Design Principles for Manageable Networks

Jiayue He
Dept. of EE, Princeton University
Email: jhe@princeton.edu

Jennifer Rexford
Dept. of CS, Princeton University
Email: jrex@cs.princeton.edu

Mung Chiang
Dept. of EE, Princeton University
Email: chiangm@princeton.edu

***Abstract*— As networks grow in size and complexity, network management has become an increasingly challenging task. We argue that "bolting on" network-management functions to the existing architecture is the wrong approach. Instead, networks should be designed with management in mind from the beginning. We focus on two important aspects of network management: tomography and optimization. We present principles with illustrative examples of recent research that pinpoint why network management is difficult and show how changes to existing protocols and architectures can lead to manageable networks. We also discuss the trade-offs between making network management easier and the overhead these changes impose.**

I. Introduction

Network management is the continuous process of monitoring a network to detect and diagnose problems, and of configuring protocols and mechanisms to fix problems and optimize performance, as in Figure 1. Unfortunately, today's network architectures were not designed with these tasks as a main priority. As a result, managing data networks is, at best, a black art practiced by an increasingly overwhelmed community of network operators. We observe that the design of protocols, control mechanisms, and monitoring systems *induces* the problems that network operators must solve. Rather than just retrofitting network management on the existing infrastructure, we advocate designing network architectures with management in mind in the first place. The key idea is to design protocols and architectures to induce network-management problems that are easy to solve.

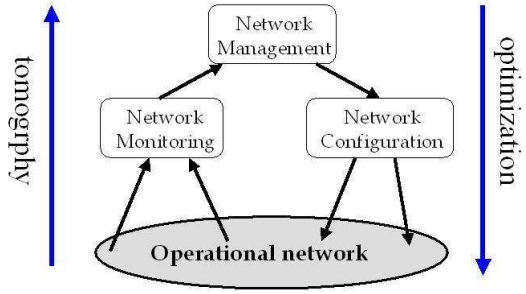

Fig. 1. An overview of network management system.

Traditionally, network management has been largely impenetrable to the research community since many of the problems appear both complex and ill-defined. During the past few years, the research community has made tremendous progress casting many important management tasks as tomography and optimization problems:

- *Network tomography* involves inferring important properties of the network from the available measurement data, which become the inputs to network management. For example, operators typically infer the traffic matrix—the offered load between the ingress and egress points in the network—from link load statistics.
- *Network optimization* involves configuring tunable parameters to optimize an objective function under resource constraints, which forms the outputs from network management. For example, operators typically set the link weights for shortest-path routing protocols like OSPF or IS-IS to minimize network congestion.

To make general ideas we advocate more concrete, we will focus on tomography and optimization problems in this paper. Recent research on these subjects has lead to useful tools for network operators. However, today's monitoring systems and network protocols can still often lead to network tomography and optimization problems that are exceptionally difficult to solve. For example, inferring the traffic matrix is highly underconstrained [1], [2]. Similarly, optimizing the link weights is NP-hard and the best heuristics still could deviate from the optimal solution significantly [3]. Many tasks in network management remain an art.

In this paper, we argue that the difficulty of solving the key tomography and optimization problems is an indication that we need to revise the underlying protocols, or even the architectures, that lead to these problem formulations in the first place. We advocate the design of *manageable networks*—network architectures that lead to easy-to-solve tomography and optimization problems. Indeed, the key difference between "network management" and "manageable networks" is that the former refers to solving a given problem (induced by the existing protocols and architectures) while the latter involves formulating the "right" problem (by changing protocols or architectures accordingly) that would be easy to solve later.

The changes to protocols and architectures can range from minor extensions to clean-slate designs. In general, the more freedom we have to make changes, the easier it would be to create a more manageable network. On the other hand, the resulting improvements in network management also need to balanced against other considerations such as *scalability* and *extensibility*, and must be made *judiciously*. There can be a trade-off between making network-management problems easier by changing the statement of the problem and the implementation costs which result. To make design judgements, it is essential to quantify these trade-offs.

