Statistical Science
2004, Vol. 19, No. 3, 472–498
DOI 10.1214/088342304000000206

Sampling for Passive Internet Measurement: A Review

Nick Duffield

Abstract. Sampling has become an integral part of passive network measurement. This role is driven by the need to control the consumption of resources in the measurement infrastructure under increasing traffic rates and the demand for detailed measurements from applications and service providers. Classical sampling methods play an important role in the current practice of Internet measurement. The aims of this review are (i) to explain the classical sampling methodology in the context of the Internet to readers who are not necessarily acquainted with either, (ii) to give an account of newer applications and sampling methods for passive measurement and (iii) to identify emerging areas that are ripe for the application of statistical expertise.

Key words and phrases: Traffic measurement, network management, sampling methods, estimation, packets, flows.

1. INTRODUCTION AND MOTIVATION

1.1 A Brief History of Internet Measurement

Why measure the Internet? The Internet is a large distributed system that comprises the network infrastructure, the hosts it connects, the traffic (in the form of packets) generated by the hosts and the protocols that govern the transmission of packets between hosts across the network. The variability of the underlying traffic demands and the complexity of their interactions present a challenge for service providers, who must ensure that network resources are matched adequately with demands. The aim of network measurement is to provide the data for network control, enabling the service provider to characterize the state of the network, the demands of traffic and its consumption of network resources, and the performance experienced by traffic on the network. Measurement systems monitor the system response to reconfiguration to determine if corrective actions are required. Actions operate over a range of time scales, from the deployment of new network infrastructure over a period of months, through tracing network attacks over minutes or hours, to control of traffic flows in close to real time, for example.

Too little data or too much? Internet service providers have in one sense too little data at their disposal, while in another sense they have too much. Too little, because it is not always possible to directly measure quantities of interest, these being only components in aggregate measurements. For example, troubleshooting packet loss requires knowing network performance on individual links, while in practice it may only be feasible to measure performance between two hosts, that is, the composite performance along a path that comprises several links.

A recent response to the problem of "too little data" was to develop tomographic methods for inferring individual components from collections of aggregate measurements. When troubleshooting loss, correlating performance measures along intersecting network paths reveals the performance on the intersection of those paths. Network tomography is reviewed in another paper in this issue; see also [1] and [16].

The "too much data" problem is that the volumes collected are truly enormous. A large service provider may collect data from tens of thousands of network interfaces. A single high speed network interface could in principle generate hundreds of gigabytes of (unsampled) flow statistics per day if fully utilized, while

Nick Duffield is a member of AT&T Labs-Research, Florham Park, New Jersey 07932, USA (e-mail: duffield@research.att.com).

the whole network might generate several gigabytes of simple network management protocol (SNMP) statistics per day. (See Section 3.1 for a description of flow and SNMP statistics.) Furthermore, the rate of data collection is growing, due to the requirement for ubiquitous fine grained measurements. As a result, routers and switches are being equipped with increasingly sophisticated measurement capabilities, providing ever more data to the service providers. Let us see why this has happened.

Why was comprehensive measurement not built into the Internet from the start? The current need for measurement capabilities stems from their relative lack of importance in the original design of the Internet. One of the strengths of the Internet is that endpoints do not need visibility into the details of the underlying network that connects them in order to transmit traffic between one another. Rather, the functionality required for data to reach one host from another is separated into layers that interact through standardized interfaces. For example, the transport layer provides a host with the appearance of a conduit through which traffic is transferred to another host: lower layers deal with routing the traffic through the network and the actual transmission of the data over physical links. Because of layering, there are scant mechanisms built into the standard Internet protocols that enable measurement of the network interior from endpoints: what happens in the interior is the business of the lower layers. (A basic diagnostic tool, `traceroute`, is an inspired hack that coerces the network interior into revealing some link level detail.)

The original Best Effort service model of the Internet reinforced the lack of detailed measurement capabilities. Best Effort offered no hard performance guarantees to which conformance needs to be measured. Basic robustness of connectivity—the detection of link failures and rerouting traffic around them—is a task of the network layer, and so need not concern the endpoints. Service is uniform for all traffic so measurement capabilities in routers need not differentiate among different traffic flows. Today only aggregate loss and utilization statistics are ubiquitously reported by router interfaces.

1.2 Data Volumes and the Need for Sampling

New services need differentiated measurements. The need for detailed measurements stems from the current movement of services beyond the original Best Effort model. Increasingly, service providers need to characterize fine scale traffic demands—even down to the level of individual customers—to better match available resources to them and so insulate traffic from the underlying variability of network conditions, for example, by segmenting resources. Measurement of the bytes used by each customer, perhaps differentiated by application, supports usage sensitive pricing. Customers want to verify conformance to agreed network performance targets by their own traffic. Service providers want to measure individual link performance to identify congestion and take corrective actions before performance targets are violated. The demands of real time applications require that measurement take place at fine time scales. The fact that detailed measurements are not facilitated in the original Internet design is reflected by the existence of a large number of research projects devoted to enhancing its measurement capabilities.

The need for data reduction. For many applications, measured data must be transmitted to collection points for storage and analysis. The massive volume of data has cost ramifications for the collection infrastructure. First, processing and storage resources on the routers and switches are comparatively expensive and scarce in practice; they are already employed in the regular work of routing and switching packets. Second, the transmission of measured data to the collection points can consume significant amounts of network bandwidth. Third, sophisticated and costly computing systems are required for analysis and storage of the data. These three factors motivate data reduction. However, there is an inherent tension between reducing data, on the one hand, and supplying sufficiently detailed measurements for applications, on the other. This tension is most evident at the observation point, where resources are typically the least available.

Data reduction is preferably carried out online in a single pass through the traffic stream to avoid buffering and reprocessing. Three methods are commonly employed:

- *Aggregation*. The combination of several data into a single composite, the components of which are then discarded. Aggregation is commonly additive, for example, finding the total traffic from a set of sources or over a time interval. Aggregates are used to provide a compact data summary when it is acceptable to lose visibility of the aggregate's components.
- *Filtering*. Selection of data based on the data values; unselected data are discarded. For example, traffic from a given source is selected. Filtering is useful to

drill down to a subset of traffic of interest, once that subset has been identified.

- *Sampling*. Random or pseudorandom selection of data; unselected data are discarded. For example, simple random sampling of packets. (There is overlap between filtering and sampling: implementations of sampling may be filters by the above definitions, albeit with an exceedingly complex selection rule.)

The key property that distinguishes the three methods is that filtering and aggregation require knowing the traffic features of interest in advance, whereas only sampling allows the retention of arbitrary detail while at the same time reducing data volumes.

Away from the observation points—at the collection point or at intermediate points in the collection infrastructure—the constraints for data reduction may relax somewhat, allowing more working storage and even the possibility of performing multiple passes through batches of data. This opens the door to employing other reduction methods in addition to aggregation, filtering and sampling.

1.3 Challenges in Sampling and Analyzing Sampled Network Data

There are a number of statistical challenges in sampling and analyzing network measurements:

- *The majority of available data already have been sampled during collection.* For the reasons described in the previous section, raw unsampled data are increasingly difficult to come by, so it is natural to ask, What does the sampled data tell us about the original network traffic? One of the recurring themes of this review is how to infer original traffic characteristics from sampled measurements.
- *Implementations of sample designs may be limited by technology and resources.* Technological constraints may limit the ability to use the sample design that is ideal from the purely statistical point of view. Equipment vendors may implement different realizations that approximate the ideal. What are the ramifications for statistical analysis and how do the results of analysis depend on the implementation details?

  Measurements themselves travel from the observation point (e.g., a router in the network) through a number of subsystems to the eventual data repository, possibly with some preprocessing or aggregation on the way. Each stage in the journey presents an opportunity for sampling. At which stage is sampling best performed?
- *The best choice of sample design depends on traffic characteristics.* Experimental studies show that network traffic exhibits dependence and rate fluctuations over multiple time scales, leading to heavy-tailed distributions for some traffic statistics. Sample design needs to take account of such behavior, for example, to control estimation variance.
- *The best choice of sample design depends on the statistics needed by applications.* There is no general agreement on which set of traffic statistics is most useful for network management. Whereas it is possible to optimize the sample design with respect to estimation of a given set of statistics, the design may be suboptimal for another set of statistics that could play an important role for some future application. For this reason, analyzing the trade-offs between statistical efficiency and flexibility is an important task for sample design.

1.4 Summary and Outline

This paper reviews the current practice and proposals for collecting sampled traffic measurements, together with the technological factors that constrain the manner is which collection is performed, and their interplay with the requirements of applications that use the measurements.

Section 2 describes the information available in packets and its uses for managing networks. A summary of protocols used to transmit data across the Internet is provided in the Appendix. Section 3 describes the two main technologies for collection of detailed passive measurements: packet monitoring, which reports in individual packets, and flow monitoring, which reports on temporal aggregates of flows of related packets.

Section 4 describes the classical sample designs (simple random, systematic and stratified; count and time based) that have been employed for traffic sampling. Some experimental work that compares their relative effectiveness in packet sampling is discussed, along with limitations that arise from traffic burstiness. Section 5 describes the statistical ramifications of employing estimated usage from sampled packet and flow measurements. Measurements may be omitted deliberately through sampling or accidentally due to loss in the measurement infrastructure that recovers and processes the measurements. Different techniques to account for these omissions during estimation are described.

Sampling flow records presents a challenge, because a small proportion of flows contain a large proportion of the traffic. Usage estimates based on sampled

flow records are therefore very sensitive to omissions of records of large flows. Section 6 discusses sampling methods that aim to reliably capture the usage of longer flows: the first method by correlated sampling of the packet stream during flow formation and the second method by sampling the completed flow records with probability proportional to size. A third approach, uniform sampling of the packet stream prior to formation of flow records, reduces the consumption of router resources during measurement. Usage-based billing is the application most sensitive to sampling variability in usage estimates. This section concludes with a discussion of how billing and sampling can be coupled to ameliorate the worst impact of sampling variability.

Sampling trades off the opposing goals of controlling estimation accuracy and sample volumes. Offered traffic loads vary systematically over daily and weekly periods, and in response to network reconfiguration. In practice it is necessary to adapt sampling rates to these variations so as to maintain the desired trade off. Section 7 discusses four recent proposals for dynamic adaptation in the collection of sampled packet and flow statistics. Application level flow statistics, including the frequency and lengths of flows, have been used in a number of measurement-based applications. Section 8 outlines some methods by which these statistics can be recovered from the flow records generated from a sampled packet stream, as currently collected by some routers.

The ability to sample all of a subset of packet that shares some common property or sample the same packet at multiple points, enables a new set of powerful measurement-based applications, including passive performance measurement and route troubleshooting. This ability can be realized through hash-based sampling, in which packet selection depends on a hash of the packet content. This is described in Section 9. Although hashing is deterministic, selection appears almost random if a strong hash function is used and if there is sufficient variability among the content of different packets.

Simple random sampling is easy to implement and the sampling operation itself requires virtually no additional storage. Relaxing these requirements somewhat opens new possibilities for data reduction, especially at the collector where resource constraints are less stringent than at the collector. Section 10 describes recent approaches for forming compact approximate representations of streams of objects. One focus is on employing data structures that construct compact representations of the dominant traffic components with a prescribed accuracy and in one pass through the data. In data squashing, a full set of data records is replaced by a weighted set of elements of the same form, although the most compact representations require more than one pass through the data.

Packet sampling methods for routers and other devices are currently being standardized within the Internet Engineering Task Force. Some vendors already support packet sampling operations. Should distinct sampling designs be different standard sampling operations or can they be regarded as just different implementations of a single standard sampling function? These matters are discussed in Section 11. Section 12 concludes with a look to future challenges for data reduction in passive measurement.

2. INFORMATION IN TRAFFIC AND THE USES OF MEASUREMENT

2.1 Internet Protocols and their Functions

Data are transmitted between hosts across the Internet in packets constructed according to a layered set of protocols that specify the packet layout and the function of the packet content; this is described more fully in the Appendix. Each protocol is represented in the packet through content that comprises the payload (which comprises the data to be mediated by the protocol) and the protocol header (which comprises control information). The content of one protocol may form the payload of a lower layer protocol; this is called encapsulation.

In passive packet measurement we are primarily interested in the information contained in the protocol headers. Network protocol headers are used by routers to forward the packet toward its destination. Most often the network protocol used is the Internet protocol (IP) [82], although encapsulation by the connection oriented multi protocol label switching (MPLS) [4] is becoming more common. The network protocol encapsulates the transport protocol used by end-hosts to control end-to-end transmission and to direct packets toward the intended application at the end-host. Currently, two transport protocols predominate: the user datagram protocol (UDP) [81] and the transmission control protocol (TCP) [83]. The transport protocols encapsulate the data to be transmitted, which may itself be generated through application level protocols such as the hypertext transfer protocol (HTTP) [6, 43] used for Web transfers.

