed in Tab. II. The average path lengths refer to shortest path routing with unit link weights. We note that the Level 3 network is much more densely connected than the other networks. Sprint has the most sparse topology, and in particular has only two trans-atlantic links connecting the two halves of the network.

	POPs	**Links**	**Link delays**	**Avg path length (hops)**
Tiscali	29	73	0.1 - 8.9 ms	1.87
Sprint	32	64	0.1 - 42.0 ms	3.04
Level3	46	268	0.3 - 38.5 ms	1.93

TABLE II
Rocketfuel topologies.

For each topology, we create a base TM describing the long-term expected traffic from each source to each destination in the network. The TM is generated by using a simplified gravity model as described in [19]. Each node in the network is given a weight corresponding to the population of the urban region that it represents.[7] The relative traffic demand between two nodes is then determined by the product of the weight of the two nodes. Each traffic matrix element is then multiplied by a load scaling factor to determine the absolute amount of traffic between the two nodes. While this model does not necessarily give the same traffic pattern that is observed in the real networks (it does not consider that some nodes function as gateways to other networks, giving a higher weight than the population alone would indicate), it gives a traffic pattern that can intuitively be related to real-world metrics.

The real-world capacities of the links in our networks are not publicly available. In our model, a link can have three distinct capacities; 100 Mbps, 400 Mbps or 1600 Mbps. These numbers capture how bandwidth is typically available in distinct quantities, and they are large enough compared to the intensity of individual flows to allow for a significant degree of multiplexing. To assign link capacities, we start by assigning the lowest link capacity to all links. We then calculate the utilization of all links with our base TM, using shortest path routing with three different link weights: the inverse of the link capacity, the propagation delay, and unit link weights (hop count routing). We identify the link with the highest utilization, and increase the capacity of this link, if it does not already have the maximum capacity. We repeat this process until we achieve a realistic mix of high- medium- and low capacities.

B. Dynamic input traffic

We model flow arrivals as a global Poisson process, and we control the input load in our simulation by varying the inter-arrival time of this Poisson process.

The size of a flow is drawn from a truncated Pareto distribution with a scaling parameter 1.3. The minimum flow

[7] Population numbers are taken from `http://www.citypopulation.de/`

size is 8 MB, while the max flow size is 8 GB. The *rate* of the flow can take three distinct values - 0.5 Mbps with a probability of 30%, 1 Mbps with a probability of 60%, and 10 Mbps with a probability of 10%. These values give flow durations ranging from a minimum of 6.4s to a theoretical maximum of more than 35 hours. Our parameter settings are based on the analysis of a packet trace presented in [17]. By focusing on these flow sizes and intensities, we capture most of the traffic observed in the Internet, while disregarding a large number of small, short-lived flows that do not contribute much to the total traffic. Finally, each flow is randomly assigned to a source-destination pair based on the probabilities given by the TM.

C. Performing an experiment

For each scenario defined by a topology, a routing and load balancing method, a TM and a load scale factor, we simulate two hours worth of traffic in the network. Because heavy-tailed flow size distributions have difficult convergence properties, we describe in detail how we perform our measurements.

For a given scenario, we first let the simulation run for a time period until the initial transient is over, i.e., at the end of the initial increasing trend in the number of active flows. After every interval of 1s, we calculate the current cost Φ_t in the network. We also maintain a smoothed version of the cost $\Phi^{'}_t = 0.999 * \Phi^{'}_{t-1} + 0.001 * \Phi_t$. We define t_0, the time when the initial transient period ends, as the time when $|\Phi^{'}_t - \Phi^{'}_{t-1000} < \epsilon|$. In our experiments, we use $\epsilon = 0.001$. The spilloverThreshold θ is set to 0.7.

After time t_0, we run the simulation for a fixed duration of 2 hours. This period is more than long enough for the average cost Φ_{avg} to stabilize, at least for all simulations we have run. In every sampling interval of 1 second, we measure the utilization of each link in the network, and calculate the network cost Φ_t. For every flow that is routed, we measure the path stretch experienced by the flow.

With this kind of simulation, we are not able to model what happens when the offered load exceeds the capacity of the network. If routing a flow through the network results in higher load than the capacity on a link, we completely remove this flow from the simulation, and continue to route the next flow. We count such ”flow removals” as failures of a given routing/load balancing scheme to handle the offered load. We say that a given scenario can be *successfully routed* if less than 1% of flows are removed in this manner. The fraction of successfully routed scenarios under different challenges to normal operations is one of our performance metrics.