In the next two sections, we introduce *design principles* for manageable networks through illustrative examples in tomography and optimization, respectively. Most of the examples focus on incremental changes to the existing Internet architecture, with one example of clean-slate design. Our examples focus primarily on intradomain routing, where tomography and optimization problems are relatively easy to formulate and promising advances have been made in recent years. These examples are included here mainly to serve as illustrations of general principles, and certainly do not constitute an exhaustive list. In the Section IV, we discuss other aspects of networking, such as interdomain routing and active queue management, where the problems are even more challenging. We conclude in Section V with further discussions on future research opportunities.

II. Designing for Tomography

Tomography provides *inputs* to network management by solving inverse problems where network properties that cannot be observed directly are inferred from other measurement data. Most of the successful approaches thus far use a linear model $y = Ax$, where y is a vector of measurements, A is a routing matrix, and x is a vector of network properties; the goal is to infer either A or x from y. Often, these inverse problems are underconstrained, where the number of unknown variables is larger than the number of constraints imposed by the available measurements. In contrast to the process of solving a given tomography problem, *designing for tomography* explores ways to change the system to make inverse problems easy to solve. This can be achieved by reducing the sources of uncertainty in the measurement data or increasing the number of constraints. The first principle is introduced through an example of inferring link properties from path measurements, and the second principle through traffic-matrix estimation.

A. Reducing the Uncertainty Across Observations

Measurement data collected at different times do not observe the network under the same conditions, leading to uncertainty about the network state. This problem is exemplified by the difficulties of inferring link-level properties from active probes of end-to-end paths. The goal of inferring the link-level properties is to identify the link (or the set of links) responsible for dropping or delaying packets due to congestion, bit errors, failures, misconfiguration, or the presence of an adversary. Determining the faulty link in a system is necessary step towards diagnosing and fixing the problem.

Inline with the syntax for general tomography problems, y_p is the sum of link properties (*e.g.* delay) along the path used by probe p, $A_{p,\ell}$ is 1 if the probe p traverses link ℓ and 0 otherwise, and x_ℓ represents the performance properties of link l. Since active probes are sent at different times, the tomography problem must include a time-dependent error term ϵ, leading to the equation $y = Ax + \epsilon$.

One potential solution to tomography is to use maximum likelihood estimation. The following example highlights potential challenges with this approach. Consider the topology in Figure 2, where we adopt the labelling convention that link i connects node i to its parent node. Using normal unicast probes means each probe packet would be sent from node 0 to nodes 4, 5, 6, and 7 at separate times. Comparing the four end-to-end measurements can shed light on the link responsible for performance problems. For example, if probe packets do not reach nodes 4 and 5, while they do reach nodes 6 and 7, then link 2 is the likely culprit. However, since the probes are sent independently and link properties vary with time, a probe packet to node 4 could be dropped while a probe packet to node 5 is received even if link 2 is sometimes faulty. Maximum likelihood estimation can narrow down the possible failure locations based on repeated measurements, but relies on knowing the stochastic model for ϵ.

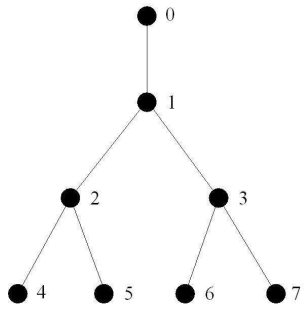

Fig. 2. A multicast-tree with 4 receivers.

Changing the underlying system to reduce the uncertainty can significantly simplify the tomography problem. For example, suppose the network supports multicast delivery of the probe packets [4], [5]. Then, node 0 would multicast a single probe packet to all four end nodes simultaneously. In this case, if link 2 drops the probe, then node 4 and node 5 would *both* lose the packet. Sending a multicast probe provides an extra degree of coupling between the observations by the four receivers. Although not widely used in today's Internet for a variety of technical and business reasons, IP multicast is supported by the routers. Enabling the use of multicast within a single domain, simply for the purpose of sending probes, would be easy. In addition to reducing the uncertainty introduced by measurement process, multicast probes consume less bandwidth since they traverse each link only once and therefore also reduces the measurement overhead. With or without multicast, the key idea is that *reducing the uncertainty across observations* leads to easier tomography problems.

