Passive Traffic Measurement for IP Operations

Matthias Grossglauser Jennifer Rexford
AT&T Labs - Research
180 Park Avenue
Florham Park NJ 07932, USA
{mgross,jrex}@research.att.com

1 Introduction

Managing a large communication network is a complex task handled by a group of human operators. These operators track the state of the network to detect equipment failures and shifts in traffic load. After detecting and troubleshooting a problem, the operators may change the configuration of the equipment to improve the utilization of network resources and the performance experienced by end users. The ability to detect, diagnose, and fix problems depends on the information available from the underlying network. These tasks are easier if the equipment provides reliable and predictable service and retains detailed information about the resources consumed by the ongoing data transfers. In diagnosing potential performance problems, operators benefit from having an accurate and timely view of the flow of traffic across the network. As part of fixing a problem, operators need to know in advance how changes in configuration might affect the distribution of traffic and the performance experienced by end users.

The goals of network operators are in direct conflict with the design of the Internet Protocol (IP). The primary goal in the design of the ARPAnet in the late 1960s was to make efficient use of existing data networks through the use of packet switching [1]. End hosts divide data into packets that flow through the network independently. In forwarding packets toward their destinations, the network routers do not retain information about ongoing transfers and do not provide fine-grain support for performance guarantees. As a result, packets may be corrupted, lost, delayed, or delivered out of order. This complicates the efforts of network operators to provide predictable communication

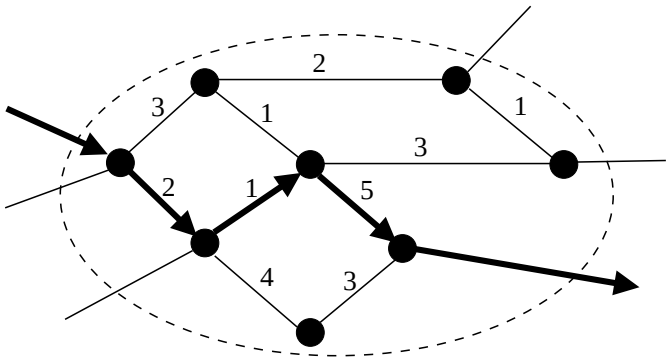


Figure 1: Shortest path routing within a domain based on OSPF/IS-IS link weights

performance for their customers. Rather than having complexity inside the network, the end hosts have the responsibility for the reliable, ordered delivery of data between applications. Implemented on end hosts, the Transmission Control Protocol (TCP) plays an crucial role in providing these services and adapting to network congestion. Inside the network, the routers implement routing protocols that adapt to equipment failures by computing new paths for forwarding IP packets. These automatic and distributed reactions to congestion and failures make it difficult for network operators to detect, diagnose, and fix potential problems.

The commercial realities of today's Internet introduce additional challenges for network operators. The Internet consists of nearly 12,000 Autonomous Systems (ASes), where each AS is a collection of routers and links managed by an single institution, such as a company, university, or Internet Service Provider (ISP). Within an AS, the routers communicate via an intradomain routing protocol such as OSPF or IS-IS. The operators configure each unidirectional link with an integer weight, and the routers distribute these weights and use them to compute shortest paths through the network, as shown in Figure 1. Neighboring ASes use the Border Gateway Protocol (BGP) to exchange information about how to reach destinations throughout the Internet, without sharing the details of their network topologies and routing policies. Much of the traffic flowing through the Internet must traverse multiple ASes. As such, the operators of any individual AS do not have end-to-end control of the distribution of traffic. The lack of end-to-end control makes it difficult for operators to detect and diagnose performance problems, and predict the impact of changes in network configuration.

Traffic measurement plays a crucial role in providing operators with a detailed view of the state of

their networks. Operators detect congestion based on periodic summaries of traffic load and packet loss on individual links. By sending active probes between pairs of points in the network, operators can identify parts of the network that exhibit high delay or loss, or detect routing anomalies such as forwarding loops. Measurement also helps identify which traffic is responsible for network congestion, and whether the additional packets stem from a denial-of-service attack, a popular event or service on the Web, or a routing change caused by an equipment failure or a reconfiguration in a different AS. Finally, measurement can assist operators in evaluating possible changes to the network configuration. Operators may alleviate congestion by changing the parameters that control the allocation of resources in the network. For example, changing one or more OSPF/IS-IS weights would trigger the selection of new paths, which could result in a more efficient flow of traffic through the network. Operators may also change the configuration of the buffer management or link scheduling parameters, or install a packet filter to discard malicious traffic.

Diagnosing performance problems and evaluating the effectiveness of corrective actions depends on having accurate information about the flow of traffic through the network. In Section 2, we present three models that provide a network-wide view of the traffic in an AS. We define the path, traffic, and demand matrices and describe how operators can use these models to diagnose and fix performance problems in their networks. Populating these models requires collecting extensive traffic measurements from multiple locations in an operational network. However, most IP networking equipment does not include the necessary support for collecting this data. In Section 3 we survey existing techniques for collecting traffic statistics in IP networks. We present a brief overview of SNMP/RMON, packet monitoring, and flow-level measurement. Constructing a network-wide view of the traffic requires new measurement techniques or the careful combination of several types of measurement data from multiple locations in the network. In Section 4, we describe recent research work on computing path, traffic, and demand matrices. We conclude the paper in Section 5 with a summary and a discussion of future research directions.

2 Network-Wide Traffic Models

Network-wide representations of the traffic are necessary to drive network-wide control actions, such as routing changes, traffic classification and filtering, and capacity planning. In this section, we

introduce three canonical spatial representations of the traffic— the *path matrix*, the *traffic matrix*, and the *demand matrix*—and discuss the application of these models in managing an IP network. These representations cover a spectrum from *descriptive*, i.e., capturing precisely the current state of the network and the current flow of traffic, to *predictive*, i.e., enabling the prediction of the state of the network after taking hypothetical control actions.

2.1 Path, Traffic, and Demand Matrices

To motivate these three canonical representations, consider how a network operator performed traffic engineering if he had complete, end-to-end control over the Internet. There are two natural representations of the traffic that would help the operator detect and diagnose problems and evaluate potential control actions. The first representation is the *path matrix*, which specifies a data volume $V(P)$ for every path P between every source host S and destination host D[1], as shown in Figure 2(a). The path matrix captures the current state and behavior of the network, by providing a detailed description of the spatial flow of traffic through the network. The second representation is the *traffic matrix*, which specifies a data volume $V(S, D)$ per source-destination pair (S, D), as shown in Figure 2(b). The traffic matrix captures the *load* on the network, which ideally is *invariant* to the state and behavior of the network. As such, in combination with a network model, the traffic matrix allows the operator to predict the flow of traffic through the network after hypothetical changes have been made. The operator can decide on a control action that will have the desired impact on the traffic flow.

In practice, a network operator only has control over a small portion, or Autonomous System (AS), of the Internet. Capturing the traffic in a single AS suggests a slightly richer set of models. The first two representations are directly inspired by the path and traffic matrix, with the caveat that the path matrix only covers the set of paths within the AS and the traffic matrix is expressed between ingress-egress nodes rather than source-destination hosts. These two representations would suffice in a situation where interdomain and intradomain routing were completely decoupled, i.e., if local control actions could have no impact on where traffic enters and leaves the AS. Under this assumption, managing a single AS is conceptually similar to managing the Internet end-to-end,

[1]Strictly speaking, given that packets can be dropped at routers inside the network, the path matrix should also include every path from a source S to an internal node (router) R.