2.2 The Information in Protocol Headers

Packet protocol headers are of central interest in network management for three reasons:

- *Protocol headers determine packet treatment by the network.* For example, the packet's routing and its priority relative to other packets. The ability to measure the rates of traffic classified according to header fields is necessary to understand patterns of use of network resources by traffic and how they respond to changes in network configuration.
- *Attribution of packet origin.* The IP header contains the source and destination IP address of the packet. In combination with routing, topology and other information, the addresses can be used to attribute the host or administrative entity responsible for the packet's presence. Some network attacks obscure their origin by forging the IP source address of attacking packets; this is known as *spoofing*. Methods that alleviate the problem of identifying the origin of a spoofed attack by measuring packet *paths* are discussed in Section 9.
- *Characterization of applications.* UDP and TCP protocol headers include a port number field that is used to direct the packet to the intended application at its destination; see [87]. The values used are standardized or conventional for some applications, and so the responsible application can often be inferred by inspection of the packet's port numbers. For example, some networks block access to the Internet by file sharing applications, based on port numbers used by the applications. (On the other hand, conventions on port usage may be ignored precisely to evade such characterization.) Security applications can identify packets involved in a network attack by matching header fields to known signatures of known attacks. For example, the Slammer worm is characterized by packets of a certain length that employ a specific port [74]. The state of some protocols, for example TCP, may be tracked through observation of flags set in their headers.

The actual application data communicated in packets by the host are usually not of interest for network management since, in most cases, the functioning of the network is not correlated with the values the data take. However there are some exceptions, including (i) data contained in packets of routing protocol, which are useful for understanding the routing state of the network, and (ii) URLs contained in the payload of certain HTTP packets, which are useful for determining the usage of content resources at Web servers.

2.3 Applications of Usage Measurements

Many network management applications employ measured traffic usage, in packets or bytes, that is differentiated according to header fields into classes at some granularity that depends on the application requirements. A selection was described by Cáceres et al. [8] and in the review by Grossglauser and Rexford [49]. Here are some examples:

- *Service development.* Service providers track the growth of new applications (as identified by TCP/UDP port numbers) and identify potential new customers that use them (from packet IP addresses not administered in their network).
- *Heavy hitters.* Determining the dominant components within a class of traffic, for example, the most popular websites, based on the IP destination address of HTTP requests.
- *Security applications.* Detecting usage indicative of network intrusions, including changes of patterns of usage of specific protocols and TCP/UDP ports, and most active hosts and networks involved; see [86] and [94] for applications of sampling methods to these topics.
- *Network engineering.* A service provider determines the intensities of traffic between sets of source and destination addresses that is carried over a congested link. This information could be used to examine the feasibility of rerouting portions of the traffic away from the congested link; see [39, 40].
- *Chargeback.* A corporate intranet apportions its costs to constituent organizations, based on usage that originates in the organizations' IP address ranges.
- *Customer billing.* A service provider charges customers, as identified by IP address, for byte usage. The rate of charge may depend on application type (as identified by TCP/UDP port numbers). Charging based on remote address (e.g., whether on or off the provider's network) also has been proposed (see, e.g., [68]). Information on some commercial examples of the use of flow records for billing purposes can be found in [13]. The use of packet samples for billing is proposed for InMon's sFlow [55].

The accuracy requirements of these applications are quite different; the list above is (roughly) ordered by stringency, where customer billing is the most stringent. There are strong legal reasons for not overbilling customers and it would not be good customer relations. (Some regulatory environments may prohibit billing on

estimated usage.) On the other hand, network management applications can probably tolerate errors of quite a few percent in estimating usage by a class of traffic. The ramifications of sampling for the estimation of network usage are a central theme of this review.

2.4 Other Measurement Applications

Although estimating per class rates from sampled measurements at a point forms a substantial part of this review, sampling is also compatible with measurement or inference of other traffic properties. This review describes two examples:

- *Path measurement.* Section 9 describes a method to measure entire paths of packets through the network. As well as determining per class usage along paths, this method enables the *passive* measurement of network path performance, route troubleshooting and network attack tracing.
- *Traffic structure.* Sampling can present a challenge to the determination of detailed structural properties of traffic (such as the duration of traffic flows) or the composition of application transactions (such as a Web browsing session) in terms of multiple application flows. An inference method whereby sampled measurement are used to infer detailed structural properties of the original unsampled flows of traffic is described in Section 8.

Passive measurement has been used to reveal complex temporal properties of network traffic (see, e.g., [67]). These measurements usually have been performed on special purpose monitors that capture all traffic on a network link. It remains a challenge to determine the extent to which the perception of these properties, and changes in them, is compatible with routine sampled measurement.

3. PASSIVE TRAFFIC MEASUREMENT

3.1 Active and Passive Measurement

Network measurements are commonly divided into two categories: active and passive. The two types of measurement generally focus on different aspects of network behavior. In active measurement, probe packets are sent between hosts. Since probes can be launched from any accessible host, active probing is well suited to end-to-end performance measurement. End-to-end packet loss can be inferred from gaps in probe sequence numbers observed at the destination, while end-to-end delay is determined by comparing (suitably synchronized) time stamps placed in each probe by the source and destination. Packet content is of interest insofar as it influences performance characteristics, such as through differential treatment by routers of packets based on their IP header fields (e.g., the type of service field).

Common active measurement tools such as `ping` and `traceroute` allow users to measure roundtrip performance from a host without requiring privileged access to routers in the network interior. (Although `ping` and `traceroute` do require the destination to respond to Internet Control Message Protocol (ICMP) packets, an ability which may be administratively disabled.) Bulk throughput can be estimated using the `treno` tool [71], which creates a probe stream that conforms to the dynamics of TCP.

In passive measurement, routers or other hosts measure existing traffic passing through or destined to them. Passivity requires both that the packet content is unaltered by measurement and that the operation of the measurement infrastructure introduces at worst negligible disruption to the normal passage of traffic in the network. For example, a router must not deplete its resources by measurement to the extent that its normal functions of routing and forwarding packets are impaired. Of course, exporting measurements to a collector consumes network resources, causing a least a small perturbation in the traffic patterns, but measurement traffic ought not to consume more than a fair share of bandwidth under normal operating conditions. (On the other hand, during overload conditions, it may be desirable for measurement traffic to claim more than its fair share of bandwidth so as to maintain collection of measurements.) Data reduction before transmission clearly has a role to play in limiting the measurement transmission bandwidth.

The rest of this section is devoted to passive measurement. Three forms of passive measurement are described: SNMP data in Section 3.2, packet monitoring in Sections 3.3 and 3.4, and flow statistics in Section 3.5.

3.2 MIBs and SNMP Statistics

Passive network measurements are commonly collected in three ways: (1) polling management information base (MIB) data from routers, (2) packet monitoring and (3) flow monitoring. In the first of these, routers keep coarse grained statistics in an MIB. A particular version, MIB-II, is standardized and hence available across many network elements. However, the information is only available in a form which is spatially highly aggregated: counters of packets and bytes

transmitted and lost at an interfaces. These quantities are recovered by polling the routers using SNMP, the simple network management protocol [9]. (Although the name refers to the protocol, it commonly connotes these statistics.) The SNMP statistics are commonly polled every 5 min, although polling at intervals down to a few seconds is claimed to not impair router performance; see [18]. Another standardized MIB, RMON [98], was designed for remote monitoring. An RMON agent operating in a network device can be configured to compute traffic statistics, and recognize and respond to defined network conditions, for example, by capturing packets or raising alarms. This versatility makes RMON complex, and implementations are limited to low speed interfaces. However, it is not well suited to the continuous measurement and export of detailed traffic data. In the remainder of this section, we describe packet and flow monitoring currently used for this purpose.

3.3 Packet Monitoring

Packet monitoring entails passively copying a stream of packets, then selecting, storing, analyzing and/or exporting information on these packets. Until recently packet monitoring was performed exclusively by special purpose hosts installed in the network; see, for example, [3, 8, 21] and [42]. A copy of the packet stream is brought to a monitor in one of three ways: by copying the physical signal that carries the packets (e.g., with an optical splitter) and bringing the signal to an interface on the monitor; by attaching the monitor to a shared medium that carries the traffic; by having a router or switch copy packets to an interface to which the monitor is attached.

Packet monitors have to cope with some formidable demands on their resources, particularly on the processing bandwidth needed to work at the full line rate of increasingly high speed links. Restricting data capture to some initial number of bytes of the packet is a common way to control data bandwidth at the monitor. This is a reasonable solution, since the IP header and other protocol header information is located at or near the start of the packet. Even so, widespread continuous collection, transmission and storage of unreduced packets has been infeasible for a number of years due to the immense volumes of data relative to the capacity of systems to collect them; see [2] for an early reference and [72] for a recent one. Collection of full packet header traces is feasible only for limited durations. Instead, for applications that require continuous monitoring over an extended period, it is common to perform analysis at or near the monitor by forming flow records (see Section 3.5 below) or other aggregate statistics (see [3]), or a more general stream querying functionality (see [21]). Collection of packet IP and transport headers is commonly performed using `tcpdump` [57] or its variant `windump` [99]. Depending on the traffic load and processing power at the measurement host, these tools may also be able to capture parts of the packet payload.

3.4 Embedded Packet Monitoring in Network Elements

Deployment of packet monitors is limited by equipment availability and administrative costs. A more recent approach to packet monitoring was to embed the passive measurement functionality within network elements such as routers and switches. Once packet monitoring capabilities become available in network elements, packet measurement can become ubiquitous in the network. However, little or no capabilities for measurement analysis are expected to be available in routers and switches, because they generally lack the additional computational resources for this purpose. Instead, some form of data reduction is required, both in the selection of information from packets and in the selection of packets to be reported on. Some packet sampling capabilities are becoming available in routers, for example, sampling in InMon's sFlow [80].

Packet selection capabilities for network elements are currently being standardized by the Packet Sampling (PSAMP) Working Group of the Internet Engineering Task Force (IETF); see [53]. The aim of this work is to define a set of packet selection capabilities which are simple enough to be ubiquitously deployed, yet rich enough to support the needs of measurement-based network management applications. Although specific selection operations are yet to be finalized, it is likely that this will include filtering and various forms of sampling.

3.5 Flow Records

A flow of traffic is a set of packets with a common property, known as the flow key, observed within a period of time. Many routers construct and export summary statistics on packet flows that pass through them. Ideally, a flow record can be thought of as summarizing a set of packets that arises in the network through some higher level transaction, for example, a remote terminal session or a Web-page download. In practice, the set of packets that are included in a flow depends on the algorithm used by the router to assign packets to flows. The

flow key is usually specified by fields from the packet header, such as the IP source and destination address and TCP/UDP port numbers. Flows in which the key is specified by individual values of these fields are often called *raw* flows, as opposed to *aggregate* flows in which the key is specified by a range of these quantities.

Flow statistics are created as follows. When a packet arrives at the router, the router determines if the flow is active, that is, if statistics are currently being collected for the packet's key. If not, it instantiates a new set of statistics for the key. The statistics include counters for packets and bytes that are updated according to each packet matching the key. When the router judges that the flow is terminated, the flow's statistics are exported in a flow record and the associated memory is released for use by new flows. A router commonly terminates a flow if one of the following criteria is met: (i) inactive flow or interpacket timeout, that is, the time since the last packet observed for the flow exceeds some threshold; (ii) protocol level information, for example, a TCP FIN packet that terminates a TCP connection; (iii) memory management, that is, termination to release memory for new flows; (iv) active flow timeout, that is, to prevent data staleness, flows are terminated after a given elapsed time since the arrival of the first packet of the flow.

Flow definition schemes have been developed in research environments (see, e.g., [3] and [14]) and are being standardized by the IP Flow Information Export (IPFIX) [54] Working Group of the IETF. Examples of flow definitions employed as part of network management and accounting systems can be found in Cisco's NetFlow [12], QoSient's Argus [84], Riverstone's LFAP [88] and XACCT's Crane [101]. A flow record typically includes the properties that make up a flow's defining key, the arrival times of the first and last packets, and the number of packets and bytes in the flow.

Flow records yield considerable compression of information, since a flow is summarized in a fixed length record, regardless of the number of packets in the flow. The trade-off is loss of detail of the timing of packets within the flow. The compression factor depends on the composition of traffic: it is greater for long flows and smaller for short flows. For traffic mixes observed in backbone traffic, byte compression factors for IP and transport headers versus NetFlow records of 25 or more are commonly attainable.

4. SELECTION OPERATIONS FOR SAMPLING

In this section we describe how classical sampling methods are currently applied to the sampling from streams of packets or flows. At a formal level it makes no difference whether we sample packets or flows; we use the term "object" to denote either. We examine how the choice of sampling design is influenced by both the technological setting and the underlying statistical properties of the objects to be sampled. We also report on studies that compare the accuracy of some of the sampling methods.

Sample designs for packet sampling were categorized initially by Amer and Cassel [2] and employed in subsequent work; see [15,103]. We employ this categorization in parts and also codify definitions to some measurement terminology in common use. Although the term "sampling" is widely used, we often use the wider term "selection" because some of the methods described here do not entail sampling except in a degenerate sense. Rather, we wish to include methods that are deterministic in the sense that the selection decision for an object is determined exactly by its content, that is, with no randomness. In the terminology emerging in the PSAMP Working Group of the IETF [53], such deterministic selection operations are called filters; all other selection operations are called sampling operations. This is intuitive in the sense that if the selection of an object is not determined entirely by its content, there must be some form of sampling taking place.

