UNIVERSITY OF OSLO
Oslo University College

High Speed Network Sampling

Master thesis

Ole Arild Rindalsholt

23rd May 2005

Abstract

Classical Sampling methods play an important role in the current practice of Internet measurement. With today's high speed networks, routers cannot manage to generate complete Netflow data for every packet. They have to perform restricted sampling. This thesis summarizes some of the most important sampling schemes and their applications before diving into an analysis on the effect of sampling Netflow records.

Contents

Acknowledgements

I would first of all like to thank Mark Burgess for making this masters degree at all possible – without him, I would not have been writing this thesis. I would also like to thank my supervisor Tore M. Jonassen and the rest of the Oslo University College teaching staff for making me into the well-educated man I am today. Stig Jarle Fjeldbo and Håvard Wik Thorkildssen for general chit-chat and academic conundrums, Raheel A. Chaudhry for endless entertainment and the rest of the class for making school a fun place to be. My mother and father, and last, but not least, my girlfriend Sheila for sticking up with me and my weird habits.

Oslo, May 2005

Ole Arild Rindalsholt

1 Introduction

1.1 A Brief History of Internet Measurement

For many Internet service providers one problem appears when they want to analyse their data; they have either too little or too much data at their disposal. Too little in the sense that they not always can measure the interesting quantities, quantities that might be part of an aggregated measure. For instance when trying to troubleshoot packet loss, it is necessary to know the performance on each link, while it in reality might only be possible to measure performance between two hosts even though packets between them might traverse several different paths through the network.

On the other hand, the amounts of data being collected are huge. One high speed interface could generate hundreds of gigabytes of unsampled data each day, and a large service provider might have several thousand network interfaces to monitor. On top of that comes statistics generated by protocols such as SNMP, which could also generate gigabytes of data each day. And the rate of data collection is increasing, due to demand for both faster lines and even more detailed measurements. As a result, advanced measurement devices are being equipped on the routers and switches, leaving the service providers with even more data.

The reason for this current lack of measurement capabilities lies in the original design of the Internet. It is designed so that the endpoints do not need any details about the network connecting them, the functionality for transmitting data from one host to another is divided into layers communicating through standard interfaces. Because of this layering, there are barely any measurement mechanisms embedded in the Internet protocols, leaving the interior of the network to the lower layers, unable for the endpoints to monitor. In order to reveal some of the details we need to use some sort of hack. (*traceroute* is such a hack)

The Best Effort model of the Internet only added to the problem when trying to do detailed measurements. Since it didn't offer any solid performance guarantees, there was no need to build in ways of confirming these guarantees through measurements. Connectivity, link failures and rerouting should not need to concern the endpoints, leaving them happily(?) unaware and leaving the problem to the network layer. Today only aggregate loss and utilization statistics are ubiquitously reported by router interfaces.

1.2 Data Volumes and the Need for Sampling

As we have seen, the need for detailed measurements are caused by services that need more details and control than the Best effort model of the Internet can offer. Service providers need to characterize traffic down to the level of individual customers in order to be able to keep their service level agreements. This detailed monitoring could also be used for billing customers based on usage. Also, real time applications forces the service providers to be able to react to congestion and take corrective action before their agreements are violated, demanding measurements at a fine time scale.

However, collecting and processing the data from these measurements is not as easy as it sounds. The data often needs to be transported to collection points for storage and analysis, and involves costly operations for the network. It consumes resources on the routers and switches, already loaded with regular routing and switching. In addition comes the network bandwidth spent on this transport. And on top of that is sophisticated and expensive data equipment needed for analysis and storage of the data. We see the need for some kind of data reduction. But how much data can be reduced while still supplying detailed enough measurements for applications? This is a constant tug a war, especially where resources are scarce, typically the observation point.

Reducing the data is usually carried out online in a single pass through the data stream in order to avoid buffering and reprocessing. There are currently three common methods:

- *Aggregation*. Combining several data into a single composite, and then discarding the original components. This could for instance be the total traffic from a set of sources or over a time interval. Aggregation provides us with compact data summaries when we are allowed to lose visibility of the aggregate's components.

- *Filtering*. Selecting data based on the data values, discarding everything that is not selected. This could for instance be selecting data only from a certain source or destination, or even both. This is most useful when we have identified the interesting data and want to remove everything else.

- *Sampling*. Random or pseudorandom selection of data, where the unselected data is discarded. This could for instance be a simple random selection of packets.

The main difference between these methods is that aggregation and filtering requires some prior knowledge of the data, while sampling both retains the arbitrary details and reduces the amount of data.

When moving from the observation point and to the collection point or intermediate points in the collection infrastructure, we allow more working storage and maybe even the possibility of multiple passes on the data. This makes it possible to deploy other reduction methods in addition to the three mentioned; aggregation, filtering and sampling.

1.3 Challenges in Sampling and Analysing Sampled Network Data

But there are also some statistical challenges when sampling and analysing network measurements:

Most available data has already been sampled during collection. As we have seen earlier, there are good reasons for decreasing the amount of data as soon as possible, and as a result of that, unsampled raw data is hard to find. We need to find out what the sampled data tell us about the original data.

Technology or resources may limit implementations of sample designs. Even though a sample design might seem ideal from a statistical point of view, technological constraints may limit the ability to use it. Different approaches are implemented to approximate the ideal, an ideal

that might be considered subjective in each case. Since the measurements might travel through several subsystems on its way from the observation point to the data repository, each stage gives us an opportunity for sampling. The problem is to find out at what stage the sampling is best performed.

The choice of sampling design should be based on traffic characteristics. Studies show that some traffic statistics lead to heavy-tailed distributions, a factor that needs to be taken into consideration when designing a sampling scheme.

The choice of sampling design should be based on the statistics needed by applications. On general, there is no set of traffic statistics that is most useful for network management. Sample design could be optimal with respect to one set of statistics, while still suboptimal for another set of statistics that could play an important role for a future application. Because of this, trade-offs between flexibility and statistical efficiency plays an important role and needs to be analysed thoroughly for a sample design.

2 State of the Art

2.1 Internet Protocols and their Functions

The packets travelling across the Internet are constructed according to a layered set of protocols specifying the packet layout and the function of the packet content. The content of one protocol may form the payload of a lower layer protocol, encapsulating the higher level packets.

When we are using passive packet measurement we are mainly interested in the protocol header information. The information in the network protocol headers are used by routers to forward packets towards their destination. The Internet protocol (IP) [35] is by far the most common network protocol, even though multi protocol label switching (MPLS) [6] is becoming increasingly popular. These protocols are network protocols encapsulating the transport protocols used by end-hosts to supervise transmissions and direct the packets toward the correct end-host application. There are currently two transport protocols that are dominating; the transmission control protocol (TCP) [35] and the user datagram protocol (UDP) [34]. These transport protocols again encapsulate the data to be transmitted, data that may very well be generated through application level protocols. This could for instance be the hypertext transfer protocol (HTTP) [4, 21] that is used for Web transfers.

