Deriving Traffic Demands for Operational IP Networks: Methodology and Experience

Anja Feldmann*, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True

AT&T Labs – Research; Florham Park, NJ 07932
{anja,albert,lund,reingold,jrex,ft}@research.att.com

ABSTRACT

Engineering a large IP backbone network without an accurate, network-wide view of the traffic demands is challenging. Shifts in user behavior, changes in routing policies, and failures of network elements can result in significant (and sudden) fluctuations in load. In this paper, we present a model of traffic demands to support traffic engineering and performance debugging of large Internet Service Provider networks. By defining a traffic demand as a volume of load originating from an ingress link and destined to a set of egress links, we can capture and predict how routing affects the traffic traveling between domains. To infer the traffic demands, we propose a measurement methodology that combines flow-level measurements collected at all ingress links with reachability information about all egress links. We discuss how to cope with situations where practical considerations limit the amount and quality of the necessary data. Specifically, we show how to infer interdomain traffic demands using measurements collected at a smaller number of edge links — the peering links connecting to neighboring providers. We report on our experiences in deriving the traffic demands in the AT&T IP Backbone, by collecting, validating, and joining very large and diverse sets of usage, configuration, and routing data over extended periods of time. The paper concludes with a preliminary analysis of the observed dynamics of the traffic demands and a discussion of the practical implications for traffic engineering.

1. INTRODUCTION

The engineering of large, IP backbone networks faces a number of difficult challenges. Owing to the astonishing success of Internet applications and the continuing rollout of faster access technologies, demand for bandwidth across backbones is growing explosively. In addition, shifts in user behavior, publishing of new Web content, and deployment of new applications result in significant fluctuations in the volume of traffic exchanged between various hosts in the Internet. Furthermore, changes in routing policies and failures of network elements can cause sudden fluctuations in how traffic flows through the backbone. This leaves network operators in the difficult situation of trying to tune the configuration of the network to adapt to changes in the traffic demands. The task is particularly daunting since the Internet Service Provider (ISP) responsible for administering the backbone does not have end-to-end control of the path from the source to the destination. The majority of traffic in the Internet travels across multiple administrative domains.

The networking community has responded with research and development on increasing link and router capacity and providing a more easily configurable infrastructure. However, relatively little attention has been given to the systems needed to guide the operation and management of the improved infrastructure. In particular, there has been very little work on models for traffic demands or on techniques for populating these models from network measurements. Most existing measurement techniques provide views of the *effects* of the traffic demands — poor end-to-end performance (e.g., high delay and low throughput) and heavy load (e.g., high link utilization and long queues). These effects are captured by active measurements of delay, loss, or throughput on a path through the network [1], or passive monitoring of individual routers and links [2, 3].

However, managing an ISP backbone begs for a *network-wide* understanding of the flow of traffic. An accurate view of the traffic demands is crucial for a number of important tasks, such as debugging performance problems, optimizing the configuration of the routing protocols, and planning the rollout of new capacity. In particular, the recently-formed IETF working group on Internet Traffic Engineering recognizes that (i) accurate demand models are crucial for effective traffic engineering of IP networks, but (ii) developing such models and populating them via appropriate measurements are open problems [4, 5]. These are precisely the topics we address in this paper. As far as we know, no comparable study of the network-wide traffic demands in an ISP backbone has been conducted to date.

How should traffic demands be modeled and inferred from network measurements? At one extreme, IP traffic could be represented at the level of individual source-destination

*Anja Feldmann has joined Universität des Saarlandes in Saarbrücken, Germany (anja@cs.uni-sb.de).

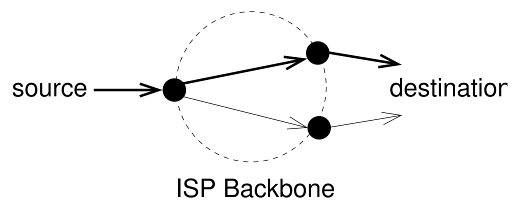


Figure 1: An IP traffic demand is naturally modeled as a point-to-*multipoint* volume because the destination is reachable via *multiple* egress links.

pairs, possibly aggregating sources and destinations to the network address or autonomous system level. Such an *end-to-end* traffic matrix would lend insight into the fluctuations in load over the Internet across time. However, representing all hosts or network addresses would result in an extremely large traffic matrix. In addition, no single ISP is likely to see all of the traffic to and from each network address, making it virtually impossible to populate such a model.

Alternatively, IP traffic could be aggregated to point-to-point demands between edge links or routers in the ISP backbone, an option suggested in [6] in the context of MPLS-enabled networks. However, this approach has fundamental difficulty in dealing with traffic that traverses multiple domains. A given destination (network address) is typically reachable from multiple edge routers, as shown in Figure 1. As a result, IP traffic demands are naturally modeled as *point-to-multipoint* volumes. This is a simple, but crucial difference between IP networks and connection-oriented networks (such as Frame Relay), where demands are naturally modeled as point-to-point volumes. The set of egress links depends on the ISP's routing policies and the BGP advertisements received from neighboring domains. The selection of a unique link from this set depends on intradomain routing information. In the example, suppose the traffic exits the network via the top egress link. A link failure or a change in the configuration of intradomain routing could cause the traffic to move to the bottom egress link. A change in the ISP's interdomain policies or the withdrawal of a route advertisement from a neighboring domain could also alter the flow of traffic. Modeling interdomain traffic as point-to-point would couple the demand model to the internal routing configuration, making it difficult to predict how such changes would affect network load; the routing change itself could have a major impact on the point-to-point demands.

In this paper, first we propose a simple traffic demand model, which effectively handles interdomain traffic. As discussed in Section 2, the model is invariant to changes in the internal routing configuration, and as such provides a sound basis for traffic engineering. Our demand model allows us to predict how changing the internal routing configuration impacts the distribution of load on internal links. Second, we provide a methodology for populating the model from usage measurements collected at ingress links and reachability information collected at egress links. Third, we consider how to apply the model when practical considerations severely limit the location of usage measurements to a much smaller number of edge links. Specifically, in Section 3, we propose a methodology for populating the interdomain demand model when usage measurements are limited to the links to neighboring service providers, coping (in particular) with not having usage measurements at customer access points.

Next, we describe our experiences applying the methods of Sections 2 and 3 in a large, operational ISP network — the AT&T IP Backbone. This is where we must confront practical limitations in the usage, configuration, and routing data available in today's IP networks. In Section 4, we describe the challenges of processing router configuration files, forwarding tables, flow-level measurements, and SNMP data collected from multiple locations in the network over an extended period of time. In particular, we highlight how we addressed several practical constraints that arose in processing the large (and lossy) flow-level measurements. In Section 5, we present results showing the effectiveness of the techniques in Section 2 and 3. We show that the data sets collected at multiple times and locations are remarkably coherent, and present a detailed explanation of the occasional inconsistencies that arise from network dynamics.

Our analysis of the measured demands focuses on the time scale of tens of minutes to hours or days. Traffic engineering tasks occur on this time scale [7], where fundamental shifts in user behavior and changes in network routing introduce traffic variability beyond statistical fluctuations. On a smaller time scale, Internet traffic fluctuates in reaction to bursty user behavior and congestion control mechanisms. In populating our demand model, we focus on large aggregates of traffic, rather than the dynamics of individual flows. The distribution of traffic through the network is sensitive to the dynamics of interdomain routing, which may change the set of egress points for a particular destination. Our demand model can be applied to investigate the *impact* of such changes in reachability information, due to network failures, reconfigurations, or policy changes.

2. TRAFFIC DEMANDS

This section presents a brief overview of ISP backbone architectures and routing protocols. We also propose a model for IP traffic demands, and discuss its application to several important traffic-engineering tasks. Then, we describe how to compute these demands from flow-level measurements at ingress links and reachability information about egress links.