4.1 Model for Objects under Measurement

We start with a description of the objects to be measured and their attributes. The stream of objects passes an observation point at which they are sampled. The stream is modeled as a (marked) point process that comprises the sequence of pairs $\{(t_i, c_i) : i = 1, 2, \dots\}$. Objects are indexed by the count i of their order of arrival at the observation point and t_i is the time of observation of object i, so that $t_i \le t_{i+1}$. The mark c_i is the content of object i and takes its value in some set C. We can regard C as a set $\{0, 1\}^\ell$ of binary vectors of some length ℓ. It is the entirety of the object under measurement, for example, header plus payload for a packet. Our two examples of objects are packets and flow records. Generally, these objects can have variable length; in each case ℓ is taken to be the maximum possible number of bits that can be occupied by the object.

4.2 Selection and Triggers

Describing a given selection method entails specifying the rule by which the counts i of selected objects are derived from the full observed stream of objects. The structure of the rules used in practice often makes it convenient to use a two-step specification. The first step is to specify a set of triggers. A trigger is an event or time at which some type of packet selection is initiated (the trigger is said to "fire"). In the second step, when a trigger fires, a number of the objects are then selected from the stream according to a specified rule over some period of time. No further objects are selected until the next trigger fires.

We consider two types of triggers: count driven and time driven. Count-driven triggers form an increasing sequence of counts $\{i_n : n = 0, 1, 2, \dots\}$, denoting the object at which each trigger fires. Likewise, time-driven triggers form an increasing sequence of times $\{\tau_n : n = 0, 1, 2, \dots\}$. In the simplest cases, the objects selected are those at which the count-based triggers fire. An example of a time-driven trigger is one in which the triggers form a Poisson process, each trigger initiating selection of the first object (if any) that is observed before the next trigger fires. In a more complex example, triggers fire upon observation of an object with a specified content (see Section 4.3 on filtering), after which any objects observed in the next t seconds are selected. Triggers and selection of this type are configurable in RMON probes [98].

Ideally, the model of the trigger process is sufficiently flexible to allow response to specified patterns of multiple packets arriving in an interval. In practice, two technological requirements can constrain such temporal dependence of triggers and packet selection. First, some network control applications require measurements to be transmitted to them as soon as possible after the measurement is made. For this reason we usually assume that triggers are adapted to the object stream in the sense that the selection decision for a given object cannot be contingent on future objects in the stream. More formally, we want to be able to determine whether trigger n has already fired by arrival of object i (i.e., $i_n \le i$), from inspection of i, the first i objects $\{(t_j, c_j) : 1 \le j \le i\}$, and possibly some random variable whose distribution depends only on the first i objects. The second requirement is that dependence on the past does not require use of large amounts of storage. A simple example that fulfills this requirement is independent sampling. In general, any influence of the past on the current sampling decision should be through some small number of variables that are easily stored.

4.3 Filtering

Filtering is the selection of the objects based only on their content. The object i is selected if its content c_i lies in a specified subset of the content set C. Most commonly, filters are specified as mask/match operations. These are specified by two members c_{mask} and c_{match} of the content set C, which is regarded as a set of binary strings of a fixed length. The mask c_{mask} is applied to the content c_i by taking their bitwise logical AND. The object is selected if the result is equal to c_{match}.

Objects with variable format present challenges for filtering, since a given field of interest may not have a constant offset from the start of the object. A specific example concerns IP packets with options. The header of an IP packet with options is longer than in a packet without options. Fields in the IP payload, such as TCP/UDP port numbers, occur at a greater offset from the start of the packet. Applying a fixed length mask/match to the whole packet under the expectation that IP options are not present results in misalignment of the mask with its intended target in a packet with options. One approach in this case is to parse the packet for the desired fields and apply the relevant mask/match to each field separately. The packet is then selected if it passes each constituent filter. However, such parsing increases the computational cost for filtering.

4.4 Uniform Sampling

This section reviews classical sampling methods—systematic, simple random and stratified sampling—and their common applications in passive Internet measurement.

4.4.1 *Systematic sampling.* In count-based systematic sampling, the triggers are $i_n = nN + i_0$, the occurrence of objects with integer period $N > 0$. In the simplest case, the object i_n is selected. More generally, $M \le N$ objects $i_n, \dots, i_{n+M-1}$ are selected. An example is the capture of subsets of successive packets; see [59] for specification of a commercial implementation. Sampling of consecutive packets may be useful for understanding the detailed dynamical behavior of packet streams. In time-based systematic sampling, triggers fire at times $\tau_n = nT + \tau_0$. Selection takes place after each trigger has fired, for example, selection of the next arriving object or all objects arriving within a time t of the trigger.

Systematic sampling is very straightforward to implement: Set a counter to the sampling period, decrement on each packet, select a packet on reaching zero,

then reset the counter and repeat. However, systematic sampling is vulnerable to bias if the objects being sampled exhibit a period which is rationally related to the sampling period, since samples are taken only at a discrete set of phases within the period. Potential sources of periodicity are timers in protocols and periodically scheduled applications. A further drawback is that periodic sampling is to some extent predictable and, hence, open to deliberate manipulation or evasion; see [77] and [78] for further discussion.

4.4.2 *Random additive and simple random sampling.* The potential problems of systematic sampling are avoided by suitable use of random additive sampling. Here, the intervals between successive triggers are independent random variables with a common distribution. (Periodic sampling is a degenerate case where the random variable takes a constant value.) The advantages of random additive sampling for Internet measurement were highlighted by Paxson [77]. It avoids synchronization problems. Choosing the intervals to be geometrically distributed (for count-based sampling) or exponentially distributed (for time-based sampling) avoids predictability. Furthermore, Wolff's Poisson arrivals see time averages (PASTA) property [100] ensures that any empirical mean over sampled objects is an unbiased estimator of the corresponding population mean. For these reasons, random additive sampling is recommended in standards for performance metrics; see [78].

A simple implementation of random additive sampling is to generate, immediately following a given trigger, the length of the interval until the next trigger. However, when the interval distribution has unbounded support—for example, with the exponential or geometric distribution—some generated intervals will not fit in storage unless a cutoff is applied. The special case of geometric random additive sampling with mean intersample count m is equivalent to simple random sampling with probability $1/m$. It can be implemented by making a sampling decision for each object, although this is computationally more costly than generating random intersample times.

4.4.3 *Uniform stratified random sampling.* In stratified random sampling, objects are assigned to strata according to an attribute; then a number of elements are drawn randomly from each stratum. Stratification reduces the variance of single packet statistics if the variance between strata is greater than the variance within strata. In uniform stratified sampling, the number of elements in each stratum is the same, as is the number selected from each stratum. Thus the marginal selection probability for each object is identical, but some combination selections are disallowed. Criteria for choosing strata, including optimal selection of stratum width, were discussed by Cochran [17] and Krishnaiah and Rao [63].

Uniform stratification based on packet count was examined by Claffy, Polyzos and Braun [15]. Each group of N successive packets forms a stratum, then some number $M < N$ of each are drawn at random. Note that this form of sampling is adapted in the sense of Section 4.2, because, since the size of the stratum is fixed, the random positions of the selected packets can be generated in advance for each stratum.

4.5 Comparison of Uniform Packet Sampling Methods

The relative strengths of the uniform sampling methods have been investigated by several authors in the context of packet sampling. Amer and Cassel [2] discussed the maintenance of estimates of mean and variance of packet characteristics, such as packet length, from sampled measurements. Claffy, Polyzos and Braun [15] used packet traces to compare distributions of packet lengths and interarrival times of the full traces with those obtained by sampling their time series. Accuracy was compared using a chi-squared measure of discrepancy; see also [78, 81]. Time-based sampling was found to be uniformly less accurate than count-based sampling. This is due to traffic patterns: It is well established that Internet traffic is bursty over a range of time scales (see, e.g., [67]). Consequently there can be inhomogeneous bursts of many packets with small interarrival times. Time-based sampling methods more easily miss these than a count-based method, and estimators built on them have higher variance. On the other hand, accuracy among the count-based sampling methods was very similar. Recent work by Zseby [103] compared sampling methods in the context of validation of service level agreements for packet delay.

Although selection probabilities for single packets are identical for the three methods, selection probabilities for multiple packets are generally not. For example, back-to-back packets that share a common property (e.g., a flow key) are never sampled using systematic sampling. The use of different packet sampling methods prior to flow formation is discussed further in Section 6. The ramifications of forming standards for packets for packet sampling are discussed in Section 11.

4.6 Nonuniform Probability Sampling

Nonuniform probability sampling is a design in which objects are sampled with a probability that can depend on their content. This design can be used to weight sampling probabilities in order, for example, to boost the chance of sampling objects that are rare but important. For example, a small number of flow records are important because they report a large fraction of the traffic; this application is treated in more detail in Section 6.3.

Assume the content of each object i contains a quantitative field x_i and that the object is selected with probability p_i. With nonuniform p_i, some generalization of the usual sample mean is required to form an unbiased estimator of the population mean $\overline{x}$ of x. Suppose that out of a population of size N, objects with sizes $x_1, \ldots, x_n$ are randomly selected. The Horvitz–Thompson estimator $N^{-1} \sum_{i=1}^{n} x_i / p_i$ is an unbiased estimator of $\overline{x}$ when all $p_i > 0$; see [52]. If N is not known, it has an unbiased estimator $\sum_{i=1}^{n} 1/p(x_i)$. Uniform sampling is special case: The expected value of n/N is the uniform sampling probability. The Horvitz–Thompson estimator has the useful property that its variance itself has a relatively simple unbiased estimator. However, although unbiased, this variance estimator can have large enough variance to take negative values, depending on the degree of correlation in selection of objects. Related variance estimators that trade off various drawbacks were described by Thompson and Seber [95].

A particularly attractive form of nonuniform probability sampling is *probability proportional to size* (PPS) sampling. Here it is assumed that some auxiliary size variable y_i is known for all N objects from which selection takes place. Choosing p_i proportional to y_i is attractive because if x_i is roughly proportional to y_i, then the terms in the Horvitz–Thompson estimator will be roughly constant, leading to low variance. In the networking context, an example is estimating the mean of the packets x_i from sampling flows from PPS sampling based on the numbers of bytes y_i.

If all objects are already available in a data base, we could, in principle, sample using probabilities $p_i = y_i / \sum_{i=1}^{N} y_i$. This is not suitable for sampling from a stream of objects if sampling is to be adapted in the sense of Section 4.2, since object sizes are not known in advance. Section 6 describes a method for conditionally independent size-dependent sampling of flow records that is adapted; some nonadapted sampling methods better suited to sampling from data bases are listed in Section 7.

4.7 Emulation of Random Sampling and the Generation of Pseudorandom Variates

Although pseudorandom number generators with well understood properties have been developed (see, e.g., [65]), they may not be the method of choice for implementations of random sampling in settings where computational resources are very scarce. For example, in a router, the budget for computational operations per packet is very small.

An alternative approach is to use object content to generate pseudorandom variates. For this reason, some implementations of random sampling can actually be regarded as filters, albeit complex ones. The statistical ramifications of this approach need to be carefully examined. For example, unexpected dependence in the content of different objects can lead to dependence in sampling. If the mechanism by which content translates to the pseudorandom variates is not sufficiently complex, it may be possible to subvert sampling with a stream of appropriately crafted objects. An example in Section 9 shows how both of these potential problems can be ameliorated.

Bear in mind the difference between nonuniform probability sampling and the generation of pseudorandom variates from content. The former is a sample design where the selection probabilities depend on content; the latter is an implementation choice to exploit object content to emulate random choice (e.g., simple random sampling).

4.8 Aggregation

Section 1.2 listed aggregation as one of the alternative methods for data reduction. Although aggregation is not a selection operation, for completeness we formalize the most common notion of aggregation employed in Internet measurement. Typically this involves aggregating the byte or packet usage of a set of objects that share a matching key.

We view the contents of each object as including two parts (a, c), where $a \in A$ is the field to be aggregated over and c is an attribute that is to be combined through addition (e.g., a byte size). Given a set of objects with contents $\{(a_i, c_i)\}$ and a partition of A into grains $\{A_n\}$, the aggregates take the form $(A_n, \sum_{i\,:\,a_i \in A_n} c_i)$ for n such that $\{i : a_i \in A_n\}$ is nonempty. In other words, for each grain A_n present in the set of objects under aggregation, the aggregate records the total of the attribute values c_i present in all objects i for which $a_i \in A_n$.

As an example, the aggregate represents the total of the bytes c_i reported in IP flow records i, broken down

according to the destination IP routing prefix. Here, a_i is the destination IP address of the flow, and $\{A_n\}$ is the covering of the routeable IP address space according to an IP prefix.

5. PACKET SAMPLING AND VARIABILITY IN THE ESTIMATION OF NETWORK USAGE

This section discusses renormalization of usage estimates from reports on uniformly sampled packets and how to take into account the effect of loss of reports in transit.

5.1 Renormalization of Measured Usage with Target Sampling Rate

We saw in Section 4.6 that each sampled value, such as a packet or flow length, must be renormalized through division by its selection probability so as to obtain an unbiased estimator of the original. Let there be M packets in a given class, of which m are selected when sampling independently at rate p. We form an unbiased estimate $\widehat{M}_1$ of the total packets M in that class by $\widehat{M}_1 = m/p$. Bytes are estimated similarly. Let the M original packets in the class have sizes $b_1, \ldots, b_M$ with total $B = \sum_{i=1}^{M} b_i$. Then $\widehat{B}_1 = p^{-1} \sum'_{i=1,\ldots,M} b_i$ is an unbiased estimator of B, where $\sum'$ denotes the random sum over sampled packets only.