2.1.1 The Information in Protocol Headers

There are mainly three reasons why packet protocol headers are so interesting for network managers:

- *Protocol headers determine how the network should treat the packet.* This could for instance be how to route the packet or how high priority it should have.

- *Attribution of the packet origin.* Used in combination with other information such as routing and topology, the source and destination addresses in the IP header fields can be used to attribute who is responsible for the packets presence.

- *Application characterization.* Both TCP and UDP protocol headers contains a port number field used to direct packets to the correct application at its destination [39]. Since these port numbers are standardized or conventional for several applications, an inspection of the packet's port numbers can often infer the responsible application. Some networks use this information to block file sharing applications from the internet, basing it on the applications port numbers. (This has back-fired in some sense, forcing the applications to ignore the conventions to evade this characterization.) Security applications can identify networks attacks by matching the packet header fields to known signatures of known attacks. For protocols such as TCP, states can be tracked through flags set in the headers.

The actual payload of the packets sent and received by the host are usually not interesting from a network management point of view. This usually does not affect the performance of the network, however there are some exceptions. One is data from routing protocol packets, allowing us to understand the routing state of the network. Another is the URLs in certain HTTP packets, that helps us determine the usage of content resources at Web servers.

2.1.2 Applications of Usage Measurements

Most network management applications employ measured traffic usage, either in packets or bytes, based on the header fields. These data are differentiated into classes at some granularity that depends on the application requirements. Here are some examples:

- *Deployment of services.* By identifying applications by their TCP/UDP port number, service providers track the growth of new applications and identify potential new customers using them.

- *Heavy hitters.* Determining dominant components within a traffic class, for instance the most popular web pages based on the IP destination address inn the HTTP requests.

- *Security applications.* Detect usage indicatives of intrusions in the network, that could be changes in patterns and usage of specific protocols and ports, or most active hosts and networks. See [38] and [42] for more.

- *Network engineering.* By examining the intensities of traffic between a set of source and destination addresses carried over a congested link, a service provider could get the information needed to determine whether a rerouting of traffic to avoid the congestion is possible. [17, 18]

- *Chargeback.* A corporate intranet might spread their costs to their subdivisions, based on usage from the subdivisions IP range.

- *Customer billing.* A service provider might choose the same approach, charging customers for their usage, identified by their IP ranges. The charging could be application dependant, based on TCP/UDP port numbers. InMon's sFlow [25] proposes the use of packet samples for billing.

Even though the applications above all are based on usage measurements, the accuracy requirements are quite different. The list above is roughly ordered by stringency, customer billing being the most stringent. Overbilling customers would both be illegal and bad for business. Network management applications though, could probably tolerate errors of several percent when estimating usage.

2.2 Active and Passive Measurement

We usually divide network measurements into two categories; active and passive measurements. These two generally focus on different aspects of network behaviour. Active measurement sends probe packets between hosts, making it well-suited for end-to-end performance measurement since probes can be sent from any accessible host. By observing

sequence numbers at the destination, packet loss can be inferred, while end-to-end delay can be determined by comparing synchronized probe time stamps at the source and destination. Packet content could also be of interest, often used to differentiate treatment by the routers based on their IP header (type of service) fields.

Tools such as *ping* and *traceroute* are popular active measurement tools allowing the users to measure roundtrip performance without privileged access to the network interior. The only requirement is that the destination responds to Internet Control Message Protocol (ICMP) packets, something that is not always the case. Less known are perhaps tools for estimating throughput, such as the *treno* tool [31], using a stream of probes to conform to the dynamics of TCP.

Passive measurement, on the other hand, uses the routers or other hosts to measure already existing traffic passing through or destined to them. No packet content should be altered by this measurement, and the disruption to the normal passage of traffic through the network should at worst be negligible. A router should for instance not spend all its resources measuring, letting its normal tasks of routing and forwarding suffer. Exporting measurements to a collector would of course consume some network resources and cause a small perturbation in traffic patterns, but this traffic should not consume more than its fair share of bandwidth under normal operating conditions. During overload conditions though, it would be desirable to allow it to claim what it needed to maintain measurement collection. We can see that data reduction before transmission clearly plays an important role when we want to limit the measurement transmission bandwidth.

2.2.1 MIBs and SNMP Statistics

We usually collect passive measurements in three ways: By (1) polling management information base (MIB) data from routers, (2) monitoring packets and (3) flow monitoring. A MIB database contains course grained statistics from the routers, however the information is highly aggregated; only counting packets and bytes transmitted and lost at an interface. These values are polled from the routers using SNMP, the simple network management protocol [6]. RMON [43] is a standardized MIB designed for remote monitoring, which can be configured to compute traffic statistics, and recognize and respond to predefined network conditions by capturing packets or raising alarms. However, this complexity limits its implementations to low speed interfaces, and the MIB approach is not well suited to the continuous measurement and export of detailed traffic data. Packet and flow monitoring is currently used for this purpose.

2.2.2 Packet Monitoring

Packet monitoring describes an operation where we passively copies a stream of packets and then select, store, analyse and/or export information on these packets. Traditionally, packet monitoring was exclusively performed by special purpose hosts in the network; see [2, 5, 10, 20]. A monitor will receive a copy of the packet stream in one of three ways: either using an optical splitter to copy the physical signal carrying the packets and bring this signal to one of the monitor interfaces, attach the monitor to a shared medium carrying the traffic; or by using a router or switch to copy packets to a monitored interface.

Packet monitors have high demands when it comes to resources, especially when it comes to processing bandwidth needed to work at the full line rate of high speed links. Capturing only some initial bytes of the packets is a common way to restrict the data capture and control the data bandwidth at the monitor. Since the IP header and other protocol headers are located near the start of the packets, this makes a reasonable solution. But still, continuous collection, transmission and storage of unreduced packets have not been feasible for a number of years due to the immense volumes of data compared to the capacity of systems collecting them, [1, 32]. Full packet header traces is only feasible for limited durations, and instead, applications requiring continous monitoring for a longer period tend to perform analysis at or near the monitor by forming flow records (see section 2.3 below) or other aggregate statistics [2], or even a more general stream querying functionality [10]. The packet IP headers are commonly collected using *tcpdump* [26] or the Windows version *windump* [44]. These tools may also be able to capture parts of the packet payload, depending on the traffic load and processing power at the host.

