On the Stability of Adaptive Routing in the Presence of Congestion Control

Eric J. Anderson and Thomas E. Anderson
Department of Computer Science and Engineering
University of Washington
eric,tom@cs.washington.edu

Abstract— **Efficient use of network resources has long been an important problem for large-scale network operators. To this end, several recent research efforts have proposed automated methods for optimizing routes based on traffic measurements. However, these efforts have not considered the stability of the dual feedback control mechanisms of adaptive routing and congestion control, when operating together. In this paper, we demonstrate that an important class of adaptive routing algorithms can yield stable optimal routes in the presence of congestion control, provided that either the congestion control mechanism is fair or the network workload behaves under reasonable constraints. We further show that one or the other of these assumptions is necessary for this class of adaptive routing algorithms – otherwise, unstable, sub-optimal routes may result in some pathological cases.**

I. Introduction

Efficient use of network resources has long been an important problem for large-scale network operators. By exploiting redundant paths in response to changing workload conditions and router/link failures, adaptive routing has the potential for significantly improving network performance. Although some believe that adaptive routing is unnecessary, arguing that many networks are under-loaded by design, a recent study at a major service provider showed spikes of over 90% load on some links over periods of many minutes and spikes over 50% load over periods of hours [1].

As a result, automated mechanisms for adaptive routing have attracted significant research attention over the last thirty years, resulting in substantial progress in theory [2], [3], [4], more practical systems [5], [6], [7], and techniques for adding adaptive routing to existing routing protocols [8], [9], [10]. Despite this, there has been little progress towards the deployment of adaptive routing in operational networks; instead, large networks typically rely on human traffic engineers to optimize their systems. In part, this lack of progress is due to a well-founded fear among network operators that automated adaptive routing will be unstable in practice, yielding both sub-optimal performance and poor reliability due to routing oscillations. In fact, early experience with adaptive routing in the ARPAnet showed exactly these effects [5].

In this paper, we focus on a specific sub-problem in this area: whether adaptive routing algorithms can be designed to be stable in the presence of congestion control. By *stability* we mean the notion – made precise in section II below – that the repeated operation of the adaptive algorithm quickly converges to a good solution, from which the system will not substantially vary except as traffic changes. This is an interesting question due to the interaction between the feedback control mechanisms of adaptive routing (adjusting routes in response to changes in traffic load) and congestion control (adjusting the traffic load in response to changes in network capacity, i.e., routes). With congestion control, an inefficient initial routing configuration can become self-confirming, as the routing mechanism sees only the traffic that the congestion control algorithm allows into the network. In fact, we show by proof and counterexample that some adaptive routing mechanisms are stable and optimal when combined with fair congestion control mechanisms, but are potentially unstable and/or reach sub-optimal solutions when combined with other less friendly congestion mechanisms.

Earlier work on adaptive routing has sidestepped this issue by assuming that the workload is "quasi-static" – inelastic to routing changes and measured over a long enough interval (e.g., a day) so that all pent-up demand can be met by the network. By contrast, we are interested in the limits of routing adaptation to short term events, such as link failures and temporary increases in load. Over shorter time scales, host congestion control can significantly reduce measured load, and user studies have shown that web surfing declines when the network is slow [11]. To a network operator charging for bandwidth use, this reduction in demand represents a lost opportunity for revenue.

We restrict the class of algorithms under consideration in two significant ways to make our analysis more tractable. First, we assume that the adaptive routing algorithm uses as input a point-to-point traffic matrix measured at the network edge, instead of interior measurements of link utilization. Traditional adaptive routing systems such as ARPAnet [5] have been based on interior link utilizations, as these are easier to measure. However, such systems are inherently less stable and take longer to converge than systems based on measurements of a traffic matrix, since routing changes can directly affect link utilizations, and link utilizations directly affect routing decisions. Gallager was able to design a "stable" adaptive routing algorithm based on link utilizations, but his analysis assumes infinite queues inside the network and no congestion control [12]. We could find no obvious way to extend the proofs presented in this paper to Gallager's algorithm.

By contrast, a traffic matrix provides the measured demand from every point in the network to every other point; in the absence of congestion control, this can be used to compute optimal routes using linear or non-linear optimization algorithms [13], [14]. Further, it is not unreasonable to assume the availability of this information, at least for optimizing routes within a single service provider [15]. Although many legacy pieces of network equipment lack the capability to collect and report this information, most new routers from Cisco [16], Juniper [17], and Foundry [18] have added this capability, and over the long term, we expect this support to become standard.

Second, we restrict our discussion to the class of adaptive routing algorithms that have the property of minimizing the maximum utilization across any link in the network (the so-called "min-max" property) [13]. This class of algorithms includes not only the simple min-max algorithm using a linear constraint solver, but also other solutions that optimize routes through the network without affecting the min-max property. As just one example, the authors have developed an adaptive routing algorithm that optimizes for path length while preserving min-max; this algorithm is as efficient as shortest path under low to moderate load, and as efficient as min-max when the network becomes congested [19]. As we discuss in more detail in Section IV, there are often multiple solutions that preserve the min-max property; to avoid useless route flapping, we restrict ourselves to those min-max algorithms that contain systematic tie-breaking rules.

Of course, adaptive routing algorithms that preserve min-max are not the only ones worth consideration; for example, some algorithms attempt to minimize average link utilization or minimize average queueing delay. Our proof techniques, however, do not extend to these other criteria, unless the algorithm also preserves the min-max property. Finally, we note that our work is somewhat, but not completely, independent of the mechanism used to implement adaptive routes. Both source routing, as in MPLS [20], [21], and destination-based fractional routing [22] are general-purpose enough to be able to implement the min-max property for all networks and workloads. However, many service providers instead use OSPF or IS-IS shortest path algorithms for computing routes, adjusting weights on each link to achieve traffic engineering objectives. As recent work has shown, computing a set of link weights to optimize for min-max, or even approximating min-max to within a constant factor, is NP-hard for general networks [8]. Thus our work does not extend, at least efficiently, to link weight setting algorithms, despite their attractiveness as a transition path to deploy adaptive routing on legacy systems. A full discussion of the practicality of implementing min-max based adaptive routing is beyond the scope of this paper; in the associated thesis, we show that min-max based routing can be implemented efficiently, reliably, and with substantial performance gains relative to load-insensitive routing for typical network topologies and workloads [19].

The key insight for our work is that routing algorithms that preserve the min-max property (henceforth referred to as "min-max" for notational convenience) leave the maximum headroom for congestion control to expand into newly improved routes, thus avoiding the self-confirming property of poorly chosen routes. Our results show that the following hold in the presence of congestion control, provided that the workload can be feasibly routed:

1) If the current routing configuration is nearly optimal, min-max algorithms converge immediately to the optimal configuration.
2) If the congestion control algorithm is fair, min-max algorithms converge to an optimal, stable solution from any initial configuration, regardless of congestion.
3) Given weak assumptions about the congestion control algorithm, min-max algorithms converge under conditions of reasonable network congestion.

