NetScope: Traffic Engineering for IP Networks

Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, and Jennifer Rexford
Networking and Distributed Systems
AT&T Labs – Research
{anja,albert,lund,reingold,jrex}@research.att.com

Abstract

Managing large IP networks requires an understanding of the current traffic flows, routing policies, and network configuration. Yet, the state-of-the-art for managing IP networks involves manual configuration of each IP router, and traffic engineering based on limited measurements. The networking industry is sorely lacking in software systems that a large Internet Service Provider (ISP) can use to support traffic measurement and network modeling, the underpinnings of effective traffic engineering. This paper describes the AT&T Labs *NetScope*, a unified set of software tools for managing the performance of IP backbone networks. The key idea behind NetScope is to generate global views of the network, on the basis of configuration and usage data associated with the individual network elements. Having created an appropriate global view, we are able to infer and visualize the network-wide implications of local changes in traffic, configuration, and control. Using NetScope, a network provider can experiment with changes in network configuration in a simulated environment, rather than the operational network. In addition, the tool provides a sound framework for additional modules for network optimization and performance debugging. We demonstrate the capabilities of the tool through an example traffic-engineering exercise of locating a heavily-loaded link, identifying which traffic demands flow on the link, and changing the configuration of intra-domain routing to reduce the congestion.

1 Introduction

Traffic engineering aims to optimize network performance through three integrated activities – measuring traffic, modeling the network, and selecting mechanisms for controlling the traffic. Unfortunately, large Internet Service Providers (ISPs) have few software systems and tools to support traffic measurement and network modeling, the underpinnings of effective traffic engineering. Seemingly simple questions about topology, traffic, and routing are surprisingly hard to answer in today's IP networks. A tremendous amount of work has gone into developing mechanisms and protocols for controlling traffic. Indeed, most of the work in the Internet Engineering Task Force (IETF) concerns the aspect of traffic engineering concerned with traffic control. By comparison, little work has gone to support traffic measurement and network modeling in operational networks. (Notable exceptions in the IETF include the IPPM and RTFM working groups.) Unless control mechanisms are driven by the appropriate measurements and understanding from well-tested models, the benefit of the controls will be limited.

This paper describes the AT&T Labs *NetScope*, a unified set of software tools for traffic engineering in operational IP backbone networks. The key idea behind NetScope is to generate global views of the network, on the basis of configuration and usage data associated with the individual network elements. Having created an appropriate global view, we are able to infer and visualize the network-wide implications of local changes in traffic, configuration, and control. NetScope provides an extensible and powerful platform for "what-if" traffic-engineering investigations. Using NetScope, a network provider can experiment with changes in network configuration in a simulated environment, rather than in the operational network. In addition, the tool provides a sound framework for additional modules for network optimization and performance debugging.

In order to illustrate some of the capabilities of NetScope, we focus on one compelling application: the problem of configuring intra-domain routing based on the underlying network topology and traffic demands. This application drives the choice of an appropriate topological view, as well as the methods for traffic aggregation and routing. The task can be decomposed into several steps, each drawing heavily on the underlying modules and the flexible visualization environment. By joining topology and usage data, NetScope displays the utilization of each of the links, and identifies the most heavily-loaded link. Then, NetScope's integrated view of traffic, topology, and routing enables us to identify which source-sink pairs are responsible for the congestion. Based on this information, we change the configuration of intra-domain routing (by changing an OSPF link weight) to direct some of these traffic demands away from the congested link. NetScope then recomputes the routes and the resulting load on each link, providing an estimate of how the traffic would flow after the change.

1.1 Network Engineering Constraints

Before plunging into details, let us first consider the factors that are driving the need for better network engineering tools:

- **Service quality:** Increasingly, customers are demanding more stringent assurances with regards to performance, reliability, and security, in the form of Service Level Agreements (SLAs). Customers are developing certification and ongoing testing procedures, to assure compliance to these SLAs. Applications are emerging, such as IP Voice, that by their nature require high quality data transport, as measured by delay, loss, and jitter. Since IP networks do not have explicit support for performance guarantees, network operators must carefully coordinate the operation of the individual network elements to realize customer SLAs.

- **Interdependent tunable parameters:** At present, vendors of network components provide ISPs with little or no low-level control over the basic mechanisms responsible for packet scheduling, buffer management, and path selection. Instead, backbone providers are forced to understand a large set of inter-related parameters, each to some degree affecting configuration and operation. To date, an ISP must manage its backbone network, and complicated peering relationships with neighboring providers, by tuning these knobs through a combination of intuition, testing, and trial-and-error.

- **Network growth:** Individual backbone networks are growing rapidly in size, speed, and scope, and the industry is consolidating many disparate networks into larger integrated networks. As a result, network management functions that could once be handled by a small group of people, based on intuition and experimentation, must now be supported by effective traffic-engineering tools that unite configuration and usage information from a variety of sources.

- **Traffic variability:** Internet traffic is complex. Offered loads between source-destination pairs are typically unknown. The distribution of IP traffic often fluctuates widely across time. This introduces significant complexity to network traffic engineering, without quieting customer demands for predictable communication performance. Detecting and diagnosing shifts in user demands, and the performance implications, requires continual vigilance in network operations. Effective traffic-engineering tools should support rapid identification of potential performance problems and a flexible environment for experimenting with possible solutions.

The complexity of managing an ISP network stems from some of the same fundamental issues responsible for the success and growth of the Internet. IP networks forward datagrams, and lack the basic notion of a call or a virtual-circuit which provides the basic hook for measurements and provisioning in circuit switched, Frame Relay, and ATM networks. As a result, load, reliability, and performance are harder to characterize and track. In addition, IP networks self-configure, via a set of interrelated, dynamic intra-domain and inter-domain protocols, guided by the local configurations of network elements. Thus, databases that are meant to hold authoritative descriptions of the network can diverge from the network reality, and small changes in one part of the network can have a significant impact on the flow of traffic.