2.1 ISP Backbone Networks

An ISP backbone network consists of a collection of IP routers and bi-directional layer-three links, as shown in Figure 2. *Backbone* links connect routers inside the ISP backbone, and *edge* links connect to downstream customers or neighboring providers. Edge links are divided into *access* links and *peering* links. 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. Depending on the contractual relationships, the ISP may or may not allow a pair of peers to communicate across the backbone.

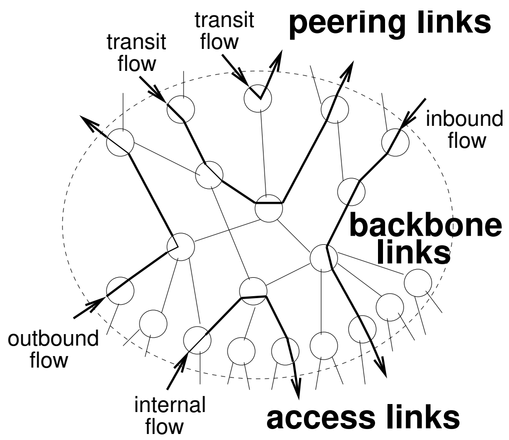


Figure 2: Traffic flows in an ISP backbone

The traffic enters the backbone on an *ingress* link and leaves on an *egress* link. Figure 2 illustrates the possible scenarios. For example, *internal* traffic travels between two access links, whereas *transit* traffic travels between two peering links. *Inbound* traffic travels from a peering link to an access link, and *outbound* traffic travels from an access link to a peering link. Much of the traffic in the Internet must travel through multiple domains en route from the source to the destination. Hence, most of the traffic in an ISP backbone enters or leaves the network on a peering link. As such, an ISP rarely has complete control of the entire path from the source to the destination. Even for internal traffic, the customer exercises control over how the traffic enters the ISP backbone, and how the traffic travels from the egress link through the customer's network to the destination host.

The path traveled by an IP packet depends on the interplay between interdomain and intradomain routing. The ISP employs an intradomain routing protocol, such as OSPF, IS-IS, or MPLS, to select paths through the backbone between ingress and egress links. Under OSPF and IS-IS, the routers exchange link-state information and forward packets along shortest paths, based on the sum of link weights chosen by the ISP. Typically, customers and peers do not participate directly in these protocols with the ISP. Communicating across domains requires coordination with customers and peers to exchange reachability information. Interdomain routing operates at the level of a network address, or prefix, consisting of an IP address and a mask length (e.g., 135.207.119.0/24 has a 24-bit mask that specifies a block of 256 contiguous addresses). An ISP typically uses static routes to direct traffic toward customers who have a fixed set of network addresses and do not participate in an interdomain routing protocol. The Border Gateway Protocol (BGP) is used to exchange dynamic reachability information with the remaining customers and neighboring providers.

2.2 Demand Model

The interplay between intradomain and interdomain routing has important implications for how we define a traffic demand. The ISP network lies in the middle of the Internet, and may not have a direct connection to the sender or the receiver of any particular flow of packets. As such, a particular destination prefix may be reachable via *multiple* egress links from the ISP. A multi-homed customer may receive traffic on two or more links that connect to different points in the ISP backbone. Likewise, an ISP may have multiple links connecting to a neighboring provider. When the ISP learns multiple routes to the same destination prefix, the ultimate decision of which route to use depends on the BGP route-selection process. The decision process involves multiple steps to select from the set of advertised routes. First, the ISP can apply import policies to prefer one route over another. For example, the ISP may prefer routes via one neighbor over another. Then, the ISP considers the length of the path, in terms of the number of autonomous systems involved, followed by several other criteria [8].

Later in the tie-breaking process, the selection of a route (and the corresponding egress link) depends on information from the intradomain routing protocol. For example, suppose the BGP selection process results in two routes leaving the ISP backbone on the east and west coasts, respectively. The egress link for a particular packet would depend on where this traffic entered the network. The packet would travel to the "closest" egress link, where closeness depends on the intradomain routing parameters. Finally, if both egress links have the same shortest-path cost, the tie is broken by comparing the identifiers of the two routers responsible for advertising these routes. The dependence on intradomain routing implies that a change in the backbone topology or routing configuration could change which egress link is selected. Similarly, if traffic enters the backbone in a different location, the egress link could change.

To be practical, our representation of traffic demands should enable experimentation with changes to the network topology and routing configuration. Hence, we associate each traffic demand with a *set* of egress links that could carry the traffic. The set represents the outcome of the early stages of the BGP route-selection process, before the consideration of the intradomain protocol. This is in contrast to models that use a multipoint set to capture uncertainty in the distribution of customer traffic across a set of different destinations [9]. In our model, the selection of a particular egress link within the set depends on the configuration of intradomain and interdomain routing. The ISP typically has very limited control over the selection of the ingress link of the traffic. The selection of the ingress link depends on the routing policies of other autonomous systems and directly-connected customers. For our initial work on computing and analyzing the traffic matrix, we do not attempt to model how the ingress link is selected. Our model of a traffic demand consists of an ingress link, a set of egress links, and a volume of load.

2.3 Traffic-Engineering Applications

The volume of load represents the quantity of traffic distributed from the ingress link to the set of possible egress links, averaged over some time scale. This introduces issues of both spatial and temporal aggregation. On the one extreme, it is possible to compute a separate demand for each source-destination pair that exchanges traffic across the backbone. On the other extreme, there could be a single demand for all traffic with the same ingress link and set of egress links. The appropriate choice depends on the application. For example, consider the task of optimizing the configuration of intradomain routing to balance network load [10]. This application could combine all traffic with the same ingress link and set of egress links into a single demand. Changes in intradomain routing could affect the selection of the egress link for each demand. The details of which packets or flows contribute to the demand are not important. Minor extensions to this approach could define a separate demand for each traffic class under differentiated services. This would enable the route optimization to consider the load imparted on each link by each traffic class.

As another application, suppose an ISP is considering a change in its BGP import policies for routes to a particular destination prefix belonging to another provider. A destination prefix that receives a large amount of traffic could result in heavy congestion on one or more peering links. Redirecting this load to a different set of egress links could alleviate the congestion. BGP route advertisements, or entries in the BGP tables, could be used to determine the egress links for a destination prefix. A change in BGP import policies, such as filtering a route advertisement or assigning different local preference values, would change the set of egress links associated with this destination prefix. Similarly, network failures, policy changes in other domains, and even network congestion could result in fluctuations in the BGP reachability information [11, 12]. These intentional and unintentional changes would result in a new traffic demand. To experiment with different sets of egress links, the ISP would need to know which traffic is associated with this particular prefix. For this application, traffic destined to this prefix should not be aggregated with other traffic with the same ingress link and set of egress links.

An ISP may also need to predict the effects of adding or moving an access link. For example, the ISP could rehome an existing customer to a different edge router. In this situation, all outbound demands associated with this customer should originate from the new location, and all inbound demands would have a new set of egress links to reflect the rehomed access link. This would enable the ISP to predict how rehoming the customer would affect the load on the backbone. Similarly, an existing customer may request a new access link for higher bandwidth or fault tolerance. The new link could be added to the set of egress links for inbound demands. The ISP may also have information about how the customer would direct outbound traffic to its access links. This would enable the ISP to predict what portion of the existing outbound traffic from this customer is likely to enter the network on the new access link. Finally, the ISP may need to estimate the effects of adding a new customer. In some situations, the ISP may have information that can aid in predicting the demands. For example, a customer that hosts Web content may have server logs. The traffic statistics could be aggregated to the client prefix level [13] to predict the outbound demands for the new access link.