Further, we show by counterexample the converse – that min-max algorithms do not achieve optimal performance in the presence of congestion control without either the assumption of fair congestion control or limited workloads.

The rest of the paper presents our results in more detail. We first define the problem more formally in Section II. We then present our principal results in Section III. In Section IV, we consider how to resolve route oscillation for networks where min-max does not specify a unique solution. Finally, in Section V, we summarize our findings.

II. Problem definition

Broadly, in this work we consider traffic-aware adaptive routing algorithms that (after initialization) perform the following three steps:

1) Measure some properties of the network, such as traffic flow or link utilization.
2) Compute a set of routing tables, based on the measured properties and the current system state.
3) Operate the network under the given set of routing tables.

In principle, the process can be repeated ad infinitum.

A. An example of undesirable behavior

Consider the example network displayed in Figure 1. This example is loosely based on the transcontinental network in the early ARPAnet [5]. In the Figure, all traffic moves from left to right, and all links have capacity equal to one unit of traffic per period.

Suppose the network is initialized with simple shortest-path routing tables, and suppose weights are assigned to each link according to the following rule:

- less than 40% utilization, $w = 1$
- between 40% and 60% utilization, $w = 2$
- over 60% utilization, $w = 3$

Suppose further that the four nodes $A, B, C,$ and D each originate one-quarter unit of traffic per period, all destined for node Z. All links in the network, except perhaps links L_{upper}, L_{lower} and $E \rightarrow G$, will have utilization of less than 40% throughout, and hence for these links $w = 1$ throughout the operation of the algorithm. Initially, the shortest-path routing sends three quarter-units of traffic (from A, B, C)

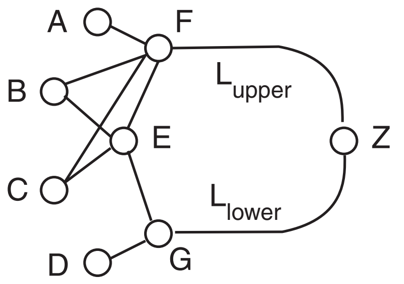

Fig. 1. An example of an oscillating network. Each of the four nodes on the left sends an equal amount of traffic to the node on the right. The weight assigned to the upper and lower links is equal to the number of flows that traverse that link. The network is initialized with shortest path routing. The result is oscillation.

through L_{upper} and one quarter-unit through L_{lower}. Then, the algorithm assigns weights of $w = 3$ and $w = 1$ respectively to those links for use in the subsequent routing calculation. In the second step, the paths to the destination Z are recomputed. The new weighted shortest path routing leaves the traffic originating at A and at D unchanged, but sends traffic from nodes B, C through L_{lower} instead of through L_{upper}. Based on that new routing, utilization of the links L_{upper} and L_{lower} changes. Then, weights are recomputed to be $w = 1$ for L_{upper}, $w = 2$ for the link from E to G, and $w = 3$ for L_{lower}. The paths computed using these weights are identical to the original routing; thereafter, the system oscillates. In each configuration, there is congestion on either the upper link or the lower.

A routing with less congestion can be obtained by routing half the traffic through the upper link and half through the lower. This routing can be achieved through link weights, for example, by changing the weight on the links joining B to node E and C to node F to $w = 3$. In that case, the traffic sourced at A, B is routed through L_{upper} while that at C, D through L_{lower}. In general, as we see in this example, the best set of link weights for a given traffic pattern may require changes to weights on links that themselves experience no congestion.

These problems are not just theoretical; the adaptive routing scheme used in the early ARPAnet exhibited route instability in practice [5], [2]. Contributing to the instability were several factors, including the use of instantaneous queue length (itself rapidly varying) as a measure of utilization, and a topology that relied on two transcontinental links (loosely modeled in Figure 1 above). The observed instability was significantly reduced in later versions of the adaptive routing algorithm, through the use of *damping* (or hysteresis) and thresholding. However, damping reflects a direct tradeoff between stability on the one hand and responsiveness, the system's ability to react to changed circumstances, on the other [2]. The tuning of these parameters was both delicate and specific to the topology. The ARPAnet designers summed up their experience as follows [5]:

> These [parameter] values ... are not necessarily appropriate for all network topologies.

Subsequent research confirmed that stability is a significant issue with link utilization based schemes [9].

B. *A definition of stability*

We identify four distinct (but related) desirable properties of a "stable" adaptive system:

1) The system should converge to some state: over time, if the input does not change or makes only small changes, the system behavior should not exhibit large fluctuations.
2) The state the system converges to should be "near-optimal", achieving a routing that produces good overall system performance.
3) The system should not exhibit "route oscillation", i.e., not only broader system behavior but the routing tables themselves should converge.
4) These properties should hold independent of the network topology.

Note that stability and convergence without optimality is trivial – static routing along the shortest number of hops is stable and converges quickly, but it does not always yield an efficient solution for various workloads and topologies. Among adaptive routing systems, the experience of link weight schemes indicates that link utilization based routing does not always exhibit favorable stability properties. We show in the next section that min-max adaptive routing algorithms, by contrast, do have favorable stability properties.

III. Convergence

In this section, we develop a formal model for analyzing the stability of adaptive routing systems that operate in the context of congestion control. The adaptive routing systems we discuss here take as input measured point-to-point traffic. This measured traffic can be seen as an estimate of some underlying "true demand". To analyze convergence, we make the formal assumption that this "true demand" is fixed. Then we can identify an *optimal* routing as the routing produced by the min-max algorithm when presented with this "true demand". We measure the performance of an algorithm by comparing its routing to this optimal routing.

Initially, we we will assume that under the optimal routing the "true demand" can be feasibly routed, at least when the entire measurement interval is considered. We remove this assumption in Section III-I. Even with this assumption, however, given an arbitrary starting point, there may be congestion. If the workload rapidly changes, for example, the previous routing may be ill-adapted to the new demand. In this event, congestion control mechanisms will inhibit the observed traffic, and measured traffic may fall short of the true demand. The adaptive routing system will be presented with this measured traffic, and under repeated application of the cycle of routing and measurement will attempt to improve network performance. In general, therefore, we must determine whether and how quickly a min-max algorithm adapts to congestion from an initial sub-optimal routing.

In practice, observed traffic may fail to equal "true demand" for any of several reasons. In addition to congestion control, these include router buffering, link errors, stochastic fluctuations, and users reducing their demand when faced with a slow network. We restrict our formal model and our analysis to the effect of congestion control.