1.2 NetScope Tool Overview

Managing the performance of a large ISP backbone network requires advanced tools to support operations, engineering, and design. Section 2 presents background material on IP backbone networks and routing protocols. The key components of the NetScope toolkit are shown in Figure 1:

- **Configuration:** The configuration module extracts a network-wide view that includes the topology, link capacities, customer address, peering connections, route configuration, and layer-two connectivity. This information can be extracted from the configuration files of each router, as well as the IP forwarding tables and inter-domain routing information, in an operational ISP network.

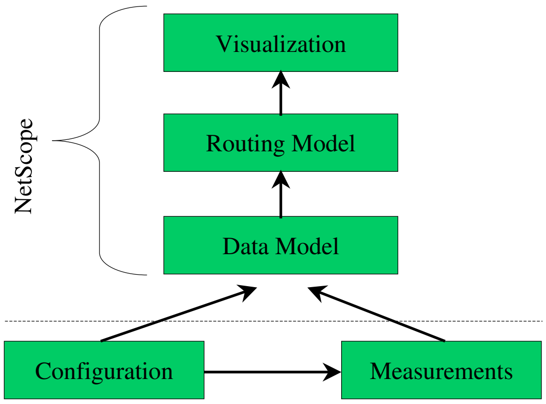

Figure 1: NetScope software architecture

- **Measurement:** The traffic demands are determined based on detailed measurements at the periphery of the network, where traffic enters and leaves the backbone. These measurements can be aggregated to the level that permits efficient and accurate modeling of inter-domain and intra-domain routing. The measurement module associates the traffic demands with the appropriate routers and links, drawing on information from the configuration module.

- **Data model:** A novel feature of NetScope is that it combines diverse network configuration information with diverse network measurements in a joint data model. Attributes of routers, links, and traffic demands are derived from the configuration and measurement modules. To facilitate experimentation with new network designs and projected traffic demands, the tool uses simple flat files as input to the data model.

- **Routing model:** The routing module combines the network topology and traffic demands by modeling how traffic would travel through the network, based on the intra-domain and inter-domain routing protocols. The model captures the selection of shortest paths to multi-homed customers and peers, the splitting of traffic across multiple shortest-path routes, and the multiplexing of layer-three IP links on layer-two trunks.

- **Visualization:** The visualization module displays the network's layer-three and layer-two connectivity, and provides convenient access to selected network configuration information and traffic statistics. The module supports coloring and sizing of routers and links based on a variety of statistics, and computes histograms, tables, and time-series plots. In addition, the environment allows changes to the network configuration to support "what-if" experiments.

The dotted line in Figure 1 is very significant – the network topology and traffic statistics could derive from a wide variety of sources. For example, the topology could be extracted from configuration files and forwarding tables, tracked in real-time by monitoring routing protocol traffic, or projected based on a proposed network design. Similarly, the traffic data could derive from network measurements, customer subscriptions, or projected demands. Section 3 describes the data model and how it can be populated from configuration and measurement information. The routing and visualization modules are described in Section 4 and Section 5, respectively.

NetScope is able to track change while maintaining focus on the network features of interest to the network architects and operators. All aspects of the network infrastructure can and must evolve. New routers, line cards, and versions of the element control software are regularly introduced. Basic architectural changes (such as the introduction of MPLS) occur less frequently. Higher level modules of the software (above the line in Figure 1) can and must evolve to accommodate basic architectural changes. The lower level modules of the software (below the line in Figure 1), responsible for parsing the raw data associated with network elements and building higher level abstractions, are designed for simplicity and extensibility to accommodate frequent, incremental change in the network. Extending these modules to cope with new network features as they become of interest to network architects is a much easier task than building and maintaining a tool that aims to handle all possible combinations of features and policies.

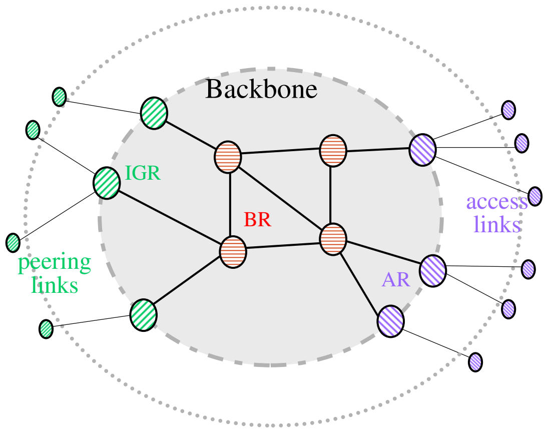

Figure 2: Logical view of an IP backbone network

In designing NetScope, we have introduced particular ways of representing the network topology, intra-domain and inter-domain routing, and the offered traffic. The work described in this paper focuses on *deriving* these models, and defining their basic parameters. The tool does not at present support real-time updates to configuration data. Such changes do not occur very frequently, and hence it is reasonable to expect some amount of stability of the network in between failures and reconfigurations. In fact, NetScope is designed to study precisely these kinds of events by evaluating how they would affect the flow of traffic in an ISP backbone. Unlike the network model, Internet traffic does fluctuate on a variety of time scales, with important implications for traffic engineering. Congestion control mechanisms, such as TCP, introduce variability on the small time scale of seconds [1], while variation in user demands introduces burstiness on multiple time scales [2]. However, on time scales beyond a few hours, fundamental shifts in user demands and changes in the network topology introduce variability beyond statistical fluctuations. Offered load typically changes with the time of the day, or day of the week, or in response to a network failure or reconfiguration. In the traffic engineering example in this paper, we focus on this larger time scale.

2 Internet Backbone Networks

The Internet is divided into a collection of autonomous systems (ASs). Routing through the Internet depends on protocols for routing between ASs, called Exterior Gateway Protocols [3, 4], and protocols for routing within individual ASs, called Interior Gateway Protocols [5]. Each AS is managed by an Internet Service Provider (ISP), who operates a backbone network that connects to customers and other service providers. After describing the architecture of ISP backbone networks, we discuss intra-domain and inter-domain routing.