References

[1] Bob Albrightson, J.J. Garcia-Luna-Aceves, and Joanne Boyle. EIGRP - a fast routing protocol based on distance vectors. In *Proceedings Networld/Interop*, May 1994.

[2] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. In *Proceedings of ACM SIGCOMM*, 2003.

[3] Debashis Basak, Hema Tahilramani Kaur, and Shivkumar Kalyanaraman. Traffic engineering techniques and algorithms for the Internet. Technical report, Rensselear Polytechnic Inst., 2002.

[4] Dana Blair. Cisco. Personal communication.

[5] Cisco. *Enhanced Interior Gateway Protocol*. http://www.cisco.com/application/pdf/paws/16406/eigrp-toc.pdf.

[6] Anwar Elwalid, Cheng Jin, Steven H. Low, and Indra Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings INFOCOM*, pages 1300–1309, 2001.

[7] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. REPLEX – dynamic traffic engineering based on Wardrop routing policies. In *Proceedings CoNEXT*, Lisboa, Portugal, dec 2006.

[8] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings INFOCOM*, pages 519–528, 2000.

[9] Bernard Fortz and Mikkel Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756 – 767, May 2002.

[10] Ruomei Gao, Constantine Dovrolis, and Ellen Zegura. Avoiding oscillations due to intelligent route control systems. In *Proceedings, INFOCOM*, Barcelona, Spain, 2006.

[11] R.J. Gibbens, F.P. Kelly, and P.B. Key. Dynamic alternate routing-modeling and behaviour. In *Proceedings 12th International Teletraffic Congress (ITC12)*, Turin, Italy, 1988.

[12] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the tightrope: responsive yet stable traffic engineering. In *Proceedings SIGCOMM*, pages 253–264, 2005.

[13] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *Proceedings SIGCOMM*, 1989.

[14] John M. McQuillan, Ira Richer, and Eric C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, pages 711–719, 1980.

[15] Deep Medhi and Karthikeyan Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann, 2007.

[16] Srihari Nelakuditi, Zhi-Li Zhang, Rose P. Tsang, and David H. C. Du. Adaptive proportional routing: A localized QoS routing approach. *IEEE/ACM Transactions on Networking*, 10(6), December 2002.

[17] Ravi Prasad and Constantine Dovrolis. Beyond the model of persistent TCP flows: Open-loop vs closed-loop arrivals of non-persistent flows. In *Proceedings 41st Annual Simulation Symposium*, April 2008.

[18] Rocketfuel topology mapping. WWW. http://www.cs.washington.edu.

[19] Matthew Roughan. Simplifying the synthesis of Internet traffic matrices. *SIGCOMM Comput. Commun. Rev.*, 35(5):93–96, October 2005.

[20] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring Internet backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings International Teletraffic Congress (ITC-18)*, 2003.

[21] Ashwin Sridharan and Roch Guerin. Making IGP routing robust to link failures. In *Proceedings of Networking*, Waterloo, Canada, 2005.

[22] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum-Delay Routing. In *Proceedings of ACM SIGCOMM*, 1999.

[23] S Vutukury and J.J. Garcia-Luna-Aceves. MDVA: a distance-vector multipath routing protocol. In *Proceedings INFOCOM*, 2001.

[24] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: traffic engineering in dynamic networks. In *Proceedings SIGCOMM*, pages 99–110, 2006.

[25] Zheng Wang and Jon Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *SIGCOMM Comput. Commun. Rev.*, 22(2):63–71, 1992.

[26] X Xiao, A Hannan, and L Ni. Traffic engineering with MPLS in the Internet. *IEEE Network Magazine*, March 2000.

[27] Dahai Xu, Mung Chiang, and Jennifer Rexford. DEFT: Distributed exponentially-weighted flow splitting. In *Proceedings INFOCOM*, Anchorage, AK, USA, may 2007.

[28] Dahai Xu, Mung Chiang, and Jennifer Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. In *Proceedings INFOCOM*, Phoenix, AZ, USA, April 2008.

[29] William T. Zaumen and J.J. Garcia-Luna-Aceves. Loop-free multipath routing using generalized diffusing computations. In *Proceedings INFOCOM*, pages 1408–1417, San Francisco, CA, USA, mar 1998.

[30] Chun Zhang, Zihui Ge, Jim Kurose, Yong Liu, and Don Towsley. Optimal routing with multiple traffic matrices tradeoff between average and worst case performance. In *Proceedings ICNP*, November 2005.