A. *Definition of the model*

We formally define a model for an adaptive routing system as follows. Consider a *routing network* $G = (V, E, T)$ of nodes $i, j, k \in V$, capacitated edges $e \in E$ each with capacity $c(e)$, and routing tables[1] $T \overset{\text{def}}{=} \{\phi_{ij}(k)\}$, where $\phi_{ij}(k)$ denotes the proportion of traffic leaving node i destined for j that is assigned to the link $i \to k$. By definition, the proportions across all outgoing links must add to one, i.e., we have

$$\forall i, j, \sum_{k \in V} \phi_{ij}(k) = 1.$$

We identify three demand quantities:

- d_{ij}, the *true demands*, which for our purposes are predefined constants of the system;
- r_{ij}, the *reference demands*, used by the min-max algorithm to compute the routing tables T; and
- m_{ij}, the *measured traffic*, which are the result of applying d_{ij} to the network $G = (V, E, T)$.

We suppose that the application of the true demands to the network G (including its routing tables) results in a *measured flow* $f_m = f_m(P)$ along each path P, whose point-to-point path flows[2] sum to the measured traffic m_{ij}:

$$\sum_{P: i \to j} f_m(P) = m_{ij} \tag{1}$$

The measured traffic is used at the subsequent step as input to the min-max algorithm, i.e., as reference demands used to compute routing tables. Formally, the min-max algorithm computes a complete *reference flow* $f_r(P)$:

$$\sum_{P: i \to j} f_r(P) = r_{ij}^{(n+1)} = m_{ij}^{(n)} \tag{2}$$

The iterative operation of the adaptive scheme can be described schematically as follows:

$$m_{ij}^{(0)} \overset{(A)}{\to} r_{ij}^{(1)} \overset{(B)}{\to} m_{ij}^{(1)} \overset{(A)}{\to} r_{ij}^{(2)} \overset{(B)}{\to} m_{ij}^{(2)} \to \ldots$$

Step A consists of equating the reference demands to the measured traffic of the preceding step, and computing a reference flow via the min-max algorithm. Step B consists of applying the true demands d_{ij} to the network whose routing tables T are derived from the reference flow.

The challenge, then, is to determine whether this process converges to the optimal routing, regardless of the topology and regardless of the starting point for $m_{ij}^{(0)}$. If so, then we call the adaptive routing system *stable*.

[1]For convenience, we refer to "routing tables" here as the formal analogue of "forwarding tables" used in network terminology.

[2]We use the term "flow" in the multicommodity flow sense of the aggregate traffic along a path, rather than an individually identified transaction such as a TCP connection. Where necessary we use "TCP flow" to denote the latter. Formally, a flow is the amount of traffic from any input to any other output along a single path in the network.

B. *Useful definitions*

In this subsection, we introduce the definitions of unrestricted flow, a network's margin of support for demands, and an optimal network G_{opt}.

Definition 1: The *unrestricted flow* f_u, defined with respect to a particular set of demands d and routing tables T, is the flow computed by assigning to each source node the demands d_{ij} and applying at each node the proportions given by T, without regard to capacity constraints in the network G. In particular, f_u may not be a feasible flow in the sense of a multicommodity flow formulation.

Whenever a reference flow determines the routing tables, the unrestricted flow along each path $P : i \to j$ is simply the reference flow along P increased by the ratio of true to reference demands between the endpoints, or d_{ij}/r_{ij}.

Suppose the unrestricted flow f_u with respect to a particular set of demands d_{ij}, taken as a whole, presents no congestion.

Definition 2: If for all edges e the unrestricted flow f_u satisfies

$$\sum_{P: e \in P} f_u(P) < (1 - \epsilon) c(e) < c(e),$$

then we say that G *supports the demands* d *with margin* ϵ.

If the unrestricted flow with respect to a particular set of demands is feasible, but not necessarily bounded away from full utilization by ϵ, we say instead that the demands can be routed:

Definition 3: If for the unrestricted flow f_u

$$\sum_{j} f_u(P) \leq c(e),$$

then G can *route* the demands d.

Finally, we denote by G_{opt} the network G with routing tables obtained by applying the min-max algorithm to the true demands d. Because min-max minimizes worst-case utilization, it produces a routing with the best possible ϵ. Thus we have the following characterization of min-max:

Proposition 1: If the network G supports a particular set of true demands d_{ij} with margin ϵ, then the network G_{opt} also supports d_{ij} with margin ϵ.

Proof: G_{opt} is G with tables produced by the min-max algorithm. The result is immediate from the definition. □

C. *Assumptions about congestion behavior*

In this subsection, we identify three basic assumptions about the system behavior under congestion. We believe these assumptions are reasonable and natural properties of congestion-controlled networks, especially for Internet routing. Using these assumptions, we prove convergence under many conditions. Later, by making more specific assumptions about the congestion control mechanism, and particularly about its fairness, we demonstrate a stronger conclusion about convergence of min-max algorithms.

Specifically, we want to identify how the system determines the measured traffic m for a given combination of routing tables T and true demands d, and especially how congestion control arbitrates among flows through network bottlenecks.

Formally, we define a *bottleneck edge* as an edge for which total flow along the edge equals its capacity.

We make the following three general assumptions about the behavior of any congestion-controlled system under conditions of significant network load:

Assumption *1:* The measured flow $f_m(P)$ is uniquely defined by the network $G = V, E, T$ and the input demands d.

Assumption *2:* On any path P, the measured flow does not exceed the unrestricted flow, $f_m(P) \leq f_u(P)$. That is, congestion control serves only to slow down the rate of traffic relative to the true demands.

Assumption *3:* On any path P, if the path contains no bottleneck edge, then the measured flow is the unrestricted flow, $f_m(P) = f_u(P)$. That is, unless there is congestion along a path, the network admits the entire flow along that path based on the true demands.

The first assumption formally requires that measured traffic at any point does not exhibit hysteresis (dependence on the existing flow). In practice, even where a congestion control mechanism offers statistical fairness, a particular newly-introduced connection may not achieve its bandwidth share instantaneously. Our assumption here ignores these time-dependent effects.

The second assumption implies that the true demands are the limit of what end hosts want to send through the network (or are permitted to send across rate-limited access links) during the measurement interval. Applying congestion control does not cause these true demands to increase.

The third assumption is valid when congestion control is applied path by path, affecting only flows through a bottleneck link and, indirectly, bottleneck links upstream and downstream of that link. This model is reasonable where each individual connection (TCP or UDP flow) follows a unique path to its destination. Most multipath routing systems attempt to preserve single path routing for individual connections [22]. If a single TCP flow is split among multiple paths transparently to the end host, however, TCP congestion control feedback may generalize from loss on one path to reduce flow along all paths in aggregate. In this case Assumption 3 is not strictly valid. However, we expect that this assumption is broadly and substantially correct, at least on average [23].

Note that these three assumptions do not address how bandwidth is allocated among competing flows through a bottleneck link, i.e., the *fairness* of the congestion control mechanism. We defer a discussion of the role of fairness in the stability of adaptive routing to Section III-H.