2.1 Backbone Architecture

ISP backbone networks consist of a collection of IP routers and bi-directional layer-three links, represented as nodes and edges in Figure 2. *Access* links connect directly to customers. For example, an access link could connect to a modem bank for dial-up users, a web-hosting complex, or a particular business or university campus. *Multi-homed* customers have two or more access links for higher capacity, load balancing, or fault tolerance. *Peering* links connect to neighboring service providers. A peering link could connect to a public Internet exchange point, or directly to a private peer or transit provider. An ISP often has multiple peering links to each neighboring provider, typically in different geographic locations. *Backbone* links connect routers inside the ISP backbone. Routers rely on input, via a configuration file, about the state of their hardware, the partitioning of resources (such as buffers), and which policies to apply to the forwarding decisions. A link is configured by entering interface definitions on all routers that are part of the link. The ISP has complete control over the configuration and operation of backbone links, by virtue of controlling the adjacent routers. Configuration of access and peering links depends on interaction with the customers and the peers, respectively.

Each router terminates a mixture of access, peering, and backbone links. To simplify the discussion, we assume that all access links terminate at Access Routers (ARs) and all peering links terminate at Internet Gateway Router (IGRs), and that all remaining routers are Backbone Routers (BRs) that only terminate backbone links. In an operational network, this split in functionality simplifies the requirements for each router. For example, an AR should provide high port density to connect to a large number of customers with various access speeds and technologies. On the other hand, a BR should provide high packet-forwarding performance. Finally, isolating peer traffic to a small set of IGRs simplifies the management of inter-domain routing policies.

ISP backbones run over an underlying facility network. This introduces multiple layers of connectivity and capacity, which has implications for traffic engineering and reliability. The layer-three link between two adjacent IP routers often corresponds to dedicated capacity at the facility level. This abstraction holds, for example, for packet-over-SONET links. However, some networking technologies, such as FDDI and ATM, introduce an intermediate switching fabric at layer two. For example, a single layer-three link may correspond to a permanent virtual circuit (PVC) that traverses one or more ATM links via switches. In fact, multiple layer-three links may share the capacity of a single layer-two link. We refer to these layer-two links as *trunks*, and the layer-two switches as *devices*. The traffic on a trunk consists of the total load imparted by each of the layer-three links that traverses that trunk. Similarly, a trunk (or device) failure causes the failure of each of the layer-three links that travels over that trunk (or device).

2.2 Routing Protocols

Within an AS, routing is determined by interior gateway protocols such as static routing, OSPF, IS-IS, and RIP. Static routes are configured manually in the router. An ISP typically uses static routes to direct traffic toward customers who have a fixed set of IP addresses and do not participate in an inter-domain routing protocol. Routing protocols such as OSPF and IS-IS are more advanced in the sense that routers exchange link-state information and forward packets along shortest paths, based on the sum of the link weights. Typically, customers and peers do not participate directly in these protocols with the ISP. As such, OSPF typically operates only over the backbone links. Since OSPF uses flooding to exchange link-state information, the protocol does not scale to large ISP backbones. Therefore, ISPs typically introduce a routing hierarchy by dividing the backbone into multiple *OSPF areas*, each running a separate instance of the protocol.

Routing between ASs is controlled by an exterior gateway protocol, BGP. BGP is a path-vector protocol that distributes routing information between routers belonging to different autonomous systems. These routers are BGP peers that advertise and withdraw routing information about particular network addresses. A network address represents a set of contiguous IP addresses by a 32-bit number and a prefix (mask) length; for example, the network address 135.207.119.0/24 has a 24-bit mask that specifies a block of 256 IP addresses. A route advertisement includes a list of the ASs in the path to a particular network address, along with other path attributes. For example,

9.2.0.0/16   192.205.31.30   0   1740 1673 1677 1675 i

describes a route to network address 9.2.0.0/16 going through a sequence of autonomous systems, starting with the peer AS 1740. The path has a metric of 0, and 192.205.31.30 is the loopback address of the associated IGR. Upon receiving a BGP advertisement, an ISP can apply local policies to decide whether to use the route. These decisions can be based on a variety of factors, such as the AS path length and local preferences for particular downstream providers. After deciding to use a route, the ISP must also decide whether or not to forward the advertisement to neighboring ASs (after adding itself to the AS path). Limiting the distribution of advertisements allows the ISP to influence what traffic enters the network, and where.

Ultimately, the router determines the path along which to forward an individual packet from the interaction of the various routing protocols. For example, the static routes specified for a customer on a particular router indicate which access link(s) should ultimately carry this traffic. Other routers in the backbone must learn that they should route traffic destined for these customer addresses to the corresponding access link(s). Similarly, the routers need to learn which peering link(s) should be used to reach each network address in the rest of the Internet. Within an ISP backbone, this information is typically distributed via IBGP (internal BGP) or the intra-domain routing protocol. By combining this information with the shortest paths computed by OSPF or IS-IS, each router can determine the appropriate outgoing link(s) for each network address. The mapping from network address to outgoing link(s) is stored in a forwarding table. For example, the entry

135.207.0.0/16   12.126.223.194   Serial2/0/0:26

corresponds to a link using card Serial2/0/0:26, with next hop 12.126.223.194 to forward packets toward the network address 135.207.0.0/16. When a packet arrives, the router performs a longest prefix match on the destination address to find the appropriate forwarding-table entry for the packet. Then, the router forwards the packet to the appropriate outgoing link; in some cases, the forwarding entry has multiple outgoing links and the router can pick a single link from this set. The next-hop router repeats the process, forwarding the packet closer to its destination.

3 Data Model

Traffic engineering requires a network-wide view of the underlying topology and an estimate of the offered traffic load. This section presents a data model that allows us to combine the network topology with traffic statistics in one data structure. We briefly discuss our approach to extracting this information from an operational network.