B. Increasing the Number of Deterministic Constraints

Network tomography problems typically do not have enough constraints, leading to many feasible solutions. A classic example is the problem of inferring the traffic matrix from link-load statistics. Each element in the traffic matrix is the offered load from an ingress router i to an egress router j, as shown in Figure 3. The traffic matrix is an important input to several network-management tasks, such as anomaly detection and traffic engineering. For example, changes in traffic load

can alert the system to the potential existence of a flash crowd, denial-of-service attack, network failure, or a change in user behavior. The traffic matrix is also useful for predicting the effects of configuration changes, such as the way link loads would change after adapting the intradomain link weights.

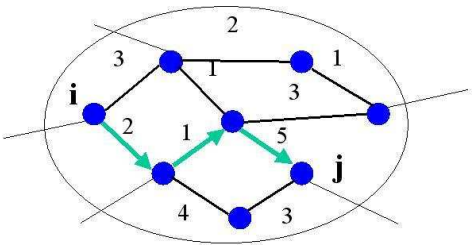

Fig. 3. Network topology with link weights for shortest path routing.

Direct observation of the traffic matrix is difficult because it requires collecting fine-grained measurements at every edge router. Instead, network operators typically infer the traffic matrix from coarse-grained link-load statistics and knowledge of the current routing configuration. The unknown traffic-matrix elements are represented by x_p, where p is an ingress-egress pair, such as (i, j) for edge nodes i and j. The current routes are represented by $A_{\ell,p}$—the fraction of traffic for ingress-egress pair p that traverses link ℓ; for example, $A_{\ell,p}$ is 1 if the path traverses ℓ and 0 otherwise[1]. The relationship between x_p and $A_{\ell,p}$ can be written in matrix form as $y = Ax$, where y_ℓ is the load imposed on link ℓ. However, inverting this relationship to compute x from y and A is difficult, because the number of links is typically much smaller than the number of ingress-egress pairs—that is, many different traffic matrices would be consistent with the observed link loads.

Estimating the traffic matrix requires a way to compensate for the limited number of constraints. Early work considered collecting the link-load statistics at several points in time and binding these observations together with statistical models; however, the statistical assumptions did not apply well to Internet traffic, leading to large errors in the resulting inferences [1]. More recent approaches draw on the notion of gravity models that make assumptions about the spatial distribution of traffic, with greater accuracy [1], [2]. Still, the errors can be relatively large, *e.g.*, 20%. Stepping back, there are several ways to change the system in the first place to increase the number of constraints:

- *Creating multiple instances of A:* Modifying the link weights in the operational network provides a way to observe the link loads under different routing matrices [6].
- *Populating some values of x:* Extending selected routers to collect fine-grained traffic statistics would enable direct observation of x_p for certain node-pairs p [7].
- *Collecting the local traffic matrix at each router:* Extending each router to collect a "local traffic matrix"—a count of the traffic between each input and output port—would impose more constraints on the global traffic matrix [8].

[1]More generally, $A_{\ell,p} \in [0, 1]$ if the network supports multi-path routing.

The first solution can be applied today, at the expense of introducing transient disruptions in the running network. The second solution is also feasible today, if some routers have been upgraded to support flow-level monitoring (e.g., Cisco's Netflow feature). The third approach introduces less measurement overhead but would require modest changes to the routers. All three are effectively *adding more constraints* (i.e., the vector x is constrained to lie within a smaller space) to make a tomography problem easier to solve.

III. Design for Optimization

Optimization produces *outputs* from network management by setting the tunable parameters that control network behavior. Solving an optimization problem involves minimizing an objective function subject to a set of constraints. Many optimization problems that arise in data networks are computationally intractable [2] or even have many local minima that are significantly suboptimal.