These assumptions have one straightforward but important immediate consequence for min-max algorithms. Because a min-max algorithm produces a linear solution to a linear program, the routing solution based on measured traffic is always no worse – in the sense of worst-case link utilization – than the optimal solution based on true demands:

Proposition 2: Suppose the network G supports a particular set of true demands d_{ij} with margin ϵ. Then any solution produced by a min-max algorithm based on any measured traffic m_{ij} supports that measured traffic with margin ϵ. Specifically, that solution's reference flow f_r on any edge e is strictly less than its capacity by the margin ϵ,

$$\sum_{P:e\in P} f_r(P) \leq (1-\epsilon)c(e)$$

Proof: From Proposition 1, the solution G_{opt} produced by min-max on true demands d_{ij} has margin ϵ. By Assumption 2, the measured demands m_{ij} do not exceed the true demands. Since min-max optimizes for link utilization, the routing obtained by min-max for the measured demands, described by the reference flow f_r, cannot have a worst-case utilization greater than margin ϵ. For G_{opt} is one such routing, with a worst-case utilization when restricted to the measured demands of no worse than margin ϵ, and min-max optimizes over all such. □

D. Immediate convergence for near-optimal starting conditions

Our first major result is that min-max will often converge to an optimal solution immediately.

Proposition 3: Suppose the network G supports the true demands d_{ij} with margin ϵ. If at any step the measured traffic $m_{ij}^{(n)}$ is within ϵ of the true demands,

$$\forall i, j,\ m_{ij}^{(n)} \geq (1-\epsilon)d_{ij},$$

then min-max converges to the optimal solution in at most two steps.

Proof: Because the measured traffic $m_{ij}^{(n)} \leq d_{ij}$, the solution computed by min-max at step $n+1$ (using $m^{(n)} = r^{(n+1)}$ as input) supports the reference demands with margin ϵ (Proposition 2). If the measured traffic is within ϵ of the true demands, then the unrestricted flow f_u is not more than $1/(1-\epsilon)$ times the reference flow. Hence the unrestricted flow never encounters a bottleneck. In this case by Assumption 3 the measured flow $f_m(P) = f_u(P)$ on all paths. Thus the traffic measured at the next step is the true demand, $m^{(n+1)} = d_{ij}$. Then at step $n+2$ min-max takes the true traffic as input and computes an optimal solution. Because the optimal routing supports the true demands, at every succeeding step the measured traffic is again equal to the true demands. □

We reiterate that this is the common case in practice. Unless a network is seriously overloaded, or the initial routing is particularly inefficient, the congestion encountered should not affect the measured traffic so severely as to distort the routing beyond that required by Proposition 3. This stability property is fundamentally stronger than for link utilization based schemes, where even if utilization is well below link capacity, the routing could develop oscillations and fluctuate without converging. (See Figure 1 and accompanying text.)

Of course, we would like to be able to show convergence regardless of topology or offered load. While we can't show this in the general case for min-max algorithms, we can demonstrate convergence under more severe workload conditions than assumed in Proposition 3. However, the stronger results are not as straightforward, as the next example shows.

E. An example of a non-monotonic network

It would be straightforward to prove convergence if total network flow increased at every iteration. However, this is not always the case when there are multiple bottlenecks per path, as we show in this subsection.

First we begin with a definition of the notion of monotonic increase of flow.

Definition 4: A network is *monotonic at step n* if either the measured traffic $m_{ij}^{(n)}$ is within ϵ of the true demands, or is in aggregate greater than the reference demands $r_{ij}^{(n)}$ by at least an amount ϵC_{min}, for some constant C_{min} of the network:[3]

$$\sum_{i,j} m_{ij}^{(n)} - \sum_{i,j} r_{ij}^{(n)} \geq \epsilon C_{min}$$

Definition 5: A network is *always monotonic* if, given $G^{(0)}$ the initial configuration, each network $G^{(n)}$ in the sequence given by the iteration of the min-max algorithm is monotonic at step n.

For networks that are always monotonic, convergence is indeed automatic:

Theorem 1: An always monotonic network converges in a number of steps proportional to ϵ^{-1}.

Proof: By the definition of monotonic networks, either the sum of the measured traffic increases at every step by ϵC_{min}, or the network given by the reference demands supports the true demands and Proposition 3 applies. The iteration cannot take more than $\lceil (\sum d_{ij})/\epsilon C_{min} \rceil$ steps. $\square$

Unfortunately, not all networks are necessarily monotonic, as the example illustrated in Figure 2 and defined in Table I indicates. This example has three demands and two bottleneck links.[4] Assume routing tables based on the reference flow indicated in Table I, 80 from each of A and B and 10 from C. This flow adds up to flow of 90 through each bottleneck link. Thus the network supports the reference flow with margin 10%. After applying the true demands to these routing tables, the unrestricted flow through each bottleneck link is $130 (= 80 + 50)$. This unrestricted flow exceeds the capacity of each of the bottleneck links. A measured flow of 50 from each source, though not necessarily "fair", satisfies each of the three Assumptions 1-3. In that case, the total measured traffic is less than the total reference flow $(50 + 50 + 50 < 80 + 80 + 10)$. See Table I.

Because the measured traffic in this example is smaller than the reference flow, and because in our formulation the reference flow is based on the measured traffic at the previous

[3]In the results that follow, C_{min} will be bounded by the minimum capacity of any link in G.

[4]The network includes an additional path from C to Z, not shown in the Figure, of large capacity and with a routing table entry at C of zero. This augmented network supports the true demands with margin 10%. We omit this link from the diagram for simplicity; the analysis is unaffected.

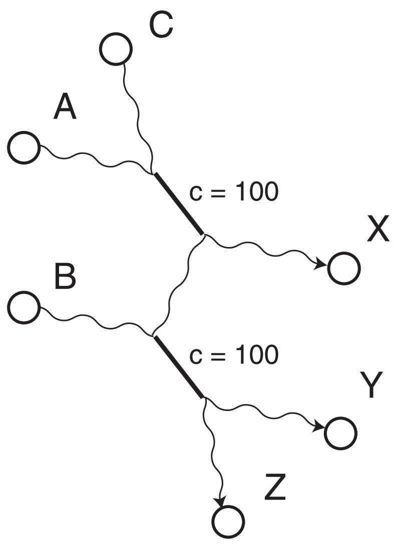

Fig. 2. An example of a non-monotonic network. The true demand, reference flow, and measured traffic for this three-commodity network are listed at Table I. The bottleneck links are indicated in heavy type. The reference flow has maximum utilization of 90%. The measured traffic has two bottleneck links. The network includes an additional path from C to Z, of large capacity and with $\phi = 0$, that is not shown in the Figure. The example illustrates a situation where the aggregate measured flow can be less than the reference flow on which the routing tables are based.

TABLE I
WORKLOAD FOR THE NON-MONOTONIC EXAMPLE OF FIGURE 2.