2.2.3 Embedded Packet Monitoring in Network Elements

Equipment availability and administrative costs are the main limiting factors when it comes to deployment of packet monitors. A recent approach is to embed passive measurement functionality within network elements such as routers and switches. This allows packet measurement to become ubiquitous in the network. Little or no measurement analysis capabilities should be expected however, since routers and switches generally lack the additional computing power needed for this purpose. We need some sort of data reduction instead, both when selecting information from packets and when selecting packets to be reported. Packet sampling capabilities are becoming available in routers, for instance InMon's sFlow [33].

The Packet Sampling (PSAMP) Working Group of the Internet Engineering Task Force (IETF) [53] is currently standardizing packet selection capabilities for network elements in order to define a set of capabilities that are simple enough to be ubiquitously deployed while still rich enough to support the needs of measurement-based network management applications. This will most likely include filtering and various forms of sampling.

2.3 Flow Records

A traffic flowed is defined as a set of packets with a common property, the flow key, observed within a period of time. Summary statistics on packet flows that pass through them are usually constructed and exported by routers. One can think of a flow record as a summary of a set of packets that comprises a higher level transaction, such as a remote terminal session or a Web-page download. The set of packets included in a flow depends on the algorithm used to assign packets to the flow by the router. The flow key is usually specified by packet header fields, for instance the IP source and destination address combined with TCP/UDP port numbers. Flows that are specified by an individual value of these fields are often called *raw* flows, in contrast to *aggregate* flows that are specified by a range of these quantities.

Flow statistics are created in the following manner. Every time a packet arrives at the router, the router determines whether or not the flow is active, that is, whether or not statistics are currently being collected for that packet's key. If not, a new set of statistics are initiated by the

packet and its key. These statistics include counters for packets and bytes updated according to the number of packets matching the key. The router decides when the flow is terminated, exports the statistics in a flow record and frees the memory for new flows. Flows are commonly terminated if one of the following criteria is met: (i) Inactive flow or interpacket timeout, in other words if the time since the last packet for the flow exceeds a threshold. (ii) Protocol level information terminating the TCP connection, such as a TCP FIN packet. (iii) Memory management, releasing memory for new flows. Or (iv) active flow timeout, terminating flows after a certain time since the arrival of the first packet of the flow.

The IP Flow Information Export (IPFIX) [24] Working Group of the IETF is currently standardizing flow definition schemes that have been developed in research environments [2, 9]. A flow record usually includes the properties defining the key, arrival times of the first and last packet, number of packets and bytes in the flow.

Flows are summarizes in a fixed length record regardless of the number of packets in the flow, and thus yields considerable compression of information. In return we loose details of the timing of packets within the flow. How well the traffic is compressed depends on the composition, greater compression for long flows and smaller for short flows. In backbone traffic mixes, byte compression factors for IP and transport headers versus NetFlow records of 25 or more are commonly attainable.

2.3.1 Sampling in the Formation and Collection of Flow Records

In order to produce flow records, we need three sets of resources involved: 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 involved in analysing the statistics at the collector. Usage of each of these resources can be controlled through sampling.

Sampling in flows is fundamentally different than packet sampling due to the statistical properties of flows. While packet size has a maximum value depending on the transmission technology, studies have shown that flow length distribution is heavy-tailed; a large proportion of the total bytes and packets in the packet stream occur in a small proportion of the flows [19]. This is making both uniform sampling and uncontrolled sampling far more problematic for flow records than for sampled packets, since omission of a single flow report can have huge impact on estimated usage.

NetFlow uses UDP to export its flow records, a transport method that is not equipped with any capability for reliable transmission. This capability is being incorporated in the relevant standards under development in the IPFIX Working Group of the IETF [24]. For now, we have to minimize report loss by locating staging collectors close to routers, for instance in switching centres housing a number of routers. The network that the reports are transmitted over should be congestion free, at least under most operating conditions. The flow records can then be retransmitted to their ultimate destination by the stagers, using a more reliable transport protocol such as TCP. The staging collectors could also perform analysis, and even with future reliable transport from the observation point, staging collectors may be desirable for scalability, since they also may perform data reduction and field queries. [13] describes such a system.

2.3.2 Packet Selection

When forming flow records, the main resource constraint is at the router flow cache where the keys of active flows are maintained. In order to look up packet keys at the line speed of the router interfaces, the cache is required to operate in fast and expensive memory. And since the routers need to carry increasingly large numbers of flows concurrently, a large cache is necessary. By sampling the packet stream before the flow records are constructed, the time available for each flow cache lookup is prolonged, allowing storage to be carried out in slower, less expensive memory.

Cisco has employed systematic sampling based on packet count for their Sampled NetFlow feature. The more recent random sampled NetFlow [37] employs stratified sampling based on arrival count. One packet is selected randomly out of every window on N consecutive arrivals, allowing two consecutive packets to be sampled. However, more than two consecutive packets are never sampled as long as $N > 1$, and longer backlogs do not develop as long as the mean rate of sampled packets can be accommodated.

If the sampling period is much larger than the typical flow length, only one packet per flow will usually be reported. If this was the case, we might as well sample packets rather than constructing flow records, saving resources at the router since there would be no need to cache the single packet flows until expiration of the interpacket timeout.

But are flow records redundant when facing packet sampling? Since web traffic is such a large component of Internet traffic and the average flow length of such traffic is quite small [19], there are many short flows. But on the other hand, there are several reasons to expect that longer flow will account for most of the traffic: (i) Internet traffic is increasingly being dominated by file sharing applications, where flows of hundreds and thousands of packets are the norm. (ii) Flows carrying Internet telephony packets should be rather long lived. (iii) Virtual Private Networks, where traffic from many sources will be seen as a single aggregate flow. So unless packet sampling periods become large compared to the length of these flows, flow statistics will continue to afford useful information in the future.

Sampling can actually *increase* the number of flows generated from a flow where the typical time between sampled packets exceeds the flow interpacket timeout T. Such flows were called sparse by Duffield, Lund and Thorup [14].

2.3.3 Cache Selection

There are several sampling schemes that 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.

The *sample-and-hold* approach of Estan and Varghese [16] uses a cache lookup for the key of each incoming packet, and if they find a match, statistics are updates as usual for a flow with matching key. If a match is not found, however, 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. This gives the probability that a flow of b bytes is not sampled at all to be $(1-p)^b$. The bytes reported for a flow will thus never exceed the actual bytes, and unless the first packet is selected, there will

be an undercount. The relative error however, will be small for larger flows and can be adjusted by adjusting p.

With sample-and-hold, it is impossible to form an unbiased estimator of the true flow sizes because the number of bytes that have not been sampled is unknown. The probability for a given flow to have been sampled, only knowing the sampled bytes, is also unknown. An exception is the special case where all the packets are the same size.

Sample-and-hold requires a cache lookup for all packets. As we mentioned earlier, this requires fast and expensive memory. The advantage of sample-and-hold compared to not sampling at all is since most of the usage occurs in a small proportion of long flows and small flows tend to not be instantiated, it is possible to reduce the size of the cache.