5.2 Renormalization of Measured Usage with Attained Sampling Rate

An alternative renormalization uses the *attained* sampling rate, which may be calculated when sufficient information is provided in the measurements. In InMon's sFlow [55], routers include the cumulative count of all packets arrived at the observation point—whether sampled or not—in each sampled packet report. By subtraction of counts, the collector can calculate the *pool size*, that is, the number of packets that arrived at the observation point between two given packets for which reports reached the collector. Let the pool size be N, of which n packets in all classes were sampled. The attained sampling rate for all traffic is n/N. Assuming this rate applies uniformly across all constituent classes, the estimate of the total packets in the class of interest is obtained by dividing the number m of sampled packet in the class by the attained sampling rate, yielding $\widehat{M}_2 = mN/n$.

Analogous estimates for original bytes can be formed. In principle the attained sampling rate could be formed using either packet counts or byte counts. In practice, the estimate derived from packet counts is preferable: the estimate derived from byte counts has higher variance due to the variability of packet sizes.

The attained loss rate between two received packets is independent of which intervening packets were lost. Thus renormalization with the attained loss rate is less sensitive to deviations from independent sampling than renormalization with the target loss rate, provided that the deviations affect all traffic classes equally.

5.3 Variance of Usage Estimates from Uniform Random Sampling

Assuming independent packet selection with probability p, $\widehat{M}_1$ has coefficient of variation $s_1 = ((1-p)/(pM))^{1/2}$, but $\widehat{M}_2$ offers some reduction in variance. Suppose that $N, M \to \infty$, with the proportion of packets M/N in the class under consideration converging to r. An application of the delta method [91] shows that the coefficient of variation of $\widehat{M}_2$ converges to $s_1\sqrt{1-r}$. The byte estimator $\widehat{B}_1$ has coefficient of variation $((1-p)\sum_{i=1}^{M} b_i^2/p)^{1/2}/B$. Since packet sizes are bounded above by the maximum transmission unit $b_{\max}$ of the link at which measurement takes place, we can usefully bound this error above by $\sqrt{b_{\max}/(pB)}$.

Since the coefficients of variation are inversely proportional to the square root of the actual usage, larger contributions to usage are more reliably estimated than smaller ones. This is useful for applications. For example, the relative error in estimating high volume contributions to network usage is smaller than for general classes. This property was exploited by Jedwab, Phaal and Pinna [58] to identify heavy hitters through packet sampled usage. Their scheme assumes that only limited storage is available for ranking traffic classes by usage. If instantiating a new class would exceed storage capacity, the ranking information is truncated by discarding all classes except a certain number of the highest ranking. The probability of misranking can be controlled to be small.

Usage-sensitive charging can also benefit from increased estimation accuracy by lengthening the billing period over which samples are collected; see [55] in the context of packet sampling and [28] in the context of flow sampling. For the purpose of cluster analysis for intrusion detection, Taylor and Alves-Foss [94] used differential packet sampling rates on traffic attributed to different applications—according to TCP/UDP port number—to take a comparable number of samples from each class.

5.4 Report Loss Viewed as Sampling

Packet reports are subject to loss at the observation point, in transmission or at the collector. Report loss can be viewed as an additional form of sampling, albeit with a priori unknown sampling properties.

To apply the correct renormalization to measured usage for unbiased estimation, the transmission rate of the reports must be known. A strength of renormalizing with the attained sampling rate as described in Section 5.2 is that the attained sampling rate between observation point and collector already combines the effect of all packet loss between the two points, whether due to sampling or packet loss.

When renormalizing usage with the target sampling rate, the report transmission rate must be determined independently. If the observation point inserts sequence numbers in reports prior to transmission, the attained transmission rate can be estimated using the method described in Section 5.2. Renormalization is then performed using the product of the target sampling rate with the attained transmission rate. For export over the wide area Internet, the pattern of report loss is expected to be bursty due to the observed burstiness of network congestion events; see [102] and references therein. Following the remarks at the end of Section 5.2, renormalization with the attained loss rate offers robustness to bursty loss provided the bursts affect all traffic classes equally.

6. SAMPLING IN THE FORMATION AND COLLECTION OF FLOW RECORDS

There are three sets of resources involved in the production of flow records: those at the router involved in processing packets to compile the flow statistics, those involved in exporting and transmitting the completed flow records to their collection point, and those used to analyze the statistics at the collector. In this section we describe how usage of each of these resources can be controlled through sampling.

The requirements for flow sampling are fundamentally different than those for packet sampling due to the statistical properties of flows. Whereas packet size is bounded by a maximum that is dependent on the transmission technology, experimental studies have shown that the distribution of flow lengths is heavy-tailed; in particular, a large proportion of the total bytes and packets in the traffic stream occurs in a small proportion of the flows (see, e.g., [41]). This makes both uniform sampling and uncontrolled sampling due to transmission loss far more problematic for flow records than for sampled packets, since omission of a single flow report can have a huge impact on estimated usage.

Currently, NetFlow exports flow records using UDP transport, which is not equipped with any capability for reliable transmission. Capability for reliable export is being incorporated in the relevant standards under development in the IPFIX Working Group of the IETF [54]. In the short term, report loss can be minimized by locating staging collectors close to routers, for example, in switching centers that house a number of routers. The local networks over which reports are transmitted are managed so as to be congestion-free, at least under most operating conditions. (Although denial of service attacks may give rise to bursts of records of small flows that, if unchecked, may lead to congestion.) The staging collectors can then retransmit the flow records to their ultimate destination over the wide area Internet using a reliable transport protocol such as TCP. The staging collectors may also perform analysis. However, even with future reliable transport from the observation point, staging collectors may still be desirable for scalability, since they can also perform data reduction and field queries; see [29] for a description of a such a system.

The rest of this section describes sampling methods that avoid the pathologies of uniform sampling of flow records. Sampling of packets prior to the formation of flow records is described in Section 6.1. Sections 6.2 and 6.3 describe nonuniform packet and flow sampling schemes that are better adapted to the statistical properties of flows.

6.1 Flow Records: Packet Selection

The main resource constraint for forming flow records is at the router flow cache in which the keys of active flows are maintained. To look up packet keys at the full line of the router interfaces requires the cache to operate in fast, expensive memory. Moreover, routers carry increasingly large numbers of flows concurrently, necessitating a large cache. By sampling the packet stream in advance of the construction of flow records, the time window available for flow cache lookup for each packet is prolonged, enabling storage to be carried out in slower, less expensive memory.

Systematic sampling based on packet count has been employed by Cisco for their Sampled NetFlow feature. The more recently introduced random sampled NetFlow [85] feature employs stratified sampling based on arrival count. As in Section 4.4.3, one packet is selected at random out of every window on N consecutive arrivals. The distinctive feature relative to periodic sampling is that two consecutive packets can be sampled.

However, since more than two consecutive packets are never sampled when $N > 1$, longer backlogs do not develop provided the mean rate of sampled packets can be accommodated.

Packet sampling versus flow records. If the sampling period is much larger than the typical flow length, then typically only one packet per flow will be reported. If this were the rule, we might just as well sample packets without constructing flow records. This would save resources at the router since there would be no need to cache the single packet flows until expiration of the interpacket timeout.

So are flow records redundant in the face of packet sampling? On the one hand, there are many short flows. Web traffic is a large component of Internet traffic, in which the average flow length is quite short, around 16 packets in one study [41]. On the other hand, there are many reasons to expect that longer flows will continue to account for most of the traffic: (i) consumer Internet traffic is increasingly dominated by file sharing and downloading applications for which flows of hundreds and even thousands of packets are the norm; (ii) flows of Internet telephony packets are also expected to be long lived; (iii) in Virtual Private Networks, traffic from many sources will be seen as a single aggregate flow in the network core. Thus, unless packet sampling periods become large compared with the lengths of these flows, flow statistics will still afford useful compression of information.

Resource usage: Sparse flows and flow splitting. Sampling can *increase* the number of flow records generated from an application flow for which the typical time between sampled packets exceeds the flow interpacket timeout T. Consider sampling packets with probability $1/N$ from a flow that comprises n packets over a duration t. If

$$\frac{t}{T} > \frac{n}{N} > 1, \tag{1}$$

then the typical time between sampled packets, tN/n, exceeds T and more than one packet is typically sampled. Such flows were called *sparse* by Duffield, Lund and Thorup [30]. These authors derived expressions for the means of the rate of production of flow statistics and number of active flows, given the statistics of original application flows. In experiments, flows from file sharing and streaming applications were found to be most susceptible to splitting: they have many packets and are comparatively long-lived.

6.2 Flow Records: Cache Selection

We now describe sampling schemes which reduce the amount of fast memory needed for flow caching by targeting sampling in such a way that cache entries tend to be instantiated only for longer flows.

In the *sample-and-hold* proposal of Estan and Varghese [36], a cache lookup is performed for the key of each incoming packet and if a match is found, statistics are updated as usual for the flow with matching key. However, if a match is not found, a new cache entry is created only with a certain probability $1 - (1 - p)^s$ for some p, where s is the size of the packet. Thus the probability that a flow that comprises b bytes is not sampled at all is $(1 - p)^b$, independent of the manner in which the bytes of the flow are divided among its packets. Clearly the bytes reported for a flow never exceed the actual bytes and, unless the first packet is selected, there will be an undercount. However, the relative error is smaller for larger flows and can be made arbitrarily small by adjusting p. Estan and Varghese [36] derived upper bounds for the expected number of flow records and lower bounds for the expected usage reported.

With sample-and-hold, it is not possible to form an unbiased estimator of the true flow sizes in the manner of Section 4.6 in general. This is because the number of bytes that have not been sampled is unknown and, hence, the probability for a given flow to have been sampled, given knowledge only of the number of sampled bytes, also is unknown. An exception is the special case that all packets have a common size; denote this by a. Then for a flow that is sampled, the expectation of the difference between the actual and sampled flow bytes is $a(1/(1 - p)^a - 1)$. Thus, adding this quantity to the reported bytes yields an unbiased estimator of the actual flow bytes.

Sample-and-hold requires a cache lookup for all packets. To function at the line speed of network interfaces, this requires that the cache be maintained in fast—and hence expensive—memory. The advantage that sample-and-hold can bring over not sampling is that, since short flows tend not to be instantiated in the cache and since most of the usage occurs in a small proportion of long flows, it is possible to reduce the size of the cache memory relative to the unsampled case.

Kodialam, Lakshman and Mohanty proposed the runs based traffic estimator (RATE) [62]. Flow cache entries are instantiated and updated only when the same key is observed in a run of two back-to-back packets. This is achieved by maintaining a register

that contains the key of the last packet observed and comparing it against the key of the next incoming packet. This scheme favors the formation of statistics for longer flows, because they have a larger chance of forming a run. The statistics of runs are used to estimate the volume of high volume traffic components, exploiting results from renewal theory to determine the trade-off between estimation accuracy and flow cache size.

In an abstract setting, Gibbons and Matias [45] proposed *counting samples* from a set of possibly repeated items of various types. Sampling is independent, but once a given type of item is sampled, a count is maintained of all occurrences of that type within the set. Taking items as packets, distinguished by a key, sample-and-hold reduces to counting samples when all packets have identical size. Manku and Motwani [70] proposed *sticky sampling*, in which packet sampling probabilities vary during the measurement period, while probabilistic reconfiguration and selective discard of cache entries favors retention of longer-lived flows.

6.3 Flow Records: Report Selection

Sampling can also be applied to completed flow records, either at the router immediately prior to export or within the infrastructure that is used to collect the flow records. We saw that uniform sampling is problematic due to the consequences of omitting records of large flows. This motivates sampling that is dependent on the size of the flow being reported. A simple approach is to discard flow records whose byte size falls below a threshold. This gives a conservative, and hence biased, measure of the total bytes and is susceptible to subversion: An application or user that splits their traffic up into small flows could evade measurement altogether. This would be a weakness for accounting and security applications.

Duffield, Lund and Thorup [28] proposed nonuniform probability sampling based on flow bytes. Given a *selection function* $p:\mathbb{N}\to(0,1]$, a flow of size x is selected with probability $p(x)$. Given a set of n flows with sizes $S=\{x_i : i=1,\dots,n\}$, let S' denote the corresponding (random) set of sizes of selected flows. As in Section 4.6, we form an unbiased estimator of the total bytes $X=\sum_{x\in S}x$ using the Horvitz–Thompson estimator $X'=\sum_{x\in S'}x/p(x)$.

What is the best choice of selection function p? The answer to this question depends on the use to which the statistics will be put. Duffield, Lund and Thorup [28] provided an answer to this question in the context of balancing the opposing constraints of keeping the variance of X' small, while at the same time reducing the number of samples $\#S'$ taken. This trade-off can be expressed through the cost $C_p=\operatorname{Var}X'+z^2\mathsf{E}\#S'$. Here z^2 is a parameter that expresses the relative importance of the aims in the balance. If z is large, C_p is kept small by keeping $\mathsf{E}\#S'$ small, while if z is small, C_p is kept small by keeping $\operatorname{Var}X'$ small. The selection function p that minimizes C_p for all S is found to be $p_z(x)=\min\{1,x/z\}$: Flows of byte size greater than the threshold z are sampled with probability 1; otherwise they are sampled with probability proportional to their byte size. This is reminiscent of PPS (see Section 4.6) except that rather than determining the constant of proportionality z from the data and having all probabilities less than 1, we parameterize z, select objects independently and allow unit selection probabilities. We call sampling with the selection function p_z optimal for the cost. Resource usage for processing the flow records is bounded: The number of sampled records per byte of original traffic is $\mathsf{E}[\#S']/X\le 1/z$. The bound is tight when all flows have size below threshold z. Further details and developments in size-dependent sampling of flow records can be found in [27, 29, 30, 32].