Ultimately the spatial aggregation of the traffic demands depends on the particular application, ranging from performance debugging and backbone traffic engineering to BGP policy changes and capacity planning. Likewise, the temporal aggregation depends on the application. Long-term capacity planning could consider the traffic on a relatively coarse time scale, whereas debugging short-term performance problems would require a more careful consideration of how load fluctuates across time. In our initial study of traffic demands, we focus on backbone traffic engineering [7]. As such, we aggregate traffic with the same ingress link and set of egress links into a single demand. In a large, operational ISP network, this results in fairly large number of traffic demands. The volume of load associated with each demand is identified by flow-level measurements at the ingress links. The set of egress links is identified based on snapshots of the forwarding tables from the routers in the operational network. Then, we compute the current set of demands over a variety of time scales and study the traffic dynamics.

2.4 Measurement Methodology

To compute the traffic demands, fine-grain traffic measurements should be collected at all *ingress* links. This enables us to identify the traffic as it enters the ISP backbone. However, collecting packet-level traces at each ingress link would be prohibitively expensive. In addition, traffic engineering does not necessarily need to operate at the small time scale of individual packets. Instead, we propose that *flow*-level statistics should be collected at each ingress link. These measurements can be collected directly by the incident router [14, 15]. A flow is defined as a set of packets that match in the key IP and TCP/UDP header fields (such as the source and destination addresses, and port numbers) and arrive on the same ingress link. The router should generate a record summarizing the traffic statistics on a regular basis, either after the flow has become inactive or after an extended period of activity. The flow record should include sufficient information for computing the traffic demands: the `input` link and the `dest` IP address to identify the end-points of the demand, the `start` and `finish` times of the flow, and the total number of `bytes` in the flow. (Any additional information in the measurement records, such as TCP/UDP port numbers or type-of-service bits, could be used to compute separate traffic demands for each quality-of-service class or application.) Sampling of the measurements may be performed to reduce the total amount of data.

Computing the traffic demands requires information about the destination prefixes associated with each egress link. The aggregation process draws on a list, `dest_prefix_set`, of network addresses, each consisting of an IP address and mask length. Each prefix, `dest_prefix`, can be associated with a set of egress links, `reachability(dest_prefix)`. In an operational network, these prefixes could be determined from the forwarding tables of the routers that terminate egress links. In particular, each forwarding-table entry identifies the next-hop link(s) for a particular prefix. This enables identification of the prefixes associated with each egress link. (The router connected to the egress links has the most detailed view of the destination prefix. Suppose a router has egress links that connect to customers that were assigned contiguous blocks of IP addresses. This egress router's forwarding table would have an entry for each prefix to direct traffic to the appropriate access link. However, the other routers in the ISP backbone, and the rest of the Internet, do not need such detailed information. As such, the edge router may advertise an aggregated network address to the rest of the backbone. In this case, information available at the ingress router would not be sufficient to identify the customer prefix and the associated set of egress links.)

```
For each flow: (input, dest, start, finish, bytes)
    dest_prefix = longest_prefix_match(dest, dest_prefix_set);
    egress_set = reachability(dest_prefix);
    start_bin = ⌊start/width⌋ * width;
    finish_bin = ⌊finish/width⌋ * width;
    if (start_bin == finish_bin)
        volume[input, egress_set, start_bin] += bytes;
    else /* Compute volume of traffic for each time_ bin */
        byte_rate = bytes/(finish - start);
        volume[input, egress_set, start_bin] += byte_rate * (start_bin + width - start);
        for (time_bin = start_bin + width; time_bin < finish_bin; time_bin += width)
            volume[input, egress_set, time_bin] += byte_rate * width;
        volume[input, egress_set, finish_bin] += byte_rate * (finish - finish_bin);
Output for each aggregate: (input, egress_set, time_bin, volume)
```

Figure 3: Computing traffic demands based on measurements at ingress links

Each flow spans some time interval (from `start` to `finish`) and contributes some volume of traffic (`bytes`). Computing traffic demands across a collection of flows at different routers introduces a number of timing challenges. The flow records do not capture the timing of the individual packets within a flow. Since traffic engineering occurs on a time scale larger than most flow durations, we compute demands on a time scale of tens of minutes to multiple hours. Consequently, we are not concerned with small variations in timing on the scale of less than a few minutes. We divide time into consecutive `width`-second bins. Most flows start and finish in a single bin. When a flow spans multiple bins, we subdivide the traffic in proportion to the fraction of time spent in each time period. For example, suppose a 10-kbyte flow spent one second in the first bin and nine seconds in the second bin. Then, we associate 1 kbyte with the first bin and 9 kbytes with the second bin. The algorithm for computing the traffic demands in summarized in Figure 3.

3. MEASUREMENT AT PEERING LINKS

Collecting flow-level measurements at each ingress link would be the ideal way to determine the traffic demands. In this section, we extend our methodology to measurements collected at a much smaller number of edge links — the links connecting the ISP to neighboring providers. We describe how to infer where outbound traffic enters the backbone, based on customer address information and a model of how traffic from each of the customer's access links would be routed across the ISP backbone.

3.1 Adapted Measurement Methodology

Collecting fine-grained measurements at every ingress link would introduce substantial overhead in a large network. ISP backbones typically include a large number of access links that connect to customers at various speeds. The routers that terminate these links often vary in functionality and must perform computationally-intensive access control functions to filter traffic to/from customers. Collecting flow-level statistics at every access link would introduce additional overhead on these routers; in fact, some of these routers may not be capable of collecting fine-grain measurements. In contrast, a small number of high-end routers are used to connect to neighboring providers. These routers typically have much fewer links with substantial functionality (including measurement functions) implemented directly on the interface cards. Collecting flow-level measurements on these links is much less difficult. In addition, throughout the Internet, the links between major service providers carry a large amount of traffic and are vulnerable to fluctuations in interdomain routing, making it very important to have detailed usage statistics from these locations.

By monitoring both the ingress *and* egress links at these locations, we capture a large fraction of the traffic in the ISP backbone. However, this introduces new complications for measuring internal, outbound, and transit traffic, as summarized in Table 1. First, monitoring the peering links does *not* capture the internal traffic sent from one access link to another. (Some customer traffic may travel over particularly important access links to and from the ISP's e-mail, Web, and DNS services. Flow-level measurements should be enabled on these access links — effectively treating these connections as peering links.) Second, computing the *outbound* demands that travel from access links to peering links becomes more difficult, since flow-level measurements are not available at the *ingress* points. Inferring how these flows entered the network is the main focus of the rest of this section. Third, measuring both ingress and egress traffic at the peering links may result in *duplicate* measurements of transit traffic. These flows should not be counted twice.

The first step in computing the traffic demands is to classify a flow as inbound, transit, or outbound, as illustrated in Figure 2. The classification depends on the input and output links at the router that measured the flow, as summarized in Table 2. We initially focus our discussion on inbound and outbound flows, and discuss transit traffic in greater depth at the end of the subsection. For inbound flows, traveling from a peering link to a backbone link, we can directly apply our methodology from Section 2, since flow-level measurements are available from the ingress link. The process for aggregating the flow records is summarized in Figure 4, skipping the details from Figure 3 of dividing the `bytes` of the flow across multiple `time_bins`.

Outbound flows require more careful handling. The flow measurements provide two pieces of information that help us infer the ingress link responsible for outbound traffic — the source IP address and the input/output links that observed the flow at the egress router. The source address in-

	Ideal	Measuring at Peering Links			
		Inbound	**Transit**	**Outbound**	**Internal**
Ingress	measurement	measurement	measurement	×	×
Egress	reachability	reachability	reachability measurement	reachability measurement	reachability

Table 1: Measurement and reachability information available at ingress and egress links

Input	Output	Classification	Action
Peer	Backbone	Inbound or multi-hop transit	Point-to-multipoint demand
Peer	Peer	Single-hop transit	Point-to-multipoint demand
Backbone	Backbone	Backbone traffic	Omit flow
Backbone	Peer	Outbound or multi-hop transit	Identify possible ingress link(s) Omit flow or compute demand

Table 2: Flow classification based on input and output links

dicates which customer generated the traffic. We can match the source address with a customer prefix and, in turn, match this prefix with a set of possible access links that could have generated the traffic. (Note that we must assume that the source address correctly identifies the sender of the traffic. Although this is typically the case, a small fraction of the packets may have spoofed source addresses; that is, the sender may put a bogus source address in the IP packet header to evade detection while attacking the destination host.) The pseudocode in Figure 4 draws on a list, `src_access_prefix_set`, of the network addresses introducing traffic at access links. Each source prefix, `src_prefix`, can be associated with a set of ingress links based on the map `sendability()`. We also retain information about the input and output links that measured the flow. This information helps us infer which of these access link(s) could have originated the traffic, as discussed in Section 3.3.