The runs based traffic estimator (RATE) [28] by Kodialam, Lakshman and Mohanty, only instantiate and update the flow cache entries when the same key is observed in a run of two back-to-back packets. They achieve this by maintaining a register containing the key of the last packet observed and comparing it against the key of the incoming packet. The approach favours longer flows, since they have a higher chance of forming such a run. The statistics are used to estimate the amount of high-volume traffic components.

2.3.4 Report Selection

Sampling can also be applied to completed flow records, either prior to export or in the collection infrastructure. Uniform sampling is problematic due to the consequences of omitting records of large flows, motivating sampling that is dependent on the size of the flow being reported. It could be as easy as discarding flows with a byte size falling below a certain threshold. However, a user or application splitting their traffic up into small flows could evade measurement altogether, a serious weakness for accounting or security applications. Duffield, Lund and Thorup [12] proposed nonuniform probability sampling based on flow bytes. Flows of byte size greater than a threshold are sampled with probability 1, otherwise they are sampled with a probability proportional to their byte size.

Discard-below-threshold and sample-and-hold never overestimate usage, and even though the customers would never be overcharged, the shortfall for the provider is unconstrained. On the other hand, optimal sampling can overestimate byte usage, a problem especially for billing applications. [12] proposes a flat rate billing for usage up to a level L, with proportionate charge only on usage above this level. This makes the billing scheme insensitive to errors in estimating small usage.

3 Methods

The original idea of this project was to study the effects that different methods of sampling have on the precision of the NetFlow data. A day's worth of network traffic was to be stored using a passive measure device, and used to generate NetFlow data with and without sampling in order to compare the results. However, as mentioned earlier, a day's worth of network traffic on a high-speed line could be several hundred gigabytes, and we soon decided that it would be more practical to work with smaller tracefiles. We ended up using two files, at approximately 5 and 2 gigabytes. These files were collected from a high-speed line using a DAG card. The commands used were *dagsnap* to capture the traffic from the card and store them to disk, and then later *dagconvert* to convert them to the well-known pcap-format. DAG cards are custom designed for passive measurements.

The next step in the process would now be to generate NetFlow records, a process proving to be harder than first assumed. For this we needed some way to create and collect these records. We soon found out that the popular *flow-tools* could be of use here. Flow-tools is library and a collection of programs used to collect, send, process, and generate reports from NetFlow data, and proved to be usable both when we wanted to collect and store the flows and later on, when we wanted to create statistics from these flow records. Creating flow records from a pcap file, however, is not yet implemented in these tools. We needed some kind of NetFlow generator.

We were mainly after open-source software, both because it is cheaper and because the experiments can more easily be reproduced. We were therefore much exited when we discovered the *nProbe* NetFlow probe. nProbe is an offspring of the perhaps more known ntop utility, and is accommodating both reading from a trace file and even containing a sampling option when creating flow records. A beta release was available to the public and we decided to use it to generate our flow records. We now had a complete setup with a pcap trace file, a NetFlow probe, and a NetFlow collector and a statistics generator in flow-tools, and started to do measurements.

Things were running very smooth, capturing flows unsampled, sampled with a factor of 10 and sampled with a factor of 100. A serious problem occurred however, when we found out that the beta release of nProbe only captured a maximum of 2000 flows, not nearly enough to provide the flow records needed for our large traces. And even though nProbe was under the GPL licence, the source code was not freely available for us to make some modifications, leaving us with "our beards in the mailbox" and several unusable measurements. Some of the measurements are included in the results though, since they illustrate some trends in packet sampling in flow records. A last-minute replacement probe became the *softflowd* utility, another NetFlow probe capable of reading from trace files, but without that all-important sampling facility.

It has later been suggested that we should have used the *Net::Pcap* module to the libpcap library. This module would have allowed us to write relatively small perl-scripts capable of handling pcap files and among other things perform sampling on these files, a feature that

would have allowed us to continue an approach similar to the one we used with nProbe. Unfortunatly, we were running out of time.

Reports were created by using a couple of flow-tools, using flow-cat to cat the file and flow-stat to generate the statistics wanted. Then, the needed tables can be plotted with most plotting tools, we used Microsoft Excel in this thesis.

4 Results

Even though the most exciting results didn't show up, we still managed to get some of our suspicions confirmed. We start by examining and comparing the unsampled flow records created from the two trace files. Even though the two files were captured on different times and one is much larger than the other, we can clearly see that they are characterizing the same type of traffic when looking at the IP Packet Size Distribution in Figure 1.

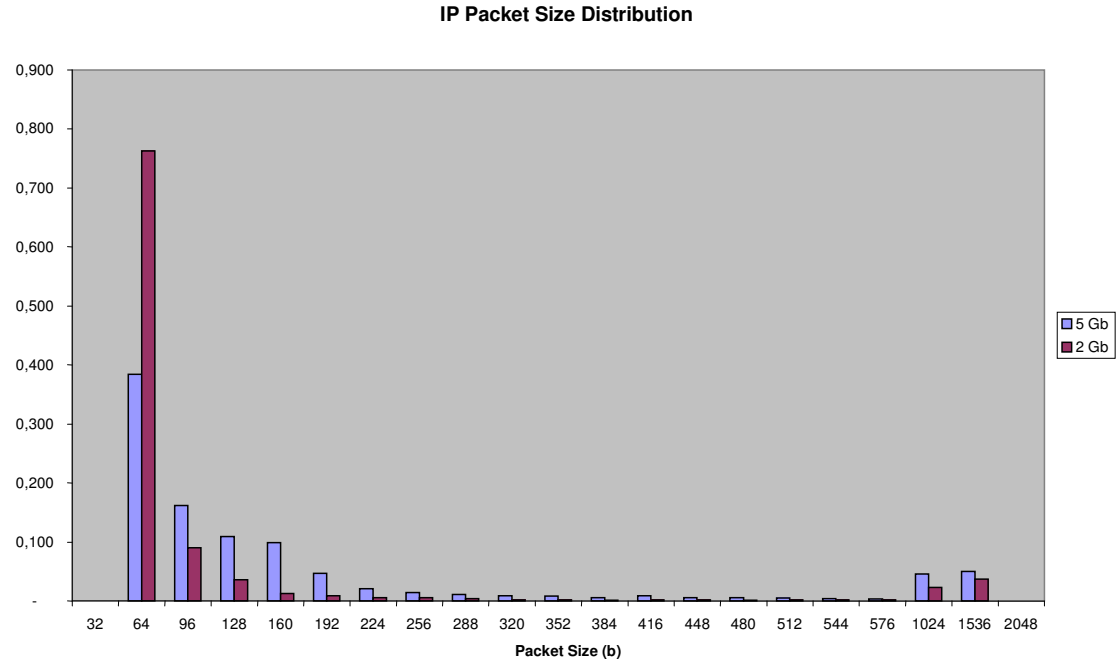

Figure 1

We see the same thing when looking at the Packets per Flow Distribution in Figure 2. The two captured files are more or less following the same patterns. These values are normalized and can thus easily be compared with another, even though the measured amount of data are quite different in the two trace files and flow records. This brings us to the amount of data saved by using the abstraction of flow records rather than trace files.