3.1 Topology Model

Our model of ISP backbones includes objects for routers, layer-three links, devices, and trunks; the latter two were defined in Section 2.1. The routers and layer-three links are connected in a topology, as shown in Figure 2. Layer-three links have several attributes that play an important role in traffic engineering. Each unidirectional link includes general information about the router originating the link, the name of the router card, the IP address of the interface, a textual description of its purpose, its capacity, and its OSPF weight. The latter two are especially important for the routing model. Some attributes are associated with both directions of a link. For example, each bidirectional link can be classified as an access, backbone, or peering link. Backbone links also belong to a particular OSPF area, which must be the same for both unidirectional links. Peering links are associated with a particular BGP peer, identified by its AS number and annotated by the IP address of the BGP peer in the remote domain.

Router attributes include the router name, the loopback IP address of the router, the type of the router (AR, BR, IGR), and the geographic location of the router in terms of city and latitude/longitude. In addition, each router includes information about which links it originates. The device attributes include the name and location of the device, and a list of trunks that originate at the device. Trunks describe the connectivity between routers and devices, and include the information about which links traverse a given trunk. Consequently, links have an additional attribute that lists the trunks they traverse (if any), as well as the routers on either end of the link.

The model is very general and its objects can be populated in a number of different ways, such as modifying an existing data model, constructing an artificial network, or extracting the information from the real network. Our approach is to extract the information from router configuration files. The topology model is populated by netdb, a network configuration debugger and database that also supports interactive queries and consistency checking [6]. Netdb processes the configuration files in two steps. The first step resolves information within a single file such as interfaces, their IP addresses, customer names, link speed, OSPF weight, and OSPF area. The second step unites information from multiple routers into a single topology using IP address information. For example a link is identified via an IP prefix. If it has two or more interface entries, we classify it as a backbone link. A link with only one IP address entry is either an access or peering link. For a peering link, the associated interface participates in a BGP peering session to a known peer. If a link is not a peering link it is an access link.

3.2 Traffic Demands

Effective traffic engineering requires not just a view of the topology but also an accurate estimate of the offered load between various points in the backbone. These estimates could be derived from customer subscriptions, traffic projections, or actual measurements. Ideally, we would like to define a *demand* in terms of the volume of load between two edge links (e.g., from an access link to a peering link). However, many customers connect to the backbone via multiple access links, and many external addresses are reachable via multiple peering links. The traffic destined for a customer could flow through a different access link depending on the configuration of intra-domain routing. Hence, traffic from the external Internet to a customer should be modeled as a demand from a peering link to a *set* of access links. Similarly, the traffic introduced by a customer should be modeled as a demand from an access link to a *set* of peering links. A set of peering links (access links) can be represented by a logical node X_i (Y_i), as shown in Figure 3. (We synonymously use X_i (Y_i) to refer to either the logical nodes or to the set of links that these nodes represent.) In this paper, we focus on demands from a peer link to

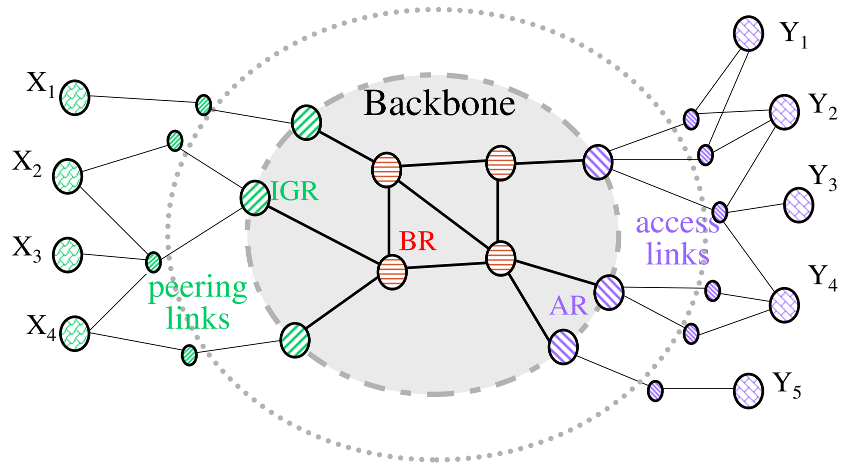

Figure 3: Network topology with logical nodes to/from sets of edge links

a set of access links, or from an access link to a set of peering links, rather than traffic between access links or between peering links.

Determining the traffic demands in an operational network requires identifying the sets of links (X_i and Y_i) and the associated traffic volumes. The sets of access links Y_i can be determined based on the *forwarding table* at each access router. Each table entry indicates a customer prefix (IP address and mask length) and the card name of the outgoing link. The card name also appears in the router configuration file, allowing the prefix to be associated with the appropriate link. By repeating this process across all of the ARs, we can determine the set of access links associated with each customer prefix. Similarly, each external prefix can be associated with a set of peering links X_i, based on information in the BGP routing tables. Each entry in the *BGP routing table* includes a prefix, an AS path, and an IGR loopback address. After identifying the IGR, it is possible to identify the appropriate peering link(s) at that router based on the next-hop AS number. The loopback address of each IGR and the next-hop AS number of each peering link can be extracted from the router configuration file.

Finally, we associate each demand with a volume of traffic. In our work on traffic engineering, we initially focus on *measured* traffic demands, rather than subscribed or projected loads. In particular, we consider flow-level measurements at the network edge, where traffic enters or leaves the network. A flow consists of a set of packets that match in all of the main IP and TCP/UDP header fields, such as source and destination IP addresses, protocol, port numbers, and type-of-service bits, and arrive close together in time. Each measurement record includes information about the traffic end-points, IP and TCP/UDP header fields, the number of packets and bytes in the flow, and the start and finish time of the flow; such information is available, for example, from Cisco's Netflow feature [7]. The source and destination IP addresses of the flow can be associated with the appropriate prefix, and matched to the corresponding set X_i or Y_i. Consequently we are able to compute comprehensive information about traffic demands by aggregating the flow level information to the level of X_i and Y_i.