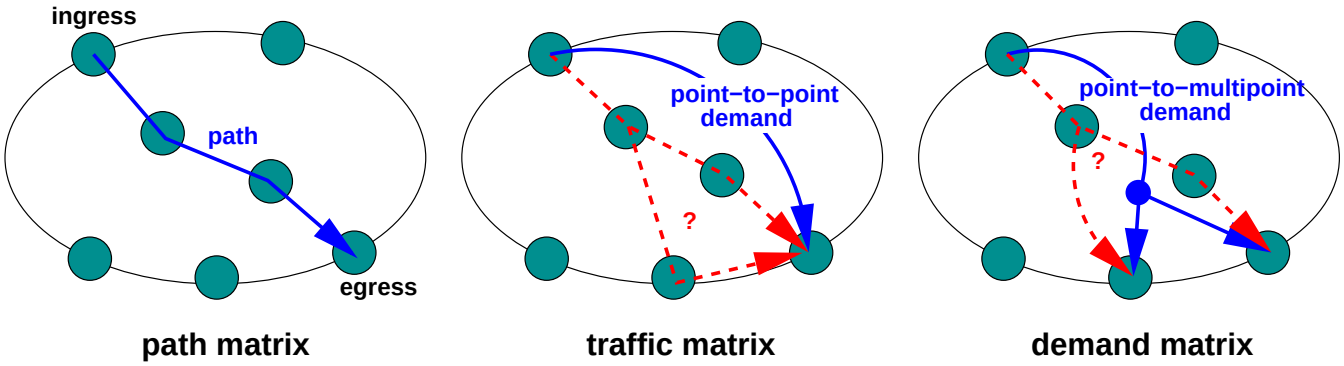


Figure 2: Network-wide traffic models: (a) path matrix (volume for every path between every ingress-egress pair); (b) traffic matrix (volume for every ingress-egress pair); (c) demand matrix (volume from every ingress to every set of egress points).

albeit at a smaller scale. However, in reality, interdomain and intradomain routing interact with each other. Routing between ASes depends on the Border Gateway Protocol (BGP) [2], a path-vector protocol that computes routes at the AS level for blocks of destination IP addresses. BGP computes a sink tree spanning ASes all over the Internet for each prefix. Each block (or, *prefix*) consists of a 32-bit address and a mask length (e.g., 192.0.2.0/24 refers to the IP addresses ranging from 192.0.2.0 to 192.0.2.255).

An AS learns routes to various destination prefixes from neighboring domains. Each router in the AS selects a *best* path for each destination prefix based on the BGP advertisements and the local routing policies. The crucial point is that the BGP routing process may select a *set* of egress points for a given prefix, rather than a single egress point [3, 4]. The selection of a single egress point at any given router depends on the relative proximity to these egress points, as defined by the configuration of the *intra*domain routing protocol, such as OSPF or IS-IS. For example, suppose that AS A advertises a route to 192.0.2.0/24 to AS B in San Francisco and in New York. Then, routers "close" (in the OSPF/IS-IS sense) to San Francisco would select the egress point in San Francisco, and routers close to New York would select the egress point in New York, as shown in Figure 3. This implies that changing the intradomain routing parameters, such as OSPF/IS-IS weights, may affect which egress point is used to reach the destination prefix from a particular ingress point. To accurately predict the effect of such a control action, the load on the network domain has to be expressed as volume $V(S, \{D_1, \ldots, D_n\})$ per ingress node S and *set* of egress nodes

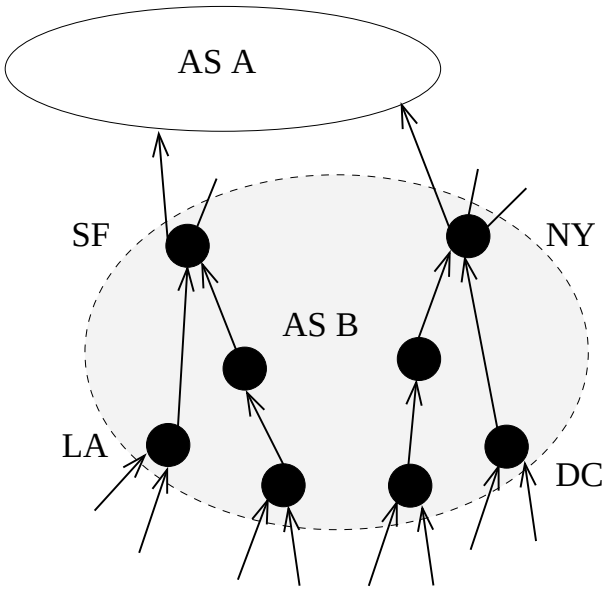


Figure 3: Forwarding traffic to the closest egress point

$\{D_1, \ldots, D_n\}$) [5]. We refer to this abstraction as the *demand matrix*, as shown in Figure 2(c)[2].

2.2 Network Operations Tasks

The path, traffic, and demand matrices provide a sound basis for detecting and diagnosing problems, and for "what-if" analysis to evaluate potential control actions. This is important for a variety of important tasks in managing an IP network.

Generating traffic reports for customers: A service provider typically generates basic traffic reports for customers, often as part of billing for access to the network. A large customer may connect to the provider via multiple access links in different parts of the country. The provider can use the traffic matrix to generate reports of the volume of traffic between each pair of access links for these customers. The path matrix can be used to demonstrate how traffic to/from the customer flows through the provider's backbone by focusing on the subset of paths that start or end at the customer's access link(s). The path matrix also allows the provider to determine if the customer's traffic traverses any congested links. This analysis allows the provider to identify the

[2]Alternatively, volume could be expressed as point to point, but over an extended topology that adds virtual, zero-weight, output links from the set of possible egress points to a virtual end node representing the set of destination prefixes sharing the same egress set.

customers that may be experiencing bad performance during a congestion event, or to demonstrate that a customer's performance problem arises in some other part of the end-to-end path, outside of the provider's network.

Diagnosing the cause of congestion: Network operators typically monitor aggregate statistics about traffic volume and packet loss on a per-link basis, as discussed later in Section 3.1. Although these measurements are sufficient to identify overloaded links, diagnosing the cause of the congestion requires fine-grained information about the flow of traffic into and out of the link. The path matrix allows the operator to determine the set of paths and their associated traffic intensities (and possibly other attributes). For example, a distributed denial-of-service (DDoS) attack would cause patterns that resemble a sink tree, with a large amount of traffic entering at multiple ingress points, and leaving at only one or a small number of egress points, probably towards a single destination address. On the other hand, congestion caused by a forwarding loop would manifest itself through the presence of cyclic paths that contain the overloaded link.

Tuning intradomain routing to alleviate congestion: Suppose that the network operators have identified a set of ingress-egress pairs responsible for congestion on one or more links in the backbone. The operators could examine the evolution of the traffic matrix over time to assess whether the heavy load for these ingress-egress pairs is a persistent (and perhaps worsening) problem and whether it arises at certain times of the day. The operators can fix the problem by reconfiguring the OSPF/IS-IS weights or MPLS label-switched paths to move some traffic off of the congested links. The traffic or demand matrix is a crucial part of evaluating potential configuration changes. For a proposed set of link weights W, say, the demand matrix allows the operator to predict the load on each link l as $\sum_{(S,D):l \in p(S,D,W)} V(S,D)$, where $p(S,D,W)$ is the shortest path between S and D under link weights W. The traffic matrix is sufficient if the routing change, say the rerouting of an MPLS label-switched path, would not effect how traffic exits the network.

Planning changes to the network topology: At a longer time scale (weeks to months), the path, traffic, and demand matrices can drive capacity planning decisions. Studying the evolution of the path matrix over time can help identify potential bottleneck links that should be upgraded to higher capacity. The traffic and demand matrices over time can be used to evaluate how the deployment of new routers and links would affect the flow of traffic in the network. The demand matrix can be used to evaluate decisions about whether and where to add another link (i.e., new

egress points) to an existing peer or customer. In addition, these predictive models can be used to evaluate the impact of adding a new customer to the network. This would involve augmenting the existing traffic or demand matrix with estimates of the traffic volumes introduced by the new customer.