Next, we discuss how our methodology applies to transit traffic that travels from one neighboring provider to another. Transit traffic falls into two categories — single-hop and multiple-hop, as shown in Figure 2. A single-hop transit flow enters and exits the ISP backbone at the same edge router, without traversing any backbone links; in this case, the flow is measured once, at this router. A multi-hop transit flow enters at one router, traverses one or more backbone links, and exits at another router; in this case, the flow is measured twice — at the ingress and egress points. The best place to capture a transit flow is at its ingress link, where we can apply the methodology of Section 2. To avoid double-counting the flow, we need to discard the flow records generated by multi-hop transit flows as they leave the network. This requires distinguishing outbound flows (introduced by an access link) from transit flows (introduced by a peering link). For a flow leaving the network, the algorithm in Figure 4 attempts to match the source IP address with customer prefix. For transit flows, this matching process would fail, and the associated flow record would be excluded.

3.2 Identifying Candidate Ingress Links

To associate each outbound flow with a set of ingress links, we must determine which source IP addresses could introduce traffic on each access link. On the surface, this problem seems equivalent to determining the set of *destination* prefixes associated with each access link. However, Internet routing is not symmetric. Traffic to and from a customer does not necessarily leave or enter the backbone on the same link. Hence, the forwarding table of the router terminating the access link may not have sufficient information to identify the source prefixes. For example, suppose a customer with two IP prefixes has two access links to the ISP. For load-balancing purposes, the customer may wish to receive traffic for one prefix on the first access link, and the rest of the traffic on the second access link. (This may involve configuring static routes for these prefixes in the edge routers that terminate the access links. Alternatively, the customer may announce these routes to the ISP using a routing protocol such as RIP or BGP.) In this example, each prefix would be reachable via a single access link. Yet, the customer could conceivably *send* traffic from either prefix via either access link. Hence, the router forwarding tables are not sufficient for identifying the source addresses that might generate traffic on an access link.

Fortunately, an ISP typically knows the IP addresses of its directly-connected customers. In fact, the customer may be assigned IP prefixes from a larger address block belonging to the ISP. In other situations, the customer already has its own block of IP addresses. As part of provisioning a new customer, the ISP configures the router that terminates the associated access link. Packet filters are specified to detect and remove traffic with bogus source IP addresses [16]. These packet filters indicate which sources might send traffic via a particular access link. The packet filters for each interface are specified in the router's configuration file. By parsing the router configuration files, we can determine which source prefixes to associate with each access link. From this information, we can determine the set of access links associated with each source prefix.

Using packet filters to identify source IP addresses is most appropriate for access links to directly-connected customers that do not connect to other service providers, or have downstream customers of their own. For customers that *do* connect to other service providers, or have downstream customers of their own, it is difficult to specify static packet filters for each source prefix on each possible ingress link. For example, when a neighboring domain acquires a new customer, traffic from these new source addresses could enter the ISP's backbone. Although the downstream provider typically performs packet filtering, these filters may not be known to the upstream ISP. This is a fundamental issue

```
For each flow: (input, output, source, dest, start, finish, bytes)
        dest_prefix = longest_prefix_match(dest, dest_prefix_set);
        egress_set = reachability(dest_prefix);
        if (input.type == peer) /* Inbound or (ingress) transit flow */
              compute volume[input, egress_set, input, output, time_bin] for each bin;
        else /* Outbound or (egress) transit flow */
              src_prefix = longest_prefix_match(source, src_access_prefix_set);
              if (src_prefix ≠ φ)
                    ingress_set = sendability(src_prefix);
                    compute volume[ingress_set, egress_set, input, output, time_bin] for each bin
Output for each aggregate: (ingress_set, egress_set, input, output, time_bin, volume)
```

Figure 4: Computing traffic demands based on measurements at peering links

that arises in the Internet due to the use of dynamic routing protocols based on destination reachability information. In these situations, our measurement methodology would argue for performing flow-level measurements at the ingress links, rather than depending on knowing the set of links where these sources could enter the ISP backbone.

3.3 Matching Flows with Routes

For inbound and transit flows, the algorithm in Figure 4 results in a point-to-multipoint demand. However, each outbound flow is associated with a *set* of ingress links, resulting in a *multipoint*-to-multipoint aggregate. Computing point-to-multipoint demands for outbound traffic requires an additional step to determine *which* access link initiated the traffic. Knowledge of intradomain routing can help resolve the ambiguity. For example, consider a source IP address that is associated with access links in New York and Chicago. Suppose the customer sends traffic to a destination with egress links in Washington, D.C., and Chicago, and the actual flow was observed leaving the backbone on a peering link in Chicago. If traffic from the New York access link would have been routed to the Washington, D.C., peering link, then the flow observed in Chicago must have originated from the access link in Chicago.

Determining whether an outbound flow could have entered the network at a given ingress link requires knowledge of the backbone topology and intradomain routing configuration at the time the flow was measured. For a given ingress link and set of egress links, we determine on which egress link the flow would exit the network. If this was not the egress link where the flow was observed, then this ingress link can be eliminated from consideration. In fact, knowing the path(s) from the ingress link to the egress link provides additional information. The flow was observed as it traveled from an input (backbone) link to an output (peering) link at the egress router. The path of the flow from the ingress link must include *both* of the links that observed the flow. Otherwise, this ingress link should be excluded from consideration. This process must be repeated for each of the possible ingress links, as shown in Figure 5.

This *disambiguation* process has three possible outcomes. First, a single ingress link could have generated the traffic. This is the ideal situation, resulting in a single point-to-multipoint demand. Second, more than one of the candidate ingress links could have generated the traffic. This would happen if multiple ingress links would send the traffic to the same egress router, and would enter this router on the same input link. (For example, a customer might have two access links in the same city, for load balancing or fault tolerance. These two access links would tend to select the same egress link.) Traffic from these access links may follow a similar path through the backbone, imparting load on some of the same links and routers. When multiple access links could have generated the traffic, the disambiguation process generates multiple demands, each with the an equal fraction of the traffic. Third, if none of the candidate ingress links could have generated the traffic, the disambiguation process has failed and the flow record is discarded. These "misses" are discussed in more detail in Section 5.2. In a similar manner, the routing model is useful for verifying the correctness of the inbound and transit demands.

The disambiguation process depends on knowing the possible paths from each ingress link to each egress link. We obtain this information from a routing model that captures the details of path selection in the ISP backbone. For each point-to-multipoint demand, the routing model determines the particular egress point as well as the path(s) through the network from the ingress link to the egress link. The set of egress links represents the outcome of the early steps of the BGP route-selection process. The routing model captures the last two steps — selection of the shortest-path egress link(s) based on the intradomain routing protocol and tie-breaking based on the router identifier. The main complexity stems from the modeling of intradomain routing. Our routing model [7] captures the details of OSPF routing in networks with multiple areas, including the splitting of traffic across multiple shortest-path routes. Snapshots of the router forwarding tables from the operational network have been used to verify the correctness of our routing software.