The trace pcap files we used for generation were *4.644.602.798* and *1.974.297.606* bytes, while the flow records of the same captures were *23.205.016* and *1.527.896* bytes respectively. The flow records are a stunningly approximated 0,5% and 0,077% of the original capture. Even more amazing is it that this was performed without any sampling whatsoever, only the abstraction of flow records. These sample captures and flows clearly states the reason for creating flows rather than transmitting trace files through the network, using unnecessary bandwidth.

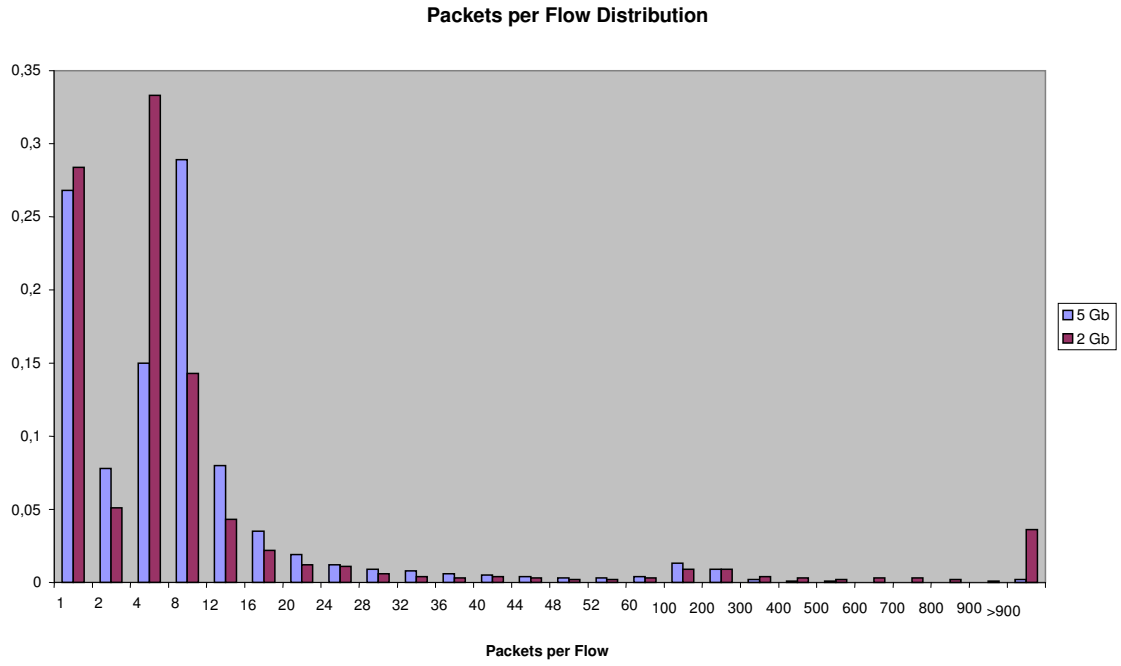

Figure 2

In Figure 3 we see a somewhat unexpected effect of the different capture sizes. The figure shows the Flow Time Distribution, a distribution telling us how long the flows are active. We see that the two captures follow roughly the same pattern, however the 2 Gb-capture have three distinctive spikes on the distribution, accounting for almost ¾ of the traffic flows.

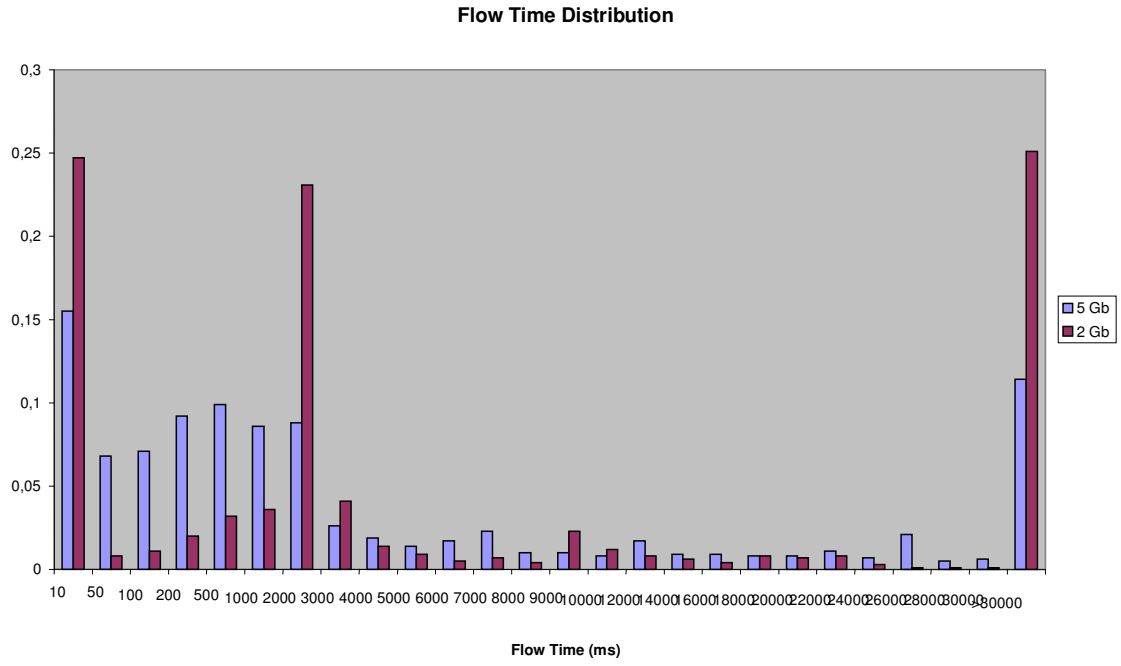

Figure 3

One would perhaps expect the 5 Gb-trace to contain most of the longer flows, since it is spanning over a longer time period and thus opens for longer flows than the 2 Gb-trace. The opposite is the case, and the reason for this probably lies in the applications and protocols used at the time. Assuming that the 5 Gb-trace is considered "normal" traffic, such spikes as we see in the 2 Gb-trace may call for a closer inspection of the trace in order to find out what is causing the disturbance.

The figures seen below are generated using nProbe. Figure 5, Figure 6 and Figure 7 shows the 5 Gb-trace expressed with 2000 flows, and no sampling, 1:10 sampling and 1:100 sampling respectively.