2.3 Practical Challenges

Several comments about the three canonical representations are in order. First, these representations rely on a decoupling into an *invariant* external or offered load, described by the traffic and demand matrices, and the spatial flow of the traffic resulting from this load, described by the path matrix. This decomposition is an idealized assumption. In fact, the Internet has many closed control loops that *adapt* the load to the network behavior itself. For example, TCP congestion control throttles the transmission rate in response to packet loss; Content Distribution Networks (CDNs) may redirect requests for Web pages to alternate servers in response to network conditions; and end users themselves adapt their behavior, perhaps abandoning a Web transfer if the response time is unacceptably long. Therefore, the decoupling inherent in the representations described above is a simplification, and the precision of predictions based on such a load characterization may have some inherent inaccuracy.

Second, the path, traffic, and demand matrices can be very large objects, especially in a large IP backbone. For example, the set of all ingress-egress pairs is on the order of n^2 in a network with n nodes; the path matrix can potentially be very much larger, depending on the frequency of route changes and the existence of multiple shortest paths between ingress-egress nodes. The demand matrix could be even larger, depending on the number of different egress sets that arise in practice. Thus, the representations as described here are mainly conceptually appealing; in most practical settings, they would not actually be fully instantiated, but only populated partially, depending on the specific operational applications they drive (e.g., DDoS attack detection, service-level agreement (SLA) verification, routing). Defining compact and useful partial representations for specific applications is an important practical challenge, which has to rely on a diverse set of expertise and technology, such as databases, statistics, and algorithms. This is beyond the scope of this paper.

Third, we have simplified the discussion of the path, traffic, and demand matrices in that we have

implicitly assumed that the only attribute of interest is the traffic volume as a function of time. In the simplest case, the traffic volume represents the average load over some time scale (e.g., on a per-hour basis); however, other bandwidth measures, such as peak rate or effective bandwidth, are also possible. In practice, the network operators may need to focus their analysis on a particular subset of traffic, such as the path matrix for Web traffic or a particular customer. Alternatively, the operators may want to have a separate element in the traffic for each application (say, based on the TCP port numbers). This adds another dimension to these representations, reinforcing our point that efficient storage and querying of such large, high-dimensional objects is a challenging and important research problem.

3 Passive Measurement of IP Traffic

This section reviews the state-of-the-art in techniques for collecting traffic measurements in large IP backbone networks. We focus on passive measurement techniques that accumulate information about the traffic as it travels through the network. First, we discuss how operators can collect aggregate traffic statistics using SNMP and highlight some of the advanced features available for local-area networks using RMON. Next, we describe how operators can collect detailed IP packet traces on individual links in the network. Then, we describe techniques for monitoring traffic at the flow level to provide relatively detailed traffic statistics with fewer measurement records. Throughout the section, we consider the limitations of each measurement technique in providing a detailed, network-wide view of the prevailing traffic.

3.1 SNMP/RMON

The Simple Network Management Protocol (SNMP) is an IETF standard for the representation of management information, and the communication of such information between management stations and management agents for the purpose of monitoring and effecting network elements [6, 7]. Both the representations, called Management Information Base (MIB), and the protocol itself are kept as simple as possible: MIB information uses a hierarchical naming structure, and the leaves of this naming tree can contain only either simple scalar variables, or tables. The communication protocol only supports three primitives: `get`, `set`, and `get-next`, i.e., reading of

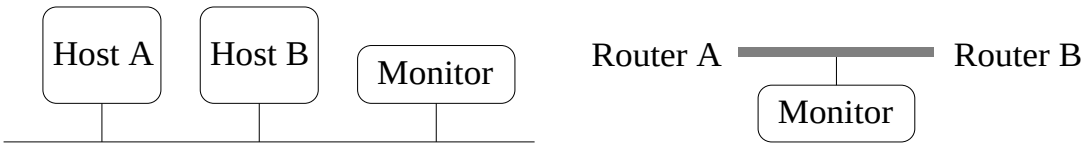
an agent variable by the management station, changing such a variable by the management station, and reading the next item in lexicographical order, which facilitates reading a table. There is also support in the protocol for automatic notifications from the agent to the management system. The first version of SNMP has since been extended in various ways; some of its original simplicity has inevitably been lost in the process [8].

The main standardized MIB containing traffic-related data is called MIB-II. It is organized in a set of groups (subtrees) that monitor the execution of various protocols on the network element, such as IP, TCP/UDP, BGP, OSPF, etc. Most of this information is either status information (e.g., the operational status of the interfaces on a router), or highly aggregated (e.g., per-interface counters of bytes and packets inbound and outbound). There is no support for fine-grained measurements required to populate directly the representations presented in the previous section.

The advantage of MIB-II is that it is almost universally supported in routers and other network elements, even on high-speed interfaces. A drawback lies in the absence of a certification authority to ensure correct implementations. Unfortunately, many vendors seem to do an insufficient job in testing MIB-II implementations, which results in measurement errors and artifacts.

RMON1 is another standardized SNMP MIB, whose goal is to facilitate remote monitoring of LANs [9]. Its main advantages are (1) to enable a single agent to monitor an entire shared LAN; (2) to endow the agent with local intelligence and memory to enable it to compute higher-level statistics and to buffer these statistics in the case of outages; (3) to define alarm conditions and the actions that should be taken in response, such as generating notifications to the network management system; (4) to define packet filtering conditions and the actions that should be taken in response, such as capturing and buffering the content of these packets.

RMON offers great flexibility in combining the above primitives into sophisticated agent monitoring functions. However, this flexibility makes a full RMON agent implementation costly. Thus, RMON has only been (at least partially) implemented for local area network (LAN) router interfaces, which are relatively low-speed. Implementations for high-speed backbone interfaces have proved to be infeasible or prohibitively expensive. Instead, router vendors have opted to develop more limited monitoring capabilities for high-speed interfaces. We next describe the two main classes of such methods, *packet monitoring* and *flow measurement*.



(a) Shared media (b) Point-to-point link

Figure 4: Tapping a link to collect packet traces

3.2 Packet Monitoring

A packet monitor passively collects a copy of the packets traversing a link and records information at the IP, TCP/UDP, or application layers. Collecting packet traces requires an effective way to tap a link in the operational network. This is relatively easy in a local area network (LAN) consisting of a shared media, such as an Ethernet, a FDDI ring, or a wireless network. As shown in Figure 4(a), the packet monitor may be a regular computer connecting to the LAN with a network interface card configured to run in *promiscuous* mode to make a local copy of every packet. Tapping a link is more difficult in a large backbone consisting of routers connected via high-speed point-to-point links. Traces may be collected by splitting each unidirectional link into two parts in order to direct a copy of each packet to the monitor, as shown in Figure 4(b). Alternatively, the router could include support for collecting packet traces. As traffic reaches the router, the interface card can make a local copy of the packets.

Copying and analyzing the entire contents of every packet is extremely expensive, especially on the high-speed links common in large backbone networks. Limiting the processing load and the volume of measurement data is crucial. Packet monitors employ three main techniques to reduce the overhead of collecting the data. First, the monitor can capture and record a limited number of bytes for each packet. For example, the monitor could record the IP header (typically 20 bytes) or the IP and TCP headers (typically 40 bytes) instead of the entire contents of the IP packet. Second, the monitor may be configured to focus on a specific subset of the traffic, based on the fields in the IP and TCP or UDP headers. For example, the monitor could capture packets based on the source and destination IP addresses or port numbers. Third, the monitor may perform sampling to limit the fraction of packets that require further processing. For example, the monitor could be configured to record information for one out of every hundred packets. Routers that support packet-level measurement typically employ sampling to reduce the overhead on the interface card

and avoid degrading packet-forwarding performance.

Packet monitoring is an effective way to acquire fine-grain information about the traffic traversing a link. PC-based monitoring platforms are quite common in LAN environments. Many of these packet monitors run the popular, public-domain `tcpdump` software [10] using the Berkeley Packet Filter [11]. Extensions to this software support the collection of application-level information inside the IP packets to analyze Web or multimedia traffic or detect possible intruders [12, 13, 14]. However, collecting packet traces on high-speed links is challenging without dedicated hardware support. Several monitors have been developed by research groups to collect traces on selected links [15, 16, 17]. In addition, several companies offer packet monitoring products that can collect traces on a variety of different types of links. Network operators can use these products to collect traces on key links that connect to important customers or services. Recognizing the importance of packet monitoring to service providers, major router vendors have begun to support packet sampling directly in their interface cards. This offers network operators an effective way to collect detailed information about the traffic on a large number of high-speed links at a reasonable cost.