6.4 Ramifications of Flow Sampling for Billing Applications

Discard-below-threshold and sample-and-hold never overestimate usage. However, although customers would never be overcharged, the shortfall for the provider is unconstrained. On the other hand, optimal sampling can overestimate byte usage since X' is an unbiased estimator of X with nonzero variance. For applications, such as billing, where overestimation may be a larger problem than underestimation, one can bias the estimator X' negatively to reduce the chance of overestimation. A convenient way to achieve this is provided by the bound $\operatorname{Var}X'\le zX$. This suggests using the biased estimator $X'_s=X'-s\sqrt{zX'}$ for the purposes of billing and $s>0$ can be thought of as the number of standard deviations by which the estimated usage is to be negatively biased.

The billing scheme can itself be adapted to sampling. Duffield, Lund and Thorup [28] proposed flat rate billing for usage up to a level L, with a proportionate charge only on usage above this level. The billing scheme is insensitive to the errors in estimating small usage. When coupled with optimal size-dependent sampling of flow records, it can be shown

that the coefficient of variation of the charge is bounded below ε provided the sampling threshold z and billing insensitivity level L obey $z \leq \varepsilon^2 L$. This property holds independently of the statistics of the flow sizes. In practice, we expect z to be constrained below through the capacity of the measurement infrastructure—recall it bounds the mean bytes reported per flow record—and we expect ε to be constrained above through policy. Thus the constraint can be realized by making the billing interval, and hence the amount of usage estimated, sufficiently large.

7. RATE ADAPTIVE AND RATE CONSTRAINED SAMPLING

Sampling trades off estimation accuracy against reduction in sample volumes. The choice of sampling parameters reflects the relative priorities attached to these two opposing goals. In practice, traffic streams exhibit both systematic and statistical variability: Internet traffic rates have both daily and weekly cycles; link failures and network reconfigurations lead to level shifts in traffic rates observed on some links; routing protocols may propagate such changes far from their point of origin. This section discusses two approaches to the problem of maintaining sampling goals within acceptable limits under varying traffic loads: adaptation of the sampling rate and rate constrained sampling.

7.1 Rate Adaptive Sampling

Rate adaptive sampling involves adjusting the sampling rate in response to the rate at which objects are selected or in anticipation of a predicted traffic rate. Drobisz and Christensen [24] developed a multiplicative adaptive scheme to manage resource usage for packet measurement in routers. They described two different controls: one based on CPU utilization and the other based on packet interarrival times. Packets are selected when a counter is decremented to zero (and then reset). Adaptation takes place by adjusting the counter through multiplication by the ratio of the actual to desired router CPU utilization or the current to average packet interarrival time. The main motivation of the study was estimation of parameters of self-similar traffic from samples. It was found that the adaptive methods produced more accurate estimates than static sampling under a given resource constraint.

Choi, Park and Zhang [11] focused on maintaining accuracy of estimates of the short-term traffic load at a router under varying traffic rates. The motivation of this study was the identification of change-points in the traffic load; the accuracy of rate estimates determines the resolution at which changes can be detected.

For the sampling of flow records, Duffield, Lund and Thorup [29] provided a rate adaptive version of the size-dependent sampling scheme described in Section 6.3. The aim is to control the rate at which samples are produced independent of the offered load. Control is effected through multiplicative adaptation of the sampling threshold z of successive time windows. If N samples are taken over a window in which the target number is M with threshold z, the threshold for the next window is zN/M. Under this iteration, the mean rate at which samples are taken converges to the target rate when the arrival process of flow records is stationary.

Hernandez, Chidester and George [50] used a predictive approach to anticipate variations in the offered load and adjust the sampling interval accordingly to meet sampling volume constraints. Their method combines linear prediction with a fuzzy logic approach that classifies the broad dynamics of the load.

7.2 Rate Constrained Sampling

Rate adaptive sampling has a number of drawbacks:

(i) Due to the inherent latency of adaptation, hard sampling volume constraints cannot be met under arbitrary statistical and systematic variation in the offered traffic.
(ii) Systematic undersampling (by arranging for the mean sampling rate to be lower than the target) is necessary to accommodate uncontrolled variations in the traffic rate.
(iii) The effective sampling probability is reduced for objects that occur during periods of high load. However, it may be precisely the objects that occur during these periods that are the most important to capture.

These drawbacks motivate sampling strategies that can select a specified number of objects during a given measurement interval. Several algorithms have been proposed. Reservoir sampling (in which one keeps a reservoir of k ongoing samples) with uniform probability was treated by Vitter [97]. For nonuniform probability sampling, an algorithm for weighted reservoir sampling with replacement was provided by Chaudhuri, Motwani and Narasayya [10]; it requires at most $O(k)$ operations per item. An algorithm for sampling without replacement that requires $O(\log k)$ operations per item was recently proposed by Duffield, Lund and Thorup [32] in a modification of the work in

Section 6.3. All these methods require only one pass through the stream of objects and only about k items need to be maintained in storage at any time. There is some penalty to pay in latency, since the identities of objects to be selected are only determined once the last object has been processed; sampling is not adapted in the sense of Section 4.2. For this reason, unless the measurement interval can be kept short, these techniques may be better suited to data base applications rather than sampling at the observation point.

8. INFERENCE OF DETAILED FLOW STATISTICS

In Sections 5 and 6 we saw how aggregate byte or packet counts can be estimated by dividing the sampled counts by the selection probability. However, understanding more detailed statistics of packet flows generated by the endpoint applications is important for a number of purposes, including:

- *Determining resource requirements for flow statistics:* for constructing them at a router and for transmitting them to a collector. From Section 6 it is clear that the requirements depend on sampling rates, the flow arrival rate and the detailed distribution of packets per flow; see [28] and [29] for further details.
- *Characterizing source traffic:* numbers of packets and bytes, and durations of individual flows and also higher levels, for example, transactions involving multiple flows that combine a set of applications such as a Web browsing session that comprises a DNS lookup followed by an HTTP transfer. Such characterizations have been performed for unsampled flows to support network design decisions (see [38] and [41]).

To support such applications, the characteristics of application level flows need to be inferred from either sampled packets or packet sampled flow records. For some problems protocol level information can be exploited. Duffield, Lund and Thorup [30] used packet sampled NetFlow statistics to estimate mean number of packets per TCP connection. The first packet in each direction of a TCP connection has a bit—called the SYN flag—set in the TCP header. Barring loss and retransmission, each direction of the connection contains exactly one SYN packet. NetFlow records include the cumulative OR of its packets' SYN flags and so indicate whether one or more packets assigned to a flow had their SYN flag set. Under packet sampling with probability $1/N$, the expected number of SYN flags seen per direction is $1/N$ times the number of connections. Accordingly, we can form a moment estimator of the number of connections by N times the number of packet sampled flows which reported a SYN flag. Dividing this into the estimated total number of packets (i.e., N times the number of sampled packets) yields an estimator of the mean number of packets per connection.

A more challenging problem is to estimate the distribution of the number of packets or bytes per flow, or the distribution of the duration of flows, from 1 in N packet sampled flow statistics. A potential difficulty is that the application level flows and the measured flows need not coincide. A single application flow may be split into several measured flows by the separate or combined actions of sampling, the inactive time-out or the active time-out. Measured flows with a common key may be combined during analysis, effectively lengthening timeouts *after* measurement has taken place.

Even after such surgery, the problem of inferring original flow statistics still remains. The simple approach of multiplying measured lengths and durations by N does not suffice in general: The number of flows will be underestimated (some flows have no packets sampled) and there will be a bias toward longer flows (these have a greater probability of being sampled). Furthermore, there is an inherent difficulty in resolving the distribution of short flows, that is, those with far fewer than N packets. These flows typically all have at most one packet sampled, regardless of their original length.

Recent work by Duffield, Lund and Thorup [31] inferred a smoothed version of the distribution of flow lengths, that is, the numbers of packets per original flow, from the measured frequencies of sampled flow lengths. The work used both maximum likelihood estimation and ad hoc methods that locate the range in which the most frequent lengths lie. Here the smoothed distribution can be thought of as representing an average over a set of original length distributions that is compatible with the measured distribution.

Hohn and Veitch [51] considered the related problem of recovering the distribution of the number of packets i per original flow from the distribution of sampled packets per flow in a model of Poisson flow arrivals. Using spectral analysis, they found the inherent limits of such inversion of the full distribution for small sampling rates, although mean flow lengths and asymptotic properties of heavy-tailed length distributions can be recovered. They also pointed out that under uniform

sampling of *flows*, the distribution f_i of original packets i per flow can be recovered straightforwardly from the distribution g_i of packets per sampled flow: they are the same. This fact can be reconciled with Section 6.3 by observing that the estimate of mean flow length $\sum_i i\widehat{g}_i$ that arises from a measured version $\widehat{g}_i$ of g_i need not be close to the mean $\sum_i i g_i$ even when $\widehat{g}_i$ is close to g_i.

9. HASH-BASED SAMPLING

Hash-based sampling offers both a convenient way to emulate random sampling through generation of pseudorandom variates, and a powerful way to consistently select subsets of objects whose contents share a common property. The basic idea is as follows. We are given a hash function h with domain the object content set C. Given some subset $S \subset h(C)$, called the *selection range*, a packet with content c is selected if the hash $h(c) \in S$. Comparison with Section 4.3 shows that hash-based sampling is in fact a filter: The object is selected if $c \in h^{-1}(S)$. However, for good choices of hash function, the inverse image $h^{-1}(S)$ is extremely complex and hence h is not expressible as, say, a mask/match filter or a simple combination of these.

9.1 Statistical Properties

In what sense can a deterministic filtering operation be used to achieve pseudorandom sampling? Two conditions must be fulfilled. First, the hash function h must be strong in the sense that it has good mixing properties: Small changes in the input (e.g., the flipping of a single bit) must cause large changes in the output. If this condition holds, any local region of potential hash inputs c becomes spread widely over $h(C)$ by the action h; hence the distribution of $h(c)$ tends to be fairly uniform even if the distribution of c is not. The fraction of objects in the clump that is selected is close to $\#S/\#h(C)$, so the target sampling rate can be tuned through appropriate choice of the size of the selection range S.

The second desirable property depends more closely on the statistics of the content. In applications, the content comprises a number of distinct fields, for example, source and destination IP address, and TCP/UDP port numbers (if present) for a packet. Although hash-based selection decisions are deterministic, they can provide the appearance of random sampling in the sense that selection decisions appear only weakly correlated with the values of individual fields. Conditions for this are (i) a strong hash function must be used and (ii) for each field f_1, another input field f_2 exists that is sufficiently variable and mutually weakly correlated: The variability of f_2 has the effect of making selection appear independent of the value of f_1.

9.2 Guarding Against Pitfalls and Vulnerabilities

Since hash-based sampling is deterministic, a concern is whether a large set of related objects can be sampled at a rate that differs significantly from the target sampling rate. An extreme case would be if the objects avoided sampling altogether, either (i) through unanticipated behavior in the hash function or (ii) because they had been deliberately crafted to have this property. The first point underlines the importance of using a strong hash function.

Hash functions based on modular arithmetic can be strong with a judicious choice of modulus; see Section 6.4 in [60]. Well-known hash functions such as CRC32 [56] and MD5 [89] may already be implemented in network elements for other purposes, making their potential reuse convenient for sampling. However, the strength of available hash functions can vary; whereas strong hash functions are typically employed for cryptographic purposes, other applications may need only a comparatively simple hash. Scarcity of processing resources may preclude using stronger hash functions in some cases. For these reasons, the statistical properties of candidate hash functions need to be evaluated, preferably on traces of actual object sequences, before use for hash-based sampling.

Can hash-based sampling be deliberately evaded? If its use becomes widespread, it is likely that well-known hash functions will be used and/or the choice of hash function will be standardized and hence well known. Nevertheless, network operators have two defenses against objects crafted to evade sampling. The first defense is through choice of the selection range, that is, S can be kept private or regularly changed, while the sampling rate—as determined by $\#S$—is held constant. However, an attacker who crafted a set of packets all with the same hash value would know that the packets would be either all selected or all not selected. A stronger defense is to employ a parameterizable hash function and keep the parameter private. In this case the set of hash values of the packets could not be predicted. Examples of parameters are the initial vector in CRC32 and moduli in hashes based on modular arithmetic.

9.3 Applications

Flowwise packet sampling. Suppose we wish to sample all packets from a random subset of flows of packets passing through a router. This is useful for understanding packet dynamics when resources are not available to collect full packet traces. One method is to employ a flow caching scheme in which newly instantiated cache entries are randomly marked. If marked, then a report is made on each packet arriving with the corresponding key; if not, no report is made. Hash-based sampling can be used to achieve the same type of packet selection without caching flow keys. Instead, the flow key of each packet is hash sampled and selected packets are reported on. Hash sampling ensures that all packets of a given key are either selected or not selected together.