4. DATA SETS FROM AT&T BACKBONE

This section describes our experiences harvesting, parsing, and joining four large data sets, each collected from multiple locations in the AT&T IP Backbone. Monitoring the peering links produces, on average, one byte of measurement data for every 138 bytes of data traffic. We describe how we join these flow-level measurements with information from router configuration files, router forwarding tables, and SNMP data to compute the traffic demands. Then, we discuss how we addressed several practical constraints in processing the large set of flow-level measurements.

```
For each aggregate: (ingress_set, egress_set, input, output, time_bin, volume)
    For each a in ingress_set
        route = Route (a, egress_set);
        if (route does not use input and output links)
            remove a from ingress_set;
    if (ingress_set ≠ φ)
        for each a in ingress_set
            dvolume[a, egress_set, time_bin] += volume/size_of(ingress_set);
    else
        count as a miss;
Output for each demand: (a, egress_set, time_bin, dvolume)
```

Figure 5: Disambiguating the set of ingress links based on routing information

4.1 Data Sets

The computation of the traffic demands draws on several different data sets, as summarized in Table 3. Router configuration files reflect the configuration of a router as part of the IP network. The file specifies the configuration of the router hardware, the partitioning of resources (e.g., buffers and link capacity), the routing protocols (e.g., static routes, OSPF, and BGP), and the packet-forwarding policies. A *global* view of the network topology and configuration can be constructed by joining information across the configuration files of the various routers in the ISP's backbone [17]. This enables us to identify all of the routers and links, and their connectivity. In addition, the joined configuration files enable us to determine the type of each link (access, peering, or backbone), as well as the packet filters associated with each access link. This information is important for aggregating the flow-level measurements. Finally, the configuration files indicate the link capacities, as well as the OSPF weight and area for each backbone link, which are necessary input for the routing model.

Each router has a forwarding table that identifies the IP address(es) and name(s) of the next-hop interface(s) for each destination prefix (e.g., "135.207.0.0/16 12.126.223.194 Serial2/0/0:26"). We use the forwarding tables to associate each destination prefix with a set of egress links. The name of the next-hop interface is joined with the name of the corresponding egress link from the router configuration files. Joining this information produces the list, `dest_prefix_set`, of destination prefixes and the map, `reachability()`, of each destination prefix to a set of egress links. In addition, the forwarding tables are used to verify the correctness of the routing model. Having a snapshot of the forwarding tables close together in time enables us to determine how each router forwarded traffic toward each destination prefix. In particular, the forwarding tables enable us to identify which subset of the backbone links would carry traffic destined to a particular prefix. These paths were checked against the routes computed by our routing model.

The flow-level measurements were collected by enabling Netflow [14] on each of routers that terminate peering links. Each router exports the measurements as UDP packets in groups of one to thirty flow records. These UDP packets are sent to a collection server. Each flow record corresponds to a collection of packets that match in their key IP and TCP/UDP header fields and were routed in the same manner (i.e., same input and output link, and same forwarding-table entry). The record includes the packet header and routing information, as well as the time (i.e., start and finish time in seconds) and size (i.e., number of bytes and packets) of the flow. Our analysis focuses on the source and destination IP addresses, the input and output links, the start and finish time, and the number of bytes. The other fields in the Netflow records could be used to compute separate traffic demands for different subsets of the traffic.

Processing a Netflow record requires associating the input and output link that observed the flow with the corresponding links in our model of the network topology. However, the Netflow record identifies each link in terms of an integer SNMP index, whereas the forwarding tables and router configuration files reference a link by its name and IP address! The SNMP index is an integer value that uniquely identifies each interface in the router. The index does not change unless the interface is moved or another interface is installed in the same router. However, this identifier is not available in the router configuration files or the router forwarding tables. Periodic polling of the router's SNMP variables allows us to determine the IP address and name associated with each SNMP index. SNMP data also includes statistics about the number of bytes carried on each link on a five-minute time scale. We used these statistics as an independent verification of the loads computed by aggregating the Netflow data.

4.2 Practical Constraints

The processing of the Netflow data introduced several practical challenges, which we briefly summarize:

Router clock synchronization: Each router synchronizes the clock of its route processor to a central server using the Network Time Protocol (NTP). However, the clocks on individual interface cards are not always synchronized, due to a historical bug in Cisco's Internet operating system. We addressed this problem by aligning the Netflow records collected on the interface cards with records from the route processor. All timestamps within the Netflow data are relative to a base clock. For each router, it suffices to adjust the base clock of the records originating each link with those originated by the route processor. In post-processing the Netflow data, we realign the base clock of each interface to match with the most recent record from the route processor. The route processor receives a relatively small number of data packets (such as routing protocol traffic and packets with expired TTL values), compared to the interface cards. Still, flow records are generated by the route processor quite

Dataset	Location	Key Fields
Configuration files	Router	Per interface: IP address, name, type (peer/customer/core), and capacity Per core interface: OSPF configuration (weight and area) Per customer interface: network addresses for packet filters
Forwarding tables	Router	Per interface: set of network addresses (IP address and prefix length)
Netflow logs	Peer link	Per flow: input and output interfaces (SNMP index), src and dest IP addresses, start and finish times, and number of bytes and packets
SNMP data	Interface	Per interface: SNMP index, IP address, name, and utilization

Table 3: Datasets and key fields used in computing and validating the traffic demands

frequently on a busy router; during a sampled 24-hour period, the interarrival time of flow records from the route processor has a mean of 0.32 seconds and a maximum of 91.4 seconds. Hence, the uncertainty introduced by correcting for timestamp problems is very small compared to the time scale of the subsequent aggregation.

Lost measurement data: Netflow records are transmitted to the collection server using UDP. As such, the measurement data is not delivered reliably. Limited bandwidth to our collection server resulted in loss of up to 90% of the UDP packets during heavy load periods. Nearly all of these packets were lost on the link connecting to our measurement server, dwarfing the losses experienced by the Netflow data in the rest of the backbone. The traffic on the link to the collection server consists mainly of the UDP Netflow data. The traffic dynamics typically introduced by TCP congestion control do not arise in this context. The dominance of UDP traffic, coupled with the limited bandwidth, results in a nearly random sampling of the measurement records. To test our hypothesis of random packet loss, we analyzed the loss patterns based on the sequence numbers of the Netflow records that arrived successfully. Detailed analysis of the loss characteristics showed that the distribution of the number of consecutive losses is consistent with assuming independent loss. Based of the assumption of random independent loss, we apply a correction factor to the received flow records to account for lost measurement data, taking in to account the fact that the loss rate varies across time and (potentially) across routers. First, we determine the loss rate during each (ten-minute) time interval for each router (and each "engine" that exports Netflow data), based on sequence numbers in the stream of flow records. Then, we assume that flows that are observed are statistically similar to other flows that ended during the same time period. We apply a correction factor based on the loss probability during that time period, corresponding to the time that the flow record was exported to the collection server. This correction factor is applied to all bytes within the flow. To verify our approach, and to select the ten-minute interval for applying the loss correction, we compared our corrected Netflow data against independent link-load statistics from SNMP. The experiments showed a good match.

Data sets from multiple time periods: Computing the traffic demands required joining four independent data sets, each collected from multiple locations in the network at different times during the day. This introduces significant challenges in joining and analyzing the data. These data sets also provide a unique opportunity to quantify the effects of routing instability on an operational network. Table 4 summarizes the data sets used in the experiments in the remainder of the section. We focus on four days in November 1999. November 3 and 4 are a Wednesday and a Thursday, respectively. November 11 and 12 are a Thursday and a Friday, respectively. These flow measurements enable us to compare traffic on two consecutive days and two consecutive weeks. Daily configuration files are used to generate the topology model. Each experiment uses the most recent forwarding tables and SNMP data available. The SNMP data is the least sensitive, since the SNMP index for each link does not change unless the network undergoes a structural change; these changes occur infrequently on the routers that terminate peering links. Independent verification assured that this did not occur during the periods of our data collection.