3.3 Flow Measurement

Flow-level measurement involves collecting aggregate traffic statistics for related packets that appear close together in time. The abstraction of a flow mimics the notion of a call or connection in circuit-switched networks. Since IP networks do not retain state about individual data transfers, the grouping of packets into a flow depends on a set of rules applied by the measurement device. The rules identify the attributes that must match across all the packets in a flow, as well as restrictions on the spacing between packets in a flow. For example, a flow could be defined as all packets that have the same source and destination IP addresses, where successive packets have an interarrival time less than 30 seconds. Alternatively, a flow could be defined as all packets that match in their IP addresses, port numbers, and protocol, and have an interarrival time less than 60 seconds. The rules for defining a flow depend on the details of the measurement device and may be configurable. The measurement device reports aggregate information for each flow, such as the key IP and TCP/UDP header fields that match across the packets, the total number of bytes and packets in the flow, and the timestamps of the first and last packets.

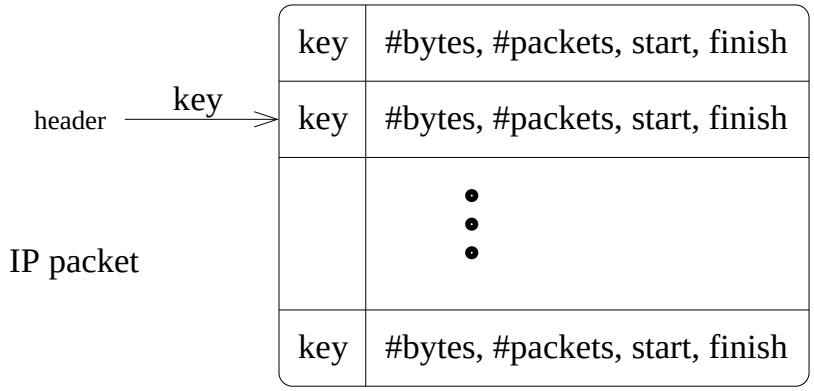


Figure 5: Accessing the flow cache with a key computed from the packet header

Similar to a packet monitor, a flow-measurement device must tap a link to receive a copy of each packet. To collect the aggregate traffic statistics, the devices maintains a cache of active flows. When a packet arrives, the device computes an index into the cache by combining the various header fields that define a flow, as shown in Figure 5. The device accesses the cache to create a new entry or update an existing entry. Each entry in the cache includes the aggregate traffic statistics, such as the total number of bytes and packets, as well as the timestamps. The device evicts an entry from the cache and exports or records the information. A flow is evicted from the cache if no additional packets have arrived for a period of time (e.g., 30 or 60 seconds), or if the cache is full. The device may also evict a flow that has been active for a long period of time (e.g., 30 minutes) to ensure that information about the long-lived transfer is made available. The arrival of additional packets with the same index would trigger the creation of a new entry in the cache and, ultimately, another measurement record. In practice, the device may combine multiple records into a single unit for transmission to a separate machine for archiving and analysis.

Flow-level measurements are available from some router vendors, most notably Cisco with its Netflow feature [18]. For lower measurement overhead on high-speed links, recent releases of Netflow include support for sampling and aggregation [19, 20]. Several commercial packet monitors can produce flow-level measurements, allowing network operators to have a mixture of router support and separate measurement devices. The Internet Engineering Task Force (IETF) has made efforts to define standards for flow-level measurement. The Real-Time Traffic Flow Meter (RTFM) group [21, 22] provided a formal definition of flows and described techniques for collecting the measurement data. More recently, the IP Flow Information Export (IPFIX) working group [23] is defining a

format and protocol for delivering flow-level data from the measurement device to other systems that archive and analyze the information. The vendor and standardization activity reflects a growing interest in the use of flow-level measurement as a practical alternative to collecting packet traces.

3.4 Direct Uses of Measurement Data

While the main focus of this chapter is on measurements to populate domain-wide traffic representations, as outlined in the previous section, we briefly discuss other, more direct, uses of the different types of passive measurement data described in this section.

SNMP/RMON. MIB-II information is useful to verify overall operational health of a network, by monitoring variables related to traffic (e.g., links utilizations, fraction of packets dropped due to checksum errors), router health (e.g., CPU and memory load of the router's central processor), and to network state and protocol performance (e.g., resets on BGP sessions). Such variables are typically monitored continuously, but on a relatively slow time-scale (e.g., minutes to hours), by a central network management system. This management system will issue alarms to human operators if monitored variables are outside their predefined ranges. For many failure scenarios, however, MIB-II information is too highly aggregated to be sufficient for diagnosis. For example, if a link is overloaded due to a denial-of-service attack, the operator requires more fine-grained traffic measurements (such as packet or flow measurements) to determine the origin of the attack.

RMON gives access to much more fine-grained measurements, and provides more flexibility in how these measurements are collected. For example, it enables a network element to collect a traffic matrix at the MAC-address (layer 2) level directly and to report the most heavy hitters to the network management system. Such a functionality would be very useful in the example of a denial-of-service attack. RMON also allows the agent to accumulate a time-series of samples of a variable, and to perform limited local processing on this time-series (e.g., alarm thresholding). RMON embodies support for monitoring of packets that match a filter criterion; fields of interest of these packets can be reported back to the NMS.

In fact, if RMON were universally deployed on high-speed interfaces, it would essentially obviate the need for additional packet and flow monitoring support. However, its complexity appears to

prohibit implementation other than in LAN environments, which relegate RMON to the edge of the network. It is not widely used by network operators.

Packet monitoring. A packet monitor can provide network operators with detailed information about the traffic traversing a link. First, the traces can be aggregated to generate reports of the volume of traffic in terms of key fields in the packet headers. For example, computing traffic volumes by source and destination IP addresses allows operators to identify the end hosts responsible for the bulk of the load on the network. Considering the traffic mix by TCP or UDP port number is useful for identifying the most popular applications (e.g., Web, peer-to-peer transfers, etc.). Second, packet traces can provide information about the performance experienced by users. For example, the traces can be analyzed to compute the throughput for individual TCP connections or source-destination pairs. With accurate timestamps, the operators can combine traces from multiple locations in the network to compute latency statistics. Flags in the TCP header may be useful for identifying when a user aborts an ongoing transfer or for detecting certain kinds of denial-of-service attacks. Third, the traces can provide insight into the fine-grain characteristics of the traffic, such as packet sizes, the burstiness of the stream of packets traversing the link, and typical transfer sizes.

Flow measurement. Flow-level measurements are useful for many of the same purposes as packet traces. A network operator may use flow-level statistics to compute traffic volumes by IP addresses, port numbers, and protocols, or basic traffic characteristics such as average packet size. An ISP may use traffic statistics at the level of individual customers to drive accounting and billing applications. However, flow-level measurements do not provide the fine-grain timing information available in packet traces. The flow measurement record provides timestamps for the first and last packet in the flow, without detailed information about the spacing between successive packets. As such, flow-level measurements are not useful for studying the burstiness of traffic on a small time scale. Flow-level measurements can provide some information about the throughput experienced by users, based on the number of bytes and the time duration of a flow. However, identifying potential causes of low throughput, such as lost and retransmitted packets, is difficult without more detailed information.