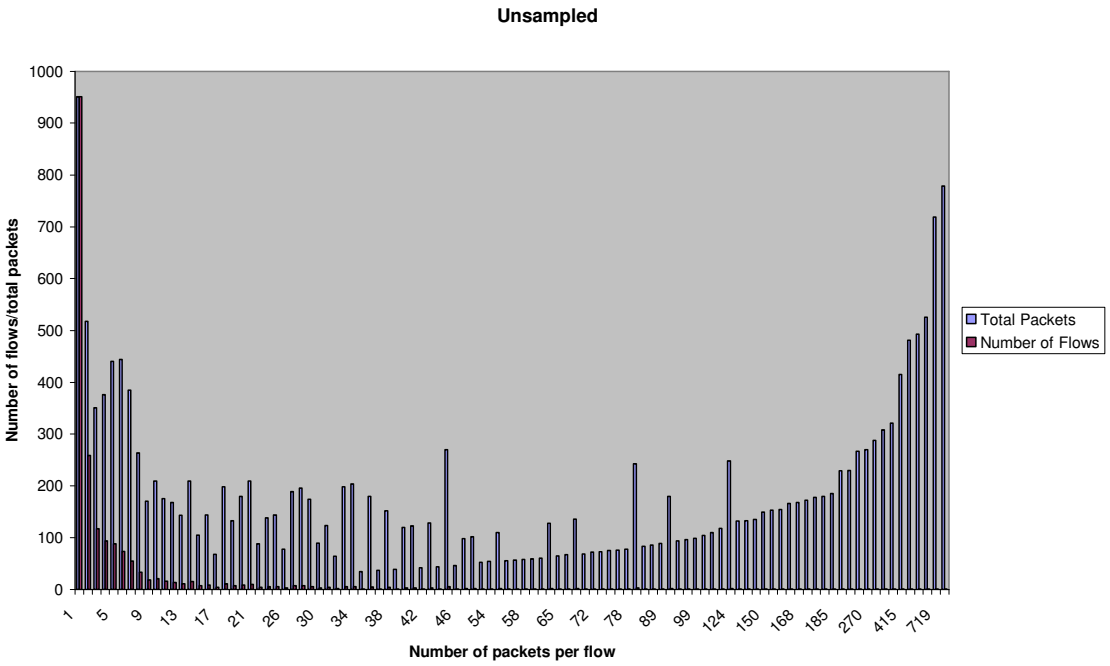

Figure 4

And even though we are using the same file as a basis, the three flows can be seen as almost separate, since they are spanning over different time intervals and share a very small portion of the data. For while an unsampled 3000 flow record barely gets to start on a high-speed link capture, the 1:10 sampled record will go 10 times longer into the capture, while the 1:100 sampled record will go 100 times as long, give the same traffic density and type of traffic. In other words, an unsampled 3 second flow record could be the same size as a record containing 300 seconds worth of traffic, sampled at a 1:100 rate. However, this abstraction comes with a cost. Flows may come and go very quickly, and while sampling every packet is taking up resources, a 1:100 sampling will miss 99 out of 100 single packet flows traversing the net. It is all about how sensitive the application using the data is.

But flow records based on sampling tend to contain more one-packet flows than similar unsampled ones. This is due to the fact that sampled flows are counting less packets than the actual flow contains unless the flow is sending in period with the sampling rate.

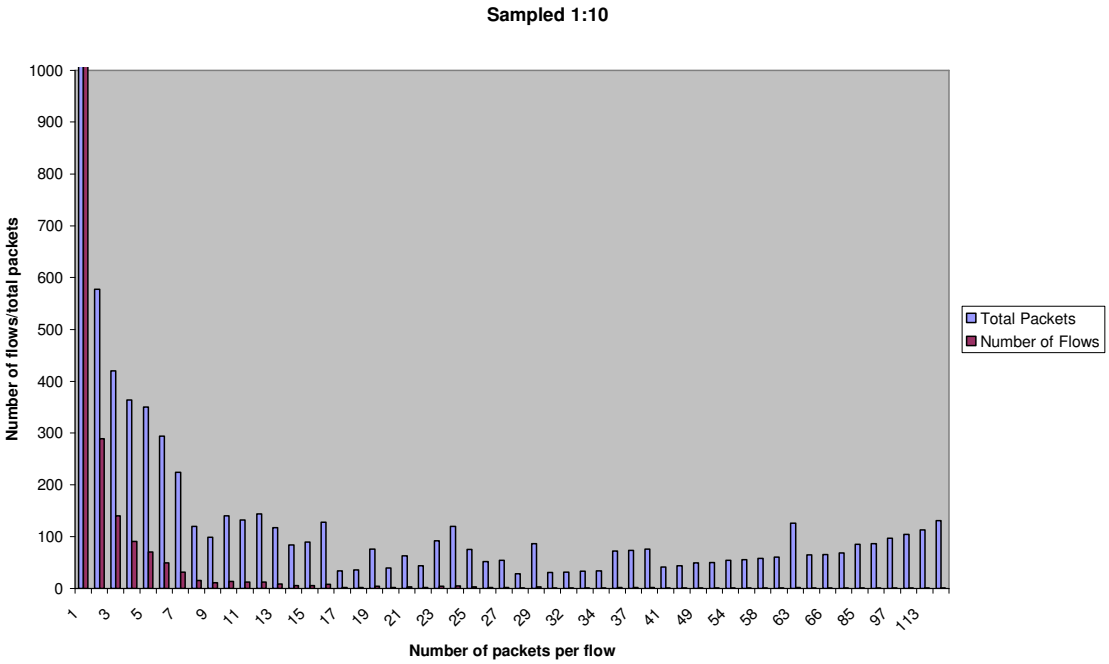

Figure 5

We are loosing both packets and even whole flows, pushing the graph more towards 1 packet per flow the higher we sample. In the extreme case where the sampling rate is so high that all flows expires during one sampling period, a flows would contain one packet, thus making the flow records obsolete to the packet capture. This is not very well illustrated in the tables, partly because the tables are normalized to a max value of 1000 flows, hiding everything above, and partly because the sampling rate is not high enough compared to the flow sizes. Sampling every 10.000 packet would most likely leave us with one packet per flow.