4 Routing

Another key feature of NetScope is that it combines the network model and the traffic measurements with an accurate model of path selection. Specifically, NetScope's routing module determines the path(s) chosen by OSPF for each traffic demand, and the load imparted on each link as the traffic flows through the network. The routing module captures the selection of shortest paths to/from multi-homed customers and peers, the splitting of traffic across multiple shortest-path routes, and the multiplexing of layer-three links over layer-two trunks. These capabilities in NetScope allow a user to explore the impact of changes in the traffic demands or in the underlying network topology.

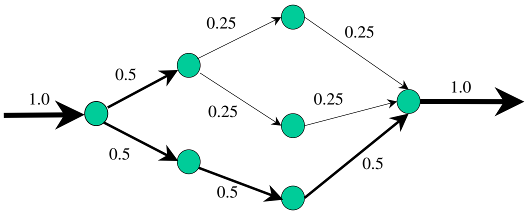

Figure 4: Traffic splitting across multiple shortest paths

4.1 NetScope Path Selection to approximate OSPF Routing

The OSPF protocol defines how routers within an area exchange link-state information and compute shortest paths based on the sum of the link weights. The link weights are static and are typically configured based on the link capacity, physical distance, and some notion of the expected traffic load. The chosen paths do not change unless a link or router failure occurs, or the OSPF parameters are reconfigured. These are rare events, particularly for the backbone links that participate in the routing protocol. As such, NetScope considers a single instance of the network topology and OSPF configuration, and does not simulate the details of the OSPF protocol, such as the flooding of link-state advertisements or the exchange of "hello" messages. Performing the path-selection computation inside the tool, rather than using the forwarding tables or traceroute results directly, facilitates experimentation with alternate OSPF configurations and different topologies.

When all of the backbone links reside in a single OSPF area, path selection simply involves computing the shortest paths between each pair of routers, based on the link weights. In a hierarchical network, traffic between two routers in the same area follows a shortest path within the area, even if the network has a shorter path that involves links in other areas. When traffic must travel between routers in different areas, the path depends on how much information each area has about its neighbors. Currently, our routing module assumes that the network does not summarize routing information at area boundaries. In the absence of route summarization, each border router reports the cost of the shortest path(s) to each of the other routers in the area, and the traffic between routers in different areas simply follows a shortest path without regard to the area boundaries. The routes are computed using Dijkstra's shortest-path-first algorithm.

4.2 OSPF Tie-Breaking

Path selection becomes more complex when there are *multiple* shortest paths between a pair of routers. Such ties arise very naturally when the network topology has parallel links between adjacent routers for additional capacity. Ties also surface when many of the links in the network have similar weights. This is sometimes done intentionally to increase the effective capacity between two end-points. The presence of multiple shortest paths allows for load-balancing of the traffic between the two end points. This is achieved by allowing the IP forwarding table to have multiple outgoing links associated with a single destination prefix. Rather than alternating between these links at the packet level, routers typically attempt to forward packets for the same source-destination pair along a single path; this reduces the likelihood that packets from the same TCP connection arrive out-of-order at the receiver. Load-balancing is typically achieved by performing a hash function on the source and destination IP addresses of each packet. The value of the hash function determines which outgoing link should carry the packet.

In theory, the details of the tie-breaking function could be modeled in the NetScope tool. However, this would significantly complicate the path-selection computation, and would require computing traffic demands at a significantly finer level of granularity. In addition, the details of the hashing function, and how the outputs of the hash function map to particular outgoing links are not specified by the OSPF protocol and, as such, depend on the vendor's implementation. Fortunately, these details are usually not important. The hash function is designed to support an even splitting of the traffic across the multiple outgoing links, especially for backbone links that carry a diverse mixture of traffic with different source and destination addresses. As such, our routing model splits traffic evenly across each of the outgoing links along a shortest path. For example, if a router has four outgoing links on shortest paths, each link would carry 25% of the traffic. The division of traffic is recursive, with the downstream routers dividing the traffic across each of their outgoing links, as shown in Figure 4.

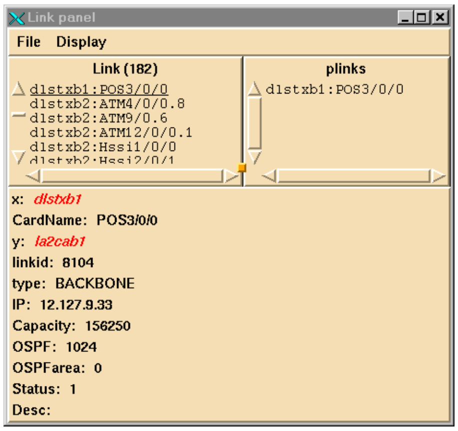

Figure 5: The Link Panel, displaying the attributes of a backbone link.

4.3 Multi-homed Customers/Peers

As discussed in Section 3.2, traffic from a customer is represented as a demand from an access link to a set of peering links (X_i). Similarly, the traffic to a customer is represented as a demand from a peering link (where the traffic was measured) to a set of access links (Y_i). The choice of a particular peering link (access link) from the set X_i (Y_i) depends on OSPF routing. A set of links can be represented as a logical node, as shown in Figure 3. Traffic travels to the closest link, along a shortest path. The chosen link, as well as the chosen path, depends on the link weights. We model the selection of the shortest path to the closest node by assuming that the logical links all have the same OSPF weight (e.g., a nominal weight of 1). Under this assumption, a shortest path to a logical node X_i (Y_i) travels through the appropriate peering link (access link).

In the end, the routing module operates on a set of demands, each traveling from one peering link (access link) to a set of access links (peering links). The module computes the set of shortest-path routes based on the topology and the OSPF configuration, and determines how the demand splits across the multiple paths. Repeating this process for each demand results in an estimate of the load imparted on each link. Then, the routing module determines the load on each trunk (layer-two link) by summing across the associated (layer-three) links. The generality of the routing model facilitates experiments with alternate topologies and OSPF configurations, as illustrated in the next section. NetScope also supports experimentation with the BGP policies for outbound traffic, by changing the sets of peering links associated with external network addresses.