4 Populating the Network-Wide Traffic Models

This section describes how the different types of measurement data discussed in Section 3 can be used to populate the network-wide traffic models described in Section 2. We present these methods in a progression from coarse-grain traffic statistics to fine-grain packet traces to illustrate how a finer granularity of measurement data allows us to relax our assumptions about the traffic and network models. In particular, we discuss the following three scenarios:

- **Assumptions on the network and traffic model:** First, we assume that only aggregate traffic statistics, such as traditional SNMP data, are available for each link. In this scenario, a class of statistical inference methods referred to as network tomography can generate an *estimate* of the traffic matrix. The inference techniques rely on assumptions about the state of the network elements and the statistical properties of the traffic.

- **Assumptions on the network model:** Second, when more fine-grained traffic measurements, such as flow or packet traces, are available from the links at the edge of the network, the traffic and path matrix can be *derived*. In contrast to network tomography, this approach does not rely on any assumptions about the characteristics of the traffic. However, the technique does need to model certain aspects of the network in order to "map" the edge measurements onto the topology.

- **No assumptions about network or traffic:** Third, we describe methods that provide a *direct* observation of the path matrix without relying on any assumptions about the traffic or the network. These methods yield the most faithful observation of the spatial distribution of traffic even in scenarios where the network state is unknown or changing. However, these techniques require additional support in routers that is not available in commercial products as of this writing.

4.1 Network and Traffic Model: Tomography

The class of methods we describe in this section apply in a setting where only aggregate traffic measurements are available. Specifically, these methods assume that the data rate on every link in

the domain is known, averaged over some time interval. The goal is to estimate the traffic matrix, i.e., the data rate for every ingress-egress pair. They rely on both a network model and a traffic model. The network model assumes that the routes from every ingress to every egress pair are stable and known over a time period over which inference is performed. The traffic model describes traffic flows as some parameterized stochastic process. Such a traffic model is central to tomography methods, as they inherently rely on multiple measurements over time to infer the missing spatial information about traffic flow.

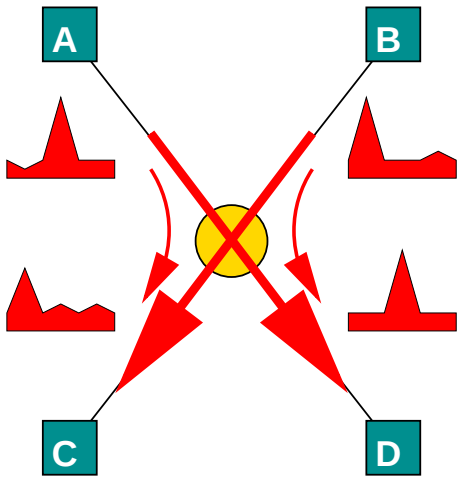


Figure 6: The evolution of aggregate link rates over time contains information about the traffic matrix.

The basic idea of network tomography is quite intuitive. Consider the simple network in Figure 6, consisting of four ingress-egress pairs AC, AD, BC, and BD, and four links, on which the aggregate rate is measured. It is quite clear that with a single measurement, nothing can be said about the traffic matrix. For example, if the data rate on each link were 1 Mbps, this does not tell us the relative contributions of AC and AD on the upper left link. However, if the evolution over time of link data rates is known, then the traffic matrix can be estimated. For example, given the link rates in Figure 6, we would intuitively conclude that AD and BD are large compared to AC and BD. Network tomography formalizes this notion and develops efficient methods to compute such estimators.

The pioneering work in this area is due to Vardi [24], and inspired by conceptually similar work in transportation networks. We give an overview of his model and of his approach to estimating the

traffic matrix.

Let the vector $Y = (Y_1, \ldots, Y_r)$ denote an *observation* of packet counts on all links, where r is the number of links in the network, and Y_i is the number of packets observed on link i. The vector $X = (X_1, \ldots, X_c)$ counts the number of packets per ingress-egress pair, where c, the number of pairs, is typically much larger than r. X can be viewed as a snapshot of the traffic matrix. The relationship between X and Y can be written in matrix form as

$$Y = AX, \tag{1}$$

where A is the routing matrix, embodying the fixed and known routing state as follows: the element $A_{i,j} = 1$ if link i is on the path of ingress-egress pair j, and 0 otherwise.

In general, the number of ingress-egress pairs is much larger than the number of links in the network (i.e., $c \gg r$). As such, a single observation of Y contains very little information about X, in the sense that the subspace $\{X : Y = AX, X \geq 0\}$ of possible X for a given observation Y is of high dimension. Vardi therefore introduces the following parameterized stochastic model. Assume that time is slotted, and that for each time slot k, we obtain an independent sample of the traffic matrix $X^{(k)}$, and hence of the link rates $Y^{(k)}$. For each k, the components of $X^{(k)}$ are independent and have Poisson distribution $\lambda = (\lambda_1, \ldots, \lambda_c)$.

$$X_j^{(k)} = \text{Poisson}(\lambda_j) \tag{2}$$

Let us consider the maximum likelihood estimator $\hat{\lambda}$ of λ. It is relatively straightforward to write down the likelihood function for λ,

$$L(\lambda) = P_\lambda(Y) = \sum_{X:AX=Y, X\geq 0} P_\lambda(X). \tag{3}$$

Unfortunately, the likelihood function is very expensive to compute, because it involves finding all natural solutions to the equation $AX = Y$. Vardi proposes an estimator based on the method of moments, which is very efficient to compute. Specifically, for large sample size K, the empirical mean $\bar{Y}$ of the observation Y,given by

$$\bar{Y} = \frac{1}{K} \sum_{k=1}^{K} Y^{(k)}, \tag{4}$$

is approximately Gaussian. Therefore, its distribution depends only on its first and second moments. By equating the empirical moments of $\bar{Y}$ with the moments predicted by the model, we obtain the equation

$$\begin{pmatrix} \bar{Y} \\ S \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} \lambda. \tag{5}$$

Here, S is the empirical covariance matrix, written as a vector, and the matrix B is a function of A.

The linear system (5) cannot be solved directly, because of inconsistencies introduced because of the noise in the empirical moments. Vardi proposes heuristics to eliminate equations from (5), and to iteratively compute an approximate estimator $\hat{\lambda}$.

Several papers build on and extend Vardi's work. For example, Tebaldi and West [25] study the tomography problem in a Bayesian setting, which allow for the inclusion of prior information into the estimate. Cao et. al. [26] consider a similar problem setting, but instead of the (discrete) Poisson traffic model, they assume that the data rate for every OD-pair is Gaussian. Their model is more flexible than Vardi's because it parameterizes the scaling relationship between the mean and variance of the Gaussian marginals.

Network tomography methods have some serious practical limitations. First, they tend to make strong simplifying assumptions about the traffic model in order to devise tractable solutions. The assumptions typically include the stationarity and ergodicity of the traffic process in order to estimate the spatial traffic matrix from multiple temporal samples. Furthermore, it is usually assumed that consecutive samples are independent. However, network traffic tends to possess much more complicated structure [27], and can be viewed as stationary only over relatively short periods of time; also, consecutive samples will not typically be completely independent. Second, even when the traffic model is accurate, it takes a large number of samples to arrive at sufficient precision. This is a direct consequence of the fact that in most cases, the number of unknowns (the elements of the traffic matrix) is much larger than the number of observables (the aggregate rate on links)[3] Third, given the size of the inversion problem (5), estimating a large traffic matrix is computationally challenging. An interesting research question therefore concerns the inference of a subset of the traffic matrix, depending on the application it is driving.

[3]Vardi presents some numerical examples using a 5-node topology that illustrate this problem.

These limitations suggest that tomography methods are not applicable to large-scale traffic management problems. However, they may very well turn out to be useful in limited applications. For example, suppose an operator wishes to obtain a "local" traffic matrix at a single router. Tomography may work well in such a limited setting, and could turn out to be an interesting alternative to collecting detailed packet or flow traces from that router. Another promising direction is to use tomography to complement other types of traffic measurement. For example, if in a network domain, only a subset of edge routers were capable of obtaining packet or flow measurements, then tomography methods may allow to infer the missing components of the traffic matrix. While such a complementary use of tomography is promising, it requires further research.