Occasionally, through mathematical techniques, an alternative representation of such difficult optimization problems may lead to effective solutions, without changing the formulation of the problem. In contrast, we advocate the engineering approach of *designing for optimization*, which focuses on formulating problems to be easier to solve, and changing the protocols accordingly. This is particularly important when the existing problem formulations are not even amenable to alternative representations, efficient approximations, or exhaustive search. Of course, the merits of these modified protocols should also be balanced with any extra overhead in practical implementation and robustness to changing network dynamics.

Changes in the protocol can lead to changes in the structure of the constraints or in the degrees of freedom. We introduce the first principle through the problem of optimizing the link weights in intradomain routing, and the second principle through the problem of selecting the egress points that direct traffic to neighboring domains. Our third and final example illustrates how embedding management objectives in the protocol can lead to new distributed protocols that implicitly solve well-formulated optimization problems.

A. Changing the Constraint Set

Often optimization problems involve integer constraints, which are not convex. For example, operators today set the link weights in intradomain routing protocols such as OSPF and IS-IS using traffic matrices as inputs (see II-B). Figure 3 shows an example network topology with an integer weight on each link. Each router learns the weighted graph and runs Dijkstra's algorithm to compute a forwarding table that directs each packet to the next hop along a shortest path. The network operator can only *indirectly* influence how the routers forward traffic, through the setting of the link weights. The objective of the optimization is to minimize the sum of link-cost functions f (often an increasing and convex function of link utilization) over all links.

[2]Sometimes because the constraint set is not a convex set or the objective to be minimized is not a convex function.

Optimizing the link weights in shortest-path routing protocols based on the traffic matrix is NP-hard, even for convex objective functions [3]. In practice, local-search techniques are often quite effective for selecting link weights [3]; however, the deviation from the optimal solution can still be large, especially under link failure scenarios and multiple traffic matrices. The crux of the problem is that these protocols direct traffic only along the shortest path, or split traffic evenly amongst multiple shortest paths. The ability to split traffic arbitrarily over multiple paths would make the constraints convex, as in MPLS. However, the downside is this approach would sacrifice the simplicity of OSPF and IS-IS, where routers compute paths in a distributed fashion based on a small amount of configuration state.

Rather than adding new technology that supports arbitrary splitting, two recent proposals advocate small extensions to OSPF and IS-IS to split traffic over multiple paths [9], [10]. Under these proposals, the routers forward traffic on multiple paths, with diminishing proportions of the traffic directed to the longer paths. Although the resulting optimization is still computationally difficult for general topologies and traffic matrices, the resulting constraints are much easier to be *approximated* by convex constraints, leading to faster solutions. In addition, the modified protocols are better than conventional OSPF and IS-IS at making efficient use of link capacities, and are more robust to small changes in the path costs. By *changing the constraint set*, [9] and [10] retain the simplicity of link-state routing protocols, while inducing optimization problems that are both easier to solve (faster running time than OSPF) and leading to more efficient solutions (substantially reduced gap to optimum).

B. Increasing the Degrees of Freedom

Today's network protocols often impose many, tightly-coupled constraints on the settings of the tunable parameters, inducing intractable optimization problems. A good example is the challenge of controlling how traffic leaves one domain en route to the ultimate destination. When a network, such as an Internet Service Provider (ISP) backbone, can reach a destination through multiple egress points, each router typically selects the closest egress point, in terms of the intradomain link weights, in a practice known as early-exit or hot-potato routing [11]. In the example in Figure 4, traffic from Dallas exits via New York City rather than San Francisco since the intradomain path cost from Dallas to New York City is smaller. Controlling where packets leave the network, and preventing large shifts from one egress point to another, is an important part of engineering the flow of traffic in the network.