Netflow Logs (all day)	Configuration Files (8pm GMT)	Forwarding Tables (4pm GMT)	SNMP Indices (8pm GMT)
11/03/1999	11/03/1999	11/04/1999	11/01/1999
11/04/1999	11/04/1999	11/04/1999	11/01/1999
11/11/1999	11/11/1999	11/14/1999	11/08/1999
11/12/1999	11/12/1999	11/14/1999	11/08/1999

Table 4: Collection times for each data set

5. EXPERIMENTAL RESULTS

In this section, we present the results from aggregating and validating the flow-level measurements collected at the peering links. Then, we discuss the application of the routing model to disambiguate and validate the demands. In both cases, we discuss the implications of the ambiguity of the ingress links for outbound flows, fluctuations in egress reachability information, and inconsistencies across the various data sets. Then, we present our initial results from analyzing the spatial and temporal properties of the traffic demands.

5.1 Netflow Aggregation

The first phase of computing the traffic demands applies the methodology in Figure 4 to the Netflow data. Typically, more than 98% of the bytes observed at the peering links can be mapped to a point-to-multipoint (inbound/transit flows) or multipoint-to-multipoint aggregate (outbound flows), as shown in the "miss" column. These mismatches stem from the three key steps in Figure 4 — (i) identifying the input and output links that observed the flow, (ii) associating the destination IP address with a set of egress links, and (iii) associating the source IP address (of an outbound flow) with a set of ingress links.

A significant fraction of the misses can be explained by step (i), as shown in the "Out 0" and "Loop" columns. Each Netflow record logs the SNMP indices for the input and output links that observed the flow. In our data sets, every Netflow record had valid input and output fields that matched with our SNMP data. Approximately 0.5–0.7% of the bytes in the network had a output link of 0, meaning that the data was delivered to the route processor. Further inspection of the raw Netflow data reveals that about 0.4% of these bytes stem from traffic actually *destined* to the router. That is, for these flows, the destination IP address is the router's loopback IP address. This traffic comes from ICMP (Internet Control Message Protocol) messages, telnet, and SNMP polling for routine operational tasks. The remaining flows with an output link of 0 correspond to unroutable traffic. For example, a packet with an expired TTL (time-to-live) field, as generated by traceroute, would fall in this category. These unroutable packets are directed to the route processor for further handling. The second category of misses ("loop") arises when a flow enters and leaves the router on the same link. These transient forwarding loops account for an extremely small portion of the total bytes in the network (e.g., less than 0.03%).

The remainder of the "misses" occur in trying to determine the ingress and egress links for a flow. As expected, the matching process is most successful when measurements are collected at the ingress link, as seen in the "Inbound (Egress)" column. Still, a small number of mismatches arise in associating a flow's destination IP address with the egress links. That is, the destination IP address does not match with any of the prefixes observed in the snapshots of the router forwarding tables. These mismatches stem from transient changes in reachability information. For example, the destination may have been temporarily unreachable when the forwarding tables were collected. Or, perhaps the destination's egress point moved from one router to another, with neither snapshot showing a route to that destination. These kinds of fluctuations in reachability information are unavoidable in dynamic routing protocols. Fortunately, they did not have a significant affect on our ability to match the flows. To verify this hypothesis, we identified the top few destinations, responsible for the majority of the missed traffic, and found that these destinations were represented in the forwarding tables collected on subsequent days.

Identifying the egress links for outbound traffic has similar challenges, as seen by the statistics in the "Outbound (Egress)" column[1]. Yet, the most challenging part of aggregating the outbound flows arises in matching the *source* IP address with one or more access links, as shown in the "Outbound (Ingress)" column in Table 5. The aggregation process identifies at least one candidate ingress link for over 99.3% of the outbound bytes. However, matching the source IP address with a set of access links does not necessarily imply that one of these links actually generated the traffic. This check does not occur until the later stage of disambiguating the set of ingress links based on the routing model.

Overall, the results are consistent across the four experiments. However, the November 11 data has a higher proportion of mismatched bytes (4.7% vs. less than 2% for the other days). These extra misses arise in two categories — egress links for inbound traffic and ingress links for outbound traffic. Both errors relate to *customer* IP addresses. Upon further inspection, most of these misses stem from a single access link. The access link was upgraded some time after 8pm GMT, when the configuration files were collected. Hence, our copy of the configuration file of the router terminating the new link had the name and IP address of the *old* link. The forwarding table was collected several days later on November 14. In this table, the next-hop entries pointing to the new link are used to direct traffic to a collection of customer prefixes. However, in our automated joining of the data sets, we did not associate these customer prefixes with the old link specified in the configuration file. Hence, these customer prefixes were unknown during the aggregation of the Netflow data. Manually associating the prefixes with the old link, and repeating the experiment, reduced the number of egress misses (for outbound traffic) from 1.019% of the bytes to 0.468% and the number of ingress misses (for inbound traffic) from 2.939% of the bytes to 0.595%, consistent with results from other days.

5.2 Route Disambiguation

The second phase of computing demands applies the methodology in Figure 5 to match the aggregated traffic with routes. The disambiguation process is primarily used to infer the ingress link associated with each outbound demand. However, we find that the procedure also provides a useful consistency check on our initial processing of the flow-level data, and aids in studying the dynamics of the other data sets involved in the computation.

Our methodology is most effective for inbound and transit traffic, where measurements are available at the ingress links. In this case, the techniques in Section 3 produce a point-to-multipoint demand. Still, our experimental results from aggregating the Netflow data are not sufficient to show that we associated each traffic demand with the *correct* ingress link and set of egress links. The routing model provides an important consistency check by verifying that traffic from the ingress link to the set of egress links would actually traverse the links that measured the flow. The results of this check are shown in the "Inbound (miss)" column in Table 6, which shows that the routing test failed for less than 1% of all bytes entering the network at the peering links. This is very promising, though not perfect. Not all changes in the set of egress links would result in a change in how the observed traffic would exit the network. Still, an error rate of less than 1% suggests that our methodology is effective for handling traffic measured on ingress links.

We expect our approach to be less effective for outbound traffic, due to unavoidable ambiguity about the ingress links. In addition, the peering links are vulnerable to fluctuations in reachability information due to the dynamics of interdomain routing between neighboring ISPs. In a small number of cases, the forwarding tables at the peering links are in-

[1]The statistics for egress matching for outbound traffic are slightly lower than the corresponding statistics for inbound traffic. This arises from the operation of our aggregation software, which does not try to identify a set of egress links for an outbound flow unless one or more possible ingress links could be identified.

Run	Miss	Out 0	Loop	Inbound Egress	Outbound Ingress	Outbound Egress
11/03/99	1.720	0.691	0.006	0.442	0.451	0.127
11/04/99	1.630	0.749	0.010	0.379	0.452	0.039
11/11/99	4.720	0.540	0.009	1.019	2.939	0.210
11/12/99	1.778	0.563	0.022	0.475	0.642	0.074

Table 5: Percent of bytes unmatched in aggregating the Netflow data

consistent with the observed flows. That is, the forwarding table suggests that the router that observed the flow would have forwarded the traffic to a different link! These inconsistencies are flagged in the "baddest" column, and account for 1.0–3.5% of the bytes leaving the network on peering links. We observed the fewest errors on the November 4th data set, where we had forwarding tables and Netflow data from the same day. To verify our hypothesis that these inconsistencies stem from fluctuations in reachability information, we inspected a single day of flow data in greater detail. The source-destination pairs in the affected demands typically moved in and out of the "baddest" category across the day, suggesting that the forwarding table entries in the operational router were changing across time.

Computing demands for outbound flows is also complicated by uncertainty about which ingress link generated the traffic. Approximately 35–45% of the bytes leaving the network at the peering links were associated with multiple candidate ingress links. Some of this ambiguity could be resolved by the routing model. In fact, between one-third and one-fourth of these bytes could be resolved to a single ingress link after applying the disambiguation process outlined in Figure 5, as seen by the "perfect" column in Table 6. With further inspection, we see that some of these demands came from customers with access links on the east and west coasts. Traffic from these access links are likely to exit the network on different egress links. However, complications arise when a customer has more than one link in the same region of the country. For example, a single customer may have two access links terminating on different routers in the same city. This offers protection from the failure of a single router without forcing customer traffic to enter the network in a different city.