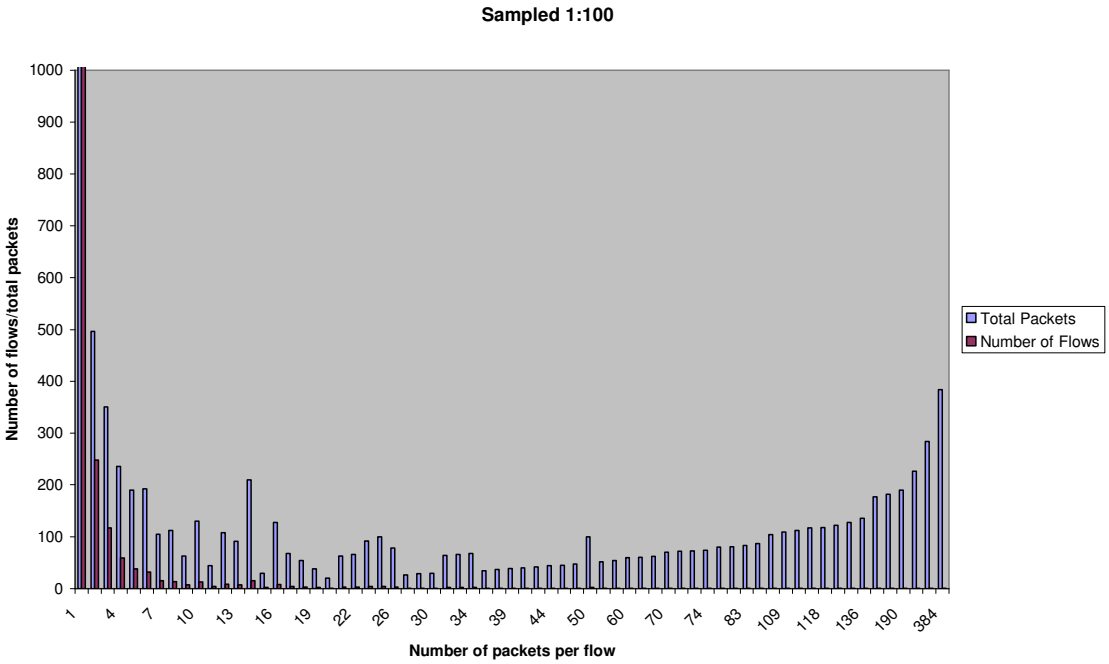

Figure 6

The patterns seen in Figure 7, Figure 8 and Figure 9 are similar to the ones in the figures above, not surprisingly since they are created in the same manner, only from the 2 Mb-trace.

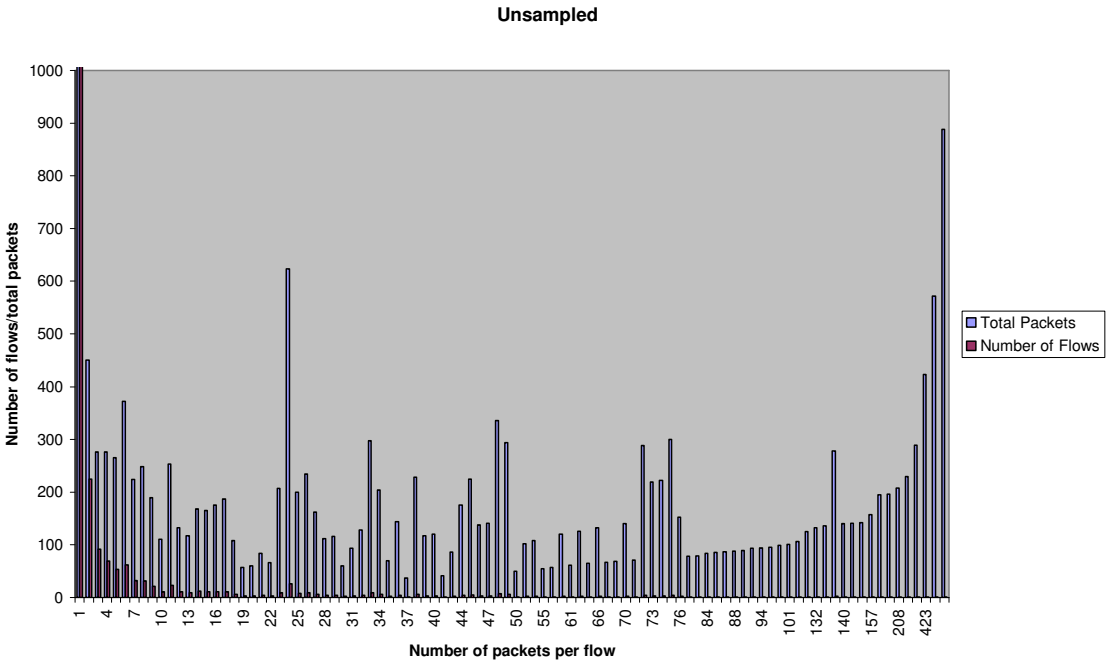

Figure 7

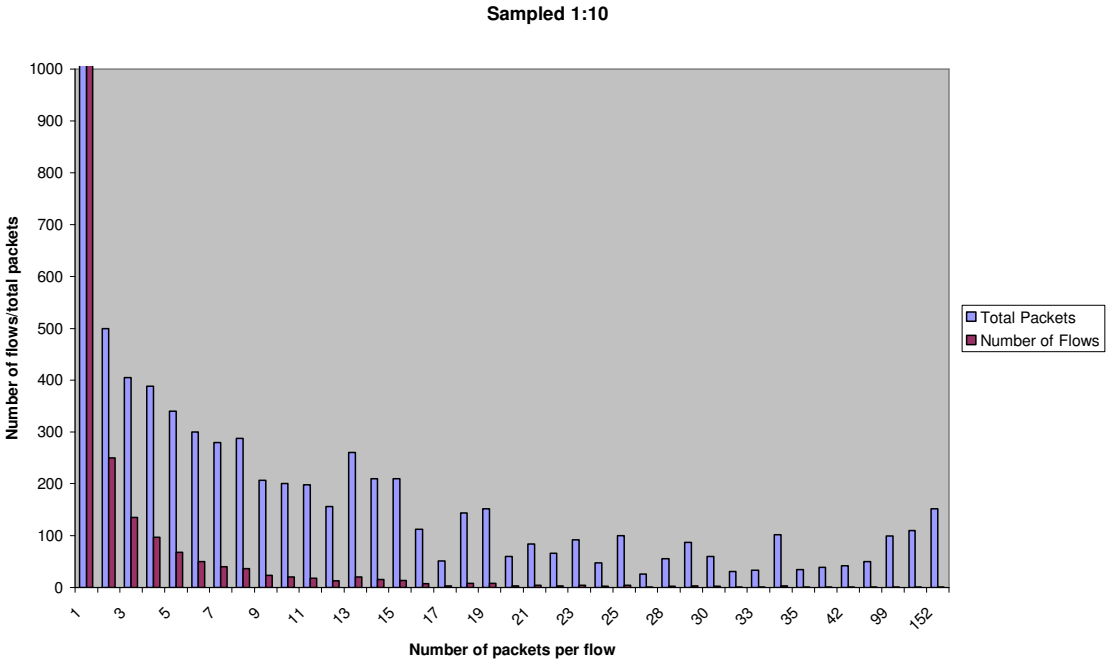

Figure 8

The same is happening here, with the tables sliding to the left as the sampling rate is getting higher. A problem for billing applications are flows that disappear due to sampling. Since they are not detected, they are assumed not to be there and are not taken into consideration when for instance calculating after a usage-based scheme. The total usage is still known though, leaving someone else to pay for the disappeared flows. Charging other customers too much however, can not be tolerated, and that is why complex sampling rules are used when it comes to sample based charging.