Since large ISPs peer typically in multiple locations, the intradomain link weights play a major role in how traffic leaves an ISP via its peering links to other providers. For example, if the traffic from Dallas encounters congestion along the downstream path from New York City, the network operators could tune the link weights to make the path via San Francisco more attractive. However, setting the link weights is highly constrained, since the weights are used to compute both the forwarding paths between the routers inside the domain and the egress points where the traffic leaves the domain. In addition, small changes in the path costs, due to weight changes or link failures, can lead to abrupt, unintentional shifts in traffic from one egress point to another. Models can capture the effects of changing the link weights on the intradomain paths and the egress points, but identifying good settings of the weights is very difficult; in some cases, no setting of the weights would satisfy all of the constraints.

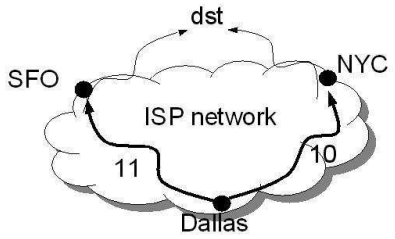

Fig. 4. Traffic from Dallas exits via New York City (with a path cost of 10) rather than San Francisco (with a path cost of 11), due to hot-potato routing

Weakening the coupling between intradomain routing and egress-point selection is the key to simplifying the optimization problem and improving network performance. Rather than selecting egress points based only on the intradomain path costs, the routers can consider, *e.g.*, a weighted sum of the path costs and a constant term [12]. Providing separate parameters for each destination prefix allows even greater flexibility, such as allowing delay-sensitive traffic to use the closest egress point while preventing unintentional shifts in the egress points for other traffic. By *increasing the degrees of freedom*, a management system can apply standard integer programming techniques to set the new parameters under a variety of constraints that reflect the operators' goals for the network [12]. Not only does the network become easier to optimize, but the performance improves as well, due to the extra flexibility in controlling where the traffic flows.

C. Embedding Management Objectives in the Protocol

Because today's protocols were not designed with management in mind, network operators have at best indirect control over system behavior by tuning the configurable parameters. With a clean-slate redesign, the protocols could allow the network-management system to specify its objectives directly. In this subsection, we explore this idea in the context of traffic engineering, with an emphasis on the interaction between congestion control and routing. In the Internet today, traffic engineering is performed assuming that the offered traffic is inelastic. In reality, end hosts adapt their sending rates to network congestion, and network operators adapt the routing based on measurements of the traffic matrix. Although congestion control and routing operate independently, their decisions are coupled. The joint system is sometimes stable, but often suboptimal [13]. In addition, traffic engineering does not necessarily adapt on a small enough timescale to respond to shifts in user demands.

Congestion control tries to maximize aggregate user utility, and as a result tends to push traffic into the network so that multiple links are used at capacity. In contrast, traffic engineering tries to prevent links from being fully utilized and to tolerate fluctuations in network conditions. One way to balance the conflicting objectives of traffic engineering and congestion control is to take a "top-down" approach by formulating an objective function that is a weighted sum of end user utility functions and network link cost functions. The capacity constraints are on the routing matrix and source rate vectors, both of which are design variables [13]. This leads to a convex optimization problem that is readily solvable through distributed algorithms.

In [13], we construct a decomposition that revisits the boundary between the network elements and the management system, while assuming end hosts still run TCP. Decomposing the optimization problem with this goal in mind, a distributed protocol is derived that splits traffic over multiple intradomain paths, where the splitting proportions depend on feedback from the links. The solution is called DATE (Distributed Adaptive Traffic Engineering). It may appear to be similar to [14], but the construction of the mechanism is "top-down" (starting from an optimization problem formulation), and considers the effects of *jointly* optimizing congestion control at the end hosts and routing over the links. The objectives of the network operator are represented by the same link-cost function f as the previous work on traffic engineering, discussed earlier in Section III-A. By *embedding the management objectives in the protocols*, the link-cost function is now autonomously applied by the links themselves as part of computing the feedback sent to the edge routers, rather than by the network-management system. As such, the network-management system merely specifies f, and does not even need to adapt the configuration of the routers over time.