5 Visualization

The NetScope visualization environment provides many ways to explore the data associated with an IP backbone network, and the ability to perform “what-if” experiments. This section gives a brief overview of the visualization environment, followed by a demonstration of using the tool. The demonstration addresses the traffic-engineering task of reducing the load on the network's most utilized link, using an artificially constructed topology and set of traffic demands.

5.1 Objects

The NetScope data model is decomposed into a set of objects, e.g., objects representing each router and each link. We associate a list of attributes with each object, as discussed in Section 3.1. The visualization environment provides a way to examine each object and see all of its attributes. For example, Figure 5 shows the Link Panel. The attributes displayed are for the link underlined in the upper left list, namely, the interface POS3/0/0 from a router in Dallas (which terminates in a router in Los Angeles). This link is implemented via packet-over-SONET technology and therefore has only one associated physical link (plink), itself. NetScope allows for easy navigation between the objects. For example, starting at a link it is straightforward to find the routers where this link terminates. Starting at a router it is easy to find the links that terminate there.

5.2 Statistics

NetScope maintains statistical information associated with objects. Each statistic is simply a value for each object of some type. For example, a link utilization statistic is a percentage associated with each link. There is no restriction on how many statistics can be associated with an object type. For example, it is possible to keep link utilization on an hourly basis for a full week. Statistics can be static (read from a file, or computed once) or dynamic (automatically recomputed as needed). NetScope has many ways of displaying statistics. For objects that have graphical representations such as links or nodes, NetScope can make the size or color of the object be proportional to the value of the statistic, providing a visual representation of the statistic. The first step of our traffic engineering task is to visualize the utilization. Figure 6 shows such a screendump from NetScope operating on our artificially constructed network topology. Each node corresponds to a router and each edge corresponds to a backbone link. The thickness and the color of the edges corresponds to the utilization on that link. The links with low utilization are thin and yellow, while those with high utilization are thick and red.

The smaller window in the figure is the Link Statistics window. This window displays information about statistics, and allows the user to visualize statistics in many ways. From this window, the user can specify which statistic (if any) should be used to determine the size or color of the objects. The statistics window also displays summary statistics for each link statistic. Under the columns labeled Min, Max, and Ave, we see the minimum, maximum, and mean value (over all links) for each displayed statistic, e.g., the ave is 8.83%.

NetScope has the notion of a current object for each type. An object becomes the current object either when it is selected or when the mouse is moved on top of it. The column labeled Current shows the utilization value for that link, with the name of the current link displayed below the menu-bar. In the statistics window in Figure 6, we see that the current link is the interface POS3/0/0 on a backbone router in Dallas, and that the utilization on that link is 29.27%. Also from this window, the user can produce other views of statistics, such as histograms (showing how many links have a utilization that falls into a certain range) or tables. It is also possible to correlate different statistics for the same object type. This can be done either via a scatter plot, or by, for example, using the color of the lines to visualize highly utilized links and using thickness of the lines to visualize links with high delay.

5.3 Locating Heavily-Loaded Links

To finish the first step of our example traffic engineering task, the NetScope user needs to identify the most heavily-loaded link. NetScope has powerful search tools that allow the user to perform complex queries on all objects via a Find panel, as shown in the example in Figure 6. Given that the maximum utilization for our example is 67.59%, the user asked the tool to locate all links with a utilization value above 60%. By passing the mouse over each link listed here one can easily find the link with the highest utilization. (There are also other ways to find this link, such as a histogram or table.) In this case, we find that the link with the highest utilization is a cross-country link from Washington, D.C. to San Francisco.

The queries allowed in the Find panel are very general and can refer to arbitrary attributes and statistics. After locating an interesting subset of data, NetScope allows the user to restrict the current view to only that subset. For example, if the user was only interested in the backbone links, the user could find that set of links and then make that the active set by clicking on the "Make Active" button. Changing the active set automatically causes several changes. The display changes to show only those links that are now active. The minimum, maximum, and average are recomputed over the new active set. Finally, the coloring and sizing of links (if turned on) will change to reflect the new range of values.

5.4 Traffic on a Link

Continuing with the traffic engineering example, we next need to identify which traffic demands contribute to the load of the highly utilized link. The link carries traffic for a collection of different demands. NetScope's data and routing models allow us to see the source and sink tree of all traffic demands using the link. Each object has a menu associated with it, with operations that apply to that object. This menu includes some general object operations such as "get information," "select," "zoom," and operations specific to the object type. The link menu, for example, includes the option for finding all traffic demands with shortest path routes that include this link. This query is executed via the demand Find Panel. By using the "Make Active" button in this window, we restrict NetScope's view to this set of demands. The statistics and their graphical visualization (e.g., the colors of the links) will be adjusted automatically.