The routing model typically cannot disambiguate traffic from two access links from the same customer in the same city. Traffic from these two access links would typically exit the network on the same peering links for most (if not all) destination prefixes, and often follows a similar path through the ISP backbone. This occurs when the intradomain path costs to and from these two access links are very similar, if not the same. In this case, successfully disambiguating the two access links is not very important! Associating the traffic with the wrong access link does not have much influence on the flow of traffic in the backbone. In fact, assuming that the traffic was split evenly across the two links, as shown in Figure 5, is quite reasonable. Customers often configure their routers to perform load balancing across their multiple access links, resulting in a nearly even division of the traffic on the two links. Overall, disambiguation to a single city accounts for 13–19% of the outbound bytes, as seen in the "one city" column in Table 6. In total, about two-thirds of the ambiguous ingress sets were resolved to one or more access links in a single city ("perfect" or "one city").

However, some customers are multi-homed to routers in different cities, and may even generate traffic from a single block of IP addresses on both links. Such multi-homing is useful for additional fault-tolerance, or because the customer has sites in multiple geographic locations. When the homing locations are relatively close to each other, the routing model may not be able to disambiguate the set of ingress links. This is a situation where additional measurement at the ingress links would be useful. Still, overall, the disambiguation process is quite successful. Only 2.5–4% of the bytes could not be associated with (one or more) point-to-multipoint demands. These results are shown in the "Outbound (miss)" column in Table 6, which includes the contribution of the "Outbound (baddest)" statistics. Based on these results, the rest of this section focuses on analyzing the statistical properties of the observed demands.

5.3 Traffic Analysis

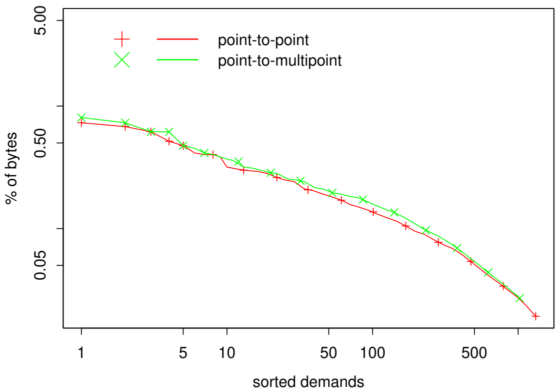
In this section, we present initial results of a statistical analysis of the traffic demands. We focus on: (i) statistical characteristics of inbound and outbound traffic, at different levels of aggregation (point-to-multipoint demands, or corresponding point-to-point loads on edge routers), (ii) time-of-day variations, and (iii) variations at coinciding time intervals within the two weeks.

A network with many access and peering links has a large number of point-to-multipoint demands. However, a very small proportion of these demands contribute the majority of the traffic. In Figure 6, we rank point-to-multipoint demands (or point-to-point loads) from largest to smallest, and plot the percentage of the total traffic attributable to each. These plots are restricted to the upper tail of the distribution, accounting for 80% of the total traffic. We refer to the particular demands (or loads) in this upper tail as the *heavy hitters*. We found the plots to be nearly linear on the log-log scale, as is characteristic of a Zipf-like distribution, where the contribution of the k-th most popular item varies as $1/k^a$, for some a. We found this general pattern to hold for all data sets and at all levels of temporal and spatial aggregation. Figure 6(b) shows greater concentration of traffic over fewer heavy hitters in outbound versus inbound traffic.

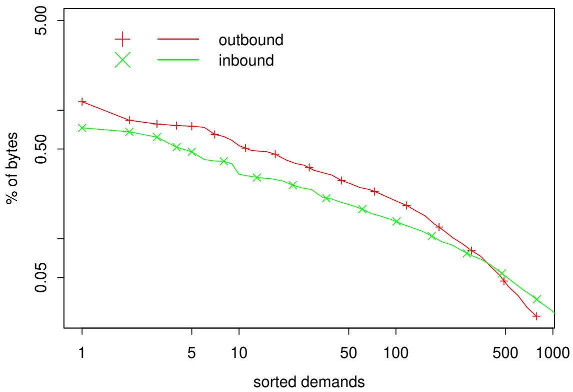
Similar trends have been seen in earlier studies that consider the load on individual links or servers. For example, link-level traces show that the distribution of traffic at the prefix and AS level follows Zipf's law [18]. Studies of the World Wide Web have shown that a small fraction of the

Run	Inbound	Outbound					
			Disambiguation				
	miss	**mult. ingress**	**perfect**	**one city**	**mult. cities**	**miss**	**baddest**
11/03/99	0.83	37.89	9.49	16.27	11.25	3.55	1.51
11/04/99	0.71	38.49	9.78	16.42	11.77	2.45	0.96
11/11/99	0.14	39.73	11.91	13.75	12.47	4.39	3.47
11/12/99	0.98	44.07	11.43	18.95	11.96	4.01	3.12

Table 6: Disambiguation statistics



(a) Inbound traffic



(b) Inbound and Outbound traffic

Figure 6: Percent bytes attributed to top ranked traffic volumes, listed in decreasing order

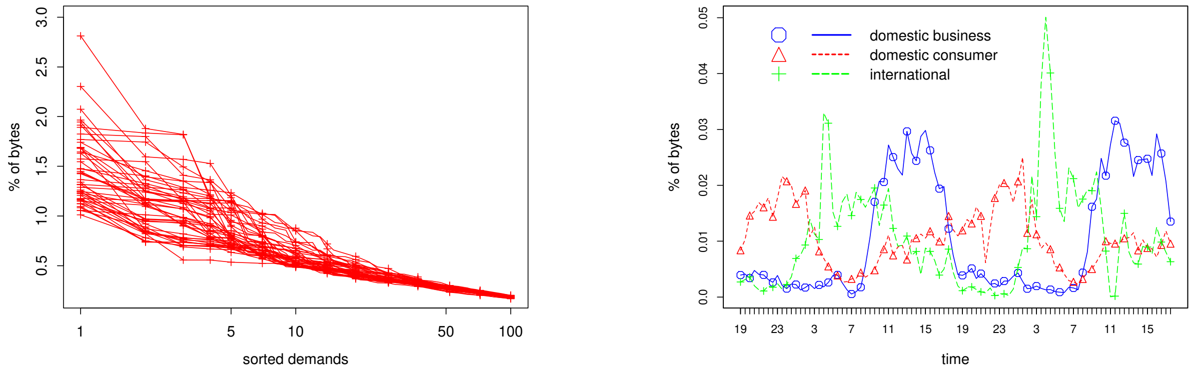
requests, resources, and servers are responsible for the bulk of the traffic [19, 20]. The small number of heavy hitters has important implications for traffic engineering. On the positive side, since the leading heavy hitters account for so much traffic, care in routing just these demands should provide most of the benefit. In addition, when measuring traffic demands, relatively coarse statistical subsampling should suffice. On the negative side, if the internal routing configuration concentrates heavy-hitter traffic on common links, through error or inherent fluctuations in the identities of the heavy hitters, the negative impact on performance could be severe. In general, the concentration of demand on a few sources opens up the possibility of large-scale network variability if these sources change behavior.

How do the top demands vary by time of day? In Figure 7(a) we plot the percentage of bytes attributable to the top 500 point-to-multipoint outbound demands over a half hour, ranked in decreasing order. The graph includes 48 curves, to cover the day. (The identities of the top 500 may change from half hour to half hour.) There is significant variation in demand sizes at the highest ranks. We have looked at the top demands more closely, and found that they may exhibit quite different time-of-day patterns. This is demonstrated in Figure 7(b), where we have plotted the time of day variation for three heavy demands entering the network in San Francisco. We informally label these three demands as domestic consumer, domestic business, and international, because they correspond of the usage patterns of consumer and business domestic dial traffic, with international traffic roughly similar to a time-shifted business pattern.