IV. Open Challenges

Our examples in Sections II and III focus on tomography and optimization problems in intradomain routing. Routing within a single domain side-steps several important issues that arise in other aspects of data networking, for several reasons:

- A single domain has the authority to collect measurement data (such as the traffic and performance statistics) and tune the protocol configuration (such as the link weights).
- The routing configuration changes on the timescale of hours or days, allowing ample time to apply more computationally intensive solution techniques.
- The tomography and optimization problems consider highly aggregated information, such as link-level performance statistics or offered load between pairs of routers.

When these assumptions do not hold, the resulting tomography and optimization problems become even more complicated, as illustrated by the following two examples.

Tomography in interdomain routing: An Autonomous System (AS) may experience equipment failures or configuration mistakes that lead to routing changes in other parts of the Internet. Identifying the responsible AS is useful to aid network operators in avoiding paths that traverse this AS in the future, or in generating reports on the robustness of different ISPs. The routing-protocol messages exchanged in the Border Gateway Protocol (BGP) provide an indirect view into the root cause of a routing change. Analysis of these messages from many vantage points can be used to infer the AS responsible for the routing change. However, this tomography problem does not map into a relatively simple, linear "$y = Ax$" formulation. Several recent studies have approached this problem [15], [16], with at best mixed success, in part because BGP-speaking routers can apply complex routing policies for selecting routes.

Optimization in active queue management: A router may apply active queue management schemes like Random Early Detection [17] to provide TCP senders with early feedback about impending congestion. RED has many configurable parameters to be selected by network operators, *e.g.*, queue-length thresholds and maximum drop probability. Unfortunately, predictive models for how the tunable parameters affect RED's behavior remain elusive. In addition, the appropriate parameter values may depend on a number of factors, including the number of active data transfers and the distribution of round-trip times, which are difficult to measure on high-speed links. Recent analytic work demonstrates that setting RED parameters to stabilize TCP is fundamentally difficult [18].

From these two examples, it is clear that there are many open challenges to network management. For example, augmenting the BGP update messages to provide more information about the reasons for routing changes may help make the tomography problem easier to solve, but identifying the right protocol changes and the incentives for ASes to deploy them remains an open question. Similarly, it is appealing to explore alternative active-queue management schemes that are easier to optimize, including self-tuning algorithms that do not require the network-management system to adjust any parameters.

The challenges are not just limited to protocols. Architecturally, the DATE example represents one extreme where most of computation and coordination is moved into the distributed protocols that run in the routers. One can consider another extreme, where the network-management systems bear all the responsibility for adapting to changes in network conditions. Both approaches redefine the boundaries between the components in Figure 1, where one moves most of the control into the distributed protocols and the other has the management systems directly specify how the routers handle packets. Determining the appropriate division of labor between the network elements and the management systems is another avenue for challenging research problems.

V. Discussions

The examples in Sections II and III demonstrate that changes in the design of monitoring systems and network protocols can make tomography and optimization problems much easier to solve. However, these examples also illustrate the limitations in our ability to design with network management in mind, including the following two:

- *Problems still remain difficult to solve:* Some of the approaches may *not go far enough* in converting an intractable problem into a tractable one. For example, the extensions to link-state routing protocols in Section III-A still induce computationally difficult optimization problems, though they do admit simpler relaxation techniques. Similarly, capitalizing on multicast probes in Section II-A still results in under-constrained problem, though the accuracy of the solutions is improved.
- *Changes to the system impose extra overhead:* Some of the approaches make the network more manageable at the expense of additional overhead, and may have *gone too far*. For example, adding flexibility in egress-point selection in Section III-B introduces more parameters that the network-management system must set. Similarly, revisiting the division of functionalities in Section III-C leads to a solution that requires feedback from the links and shaping of the TCP flows.

Moreover, the above two challenges are clearly "at odds" with each other—making network-management problems easier to solve often comes at some cost in network overhead. Characterizing a network architecture in terms of the tractability of network-management problems is just one piece of a complex design puzzle. The design of manageable networks introduces tension between the ease of network management (as measured by the tractability of the tomography and optimization problems) and the overhead of monitoring and protocol functions. Providing a deeper understanding of these trade-offs remains an important avenue for future research.