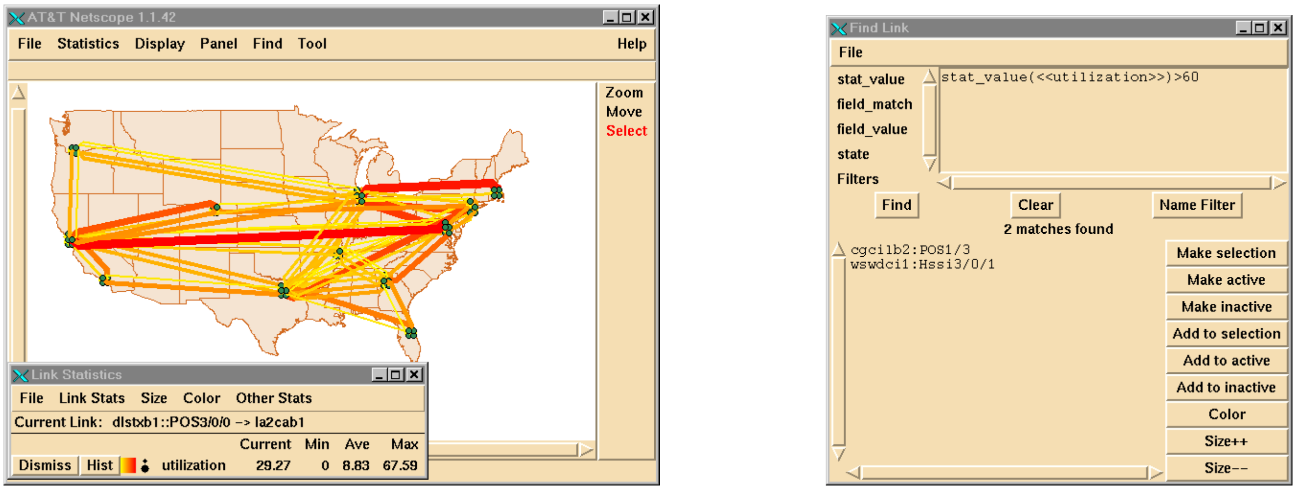

Figure 6: (Left) Link utilization displayed as size and color of links. The smaller window is the Link Statistics window. (Right) Finding all links with utilization above 60%.

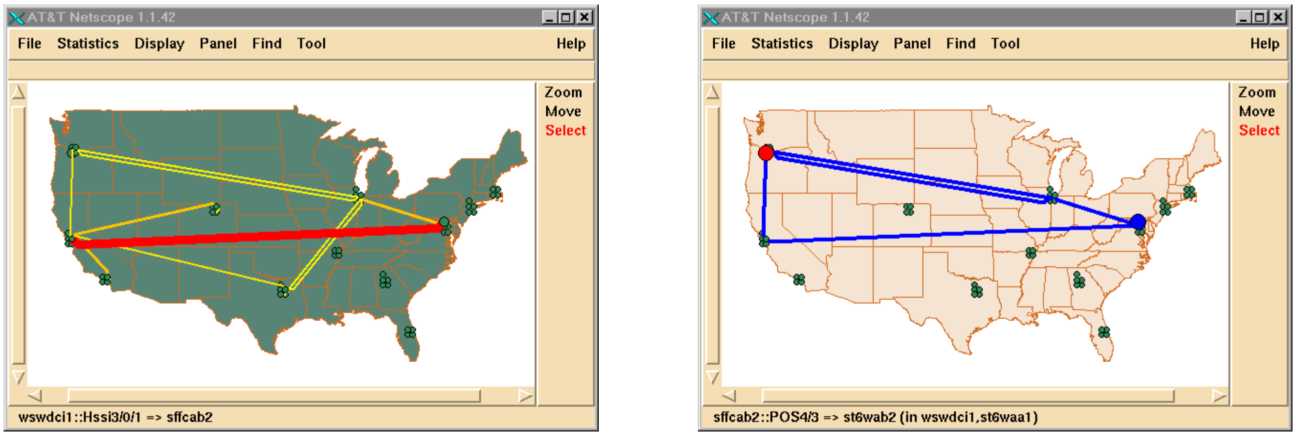

Figure 7: (Left) The utilization caused by a subset of the demands, namely all the demands that impose some traffic onto the most heavily loaded link. (Right) The routing of a demand from Washington D.C. to Seattle.

To focus our attention on the chosen demands, we activate only the links that carry one or more of these demands. The resulting view is shown in Figure 7. At first glance, it may seem confusing that the resulting graph is not a tree. However, OSPF tie-breaking introduces multiple paths for each demand, hence we get a directed acyclic graph instead of a tree. What appears to be a cycle in the part of the topology shown in Figure 7 are directed links pointing to either San Francisco or from Washington, D.C. To illustrate the impact of OSPF tie-breaking on the route selection, Figure 7 also shows how one such demand is routed through the network. This route from Washington, D.C. to Seattle uses three paths through the network, one of them using the heavily utilized link from Washington, D.C. to San Francisco followed by the link from San Francisco to Seattle. The other two routes use the link from Washington, D.C. to Chicago and then the two parallel links between Chicago and Seattle.

5.5 Changing Routes

The obvious approach to alleviate the congestion on the highest utilized link, other than increasing the capacity of the link, is to increase the OSPF weight of the link in order to change the routing. This will tend to move traffic from that link to other links. NetScope allows the user to modify OSPF weights. Upon changing the OSPF weights, NetScope recalculates all routes for all active traffic demands. The tool then updates all statistics that are based upon the traffic, including link load and utilization. In order to compare alternate settings of OSPF weights, NetScope maintains two different sets of weights, one that can be manipulated and one that acts as an anchor.

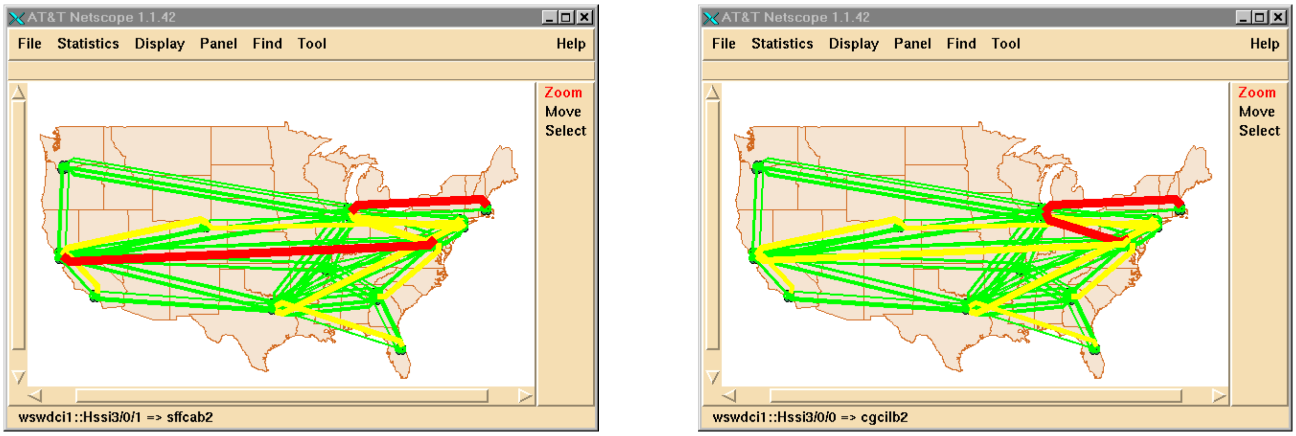

Figure 8: Increasing an OSPF weight to try to move traffic off a link. Before (left) and after (right) pictures.