To investigate change among the heavy hitters more systematically, we consider grouping the demands into quantiles (e.g., the first group corresponds to the highest ranked demands together accounting for 5% of the traffic, the second group to the remaining highest ranked demands accounting for the next 5% of the traffic, and so forth). How do the groupings change with time-of-day? Figure 8 provides a two-dimensional histogram, where the gray-scale of the i,j-th block indicates the proportion of the demands in quantile i in one time period that move to quantile j in another time period, h hours later. In Figure 8(a), the lag h is a half hour, and in Figure 8(b) it is 24 hours. The top demands (top right corner) show the least variation. In both cases, the concentration of mass along the diagonal indicates little quantile jumping. Demands in a given quantile appear in the same quantile or a nearby quantile 24 hours later. Varying h over the 24 hour interval we found the mass along the diagonal first tends to diffuse and the band widens up to $h = 12$ hours, whereupon the mass then tends to concentrate and the band narrows up to $h = 24$ hours. These preliminary results suggest a certain amount of stability in the identity of the top demands across time.

6. CONCLUSION

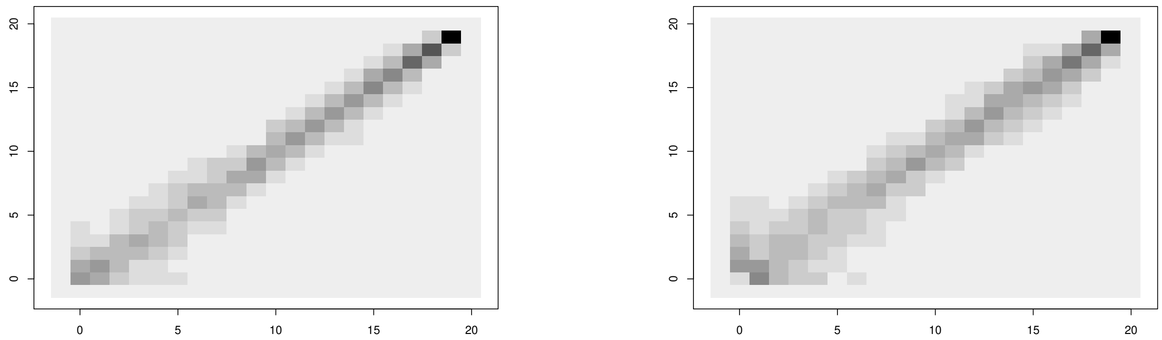
Engineering a large ISP backbone introduces fundamental challenges that stem from the dynamic nature of user be-



(a) 48 thirty-minute time intervals (Nov 4)

(b) Three demands across a two-day period (Nov 3-4)

Figure 7: Time-of-day effects in the measured traffic demands for November 3 and 4



(a) Demands at 1pm and 1:30pm (Nov 3)

(b) Demands on Nov 3 and Nov 4 (1pm)

Figure 8: Stability of the measured traffic demands across time (two-dimensional histograms)

havior and reachability information, and the lack of end-to-end control over the path of a flow. Yet, careful engineering of the network is important, since the routing configuration and backbone topology have significant implications on user performance and resource efficiency. In this paper, we propose a model of traffic demands that captures (i) the volume of data, (ii) the entry point into the ISP network, and (iii) destination reachability information. This simple abstraction facilitates a wide range of traffic engineering applications, such as performance debugging, route optimization, and capacity planning. We also present a methodology for populating the demand model from flow-level measurements and interdomain routing information, and apply our approach to a large, operational ISP network. Our analysis of the measured demands reveals significant variations in demand sizes and popularities by time-of-day, but a certain amount of stability between consecutive days.

In populating our demand model, we faced three main challenges:

- **Working with four different datasets:** Organizing access to all data sets during the same time period is difficult. Insuring their completeness and consistency posed both operational and computational challenges. Last, determining how best to join the datasets forced us to address the questions of subsampling and temporal uncertainties between the datasets.
- **Ambiguity of ingress points:** For a flow measured only at its egress link, determining the ingress link is challenging. This difficulty arises because hop-by-hop routing (based on the destination IP address) implies that downstream routers do not necessarily have (or need!) information about how packets entered the domain. In addition, the increasing decentralization of the Internet makes it difficult for any one ISP to know the source IP addresses of downstream domains.
- **Dynamics of the egress points:** Policy changes in one domain can have unforeseen implications on the

reachability information seen by other ISPs. We see evidence of this in the churn in the forwarding tables across time, and the resulting inconsistencies between the data sets. This complicated the identification the set of egress links for traffic demands.

Despite these challenges, our approach for populating the demand model performs quite well. Inconsistencies that arose could be explained by natural network dynamics.

Motivated by our experimental results, our ongoing work focuses on two main topics — real-time computation of traffic demands and detailed analysis of the measured traffic. First, we want to determine the traffic demands in the operational network on ever smaller time scales by having more immediate access to configuration, routing, and usage data. Thus far, we have considered offline processing of the multiple data sets. Moving forward, we plan to investigate monitoring architectures that would enable real-time processing of the two key, dynamic data sets — the flow-level statistics and the reachability information — based on the current state of the network. Second, we plan to devote more attention to the analysis our measured traffic demands. The network-wide view of configuration and usage data in an ISP backbone provides a rich opportunity to characterize the fluctuations in IP traffic demands. Our initial analysis suggests that these demands have interesting spatial and temporal properties with significant implications for Internet traffic engineering. Further statistical analysis of this data would lend insight into new techniques for the design and operation of IP backbone networks.

Acknowledgments

Numerous colleagues in AT&T Labs and AT&T IP Services have provided valuable feedback on this work. We would like to thank Han Nguyen for his role in formulating and supporting the NetScope project that motivated this work, and Jay Borkenhagen, Michael Langdon, Anthony Longhitano, Muayyad Al-Chalabi, and Joel Gottlieb for helping us access and understand the network measurement and configuration data. Finally, we thank Walter Willinger and Michael Merritt for valuable suggestions on how to improve the presentation of the material.

7. REFERENCES

[1] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP performance metrics." Request for Comments 2330, May 1998.

[2] W. Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Addison-Wesley, 1999.

[3] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network Magazine*, vol. 11, pp. 10–23, November/December 1997.

[4] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "A framework for Internet traffic engineering." Internet Draft draft-ietf-tewg-framework-01.txt, May 2000.

[5] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communication Magazine*, pp. 42–47, December 1999.

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

[7] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, March 2000.

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

[9] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe, "A flexible model for resource management in virtual private networks," in *Proc. ACM SIGCOMM*, September 1999.

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

[11] C. Labovitz, R. Malan, and F. Jahanian, "Internet routing stability," *IEEE/ACM Trans. Networking*, vol. 6, pp. 515–558, October 1998.

[12] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of Internet stability and wide-area network failures," in *Proc. International Symposium on Fault-Tolerant Computing*, June 1999.

[13] B. Krishnamurthy and J. Wang, "On network-aware clustering of Web clients," in *Proc. ACM SIGCOMM*, August/September 2000.

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

[15] S. Handelman, S. Stibler, N. Brownlee, and G. Ruth, "RTFM: New attributes for traffic flow measurement." Request for Comments 2724, October 1999.

[16] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing." Request for Comments 2267, January 1998.

[17] A. Feldmann and J. Rexford, "IP network configuration for traffic engineering," Tech. Rep. 000526-02, AT&T Labs – Research, May 2000.

[18] W. Fang and L. Peterson, "Inter-AS traffic patterns and their implications," in *Proc. IEEE Global Internet Symposium*, December 1999.

[19] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Trans. Networking*, vol. 5, pp. 835–846, December 1997.

[20] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, pp. 126–134, March 1999.