Furthermore, ensuring a completely tractable tomography or optimization problem is sometimes unnecessary. For example, having a rough estimate of the traffic matrix may be sufficient for important network-management tasks such as traffic engineering, especially if the routing configuration must be robust to the natural statistical fluctuations in the traffic matrix anyway. Similarly, an NP-hard problem may be acceptable, as long as reasonably accurate and readily-solvable approximations are available. Finding effective ways to quantify the acceptable error for tomography, and the acceptable amount of deviation from the optimal solution, is important for striking the right trade-offs in the design of manageable networks. There are also well-established, quantitative measures of the notions of how under-constrained a tomography problem is and how easily-solvable an optimization is. These quantitative measures can help determine *how much* the protocols and architectures need to change to better support network management. As such, we believe that the design of manageable networks can be a promising, new interdisciplinary area between the systems and theory communities.

Networks need to be managed, thus should be designed *for* the ease of management in the first place. Ultimately, the design of manageable networks raises important architectural questions about the appropriate division of functionalities between network elements and the systems that manage them. This paper represents a first step toward identifying design principles that can guide these architectural decisions. The open challenges which remain suggest that the design of manageable networks may continue to be somewhat of an art, but hopefully one that will be guided by more and more design principles. We believe that providing a new, comprehensive foundation for the design of manageable networks is an exciting avenue for future research.

ACKNOWLEDGMENT

We would like to thank Constantine Dovrolis, Nick Feamster, and Renata Teixeira for their feedback on earlier drafts. This work has been supported in part by NSF grants CNS-0519880 and CCF-0448012, and a Cisco grant GH072605.

REFERENCES

[1] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques compared and new directions," in *Proc. ACM SIGCOMM*, August 2002.

[2] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast, accurate computation of large-scale IP traffic matrices from link loads," in *Proc. ACM SIGMETRICS*, June 2003.

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

[4] F. LoPresti, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal delay distributions," *IEEE/ACM Trans. Networking*, vol. 10, pp. 761–775, December 2002.

[5] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Information Theory*, vol. 45, pp. 2462–2480, November 1999.

[6] A. Nucci, R. Cruz, N. Taft, and C. Diot, "Design of IGP link weights for estimation of traffic matrices," in *Proc. IEEE INFOCOM*, March 2004.

[7] G. Liang, N. Taft, and B. Yu, "A fast lightweight approach to origin-destination IP traffic estimation using partial measurements," *In special joint issue of IEEE/ACM Transactions on Networking and IEEE Transactions on Information Theory*, June 2006.

[8] G. Varghese and C. Estan, "The measurement manifesto," in *Proc. SIGCOMM Workshop on Hot Topics in Networking*, November 2003.

[9] J. H. Fong, A. C. Gilbert, S. Kannan, and M. J. Strauss, "Better alternatives to OSPF routing," *Algorithmica*, vol. 43, no. 1-2, pp. 113–131, 2005.

[10] D. Xu, M. Chiang, and J. Rexford, "DEFT: Distributed exponentially-weighted flow splitting." Submitted to IEEE INFOCOM, August 2006.

[11] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proc. ACM SIGMETRICS*, June 2004.

[12] R. Teixeira, T. Griffin, M. Resende, and J. Rexford, "TIE breaking: Tunable interdomain egress selection," in *Proc. CoNEXT*, October 2005.

[13] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," April 2006. Submitted. `www.princeton.edu/~jhe/research/jsac2.pdf`.

[14] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proc. ACM SIGCOMM*, August 2005.

[15] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet routing instabilities," in *Proc. ACM SIGCOMM*, September 2004.

[16] M. Caesar, L. Subramanian, and R. H. Katz, "Towards localizing root causes of BGP dynamics," Tech. Rep. CSD-03-1292, UC Berkeley, November 2003.

[17] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, August 1993.

[18] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Computer Networks*, vol. 43, pp. 633–647, December 2003.