Demand	True Demand	Reference Flow	Measured Traffic
$A \to X$	80	80	50
$B \to Y$	80	80	50
$C \to Z$	50	10	50

step, this example demonstrates that total measured traffic can decrease from one step of the min-max algorithm to the next. Thus an adaptive routing scheme based on a min-max algorithm is not necessarily monotonic.

F. Bottleneck-weighted total traffic

In the example of Figure 2, the 50 units of traffic from C contributes to two bottlenecks. In that case, adding up the total traffic does not necessarily result in a quantity that monotonically increases from step to step. However, if we keep track of the number of bottleneck links that each path traverses, we can indeed prove a related monotonicity property.

We introduce the following additional definitions and notation. Let $\mathcal{B}$ denote the set of bottleneck edges.

Definition 6: The *bottleneck-weighted measured flow* $f_b(P)$ is the measured flow $f_m(P)$ multiplied by the number of bottleneck links in P:

$$f_b(P) = f_m(P) \cdot |\{e : e \in P, e \in \mathcal{B}\}|$$

Definition 7: The *bottleneck-weighted measured increment* $f_i(P)$ is the difference between the measured flow $f_m(P)$ and the reference flow $f_r(P)$, multiplied by the number of bottleneck links in P (determined by the measured flow):

$$f_i(P) = (f_m(P) - f_r(P)) \cdot |\{e : e \in P, e \in \mathcal{B}\}|$$

Definition 8: The *bottleneck-weighted total measured traffic* is the sum of all bottleneck-weighted measured flows:

$$f_b = \sum_P f_b(P)$$

Definition 9: The *bottleneck-weighted total measured increment* is the sum of all bottleneck-weighted measured increments:

$$f_i = \sum_P f_i(P)$$

Our principal result in this section is the following:

Theorem 2: The bottleneck-weighted total measured increment is at least equal to the product of ϵ and the sum C of the capacities of all bottleneck edges. In particular, if there is at least one bottleneck edge, the bottleneck-weighted total measured increment is strictly positive.[5]

Remark. Since on any given path the measured flow may in fact decrease compared to the reference flow (as in the example in Figure 2), the bottleneck-weighted total increment is in general a sum of positive and negative terms. However, according to the theorem, the total is always positive.

Proof: The bottleneck-weighted total measured traffic, by definition, is the sum of all bottleneck-weighted measured flows. This is the sum of all measured flows on all paths containing bottleneck edges, weighted by the number of bottleneck edges, i.e., counting each bottleneck edge in the path separately. Conversely, every capacitated edge in the network is a bottleneck edge for every flow through it. Thus the bottleneck-weighted total measured traffic is simply the sum of flows through each bottleneck edge e, which because capacitated is $\sum_{e:e\in\mathcal{B}} c(e)$. Since the reference flow was computed by min-max, the network supports the reference flow with margin ϵ (Proposition 2). Thus the corresponding bottleneck-weighted total traffic of the reference flow, weighted according to the same bottleneck links, is bounded above by $\sum_{e:e\in\mathcal{B}}(1-\epsilon)c(e) = (1-\epsilon)C$. Thus the bottleneck-weighted total measured increment is at least ϵC. □

The importance of this result can be seen in its corollaries. First, we note that, while it is not always true (as shown by the Example in Figure 2) that the aggregate measured traffic increases from step to step, at least one point-to-point measured traffic amount m_{ij} must increase.

Corollary 1: At least one measured demand $m_{ij}^{(n)}$ exceeds its corresponding reference demand $r_{ij}^{(n)}$ by $\epsilon C/n^3$, where C is the sum of the capacities of all bottleneck edges, and $n = |V|$ is the number of nodes.

Proof: The bottleneck-weighted total increment is made up of terms $(m_{ij} - r_{ij})$, each weighted by a number of bottleneck links between 1 and n. The largest of these terms must be at least as large as the average, which is $\epsilon C/n^2$. The weighting factor cannot exceed n; dividing the average by n gives a lower bound on the largest value for $(m_{ij} - r_{ij})$. □

This fact has an important consequence:

Corollary 2: There is no stable state other than the optimal routing.

Proof: Any stable state for which the measured traffic has no congestion is optimal, by Proposition 3. Any stable state for which there is congestion must have $m_{ij}^{(n+1)} = m_{ij}^{(n)}$ for some n and for all i, j; but $r_{ij}^{(n+1)} = m_{ij}^{(n)}$ by construction, and by the above corollary the measured traffic must increase for at least one i, j. □

This result also offers us an easy convergence proof for a plausible (though impractical) modified min-max algorithm. Suppose we apply min-max not to measured traffic $r_{ij}^{(n+1)} = m_{ij}^{(n)}$ but instead to the greater of measured traffic and reference demand,

$$r_{ij}^{(n+1)} = \max(r_{ij}^{(n)}, m_{ij}^{(n)}).$$

Note that since both the reference and measured demands are never larger than the true demands, the reference demand can be feasibly routed. As applied to this modified algorithm, Corollary 1 now implies that the total of reference demands increases at every step by at least $\epsilon C/n^3$. Since C is bounded below by the minimum capacity edge in G, the modified scheme converges. However, this modified min-max algorithm is not likely to be a practical algorithm for Internet routing. If traffic demands change substantially, the input will erroneously reflect some mixture of new and old demands. Even if the traffic pattern itself is stationary, over time the effect of stochastic fluctuations would cause considerable distortion in the reference demands as compared to measured traffic.

Another consequence of Theorem 2 is convergence in the case where traffic encounters at most one bottleneck link. This third corollary arguably covers all but the most extreme cases of network congestion. Recall that a link is a bottleneck if and only if it is fully utilized throughout the entire measurement interval; with the exception of access links, most large networks are provisioned so that complete link congestion is normally both infrequent and temporary.

Corollary 3: If in the sequence of networks $G^{(n)}$ each path of the measured flow $f_m(P)$ never encounters more than one bottleneck link, then the min-max algorithm converges.

Proof: We show that the total measured flow increases from step to step. Under the hypothesis, all weights on bottleneck paths are one. We break down the total measured flow into paths that encounter bottlenecks and paths that do not. For the flows that do not encounter bottlenecks, Assumptions 2 and 3 imply that the measured flow cannot decrease. For the flows that do encounter bottlenecks, Theorem 2 implies the measured flow must actually increase in aggregate by the bottleneck-weighted total measured increment.