4.2 Network Model: Traffic Mapping

Developing simple and accurate statistical models of Internet traffic is difficult in practice. Packet and flow-level measurements offer a way to construct the path, traffic, and demand matrices without simplifying assumptions about the characteristics of the traffic. However, collecting fine-grain traces for every link in a large network would be expensive given the limitations of today's technology. Even if the interface cards have direct support for traffic measurement, this functionality may not be uniformly available and, in some cases, may degrade the throughput of the router. In addition, many routers do not provide effective, tunable ways to limit the volume of measurement data. Given these constraints, it is appealing to have a way to construct the path, traffic, and demand matrices based on a limited collection of fine-grain measurements. For example, monitoring every ingress point into the domain provides a way to collect fine-grain statistics for all of the traffic without enabling measurement functionality in the core of the network. However, computing the spatial distribution of traffic from these measurements depends on having an accurate model of routing in the underlying network.

Populating the network-wide models involves collecting fine-grain measurements at each ingress point and aggregating this data to the level of the path, egress point, or the set of possible egress points. The forwarding of traffic through the network depends on the IP prefix associated with each packet's destination IP address. Each router combines information from the intradomain and interdomain routing protocols to construct a forwarding table that is used to select the next-hop interface for each incoming packet. For example, a forwarding table has entries such as:

12.128.0.0/9	10.126.212.1	POS2/0
172.12.4.0/24	10.1.2.118	Serial1/0/0:1
192.0.2.0/24	10.47.35.56	POS3/0

The third entry directs packets destined for 192.0.2.0/24 to interface POS3/0 (a packet-over-SONET link in slot 3 of the router) to reach the next hop with IP address 10.47.35.56. 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. For example, the router could associate the IP address 192.0.2.38 with the prefix 192.0.2.0/24.

Most routers have a command-line interface that allows network operators to view the current state of the forwarding table; operators may run scripts that dump these tables periodically. These dumps provide the list of prefixes and their associated interfaces. The routers that connect to neighboring domains have forwarding tables that indicate the egress point(s) associated with each destination prefix. For example, suppose the Serial1/0/0:1 interface (in the forwarding table listed above) connects to a customer's network; then, the destination prefix 172.12.4.0/24 would have this interface in its set of egress points. Other routers might have forwarding table entries that direct traffic to an interface to another router inside the AS; these interfaces would not be included in the egress set. By repeating this process across all of the forwarding tables, it is possible to generate the list of egress interfaces associated with destination. Based on this information, the fine-grain traffic measurements collected at the ingress points can be associated with the longest matching destination prefix and, in turn, the set of egress points [5].

On the surface, the demand matrix may seem like an unwieldy representation that has a large number of elements. A typical forwarding table in a large IP network contains about 100,000 prefixes, and this number is increasing as the Internet continues to grow. In addition, the number of egress sets could be exponential in the number of distinct egress points. In practice, many destination prefixes have the same egress set [4]. Many organizations, such as companies or universities, announce multiple blocks of IP address to their upstream providers. Another AS in the Internet would tend to route all traffic for these destination prefixes in the same manner, since the BGP advertisements would have all the same attributes. In addition, most combinations of egress points do not arise in practice. Typically, an egress set would consist of all links to a particular neighboring domain,

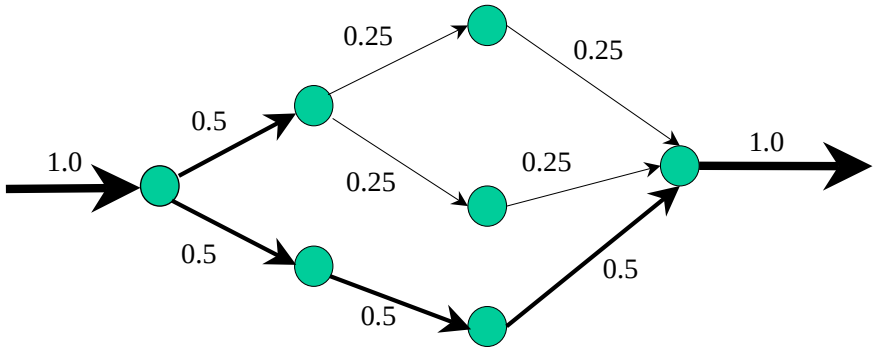


Figure 7: Traffic splitting across multiple shortest paths

such as a customer or peer. For example, a network may connect to another AS in six different locations. These six links would tend to appear together in egress sets for various destination prefixes. This tends to result in a relatively small number of different egress sets, ensuring a compact representation for the demand matrix.

The demand matrix can be used to compute the path and traffic matrices. Each element in the demand matrix corresponds to a single ingress point and set of egress points. The traffic from that ingress point travels to a particular egress point over one or more paths through the domain. The selection of a particular egress point depends on the intradomain routing configuration (e.g., OSPF or IS-IS weights), as discussed earlier in Section 2.1. The chosen egress point and the path(s) through the domain can be determined in two main ways. First, by dumping the forwarding table for each router in the domain, it is possible to identify the interfaces that would carry the packets as they travel from one router to another. Second, these paths can be computed based on a network-wide view of the topology and the intradomain routing configuration (e.g., by computing the shortest path(s) based on the OSPF/IS-IS weights) [3]. In practice, operational networks may have multiple shortest paths from the ingress point to the chosen egress point. Most commercial routers split traffic over these multiple paths, dividing the traffic evenly when multiple next-hop interfaces lie along a shortest path, as shown in Figure 7.

4.3 No Model: Direct Observation

The methods presented so far are all *indirect* to a certain degree, in that they rely on some assumptions on the traffic and on the network behavior to populate the various domain-wide measurement representations. These assumptions are embodied in the traffic and network models. Indirect measurement methods thus suffer from the inherent uncertainty associated with the physical and logical state of large, heterogeneous networks. This uncertainty has several sources:

- The exact behavior of a network element, such as a router, is not exactly known to the service provider and depends on vendor-specific design choices. For example, the algorithm for splitting traffic among several shortest paths in OSPF is not standardized.
- The network has deliberate sources of randomness to prevent accidental synchronization, e.g., through active queue management disciplines [29] or randomized timers in routing protocols [30].
- IP packets may be lost as they travel through the network. Predicting whether a packet measured in one location was successful in traversing the entire path through the domain is difficult in practice.
- Some of the behavior of the network depends on events outside of the control of the domain. For example, how traffic is routed within an Autonomous System (AS) depends in part on the dynamics of route advertisements from neighboring domains [31].
- The interaction between adaptive schemes operating at different time-scales and levels of locality (e.g., QoS routing, end-to-end congestion control) may simply be too complex to characterize and predict [32].
- Equipment failures can disrupt the normal operation of the network. Traffic measurement is one of the tools for detecting and diagnosing such problems; however, this benefit is mitigated if traffic measurement depends on the correct operation of the network.

Rather than relying on a network model and an estimation of its state and its expected behavior, *direct* methods observe the traffic as it travels through the network. In this subsection, we describe two techniques for direct observation of the path matrix—trajectory sampling and IP traceback.

4.3.1 Trajectory Sampling

Trajectory sampling [33] involves sampling packets that traverse each link (or a subset of these links) within a measurement domain. The subset of sampled packets over a certain period of time can then be used as a representative of the overall traffic.

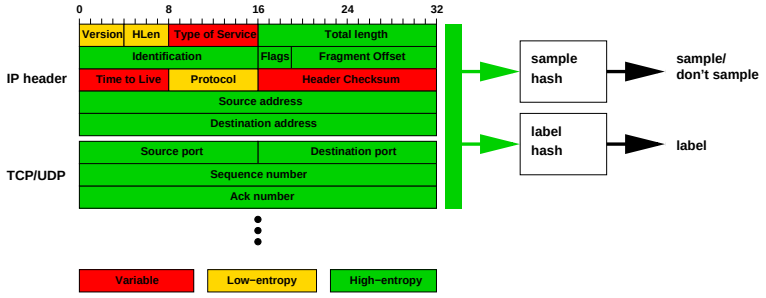


Figure 8: Trajectory sampling relies on two hash functions computed over the invariant fields of the packet's header and payload. The sampling hash function serves to decide whether a packet should be sampled or not; the label hash function stamps a unique identity on the packet for trajectory reconstruction.