Trajectory sampling. In trajectory sampling, all routers in a network hash-sample packets using an identical hash function and selection range. The domain of the hash is restricted to those fields that are invariant from hop to hop: time-to-live is excluded since it is decremented per hop. Thus a given packet is sampled either at all points on its path through the network or at none. The packet signals implicitly to the router, through its hash, whether or not it should be sampled.

The domain of the hash function needs to be wider than just a flow key, if packets are to be selected pseudorandomly within flows. A report on each selected packet is exported to a collector. The collector can reconstruct trajectories of the selected packets provided it can match different reports on the same packet and distinguish them from reports on different packets. For this purpose, reports may also contain a second distinct hash, the identification hash, of the selected packets and/or timing information.

Applications of trajectory sampling include (i) estimation of the network path matrix, that is, the traffic intensities according to network path, broken down by flow, (ii) detection of routing loops, as indicated by self-intersecting trajectories, (iii) passive performance measurement [prematurely terminating trajectories indicates packet loss and packet latencies can be determined if reports include (synchronized) time stamps] and (iv) network attack tracing of the actual paths taken by attack packets with spoofed source addresses.

Related measurement methods. Trajectory sampling was proposed by Duffield and Grossglauser [26]; further work on issues of collection and trajectory reconstruction are covered in [25]. Hashing for identification (as opposed to sampling) of packets measured at different points has been proposed for wide area ATM network measurements; see [20]. Hash-based correlation of elements in multiple packet traces during postanalysis was used by Moon and Roscoe [73].

Packet hashing for the purposes of identifying packet paths for network attack detection was proposed by Snoeren et al. [92]. Routers are to maintain digests of recently arrived packets in the form of Bloom filters [7], which comprise arrays of hashes of the invariant packet content. Bloom filters are a data structure that compactly stores sufficient information to determine whether a given packet is absent from the digested set: If the hashes of a test packet are all present in the Bloom filter, then the packet is judged to have been previously digested. Hence false positives may occur, but not false negatives. When a packet is deemed to be suspicious for some reason, the packet's path through the network can be traced back by testing its membership in the set of router Bloom filters.

An alternative trace-back approach, as proposed by Savage, Wetherall, Karlin and Anderson [90] and elaborated on by Dean, Franklin and Stubblefield [22] and Song and Perrig [93], is for routers to encode their identity within selected packets. If a packet is found to have participated in an attack, its path can be determined by decoding. One potential drawback is the lack of spare fields in the IP packet header. Instead, IP header fields, such as the identification field, must be reused. This may interfere with the operation of other applications. A case in point is trajectory sampling. Inclusion of the identification field in the hash input is attractive for trajectory sampling since it varies from packet to packet in the same flow. (Although this is not always the case in practice, it seems partly due to bad implementations.) Per hop modification of this field for trace-back breaks the trajectory semantic.

An alternative approach to trajectory sampling is to randomly mark packets upon entry to the network and then sample only marked packets in the core. This has potential drawbacks similar to those of the last paragraph: the lack of an available bit for marking in the packet header. An additional drawback is that on deployment, *all* edge routers in a network must be capable of filtering marked incoming packets; otherwise, an attacker could overwhelm the measurement system with deliberately marked packets.

10. BEYOND SAMPLING

Recall the main attractions of sampling: (i) all detail is retained from the selected objects, (ii) no additional storage is required at the observation point and

(iii) there is little or no latency in selecting objects. In practice there is scope for relaxing these properties somewhat. Whereas sampling yields a subset of exact representatives of the data—each sampled object was present in the original object stream—the conclusions drawn concerning the population are often approximate. For routine queries, there should be no objection to using an approximate summary representation of the data provided that it allows conclusions to be drawn with the same or better accuracy, or with smaller storage or computational requirements, or with smaller response time. Such a representation can be an adjunct to the complete data, rather than a replacement for it: a class of rapid queries would be supported by the summaries and detailed queries could still be performed on the full data set.

Ideally, summaries must be computable in a single pass over the object stream, or possibly some small number of passes over batches. Only limited computational and storage resources may be available, if not on the measurement device itself, then on some nearby host that mediates the transmission of data to the ultimate collector. At the collector constraints on computations may be relaxed further, due to the availability of high performance computers with extensive memory and disk or tape storage. In the last context there is also a role for permanent data reduction: working data from the recent past is reduced as it ages, prior to archiving. Barbara et al. [5] reviewed various approaches to data reduction. In this section we very briefly describe some promising approaches for compactly summarizing the massive data sets produced by passive measurement.

10.1 Summary Data Structures

The problem of developing compact summaries of objects streams has attracted much attention in the last few years. For network measurement, the usefulness of a summary depends on (i) being computable on one pass through the data stream, (ii) updates being cheap to compute, (iii) storage costs and (iv) the ability to service return queries on the dominant features of the object stream with acceptable accuracy.

Gibbons and Matias [46] defined an $f(n)$-synopsis data structure (or simply, synopsis) for a class Q of queries as a structure that can support answers to Q using $O(f(n))$ storage for a data set of size n. Here $f(n) = o(n^{\varepsilon})$ for some $\varepsilon < 1$. They reviewed a number of constructions of synopses to support queries on different characteristics of data sets, including frequency moments (e.g., the number of distinct elements, total number of elements), the most frequently occurring members and quantiles. A frequent strategy is to map data to storage using a hash function, which, on one hand, allows identification of repeated elements, and on the other hand, has statistical properties that can be exploited for estimation (e.g., by randomized algorithms). See [35] and [37] for recent approaches to counting the number of distinct flow keys, and [19] and [23] for estimating the size of dominant flows. (On the other hand, deterministic algorithms for this last problem are provided in [48].)

Recent work by Gilbert, Kotidis, Muthukrishnan and Strauss [47] describes sketches—a type of synopsis built on inner products of data with random vectors. One attractive feature is the linearity of sketches, which facilitates combination, and the geometry of the inner product, in which comparisons between data can be expressed. The authors focus on sketches built on wavelet bases. This furnishes a natural path for approximation: retention of only the dominant wavelet coefficients. Gilbert et al. gave applications to the problem of finding heavy hitters in streams of packet or flow records.

A different approach to estimating dominant traffic components was taken by Kumar et al. [64], who extended the concept of Bloom filter to keep track of the number of times a given object is presented. Due to inherent collisions within the Bloom filter (see Section 9.3), some analysis is required to estimate the frequencies of presented objects.

10.2 Data Squashing

Consider a data set that comprises n records of k fields. Data squashing entails constructing a summary data set of $m \ll n$ records with $k + 1$ fields, where the additional field is a weight w_i such that $\sum_{i=1}^{m} w_i = n$. Random sampling is a special case where the records of the squashed set are a random sample of the original records and $w_i = n/m$. However, data squashing aims to provide better accuracy than is available from simple sampling. Although there is currently no unified theory of data squashing, three approaches have been proposed; for further details and comparison, see the review by DuMouchel [33].

Closest to simple sampling is the approach of Owen [75], who selected an initial random sample of the original data and then assigned the weights by matching moments using the empirical likelihood method. The trade-off of this simple computational approach is that the squashed data set may have to be quite large to achieve the desired accuracy. Two other approaches are

based directly on likelihood. The idea is to choose the points of the squashed set and their weights so that the original and the squashed data have the same likelihood function. Madigan et al. [69] assumed a logistic regression as the data model. The likelihood is evaluated at a representative set of parameters. The idea is to cluster points with similar likelihood, then merge clusters into a single representative squashed data value, its weight being the number of points merged. The algorithm requires two passes: one to choose the logistic parameters and one to cluster and merge. DuMouchel et al. [34] argued that for any likelihood function that can be approximated by a low order Taylor series over multivariate bins, the squashed data set can be chosen by matching low order moments within each bin. Location of the squashed data points requires a constrained search within each bin. This is the most computationally intensive method of the three. On the other hand, it claims the greatest advantage for accuracy: in one example, 750,000 record data yielded a mean square error almost 500 times smaller compared with simple random sampling and the data size was reduced by a factor of roughly 100.

Clearly the computational complexities of these methods may limit the applicable domain of some of them to collectors and other back end systems. Another challenge is the immense range of values taken by categorical data such as IP addresses: there are over 10^{19} possible source–destination IP address pairs. Squashing based on the natural clustering from topology or routing may be a useful way to reduce dimensionality.

11. SAMPLING IMPLEMENTATIONS AND RAMIFICATIONS FOR STANDARDS

The different sampling methods and their implementations represent an additional source of variability for interpretation of measurements. Only under unrealistically strong assumptions (e.g., that the distribution of objects is permutation invariant) do the classical sampling methods (random, systematic, stratified) yield the same sampling distribution. In practice, objects are dependent: packets are associated, through their keys, into flows. Application and user behavior lead to recurrence of flow keys over time, for example, a day-trader checking and rechecking the price of a stock on a financial website. In the presence of dependence, different sampling methods may yield different conclusions about the population under study.

How can these differences be quantified and to what extent do they need to be taken into account when implementing sampling methods, drawing up standards for sampling in routers or designing applications that use sampled data? Currently routers sample packets using pseudorandom independent sampling [44, 55] or systematic sampling [59], depending on the equipment vendor. Should these be standardized as two distinct configurable sampling functions or can they be regarded as two different detailed implementations of some basic reference method?

Hypothesis testing can distinguish different sampling distributions and test their accuracy against the original population; see Section 4.5. Even if sampling distributions arising from different methods are statistically distinguishable, they may be close enough for some applications. Duffield, Lund and Thorup [31] found the distributions of lengths of packet sampled flows arising from periodic and independent sampling with the same probability to be statistically distinguishable. Differences in the distributions, although small, persisted even for large data sets. The differences could be attributed to the different sampling properties of flows. For example, two packets of a flow occurring back-to-back in the packet stream will never both be selected by periodic sampling, although they can be selected in independent sampling. Typical differences in the sampled distributions were only about 1% for the most common flow lengths. These differences are probably negligible for many networking management applications, although there are no generally accepted accuracy requirements for them. The relationship between the traffic properties, sampling methods and application requirements clearly needs better understanding before these questions can be settled on a scientific basis.

12. CONCLUSIONS AND OUTLOOK

Passive measurements play a crucial role in managing IP networks and understanding the properties of traffic that traverses them. The volumes of measured data are increasing for two reasons: the increase in link speeds and the requirement for detailed measurements. Resource and cost constraints necessitate data reduction, to be performed at the observation point and/or within the measurement collection infrastructure. Of the reduction methods in current widespread use (filtering, aggregation and sampling), only sampling allows the retention of arbitrary detail. To reduce the impact of sampling variance, the sampling design and the downstream applications must be well matched to the statistical properties of the underlying stream of objects. Newer aggregation and compression methods, such as

synopses and squashing, allow some trade-off between retention of detail, and computational and storage requirements.

The work reviewed in this paper suggests several areas for future research. The first concerns the development of embedded monitoring in network elements. A better understanding of the impact of choice of sampling operation on measured traffic characteristics would inform choices made in standardization. Emerging standards for packet sampling are likely to endorse mostly classical design methods, due to both the resource constraints and the depth of understanding of the underlying methods. As resources become cheaper and more readily available, some statistical analysis can be moved to routers. The choice of statistics can be standardized; the choice of algorithm can be an implementation issue over which equipment vendors could compete on price and performance.

Section 8 reported work on the general problem of inferring distributions of multipacket events (such as flows) in the original traffic from the corresponding sampled events. This can generally be framed as a problem of linear inversion under positivity constraints. The occurrence and solution of such problems is already familiar from network tomography (see, e.g., [96]). Some of the techniques applied in that area may be useful in the present case.

Distributed measurement infrastructures will provide multiple correlated observations of network events, as observed from the vantage points of different measurement devices. The volume of measurements will grow with the scale of the infrastructure. A challenge will be to provide some form of distributed data reduction across the infrastructure to eliminate redundancies between multiple network views.

Although many current measurement-based applications employ aggregate usage per traffic class, applications increasingly need to track and correlate small events. This is particularly true for detection of intrusions and network attacks, which are increasingly a composite of smaller constituents, spatially or temporally distributed so as to better evade detection. There is a large body of recent research and products that uses attack signatures and anomalous departures from customary usage patterns as signifiers of attacks; see, for example, Lee's [66] online bibliography. The task is made harder by sampling, especially when measurement must be, like the attacks, spatially and temporally distributed, and when the analysis requires correlation of different measurements. Under independent sampling the weight attached to composite events may be small. Correlated sampling that can exploit specific or generic properties of such attacks to enhance capture of multiple constituents would help boost detection rates. In some recent work related to this question, Kodialam and Lakshman [61] provided a game theoretic analysis of the sampling rates necessary to detect distributed attack traffic.

Finally, the comments of the previous paragraph raise a larger question. Until recently sampling deployed in network elements has been classical: independent or periodic uniform sampling. This was a reasonable choice, since (i) the implementations were simple, (ii) there were well-known methods available to analyze the sampled data and (iii) there was no clear benefit to using more complex methods. It is becoming evident that new sampling methods are needed both to service new applications (e.g., hash-based sampling in Section 9) and to circumvent the limits of classical methods to provide data most suited for inference of original traffic properties (long flows in Sections 6.2 and 6.3, short flows in Section 8 and network attacks in the previous paragraph). A rational choice of the best sampling method requires balancing the costs involved (including those of implementation, resource usage and estimation accuracy) together with the flexibility to provide statistics that may play a role in future network management applications.