We reduce the above description to symbols. Denote by $\mathcal{B}$ the set of paths containing a (unique) bottleneck edge. First, measured traffic can be decomposed into traffic that traverses

[5]If there are no bottleneck edges in the measured traffic, by Assumption 3 the measured traffic represents the true demand, and the min-max algorithm converges immediately.

bottleneck links and traffic that does not:

$$\sum_{i,j} m_{ij}^{(n+1)} = \sum_{P:P\in\mathcal{B}} f_m^{(n+1)}(P) + \sum_{P:P\notin\mathcal{B}} f_m^{(n+1)}(P)$$

Next, on uncongested paths, the measured traffic is the unrestricted flow (Assumption 3), which in turn is at least as large as the reference flow (Assumption 2):

$$\sum_{P:P\notin\mathcal{B}} f_m^{(n+1)}(P) = \sum_{P:P\notin\mathcal{B}} f_u^{(n+1)}(P) \geq \sum_{P:P\notin\mathcal{B}} f_r^{(n+1)}(P)$$

The measured flow through bottleneck links is, in aggregate, greater by at least $(1-\epsilon)C$ than the reference flow through those links (Theorem 2):

$$\sum_{P:P\in\mathcal{B}} f_m^{(n+1)}(P) \geq (1-\epsilon)C + \sum_{P:P\in\mathcal{B}} f_r^{(n+1)}(P)$$

The sum of all reference flows at step $n+1$ is the reference demand, which (by construction) is the same as the measured demand in step n:

$$\sum_{P:P\in\mathcal{B}} f_r^{(n+1)}(P) + \sum_{P:P\notin\mathcal{B}} f_r^{(n+1)}(P) = \sum_{ij} r_{ij}^{(n+1)} = \sum_{ij} m_{ij}^{(n)}$$

Combining these equations, we have:

$$\sum_{i,j} m_{ij}^{(n+1)} \geq (1-\epsilon)C + \sum_{ij} m_{ij}^{(n)} \tag{3}$$

Of course, if the first sum is empty, i.e., there are no bottleneck edges, then Proposition 3 applies. □

We expect that this "single-bottleneck" assumption covers a large number of cases in practice. However, we would like to show convergence regardless of workload; as the next example shows, we are unable to do so without making additional assumptions about how resources are allocated among flows when there is congestion.

G. *Limits of the min-max property*

The above discussion applies to any adaptive routing algorithm that provides a guaranteed worst-case link utilization, regardless of how the particular mechanics of congestion control allocate bandwidth on bottleneck links, except as noted in the broad assumptions in subsection III-C. It is natural to ask whether the min-max property is enough by itself to demonstrate convergence, that is, whether Corollary 3 can be extended to all configurations.

We present a counterexample that answers this question in the negative. In this example, we construct a pair of configurations that form a "two-cycle", one for which a particularly clumsy min-max algorithm, and a particularly perverse bandwidth allocation, combine to cause the system to alternate between suboptimal configurations.

Consider four demands $A \to X, B \to Y, C \to Z, D \to T$, and two bottleneck links L_1, L_2 each with capacity 100. The capacities of all other links are sufficiently large not to encounter congestion. Each demand has two alternate routing paths to its corresponding destination, as listed in Table III. In each case, one of those paths passes through both bottleneck links, and the other passes through only one bottleneck link. Specifically, for $A \to X$ and $C \to Z$, the one-bottleneck-link paths go through link L_1 only, while for $B \to Y$ and $D \to T$, the one-bottleneck-link paths go through link L_2 only.

The values for true, reference, and measured traffic used in the example are as listed in Table II. If each demand is routed along its single-bottleneck-link path, the entire set of demands can be routed with 90% utilization. In the initial configuration, however, only A and B are routed along their one-bottleneck-link paths, while C and D are routed through their two-bottleneck-link paths. The utilization according to the reference flow is 65%.[6] Upon application of the true demands to this initial configuration, congestion appears at both links. Suppose, perversely[7], that the congestion control mechanism resolves this congestion as follows: each of the demands A and B is accepted through the network in the restricted amounts of 10 units each, while each of C and D is accepted in the full amount of 45 units each. (Note that TCP congestion control in the presence of FIFO queues can yield just this kind of pathological allocation [24].) In this case, each link L_1 and L_2 experiences full utilization, i.e., 100 units. The min-max algorithm observes measured traffic of $A = 10, B = 10, C = 45, D = 45$.

Now suppose, again perversely, that at the next step min-max routes this observed traffic to meet a utilization guarantee of 65%, by routing A and B along their two-link paths, and C and D along their one-link paths. This could still preserve the min-max property if, for example, some other link in the network is at least 65% utilized. When the true demands are applied to this routing, each link L_1 and L_2 again experiences congestion. To resolve this congestion, suppose, again perversely, that this time the congestion control mechanism admits A and B in the full amount of 45 units each, but C, D only in the restricted amount of 10 units each. In this case the min-max algorithm observes traffic of $A = 45, B = 45, C = 10, D = 10$. Now suppose the min-max algorithm computes a routing for this measured traffic by assigning A and B to one-link paths and C, D to two-link paths. This routing again achieves a utilization guarantee of 65% for this second set of observed traffic.

Because the configuration resulting from the second application of the min-max algorithm is identical to the initial configuration, the cycle can repeat indefinitely. Both configurations are evidently suboptimal with respect to true traffic, since all of the true traffic can be routed within 90% utilization, while the actual configurations experience congestion at every step. Note that this example does not contradict Theorem 2, as the bottleneck-weighted total measured increment is positive at each step relative to the previous reference demand, when weighted according to the new measured bottlenecks.

One potential solution to this problem would be to restrict

[6]These specific values assigned are essentially unimportant to the outcome.
[7]But consistent with Assumptions 1-3.

TABLE II

A COUNTEREXAMPLE TO CONVERGENCE: DEMANDS.

Demand	True	Ref-1	Meas-1	Ref-2	Meas-2
$A \to X$	45	45	10	10	45
$B \to Y$	45	45	10	10	45
$C \to Z$	45	10	45	45	10
$D \to T$	45	10	45	45	10

TABLE III

A COUNTEREXAMPLE TO CONVERGENCE: ROUTING PATHS.

Demand	Path 1	Path 2
$A \to X$	L_1, L_2	L_1
$B \to Y$	L_1, L_2	L_2
$C \to Z$	L_1, L_2	L_1
$D \to T$	L_1, L_2	L_2

the class of adaptive routing algorithms further, to "cascading" min-max: after solving for a minimum worst-case utilization, identify the bottleneck link with the greatest utilization, and form a residual network by eliminating all traffic through that link. Then recursively repeat the algorithm on the residual. This approach addresses the above counterexample by requiring the minimal use of L_1, L_2. It is an open question whether the "cascading" approach can be refined to a polynomial-time algorithm (taking into account the systematic evaluation of ties), as well as whether such a modification would lead to a complete proof of convergence for all workloads.

H. Convergence for a particular congestion control model

Without stronger assumptions about the behavior of the system under congestion, we cannot infer anything about traffic demands not actually observed. As a result, we cannot prove convergence in all cases. In this subsection, we develop a model for deterministically resolving congestion, which we believe to be simple and plausible. Under this model, the min-max algorithm converges in all cases.