If packets were simply randomly sampled at each link, then we would be unable to derive the precise path that a sampled packet has followed through the domain from the ingress to the egress point. The key idea in trajectory sampling is therefore to base the sampling decision on a deterministic hash function over the packet's *content*. If the *same* hash function is used throughout the domain to sample packets, then we are ensured that a packet is either sampled on *every* link it traverses, or on no link at all. In other words, we effectively are able to collect *trajectory samples* of a subset of packets. The choice of an appropriate hash function will obviously be crucial to ensure that this subset is not statistically biased in any way. For this, the sampling process, although a deterministic function of the packet content, has to resemble a random sampling process.

A second key ingredient of trajectory sampling is that of *packet labeling*. Note that to obtain trajectory samples, we are not interested in the packet content per se; we simply need to know that *some packet* has traversed a set of links. But to know this, it is sufficient to obtain a unique packet identifier, or label, for each sampled packet within the domain and within a measurement period. Because the label is unique, we will know that a packet has traversed the set of links which have reported that particular label. Trajectory sampling relies on a second hash function to compute

packet labels that are, with high probability, unique within a measurement period. While the size of the packet labels obviously depends on the specific situation, it has been shown that labels can in practice be quite small (e.g., 20 bit). As the measurement traffic that has to be collected from nodes in the domain only consists of such labels (plus some auxiliary information), the overhead to collect trajectory samples is small.

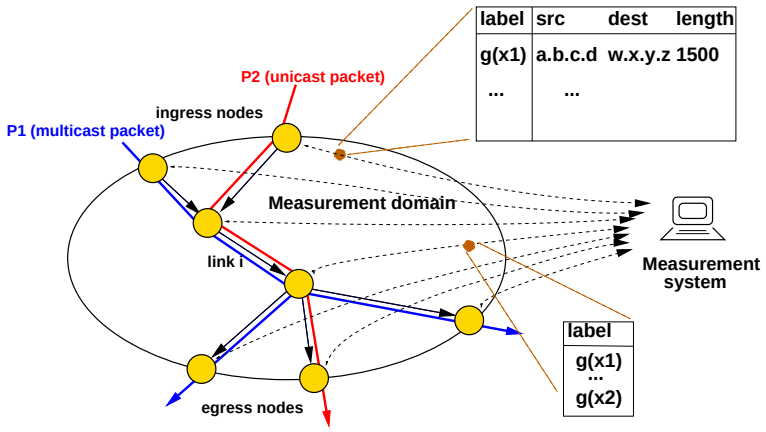


Figure 9: Schematic representation of trajectory sampling. A measurement system collects packet *labels* from all the links within the domain. Labels are only collected from a pseudo-random subset of all the packets traversing the domain. Both the decision whether to sample a packet or not, and the packet label, are a function of the packet's invariant content.

Trajectory sampling has several important advantages. It is a direct method for traffic measurement, and as such does not require any network status information. A set of trajectory samples is basically a sample of the path matrix. Trajectory sampling does not require router state (e.g., per-flow cache entries) other than a small label buffer. The amount of measurement traffic necessary is modest and can be precisely controlled. Multicast packets require no special treatment - the trajectory associated with a multicast packet is simply a tree instead of a path. Finally, trajectory sampling can be implemented using state-of-the art digital signal processors (DSPs) even for the highest interface speeds available today.

4.3.2 IP Traceback

The second domain-wide measurement method that does not rely on a network model nor a traffic model is *IP traceback* [34]. The main goal of IP traceback is detecting and diagnosing distributed denial of service (DDoS) attacks. In such an attack, a potentially large number of hosts (which may have been subverted) overwhelm a victim host, such as a web server, with so much traffic that it becomes inoperational. Defending against such an attack is difficult, because the source addresses of the attack packets are often *spoofed*, i.e., they do not correspond to the real originator of the packet. Therefore, such packets contain no information about the originator. The goal of IP traceback is to infer the originator(s) of such an attack, or to at least partially infer the paths from the originator to the victim. This allows to take countermeasures, such as blocking traffic on certain links, and isolating "hijacked" hosts used in the attack.

IP traceback is an end-to-end and interdomain solution to this problem. It is end-to-end because the victim does not require any explicit help from its network operator to infer the attack sink tree. It is interdomain because no specific coordination across domain boundaries is required to infer a sink tree spanning multiple domains. These properties enable a gradual and widespread deployment of IP traceback.

We describe IP traceback by first introducing a very crude scheme to solve this inference problem, and then successively refining it. A straightforward solution to the traceback problem would consist in having every router add its identity to a list carried with the packet. This would allow the recipient of the packet to determine the exact path followed by the packet. Of course, such a scheme would incur prohibitive processing and communication overhead.

To limit this overhead, it is desirable to allocate only a small, fixed field of each packet to IP traceback. Clearly, this prohibits fully reporting the path traversed by each packet. A possible solution consists in having packets sample the set of routers encountered along their path, and using the field to report the last sampled router to the recipient (cf. Fig. 10). In practice, this can be implemented by routers flipping a coin for each packet, and writing their IP address into the traceback field with probability p. As the defining property of a denial of service attack is that the victim receives a large number of attack packets from the same host or set of hosts, the victim can construct a histogram of router addresses observed in received packets. In the idealized case of a

single attacker and attack path, the observed frequencies of router addresses directly yield the set of routers on the path, the most frequently observed address corresponding to the router closest to the attacker, and vice versa.

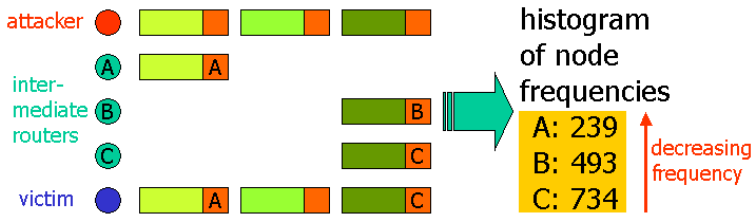


Figure 10: Node sampling. The attack path is reconstructed simply through the observed frequency of node identities.

The above approach has two problems. First, it is possible for the attacker to perturb the inferred attack path, by preloading the traceback field of all the packets it sends with some forged IP address (or possibly several addresses). This forged address will show up in the frequency histogram; the victim has no way to tell forged and real samples apart. The only way to avoid this effect would be by using a very high sampling rate of $p > 1/2$. However, this would preclude inferring reasonably long paths, as the probability for a sample to survive decreases very quickly with router distance from the victim. Second, the node-based scheme cannot distinguish multiple attack paths, as the path is simply inferred from sample frequencies.

To remedy these two problems, the scheme can be slightly extended to sample edges instead of nodes. This can be achieved through three fields, edge_start, edge_end, and dist(cf. Fig. 11). Each router again flips a coin with probability p. In case of a hit, it writes its address into the field edge_start, and sets dist to zero. The next router, because it receives a packet with dist=0, writes its identity into edge_end, and increments dist. Every downstream router will simply increment dist (provided it does not start an edge sample itself). Therefore, the receiver will effectively receive a sampled edge (edge_start, edge_end), along with the distance dist of that edge from itself.