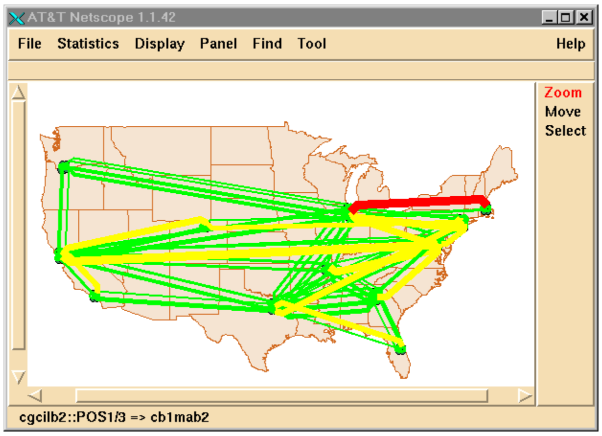

Figure 9: Decreasing an OSPF weight on a different link. This gives a better solution.

Figure 8 shows the network utilization before increasing the OSPF weight of the highest utilized link and after changing the weight. To simplify the visualization we use a different coloring scheme in this and subsequent figures. Links with low utilization (at most 30%) are green; links with medium utilization (between 30% and 60%) yellow; links with high utilization (over 60%) red. Before the change the most utilized link had a utilization of 67.59%. Changing the OSPF weight reduces the link utilization to 50.69%. Unfortunately, a link from Washington, D.C. to Chicago that previously had a fairly small utilization, now has utilization 69.70%. It is possible to explain this by going back to Figure 7. By increasing the OSPF weight on the link between Washington, D.C. and San Francisco, the path from Washington, D.C. to San Francisco to Seattle is removed from the path set for the demand from Washington, D.C. to Seattle. The same is true for the demand from Washington, D.C. to Dallas. The implication is that the traffic that used to flow on the link between Washington, D.C. and San Francisco now traverses the link from Washington, D.C. to Chicago. This illustrates one of the difficulties in managing an IP network: a small local change can have significant global impact.

Based upon this understanding of the traffic pattern, one might want to decrease an OSPF weight on a different link, so that it will attract more traffic. From the above discussion one can identify the traffic demand between Washington, D.C. and Dallas as a good target. Consequently the link between Washington, D.C. and St. Louis is a good target link. Figure 9 shows the result of such a change. This time we succeeded in decreasing the total maximum utilization to less than 61.5%. To decrease the utilization even further, a traffic engineer might want to iterate the process that we have illustrated for the now highly utilized link between Chicago and Cambridge. Alternatively, an optimization tool can automate all or part of these tasks; this requires efficient algorithms for selecting OSPF weights based on a given topology and set of traffic demands [8].

6 Conclusions and Future Work

The NetScope toolkit integrates accurate models of topology, traffic, and routing with a flexible visualization environment to support traffic engineering in large ISP networks. The power of the tool stems from the integrated data model that provides a network-wide view of configuration information and traffic statistics together with a routing model. The topology model includes the network connectivity, link capacities, and the parameters controlling intra-domain routing, derived from the router configuration files. Traffic demands are computed based on edge measurements, and are aggregated to the coarsest level that still permits accurate modeling of intra-domain and inter-domain routing. Drawing on the traffic demands and the topology model, the routing module determines the load imparted on layer-three links and layer-two trunks under a variety of network configurations. These modules, coupled with the flexible visualization environment, enable users to investigate alternative configurations off-line without disrupting the operational network. In addition, the framework readily supports additional modules for automated network optimization and performance debugging.

As part of our ongoing work, we are evolving the tool to operate on a continuous feed of topology and traffic data, to track changes in the network configuration and usage patterns. In addition, we are analyzing the statistical properties of the measured traffic demands to determine how the load between access and peering links fluctuates on various time scales. This traffic analysis is critical to determining the appropriate time scale for exercising control over the network, such as changing the configuration of intra-domain or inter-domain routing. We are also investigating how the traffic demands would change after a network reconfiguration. For example, flows regulated by congestion control mechanisms, such as TCP, may be able to transmit packets more aggressively if a routing change alleviates congestion on bottlenecked links. In addition, we are exploring the use of NetScope for traffic engineering and admission control for the mixture of traffic classes proposed for differentiated services. We believe that the NetScope approach to combining topology, traffic, and routing can serve as a general underpinning for managing the performance of large ISP networks.

Acknowledgments

Several of our colleagues in AT&T Labs and in AT&T's Internet services and operations groups have provided valuable feedback on our work, for which we are very grateful. We would like to thank Han Nguyen for his role in formulating and supporting the NetScope project, and Jay Borkenhagen, Michael Langdon, Anthony Longhitano, and Fred True for helping us access and understand the network measurement and configuration data. Thanks also to Youngmi Joo and Michael Merritt for their comments on an earlier draft of the paper.

References

[1] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proc. ACM SIGCOMM*, September 1999.

[2] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Networking*, vol. 2, pp. 1–15, February 1994.

[3] B. Halabi, *Internet Routing Architectures*. Cisco Press, 1997.

[4] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1999.

[5] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[6] A. Feldmann, "Netdb: IP network configuration debugger/database," tech. rep., AT&T Labs – Research, July 1999.

[7] Cisco Netflow. http://www.cisco.com/warp/public/732/netflow/index.html.

[8] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, March 2000.