Inspired by recent results on TCP behavior [23], we posit a "proportional bandwidth sharing" model for deterministically resolving congestion on bottleneck links. Specifically, in order to determine the measured traffic for a given reference flow, we first compute the unrestricted flow. Then we identify the most overcapacitated edge, that is, the edge with the largest ratio of utilization to capacity. (If that ratio is less than one, we admit the entire unrestricted flow, and we are done.) For each path through that edge, we scale the amount of flow along that path by the ratio of utilization to capacity, with the result that the scaled flows add precisely to full utilization. Next, fixing all scaled flows, we repeat the process for all remaining flows: identify the most overcapacitated edge; scale all hitherto unscaled flows by that ratio (calculated with respect to the residual capacity); and repeat until no edge is overcapacitated. Continuing in this way, all flows receive at least their proportionate share of bandwidth on the most congested link.

Under this deterministic model, any min-max algorithm converges. In the remainder of this subsection, we demonstrate this claim. We begin with an important observation about this congestion model.

Proposition 4: Under the above model of congestion control mechanics, let $R > 1$ be the worst case utilization percentage for the unrestricted flow. Then at least $1/R$ of each demand is routed.

Proof: In the first step in the construction of proportional bandwidth sharing described above, the worst-case scaling ratio R is identified, and the path flows routed through that bottleneck link are scaled by a percentage $1/R$. Now consider the second step, where a link with worst-case utilization ratio is identified, after fixing all the path flows routed in the first step (the residual utilization ratio). The worst-case utilization ratio so identified is no greater than R. Otherwise, the unrestricted utilization ratio (determined by adding back in the fixed flows, increased by the percentage R) would be greater than R, which contradicts the choice of the first link. Thus the scaling ratio for path flows scaled in the second step is no less than $1/R$. By an easy induction argument, the same result holds for all subsequent links. □

Based on this observation, the stability proof is straightforward.

Proposition 5: Under the above model of congestion control mechanics, the min-max algorithm converges in a number of steps $k = O(1/\log(1-\epsilon))$.

Proof: Suppose R is the worst-case ratio of the unrestricted flow for the initial conditions. By Proposition 4, the measured flow along every path is at least the ratio $1/R$ of the unrestricted flow. Hence, the sum of measured flows for any source-destination pair is at least the ratio $1/R$ of true demand. Apply min-max to the measured flow. The result, used as the reference flow in the subsequent step, is routed within headroom of ϵ (see Proposition 2). The unrestricted flow along each path is determined from the reference flow by multiplying by the ratio of true to reference demand, which ratio is no more than R. Then the total unrestricted flow through any link is never more than $R \cdot (1-\epsilon)$ times its capacity. But in that case the maximum ratio of unrestricted flow to link capacity cannot exceed $R \cdot (1-\epsilon)$. Thus when the bandwidth sharing model is applied at the second step of min-max, the worst-case unrestricted link utilization has been reduced to $R \cdot (1-\epsilon)$. At each step the worst-case link utilization is reduced by a factor of $(1-\epsilon)$. When inevitably the worst-case ratio is reduced to less than one, the convergence is immediate from Proposition 3. The convergence rate is given by $(1-\epsilon)^k < 1/R$, or $k \sim 1/\log(1-\epsilon)$, a number of steps approximately inversely proportional to the guaranteed headroom. □

I. Congested networks

In the preceding discussion, we assumed throughout that the network itself was not overloaded, i.e., that the true demands could be routed with margin ϵ for some $\epsilon > 0$. In this subsection, we investigate convergence and optimality where

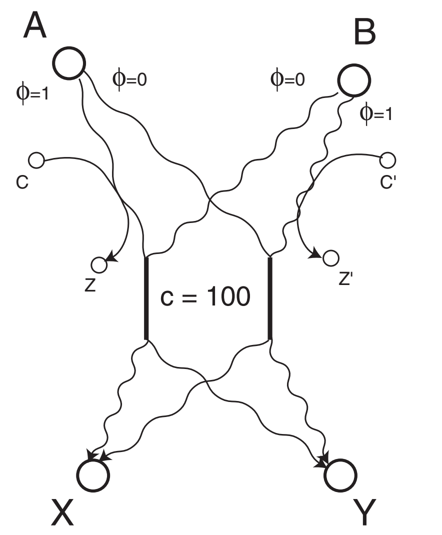

Fig. 3. An example of suboptimal convergence in a congested network. The true demands, reference, and measured traffic are listed in Table IV. If the bandwidth-sharing mechanism offers precedence to traffic $A \to X$ and $B \to Y$, then with the routing ϕ as indicated, no additional traffic from $C \to Z$ can be supported. If the routing is changed so that ϕ is reversed ($0 \leftrightarrow 1$), then the additional traffic can be supported. But the measured traffic at X and Y considered alone gives no indication of which of these two routings is preferred.

TABLE IV

Suboptimal convergence in a congested network.

Demand	True Demand	Reference Flow	Measured Traffic
$A \to X$	100	100	100
$B \to Y$	100	100	100
$C \to Z$	??	??	??

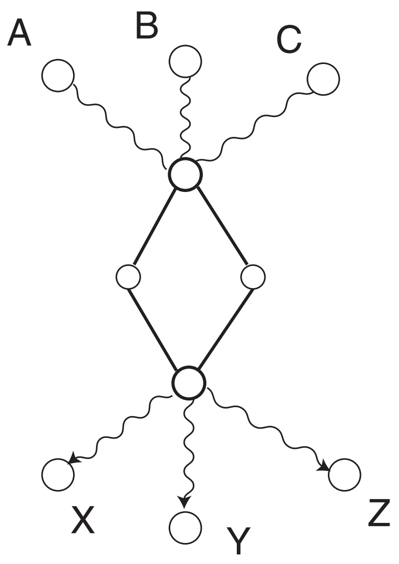

Fig. 4. An example of routing non-determinacy. The true demand, reference flow, and measured traffic for this three-commodity network are listed at Table V. The four links in the center of the network are the only potential sources of congestion. An optimal routing algorithm may assign each flow entirely to the left path, entirely to the right, or split between left and right, as long as the equality of traffic on the bottleneck is preserved.

TABLE V

Assumptions for a network illustrating degeneracy. See Figure 4.

Commodity	True	Reference	Measured
$A \to B$	50	50	50
$B \to Y$	50	50	50
$C \to Z$	50	50	50

this fundamental assumption is removed. Of course, this is an extreme case; most networks are provisioned with sufficient capacity that links are not persistently and unavoidably over-committed for extensive periods of time.