APPENDIX: THE STRUCTURE AND FUNCTION OF PACKETS IN THE INTERNET

We summarize the mechanisms by which a host (e.g., a personal computer, a Web server, a router) connected to the Internet, can send data to another such host. Data are carried by packets, which are forwarded along a network path that comprises one or more links joined by network elements such as routers or switches. The structure of the packets is defined by network protocols, most commonly the Internet protocol [82]. Internet protocol packets comprise the payload—the data to be sent—and a header, which contains the information each network element needs to determine over which link to forward the packet. Each host in the Internet possesses one or more IP addresses, which are 32 bit numbers in the currently predominant version 4 of IP. Internet protocol addresses are organized hierarchically: service providers are allocated contiguous blocks of address, which they redistribute in blocks to their customers, and so on, down to the level of individual hosts.

The IP packet header includes the IP address of both its source and destination. An IP router maintains a

routing table which determines the link over which a given packet is to be forwarded based on the packet's destination IP address. The contents of the routing tables are themselves arrived at through the exchange between routers of information concerning network topology and connectedness. The exchange of information is mediated by routing protocols, whose operation among routers is a distributed optimization that results in forwarding paths that have the lowest cost by some measure (e.g., shortest path length), under the constraint of policies set by network operators.

The IP header contains other fields, such as a checksum of the header contents (to detect bit errors) and time-to-live (TTL), an initially positive integer that is decremented by each router. When TTL becomes zero, the packet is discarded and a notification packet is sent to the packet source IP address. This avoids infinite propagation of packets around routing loops that arise from misconfiguration.

Applications running on hosts do not typically use IP directly. Rather, they interact through a transport layer that provides the appearance of a direct connection between the partner applications. In this case, the IP packets carrying data between applications have a payload which comprises a further header for the transport protocol, followed, at last, by the actual data to be transmitted. Two transport protocols are dominant. The user datagram protocol [81] provides a nearly transparent interface to the IP, but no facility to detect or recover from loss of packets in transmission, for example, due to congestion at routers when buffers that contain packets that are awaiting processing overflow. The transmission control protocol [83] provides a reliable connection between source and destination, handling detection of packet loss and retransmission, and also governing the rate of transmission in response to congestion (as indicated by packet loss). Both TCP and UDP headers contain 16 bit port numbers for source and destination, which the hosts use to direct received packets to the intended application. TCP also uses sequence numbers (for loss detection) and single bit flags that signal, among other things, the start and end of connections between hosts.

ACKNOWLEDGMENTS

Thanks go to the anonymous referees and to Walter Willinger for many useful suggestions.

REFERENCES

In order to promote wide access to the cited references, URLs for authors' versions have been included when found. Some references are also available at CiteSeer: http://www.citeseer.com, or through subscription services such as The ACM Digital Library: http://portal.acm.org/dl.cfm, and IEEE Xplore: http://ieeexplore.ieee.org, depending on the publication.

[1] ADAMS, A., BU, T., FRIEDMAN, T., HOROWITZ, J., TOWSLEY, D., CÁCERES, R., DUFFIELD, N. G., LO PRESTI, F., MOON, S. B. and PAXSON, V. (2000). The use of end-to-end multicast measurements for characterizing internal network behavior. *IEEE Communications Magazine* **38**(5) 152–159. Available at www.research.att.com/~duffield/pubs/minc.ps.gz.

[2] AMER, P. D. and CASSEL, L. N. (1989). Management of sampled real-time network measurements. In *Proc. 14th IEEE Conference on Local Computer Networks* 62–68. IEEE Press, New York.

[3] APISDORF, J., CLAFFY, K., THOMPSON, K. and WILDER, R. (1996). OC3MON: Flexible, affordable, high performance statistics collection. Available at www.nlanr.net/NA/Oc3mon.

[4] ARMITAGE, G. J. (2000). MPLS—The magic behind the myths. *IEEE Communications Magazine* **38**(1) 124–131. Available at www.comsoc.org/ci/private/2000/jan/pdf/Armitage.pdf.

[5] BARBARA, D., DUMOUCHEL, W., FALOUTSOS, C., HAAS, P. J., HELLERSTEIN, J. M., IOANNIDIS, Y., JAGADISH, H. V., JOHNSON, T., NG, R., POOSALA, V., ROSS, K. A. and SEVCIK, K. C. (1997). The New Jersey data reduction report. *IEEE Data Engineering Bulletin* **20**(4) 3–42. Available at ftp://ftp.research.microsoft.com/pub/debull/97DEC-CD.pdf.

[6] BERNERS-LEE, T., FIELDING, R. and FRYSTYK, H. (1996). Hypertext transfer protocol—HTTP/1.0. RFC 1945. Available at www.ietf.org/rfc/rfc1945.txt.

[7] BRODER, A. and MITZENMACHER, M. (2002). Network applications of Bloom filters: A survey. In *Proc. Allerton Conference*. Available at www.eecs.harvard.edu/~michaelm/NEWWORK/postscripts/BloomFilterSurvey.pdf.

[8] CÁCERES, R., DUFFIELD, N. G., FELDMANN, A., FRIEDMANN, J., GREENBERG, A., GREER, R., JOHNSON, T., KALMANEK, C., KRISHNAMURTHY, B., LAVELLE, D., MISHRA, P. P., RAMAKRISHNAN, K. K., REXFORD, J., TRUE, F. and VAN DER MERWE, J. E. (2000). Measurement and analysis of IP network usage and behavior. *IEEE Communications Magazine* **38**(5) 144–151. Available at www.research.att.com/~duffield/pubs/measure.ps.gz.

[9] CASE, J., FEDOR, M., SCHOFFSTALL, M. and DAVIN, J. (1990). A simple network management protocol (SNMP). RFC 1157. Available at www.ietf.org/rfc/rfc1157.txt.

[10] CHAUDHURI, S., MOTWANI, R. and NARASAYYA, V. R. (1999). On random sampling over joins. In *Proc. ACM SIGMOD 1999* 263–274. ACM Press, New York. Available at doi.acm.org/10.1145/304182.304206.

[11] CHOI, B.-Y., PARK, J. and ZHANG, Z.-L. (2002). Adaptive random sampling for load change detection (extended

abstract). In *Proc. ACM SIGMETRICS 2002* 272–273. ACM Press, New York. Available at parapet.ee.princeton.edu/~sigm2002/papers/p272-choi.pdf.

[12] Cisco NetFlow. Available at www.cisco.com/warp/public/732/netflow/index.html.

[13] Cisco NetFlow services and applications customer profiles. Available at www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/profiles/.

[14] CLAFFY, K. C., BRAUN, H.-W. and POLYZOS, G. C. (1995). A parameterizable methodology for Internet traffic flow profiling. *IEEE J. Selected Areas in Communications* **13** 1481–1494. Available at www.nlanr.net/Flowsresearch/Flowspaper/flows.html.

[15] CLAFFY, K. C., POLYZOS, G. C. and BRAUN, H.-W. (1993). Application of sampling methodologies to network traffic characterization. *Computer Communication Review* **23** 194–203. Also appeared in *Proc. ACM SIGCOMM 1993*. Available at www.caida.org/outreach/papers/1993/asmw/sigcomm.sampling.pdf.

[16] COATES, A., HERO, A., NOWAK, R. and YU, B. (2002). Internet tomography. *IEEE Signal Processing Magazine* **19**(3) 47–65. Available at www.ece.mcgill.ca/~coates/publications/tomoreview.ps.gz.

[17] COCHRAN, W. (1977). *Sampling Techniques*, 3rd ed. Wiley, New York.

[18] CORBATÓ, S. (1996). Backbone performance analysis techniques. In *Proc. INET 96*. Available at www.isoc.org/inet96/proceedings/d3/d3_1.htm.

[19] CORMODE, G. and MUTHUKRISHNAN, S. (2004). What's new: Finding significant differences in network data streams. In *Proc. IEEE INFOCOM 2004* **3** 1534–1545. IEEE Press, New York.

[20] COZZANI, I. and GIORDANO, S. (1998). Traffic sampling methods for end-to-end QoS evaluation in large heterogeneous networks. *Computer Networks and ISDN Systems* **30** 1697–1706. Available at netserv.iet.unipi.it/students/cozzani/cozzani/Papers/TNC98.ps.gz.

[21] CRANOR, C., GAO, Y., JOHNSON, T., SHKAPENYUK, V. and SPATSCHECK, O. (2002). Gigascope: High performance network monitoring with an SQL interface. Demonstration. In *Proc. ACM SIGMOD 2002* 623. Available at athos.rutgers.edu/~muthu/demo_02.pdf.

[22] DEAN, D., FRANKLIN, M. and STUBBLEFIELD, A. (2001). An algebraic approach to IP traceback. In *Proc. Network and Distributed System Security Symposium*. Available at www.isoc.org/isoc/conferences/ndss/01/2001/papers/dean01.pdf.

[23] DEMAINE, E., LÓPEZ-ORTIZ, A. and MUNRO, J. I. (2002). Frequency estimation of internet packet streams with limited space. *Proc. 10th European Symposium on Algorithms. Lecture Notes in Comput. Sci.* **2461** 348–360. Available at theory.lcs.mit.edu/~edemaine/papers/NetworkStats_ESA2002/paper.pdf.

[24] DROBISZ, J. and CHRISTENSEN, K. (1998). Adaptive sampling methods to determine traffic statistics including the Hurst parameter. In *Proc. 23rd IEEE Conference on Local Computer Networks* 238–247. IEEE Press, New York. Available at citeseer.ist.psu.edu/335773.html.

[25] DUFFIELD, N. G., GERBER, A. and GROSSGLAUSER, M. (2002). Trajectory engine: A backend for trajectory sampling. In *IEEE Network Operations and Management Symposium* 437–450. IEEE Press, New York. Available at www.research.att.com/~duffield/pubs/DGG01-engine.pdf.

[26] DUFFIELD, N. G. and GROSSGLAUSER, M. (2001). Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking* **9** 280–292. An abridged version appeared in *Proc. ACM SIGCOMM 2000* 271–282. Available at www.research.att.com/~duffield/pubs/DG-TS-ToN.pdf.

[27] DUFFIELD, N. G. and LUND, C. (2003). Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure. In *Proc. ACM SIGCOMM Internet Measurement Conference* 179–191. ACM Press, New York. Available at www.research.att.com/~duffield/pubs/p313-duffield-lund.pdf.

[28] DUFFIELD, N. G., LUND, C. and THORUP, M. (2001). Charging from sampled network usage. In *Proc. ACM SIGCOMM Internet Measurement Workshop* 245–256. ACM Press, New York. Available at www.research.att.com/~duffield/pubs/DLT01-usage.pdf.

[29] DUFFIELD, N. G., LUND, C. and THORUP, M. (2001). Learn more, sample less: Control of volume and variance in network measurement. Unpublished manuscript. Available at www.research.att.com/~duffield/pubs/DLT05-optimal.pdf.

[30] DUFFIELD, N. G., LUND, C. and THORUP, M. (2002). Properties and prediction of flow statistics from sampled packet streams. In *Proc. ACM SIGCOMM Internet Measurement Workshop* 159–171. ACM Press, New York. Available at www.research.att.com/~duffield/pubs/DLT02-flows.pdf.

[31] DUFFIELD, N. G., LUND, C. and THORUP, M. (2003). Estimating flow distributions from sampled flow statistics. In *Proc. ACM SIGCOMM 2003* 325–336. ACM Press, New York. Available at www.research.att.com/~duffield/pubs/DLT03-lengths.pdf.

[32] DUFFIELD, N. G., LUND, C. and THORUP, M. (2004). Flow sampling under hard resource constraints. In *Proc. ACM SIGMETRICS 2004* 85–96. ACM Press, New York. Available at www.research.att.com/~duffield/pubs/DLT03-constrained.pdf.

[33] DUMOUCHEL, W. (2002). Data squashing: Constructing summary data sets. In *Handbook of Massive Data Sets* (J. Abello, P. M. Pardalos and M. G. C. Resende, eds.) 579–591. Kluwer, Dordrecht.

[34] DUMOUCHEL, W., VOLINKSY, C., JOHNSON, T., CORTES, C. and PREGIBON, D. (1999). Squashing flat files flatter. In *Proc. Fifth ACM Conference on Knowledge Discovery and Data Mining* 6–15. ACM Press, San Diego. Available at www.research.att.com/~corinna/papers/kddsquash.ps.gz.

[35] DURAND, M. and FLAJOLET, P. (2003). Loglog counting of large cardinalities. *Proc. 11th Annual European Symposium on Algorithms. Lecture Notes in Comput. Sci.* **2832** 605–617. Springer, Berlin.

[36] ESTAN, C. and VARGHESE, G. (2003). New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems* **21** 270–313. Available at www.acm.org/sigcomm/sigcomm2002papers/traffmeas.pdf.

[37] ESTAN, C., VARGHESE, G. and FISK, M. (2003). Bitmap algorithms for counting active flows on high speed links. In *Proc. ACM SIGCOMM Internet Measurement Conference* 153–166. ACM Press, New York. Available at www.icir.org/vern/imc-2003/papers/p327-estan.ps.