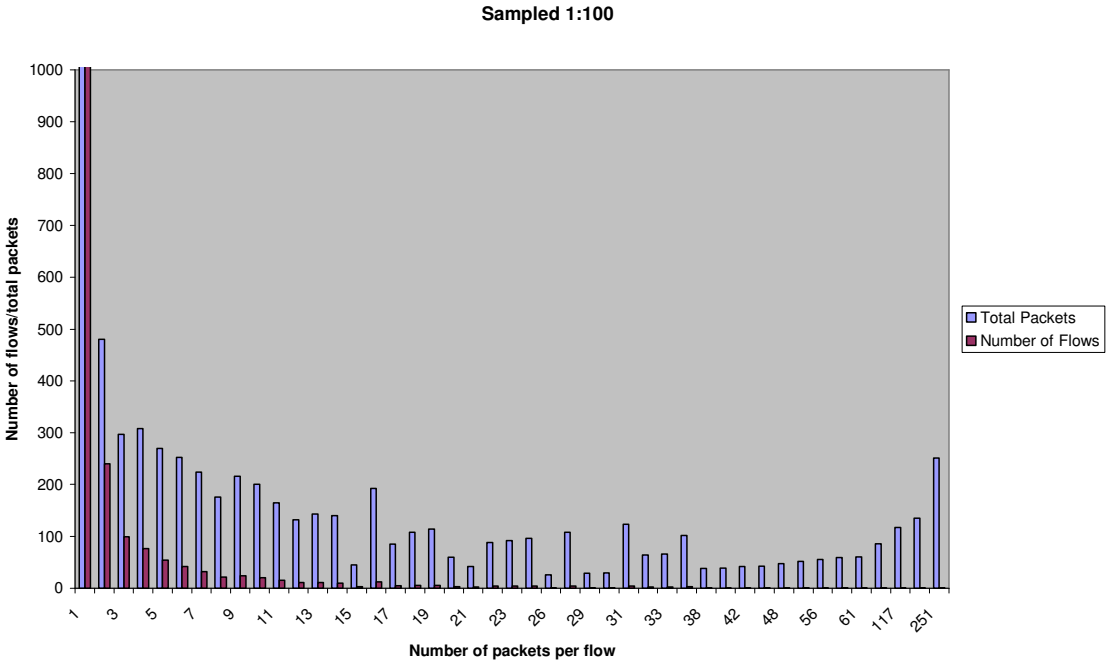

Figure 9

5 Conclusions

There is no doubt that packet sampling currently is one of the most important issues in the field of system and network administration. Due to the demand for increasingly faster lines, both commercial and non-commercial groups are surging to find a sampling scheme suitable for most applications and efficient enough to relief the struggling backbone routers and switches.

One of the major problems when working with relatively new topics like this is the lack of information. It is often required to use hacks, making the way as we go. The tools used often have to be written, re-written, modified or used in a way they were not supposed to. This way of working can be both extremely frustrating when things are not going your way, and on the other hand, extremely rewarding when things work out the way you want them to. But the relative lack of right and wrong can make hard to tell if you are pulling in the right direction, allowing you to spend hours and days on hopeless attempts. And that might have been the main reason for this thesis' failure to produce the results we were hoping for; too many time-consuming operations leading nowhere.

References

[1] 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.

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

[3] ARMITAGE, G. J. (2000).
MPLS—The magic behind the myths.
IEEE Communications Magazine 38(1) 124–131.

[4] 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.

[5] 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.

[6] 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.

[7] CHOI, BY., PARK, J., ZHANG, ZL (2002)
Adaptive Random Sampling for Load Change Detection
Proceedings of the 2002 ACM SIGMETRICS international …

[8] CLAFFY, KC., POLYZON, GC., BRAUN, HW. (1993)
Application of Sampling Methodologies to Network Traffic Characterization
Proceedings of ACM SIGCOMM

[9] 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.

[10] 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.

[11] DUFFIELD, N.G. (2004)
Sampling for Passive Internet Measurement
Statistical Science Vol. 19. No. 3, 472-498
Institute of Mathematical Statistics

[12] 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.

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

[14] 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.

[15] DUFFIELD, NG., GROSSGLAUER, M. (2001)
Trajectory Sampling for Direct Traffic Observation
IEEE/ACM Transactions on Networking,

[16] 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.

[17] FELDMANN, A., GREENBERG, A., LUND, C., REINGOLD, N. and REXFORD, J. (2000).
NetScope: Traffic engineering for IP networks.
IEEE Network 14(2) 11–19.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] GOYAL, M., GUERIN, R., RAJAN., R (2002)
Predicting TCP Throughput From Non-invasive Network Sampling
Proceedings of IEEE INFOCOM'02

[23] HERNANDEZ, EA., CHIDESTER, MC., GEORGE, AD. (2001)
Adaptive Sampling for Network Management
Journal of Network and Systems Management

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

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

[26] JACOBSON, V., LERES, C. andMCCANNE, S. (1989).
tcpdump.

Available at ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

[27] KESSLER, RE., HILL, MD., WOOD, DA. (1994)
A Comparison of Trace-Sampling Techniques for Multi-Megabyte Caches
IEEE Transactions on Computers, 1994

[28] 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.

[29] LELAND, WE., TAGGU, MS., WILLINGER, W, WILSON, DV. (1994)
On the self-similar nature of Ethernet traffic (extended version)
IEEE/ACM Transactions on Networking

[30] MANKU, GS., RAJAGOPALAN, S., LINDSAY, BG. (1999)
Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets
Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of …

[31] MATHIS, M. and MAHDAVI, J. (1996).
Diagnosing Internet congestion with a transport layer performance tool.
In Proc. INET 96.

[32] 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.

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

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

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

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

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

[38] REEVES, J. and PANCHEN, S. (2002).
Traffic monitoring with packet-based sampling for defense against security threats.
In Proc. Passive and Active Measurement Workshop.

[39] 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.

[40] SANG, A., LI, S. (2000)
A Predictability Analysis of Network Traffic
Proceedings of INFOCOM 2000

[41] SMITH, PJ., SHAFI, M., GAO, H. (1997)
Quick simulation: A review of importance sampling techniques in communications systems
IEEE Journal on Selected Areas in Communications

[42] 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.

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

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