It is straightforward to see that the attack sink tree can be fully reconstructed from edge samples. However, the edge-based scheme just described would require 64 bits for edge_startand edge_end, plus the bits required for dist. The main challenge addressed in [34] is how to compress this information so that it fits into the 16 bit identification field of the IP header. The authors propose

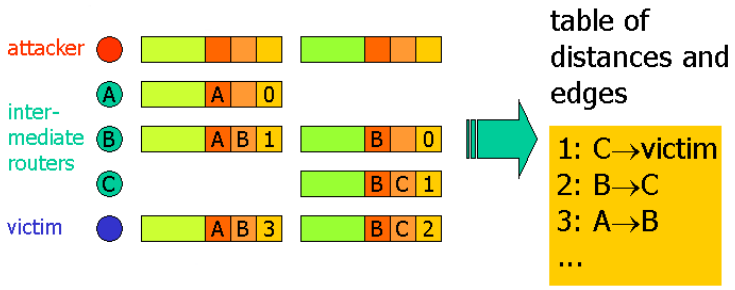


Figure 11: Edge sampling. The attack path or tree is explicitly reconstructed from sampled edges.

several clever tricks to achieve this compression.

Related proposals for the diagnosis of DDoS attacks rely on out-of-band methods (specifically, ICMP packets) to carry samples [35], and *hash-based IP traceback* [36], a method that keeps compressed path information within routers. The latter method is similar to trajectory sampling, in that it relies on a hash computed over invariant packet fields to associate unique labels with packets. In contrast to trajectory sampling, however, the method keeps track of all the packets traversing a router.

5 Conclusion

Traffic measurement plays an important role in managing large IP networks. Existing passive measurement techniques provide operators with coarse-grain traffic statistics for the entire network and fine-grain traces for individual links and routers. SNMP is widely available and provides aggregated information about link load and packet loss on the scale of minutes. Packet monitoring and flow-level measurements allow operators to have a more detailed view of the traffic; however, these techniques are not uniformly available in existing IP routers. Operators need a network-wide view of the spatial distribution of traffic to detect and diagnose performance problems, and to evaluate potential control actions. The path, traffic, and demand matrices provide a sound basis for a variety of common tasks, such as identifying the customers affected by an overloaded link and tuning the routing configuration to alleviate the congestion. Populating these traffic models requires collecting measurement data from multiple locations in the network.

The path and traffic matrices can be estimated from traditional SNMP measurements, under certain

assumptions about the statistical properties of the traffic and knowledge of routing inside the backbone. Collecting fine-grain packet or flow-level measurements at the edge of the network makes it possible to compute the path, traffic, and demand matrices without any simplifying assumptions about the traffic. With additional support for packet sampling in the routers, the path matrix can be observed directly without depending on additional information from the routers. Ultimately, IP equipment needs to evolve to include more advanced measurement functionality that supports the common tasks of network operators. Additional work on constructing network-wide representations of the traffic can help guide this process. For example, hybrid approaches for populating the models would be useful in networks that do not have uniform support for fine-grain traffic measurement. In addition, new sampling techniques may be necessary to address the challenges of collecting measurement data for high-speed links.

References

[1] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. ACM SIGCOMM*, pp. 106–114, August 1988.
`http://www.acm.org/sigs/sigcomm/ccr/archive/1995/jan95/ccr-9501-clark.html`.

[2] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, January 1999. ISBN 0201379511.

[3] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, pp. 11–19, March 2000.
`http://www.research.att.com/~jrex/papers/ieeenet00.ps`.

[4] N. Feamster, J. Borkenhagen, and J. Rexford, "Controlling the impact of BGP policy changes on IP traffic," Tech. Rep. HA173000-011106-02TM, AT&T Labs – Research, November 2001.

[5] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving traffic demands for operational IP networks: Methodology and experience," *IEEE/ACM Transactions on Networking*, June 2001.
`http://www.research.att.com/~jrex/papers/ton01.ps`.

[6] W. Stallings, *SNMP, SNMP v2, SNMP v3, and RMON 1 and 2 (Third Edition)*. Reading, Mass.: Addison-Wesley, 1999.

[7] W. Stallings, "SNMP and SNMPv2: The Infrastructure for Network Management," *IEEE Communication Magazine*, March 1998.

[8] W. Stallings, "Security Comes to SNMP: The New SNMPv3 Proposed Internet Standards," *Internet Protocol Journal (Cisco News Publications Group, San Jose, CA)*, vol. 1, December

1998.
http://www.cisco.com/ipj.

[9] D. T. Perkins, *RMON: Remote Monitoring of SNMP-Managed LANs*. Prentice Hall, September 1998.

[10] V. Jacobson, C. Leres, and S. McCanne, "Tcpdump."
ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[11] S. McCanne and V. Jacobson, "The BSD Packet Filter: A New Architecture for User-Level Packet Capture," in *Proc. Winter USENIX Technical Conference*, January 1993.
ftp://ftp.ee.lbl.gov/papers/bpf-usenix93.ps.Z.

[12] A. Feldmann, "BLT: Bi-layer tracing of HTTP and TCP/IP," in *Proc. World Wide Web Conference*, pp. 321–335, May 2000.
http://www.cs.uni-sb.de/~anja/feldmann/papers/blt_httptrace.ps.

[13] J. van der Merwe, R. Caceres, Y. Chu, and C. J. Sreenan, "mmdump: A tool for monitoring Internet multimedia traffic," *ACM Computer Communication Review*, vol. 30, pp. 48–59, October 2000.
http://www.acm.org/sigcomm/ccr/archive/2000/oct00/ccr_200010-merwe.html.

[14] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, pp. 2435–2463, December 1999.
ftp://ftp.ee.lbl.gov/papers/bro-CN99.ps.gz.

[15] C. Fraleigh, C. Diot, S. Moon, P. Owezarski, D. Papagiannaki, and F. Tobagi, "Design and deployment of a passive monitoring infrastructure," in *Proc. Passive and Active Measurement Workshop*, April 2001.

[16] R. Caceres *et al.*, "Measurement and analysis of IP network usage and behavior," *IEEE Communications Magazine*, May 2000.
http://www.research.att.com/~jrex/papers/ieeecomm00.ps.

[17] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and K. Claffy, "The architecture of the CoralReef Internet traffic monitoring software suite," in *Proc. Passive and Active Measurement Workshop*, 2001.
http://www.caida.org/outreach/papers/pam2001/coralreef.pdf.

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

[19] "Sampled Netflow."
http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s11/12s_sanf.htm.

[20] "NetFlow Aggregation."
http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t3/netflow.htm.

[21] N. Brownlee, C. Mills, and G. Ruth, "Traffic flow measurement: Architecture," RFC 2722, IETF, October 1999.
http://www.rfc-editor.org/rfc/rfc2722.txt.

[22] S. Handelman, S. Stibler, N. Brownlee, and G. Ruth, "RTFM: New attributes for traffic flow measurement," RFC 2724, IETF, October 1999.
http://www.rfc-editor.org/rfc/rfc2724.txt.

[23] "Internet Protocol Flow Information eXport (IPFIX)."
http://net.doit.wisc.edu/ipfx/.

[24] Y. Vardi, "Network Tomography," *Journal of the American Statistical Association*, March 1996.

[25] C. Tebaldi and M. West, "Bayesian Inference on Network Traffic," *Journal of the American Statistical Association*, June 1998.

[26] J. Cao, D. Davis, S. V. Wiel, and B. Yu, "Time-Varying Network Tomography," *Journal of the American Statistical Association*, December 2000.

[27] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Trans. on Networking*, vol. 2, pp. 1–15, February 1994.

[28] A. Feldmann and J. Rexford, "IP network configuration for intradomain traffic engineering," *IEEE Network Magazine*, pp. 46–57, September/October 2001.
http://www.research.att.com/~jrex/papers/ieeenet01.ps.

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

[30] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 122–136, April 1994.

[31] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet Routing Instability," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 515–528, October 1998.

[32] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 392–403, August 2001.

[33] N. G. Duffield and M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 280–292, June 2001.

[34] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 226–237, June 2001.

[35] S. Bellovin, "ICMP Traceback Messages," October 2001. Internet Draft, Work in Progress.
http://search.ietf.org/internet-drafts/draft-ietf-itrace-01.txt.

[36] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-Based IP Traceback," in *Proc. ACM SIGCOMM*, (San Diego, CA), August 2001.