[38] FELDMANN, A., CÁCERES, R., DOUGLIS, F., GLASS, G. and RABINOVICH, M. (1999). Performance of Web proxy caching in heterogeneous bandwidth environments. In *Proc. IEEE INFOCOM 1999* **1** 107–116. Available at www.research.att.com/~misha/network/proxim_full.ps.gz.

[39] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N. and REXFORD, J. (2000). NetScope: Traffic engineering for IP networks. *IEEE Network* **14**(2) 11–19. Available at www.cs.princeton.edu/~jrex/papers/ieeenet00.pdf.

[40] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N., REXFORD, J. and TRUE, F. (2001). Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking* **9** 265–279. Available at www.cs.princeton.edu/~jrex/papers/ton01.pdf.

[41] FELDMANN, A., REXFORD, J. and CÁCERES, R. (1998). Efficient policies for carrying web traffic over flow-switched networks. *IEEE/ACM Transactions on Networking* **6** 673–685. Available at www.cs.princeton.edu/~jrex/papers/ton98.ps.

[42] FELDMEIER, D. C. (1986). Statistical monitors for local area networks. In *Proc. 11th IEEE Conference on Local Computer Networks* 142–146. IEEE Press, New York.

[43] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P. and BERNERS-LEE, T. (1999). Hypertext transfer protocol—HTTP/1.1. RFC 2616. Available at www.ietf.org/rfc/rfc2616.txt.

[44] FOUNDRY NETWORKS (2004). Foundry enterprise configuration and management guide. Appendix A. Remote network monitoring. Available at www.foundrynet.com/services/documentation/ecmg/Net_Monitoring.html.

[45] GIBBONS, P. B. and MATIAS, Y. (1998). New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD International Conference on Management of Data* 331–342. ACM Press, New York. Available at citeseer.ist.psu.edu/gibbons98new.html.

[46] GIBBONS, P. B. and MATIAS, Y. (1999). Synopsis data structures for massive data sets. In *External Memory Algorithms* (J. M. Abello and J. S. Vitter, eds.) 39–70. Amer. Math. Soc., Providence, RI. Available at www.math.tau.ac.il/~matias/papers/synopsis.ps.

[47] GILBERT, A. C., KOTIDIS, Y., MUTHUKRISHNAN, S. and STRAUSS, M. J. (2001). QuickSAND: Quick summary and analysis of network data. Technical Report 2001-43, DIMACS. Available at www.math.lsa.umich.edu/~annacg/ps.files/quickdimacstr.ps.

[48] GOLAB, L., DEHAAN, D., DEMAINE, E., LOPEZ-ORTIZ, A. and MUNRO, J. (2003). Identifying frequent items in sliding windows over on-line packet streams. In *Proc. ACM SIGCOMM Internet Measurement Conference* 173–178. ACM Press, New York. Available at www.imconf.net/imc-2003/papers/p318-golab.pdf.

[49] GROSSGLAUSER, M. and REXFORD, J. (2005). Passive traffic measurement for IP operations. In *The Internet as a Large-Scale Complex System* (K. Park and W. Willinger, eds.). Oxford Univ. Press. Available at www.cs.princeton.edu/~jrex/papers/sfi.pdf.

[50] HERNANDEZ, E. A., CHIDESTER, M. C. and GEORGE, A. D. (2001). Adaptive sampling for network management. *J. Network and Systems Management* **9** 409–434. Available at citeseer.nj.nec.com/544063.html.

[51] HOHN, N. and VEITCH, D. (2003). Inverting sampled traffic. In *Proc. ACM SIGCOMM Internet Measurement Conference* 222–233. ACM Press, New York. Available at www.icir.org/vern/imc-2003/papers/thinning1.pdf.

[52] HORVITZ, D. G. and THOMPSON, D. J. (1952). A generalization of sampling without replacement from a finite universe. *J. Amer. Statist. Assoc.* **47** 663–685.

[53] IETF. Packet sampling working group charter. Available at www.ietf.org/html.charters/psamp-charter.html.

[54] IETF WORKING GROUP. Internet protocol flow information export (IPFIX). Available at net.doit.wisc.edu/ipfix/.

[55] INMON CORPORATION (2004). sFlow accuracy and billing. Available at www.inmon.com/pdf/sFlowBilling.pdf.

[56] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (1984). ISO information processing systems—data communication high-level data link control procedure—frame structure. Report IS 3309, 3rd ed., ISO.

[57] JACOBSON, V., LERES, C. and MCCANNE, S. (1989). tcpdump. Available at ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[58] JEDWAB, J., PHAAL, P. and PINNA, B. (1992). Traffic estimation for the largest sources on a network, using packet sampling with limited storage. Technical Report 92-35, Hewlett–Packard Laboratories, Bristol. Available at www.hpl.hp.com/techreports/92/HPL-92-35.html.

[59] JUNIPER NETWORKS. Configure traffic sampling. Available at www.juniper.net/techpubs/software/junos/junos63/swconfig63-policy/html/sampling-config4.html.

[60] KNUTH, D. E. (1998). *The Art of Computer Programming* **3**, 3rd ed. Addison–Wesley, Reading, MA.

[61] KODIALAM, M. and LAKSHMAN, T. V. (2003). Detecting network intrusions via sampling: A game theoretic approach. In *Proc. IEEE INFOCOM 2003* **3** 1880–1889. IEEE Press, New York. Available at www.ieee-infocom.org/2003/papers/46_02.PDF.

[62] KODIALAM, M., LAKSHMAN, T. V. and MOHANTY, S. (2004). Runs based traffic estimator (RATE): A simple, memory efficient scheme for per-flow rate estimation. In *Proc. IEEE INFOCOM 2004* **3** 1808–1818. IEEE Press, New York.

[63] KRISHNAIAH, P. R. and RAO, C. R., eds. (1988). *Sampling. Handbook of Statistics* **6**. North-Holland, Amsterdam.

[64] KUMAR, A., XU, J., WANG, J., SPATSCHEK, O. and LI, L. (2004). Space-code Bloom filter for efficient per-flow traffic measurement. In *Proc. IEEE INFOCOM 2004* **3** 1762–1773. IEEE Press, New York.

[65] L'ECUYER, P. (1988). Efficient and portable combined random number generators. *Communications of the ACM* **31** 742–751. Available at doi.acm.org/10.1145/62959.62969.

[66] LEE, W. Readings in intrusion detection (online bibliography). Available at www.cc.gatech.edu/~wenke/ids-readings.html.

[67] LELAND, W. E., TAQQU, M. S., WILLINGER, W. and WILSON, D. V. (1994). On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking* **2** 1–15. Available at dx.doi.org/10.1109/90.282603.

[68] LEVEL 3 COMMUNICATIONS, INC. (2000). Level 3 promises huge savings. *Discount Long Distance Digest*. August 2000. Available at www.thedigest.com/more/119/119-73.html.

[69] MADIGAN, D., RAGHAVAN, N., DUMOUCHEL, W., NASON, M., POSSE, C. and RIDGEWAY, G. (2000). Likelihood-based data squashing: A modeling approach to instance construction. Technical report, AT&T Labs. Available at www.stat.rutgers.edu/˜madigan/PAPERS/lds.pdf.

[70] MANKU, G. S. and MOTWANI, R. (2002). Approximate frequency counts over data streams. In *Proc. 28th International Conference on Very Large Data Bases*. Available at www-db.stanford.edu/˜manku/papers/02vldb-freq.pdf.

[71] MATHIS, M. and MAHDAVI, J. (1996). Diagnosing Internet congestion with a transport layer performance tool. In *Proc. INET 96*. Available at www.isoc.org/inet96/proceedings/d3/d3_2.htm.

[72] MICHEEL, J., BRAUN, H.-W. and GRAHAM, I. (2001). Storage and bandwidth requirements for passive Internet header traces. In *Proc. Workshop on Network-Related Data Management*. Available at wand.cs.waikato.ac.nz/pubs/6/pdf/nrdm2001.pdf.

[73] MOON, S. B. and ROSCOE, T. (2001). Metadata management of terabyte datasets from an IP backbone network: Experience and challenges. In *Proc. Workshop on Network-Related Data Management*. Available at berkeley.intel-research.net/troscoe/pubs/NRDM2001.pdf.

[74] MOORE, D., PAXSON, V., SAVAGE, S., SHANNON, C., STANIFORD, S. and WEAVER, N. (2003). The spread of the sapphire/slammer worm. Technical report, CAIDA. Available at www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[75] OWEN, A. (2000). Data squashing by empirical likelihood. Technical report, Dept. Statistics, Stanford Univ. Available at www-stat.stanford.edu/˜owen/reports/squash.ps.

[76] PAXSON, V. (1994). Empirically-derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking* **2** 316–336. Available at ftp://ftp.ee.lbl.gov/papers/WAN-TCP-models.ps.gz.

[77] PAXSON, V. (1997). End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking* **5** 601–615. Available at ftp://ftp.ee.lbl.gov/papers/vp-routing-TON.ps.gz.

[78] PAXSON, V., ALMES, G., MAHDAVI, J. and MATHIS, M. (1998). Framework for IP performance metrics. RFC 2330. Available at www.ietf.org/rfc/rfc2330.txt.

[79] PEDERSON, S. and JOHNSON, M. (1990). Estimating model discrepancy. *Technometrics* **32** 305–314.

[80] PHAAL, P., PANCHEN, S. and MCKEE, N. (2001). InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks. RFC 3176. Available at www.ietf.org/rfc/rfc3176.txt.

[81] POSTEL, J. (1980). User datagram protocol. RFC 768. Available at www.ietf.org/rfc/rfc768.txt.

[82] POSTEL, J. (1981). Internet protocol. RFC 791. Available at www.ietf.org/rfc/rfc791.txt.

[83] POSTEL, J. (1981). Transmission control protocol. RFC 793. Available at www.ietf.org/rfc/rfc793.txt.

[84] QOSIENT. Argus. Available at www.qosient.com/argus/index.htm.

[85] Random sampled NetFlow. Available at www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801a7618.html#16984.

[86] REEVES, J. and PANCHEN, S. (2002). Traffic monitoring with packet-based sampling for defense against security threats. In *Proc. Passive and Active Measurement Workshop*. Available at www.sflow.org/SamplingforSecurity.pdf.

[87] REYNOLDS, J., ed. (2002). Assigned numbers: RFC 1700 is replaced by an on-line database. RFC 3232. Available at www.ietf.org/rfc/rfc3232.txt.

[88] RIVERSTONE NETWORKS, INC. Available at www.riverstonenet.com/.

[89] RIVEST, R. (1992). The MD5 message–digest algorithm. RFC 1321. Available at www.ietf.org/rfc/rfc1321.txt.

[90] SAVAGE, S., WETHERALL, D., KARLIN, A. and ANDERSON, T. (2000). Practical network support for IP traceback. In *Proc. ACM SIGCOMM 2000* 295–306. Available at www.acm.org/sigcomm/sigcomm2000/conf/paper/sigcomm2000-8-4.ps.gz.

[91] SCHERVISH, M. J. (1995). *Theory of Statistics*. Springer, New York.

[92] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., KENT, S. T. and STRAYER, W. T. (2001). Hash-based IP traceback. In *Proc. ACM SIGCOMM 2001* 3–14. Available at www.acm.org/sigcomm/sigcomm2001/p1-snoeren.pdf.

[93] SONG, D. and PERRIG, A. (2001). Advanced and authenticated marking schemes for IP traceback. In *Proc. IEEE INFOCOM 2001* **2** 878–886. IEEE Press, New York. Available at www.ieee-infocom.org/2001/paper/476.ps.

[94] TAYLOR, C. and ALVES-FOSS, J. (2001). NATE: Network analysis of anomalous traffic events, a low-cost approach. In *Proc. 2001 Workshop on New Security Paradigms* 89–96. ACM Press, New York. Available at doi.acm.org/10.1145/508171.508186.

[95] THOMPSON, S. K. and SEBER, G. A. F. (1996). *Adaptive Sampling*. Wiley, New York.

[96] VARDI, Y. (1996). Network tomography: Estimating source–destination traffic intensities from link data. *J. Amer. Statist. Assoc.* **91** 365–377.

[97] VITTER, J. S. (1985). Random sampling with a reservoir. *ACM Trans. Math. Software* **11** 37–57. Available at doi.acm.org/10.1145/3147.3165.

[98] WALDBUSSER, S. (2000). Remote network monitoring management information base. RFC 2819. Available at www.ietf.org/rfc/rfc2819.txt.

[99] WinDump—tcpdump for Windows. Available at windump.polito.it/.

[100] WOLFF, R. (1982). Poisson arrivals see time averages. *Oper. Res.* **30** 223–231.

[101] XACCT TECHNOLOGIES, INC. Available at www.amdocs.com.

[102] ZHANG, Y., DUFFIELD, N. G., PAXSON, V. and SHENKER, S. (2001). On the constancy of Internet path properties. In *Proc. ACM SIGCOMM Workshop on Internet Measurement* 197–211. ACM Press, New York. Available at www.cs.utexas.edu/users/yzhang/papers/constancy-imw01.pdf.

[103] ZSEBY, T. (2002). Deployment of sampling methods for SLA validation with non-intrusive measurements. In *Proc. Passive and Active Measurement Workshop*. Available at www.labs.agilent.com/hosted/conferences/pam2002/proceedings/Deployment_of_Sampling_Methods_for_SLA_Validation.pdf.