Suppose now that the optimal network G_{opt} can route the true demands d_{ij}, but not within $\epsilon > 0$ for any ϵ. Under our weak Assumptions 1-3, allowing arbitrary congestion control response to bottleneck links, one can construct straightforward, albeit contrived, examples for which min-max stabilizes on a suboptimal routing. Consider the network illustrated in Figure 3 and Table IV. In this case the two listed demands $A \to X$ and $B \to Y$ cannot be routed to their destinations without causing congestion on links (1) and (2) (denoted by heavy lines). If the listed routing is the initial routing, and if congestion control will operate on bottleneck links always to give preference to $A \to X$ and $B \to Y$ over other demands, then any "true" traffic originating at C or C' will never emerge from the network. On the other hand, if the routing originating at A and B is reversed, then the additional "true" traffic can be routed.

Admittedly, this example applies a particular congestion control mechanism to bandwidth allocation on bottleneck links (essentially, giving complete priority to A and B). It is not difficult, however, to construct corresponding "diabolical" examples for other methods of bandwidth allocation. It is an open question whether some suitably refined min-max algorithm could be devised that would provide more flexible routing and convergence in the case of heavy congestion.

For example, one could modify min-max routing to spread traffic among links once extreme congestion is encountered. In similar circumstances, load-balancing methods that provide automatic splitting of traffic have been recommended [7], [22]. Specifically, one could modify the min-max algorithm to provide that where the calculation of the reference flow results in fully capacitated paths (thus indicating that optimal routing will also result in fully capacitated paths), traffic should be applied to uncapacitated paths in preference to shorter, fully capacitated paths. It is an open question whether desired convergence and fairness properties can be shown to hold under these conditions.

IV. Route Determinism

In order to address the issue of "route oscillation" in min-max adaptive routing systems it is necessary first to identify the circumstances under which a min-max routing is uniquely determined. Consider the example in Figure 4. In this example, there are an infinite number of equally optimal assignments of traffic to routes; however, linear solvers are not in general guaranteed to reach the same solution given nearly identical inputs, opening the potential for meaningless route oscillation. The example is equally valid if the amounts of flow to be routed are slightly different, or if the bottleneck is in some other part of the network, or even if a secondary optimization step is used to select for shortest paths [19]. The solution, however, is straightforward: introduce systematic tie-breaking considerations to the algorithm. For example, hysteresis could be added to the optimization criteria, to favor (with some very small positive weight compared to the min-max value) solutions that preserve the same routing as had been chosen in the previous step. Alternatively, the weights of links on the edge of the network in Figure 4 could be systematically and randomly perturbed by small amounts. We leave to future consideration the development and analysis of detailed methods aimed at reducing route oscillation in the presence of stochastic variations in traffic load.

V. Conclusion

Questions surrounding the stability of adaptive routing systems have long stymied efforts to introduce adaptive routing to the Internet. In this paper, we have addressed the sub-problem of whether adaptive routing can be designed to be stable in the presence of congestion control. We answer this question in the affirmative, showing that rapid convergence, stability, and optimality can be achieved for general network topologies and workloads, if the workload can be feasibly routed with some headroom, the adaptive routing algorithm minimizes the maximum link utilization, and the congestion control algorithm allocates bandwidth fairly among competing users. We further show by counterexample that some adaptive routing algorithms that preserve the min-max property do not converge to an optimal solution when combined with less well-behaved congestion control algorithms. The conclusions of this paper, when combined with recent advances in traffic measurement technology, hopefully will serve to reinvigorate the search for effective solutions to Internet bottlenecks using adaptive routing.

Acknowledgment

The authors would like to thank Anna Karlin and Steve Gribble for their many helpful suggestions on the work presented in this paper. This work was supported in part by DARPA contract Number F30602-00-2-0565.

References

[1] C.-N. Chuah and C. Diot, “A Tier-1 ISP perspective: Design principles & observations of routing behavior,” in *IPAM Workshop on Large-Scale Communication Networks*, 2002.

[2] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1992.

[3] L. Fleischer, “Approximating fractional multicommodity flow independent of the number of commodities,” in *IEEE Symposium on Foundations of Computer Science*, 1999, pp. 24–31.

[4] N. Garg and J. Konemann, “Faster and simpler algorithms for multicommodity flow and other fractional packing problems,” in *IEEE Symposium on Foundations of Computer Science*, 1998, pp. 300–309.

[5] A. Khanna and J. Zinky, “The revised ARPANET routing metric,” in *ACM SIGCOMM*, 1989, pp. 45–56.

[6] A. Shaikh, J. Rexford, and K. Shin, “Load-sensitive routing of long-lived IP flows,” in *ACM SIGCOMM*, 1999, pp. 215–226.

[7] S. Vutukury and J. J. Garcia-Luna-Aceves, “A simple approximation to minimum-delay routing,” in *ACM SIGCOMM*, 1999, pp. 227–238.

[8] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” in *IEEE INFOCOM*, 2000, pp. 519–528.

[9] Z. Wang and J. Crowcroft, “Analysis of shortest-path routing algorithms in a dynamic network environment,” *ACM Computer Communication Review*, vol. 22, 2, pp. 63–71, 1992.

[10] T. Ye, H. Kaur, S. Kalyanaraman, K. Vastola, and S. Yadav, “Optimization of OSPF weights using online simulation,” in *Tenth Annual International Workshop on Quality of Service (IWQoS 2002)*, May 2002.

[11] U. Manber, Personal communication, 2002.

[12] R. Gallager, “A minimum delay routing algorithm using distributed computation,” *IEEE Transactions on Communications*, vol. COM-25, pp. 73–85, 1977.

[13] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, N.J.: Prentice Hall, 1993.

[14] D. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, Mass.: Athena Scientific, 1998.

[15] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, “Deriving traffic demands for operational (IP) networks: methodology and experience,” in *ACM SIGCOMM*, 2000, pp. 257–270.

[16] Cisco Systems, “Cisco IOS netflow.” [Online]. Available: http://www.cisco.com/warp/public/732/Tech/netflow/

[17] Juniper Networks, “Internet Processor II ASIC: Visibility into network operations.” [Online]. Available: http://www.juniper.net/techcenter/app_note/350003.html

[18] InMon Corp., “InMon sFlow Probe: Media-speed gigabit route-aware monitoring.” [Online]. Available: http://www.inmon.com/probes.htm

[19] E. Anderson, “A multicommodity flow based approach to adaptive internet routing,” Ph.D. dissertation, University of Washington, 2002.

[20] M. Kodialam and T. Lakshman, “Minimum interference routing with applications to MPLS traffic engineering,” in *IEEE INFOCOM*, 2000, pp. 884–893.

[21] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS adaptive traffic engineering,” in *IEEE INFOCOM*, 2001, pp. 1300–1309.

[22] C. Villamizar, “OSPF optimized multipath (OSPF-OMP),” 1999.

[23] S. B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. Roberts, “Statistical bandwidth sharing: A study of congestion at flow level,” in *ACM SIGCOMM*, 2001, pp. 111–122.

[24] A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” *ACM Computer Communication Review*, vol. 19, no. 4, 1989.