Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Jonathan Gysel

Anomaly Visibility and Detectability in Sampled Traffic

Semester Thesis SA-2007-12
October 2006 to February 2007

Tutor: Daniela Brauckhoff
Co-Tutor: Bernhard Tellenbach
Supervisor: Prof. Dr. Bernhard Plattner

Contents

List of Figures

List of Tables

Abstract

In [1] packet and flow metrics based on entropy were proposed to detect anomalies in Internet backbone networks, since entropy measurement methods are less sensitive to packet sampling than simple volume metrics. First, we analyze several plots of measured traffic as it can be monitored on the border gateway routers of the SWITCH [8] network. The results lead us to a try of concatenating flows, which where split unintentionally by the routers.
In the second part we expand the set of flow entropy metrics that were used in [1]. A comparison of all tracked routers acknowledges the results of [1] but also shows that simple metrics like flow or IP counts cannot be neglected as they often expose an anomaly better than entropy based metrics although they are highly affected by sampling.
In all chapters the W32_Blaster worm will accompany us and provides a basis for the findings.

Zusammenfassung

Ein kürzlich erschienenes Paper [1] schlägt vor, Paket- und Flowmetriken für die Detektion von Ausnahmeerscheinungen im Internet-Backbone zu verwenden. Diese Metriken sind weniger empfindlich gegen Paketausdünnung, genannt sampling, wie einfachere volumenbasierte Messmethoden.
Die erste Hälfte des Berichts enthält verschiedene Analysen von Datenverkehrs-Aufzeichnungen, wie sie auf den Grenzroutern des SWITCH [8] Netzwerkes beobachtet werden können. Die Ergebnisse verleiten uns zu einem Versuch, ungewollte Teilungen von Flows wieder zusammenzufügen, welche von den Routern verursacht wurden.
Im zweiten Teil ergänzen wir die vorgeschlagene Auswahl (siehe [1]) von Flow-Metriken um ein paar neue. Der Vergleich aller untersuchten Router bestätigt die Ergebnisse von [1], zeigt aber auch, dass einfache Messmethoden wie das einfache Zählen der vorkommenden IPs oder die gezählten Flüsse nicht vernachlässigt werden können. Sie entlarven eine Anomalie oft besser als entropiebasierte Ansätze, trotz starker, durch sampling verursachter Verfälschung.
In allen Kapiteln wird uns der W32_Blaster-Wurm begleiten und dient uns als Basis für die Erkenntnisse.

Chapter 1

Introduction

1.1 Motivation

The research topic of anomaly detection in Internet backbone targets to recognize anomalies, such as DDoS attacks, Port Scans and Worm spreading as it is already done today in smaller scale by using intrusion detection systems (IDS). The major challenge compared to IDS is the huge amount of traffic in backbone networks that needs to be reduced. Every kind of data compression and reduction is connected with a loss or even falsification of data. Packet sampling, to mention the widest spread method, is in general highly nonlinear. Measurements on the sampled dataset show results that need not to be correlated with the measurements made on the original dataset. [4] introduces an entropy-based approach to detect anomalies, allowing to recognize events even when the characteristics is not known in advance.
In October 2006 the Communication Systems Group [9] at the Swiss Federal Instiute of Technology [10] published a paper [1] about the impact of packet sampling on anomaly detection metrics. The idea of entropy summarization was carried on and analyzed with regard to packet sampling. The results provide hope that the proposed techniques withstand sampling within an acceptable accuracy.

1.2 Problem Statement

This thesis extends the concepts proposed there. We introduce flow-IDs to tag all flows. This allows us to track a flow in all processing steps, even when packet traces are generated again. All metrics associated with flows can therefore be exactly evaluated without the need for heuristics. We compare the results of all four hardware routers of SWITCH [8], a national ISP, using an expanded set of packet and flow metrics.

1.3 Outline

Although chapter 2 recapitulates the basic ideas and proceedings, we strongly recommend reading the mentioned document [1], since the same but enhanced framework is reused. The changes in the proceeding are explained in chapter 3, called methodology. We move on to the practical implementation (chapter 4) where we find a brief summary to each added tool. The results can be found in two separate chapters, traffic analysis and Blaster Worm results (chapter 5 and 6).

Chapter 2

Theoretical Background

2.1 Sampling and Anomaly Detection Metrics

Packet capturing on backbone network links is not feasible, due to the massive amount of data that needs to be processed and stored. A more appropriate method is collecting flows, but it comes along with several weaknesses. For example, the aggregated traffic records permit no conclusion about the packet sizes, the header information is incomplete and the export from the sensor cannot be implemented near real-time. Another challenge treated later on is the premature export on idle connections.
Nevertheless, flow capturing such as Cisco's implementation named NetFlow v5 is a widespread technique to monitor large networks. To further reduce the enormous amount of acquired data, packet sampling is usually applied in advance. A router uses a random generator to pick up every packet with a certain uniformly distributed probability. The remaining packets are logged in the corresponding netflow records.
Herein lies the main topic of research: How can we detect anomalies even when the traffic is sampled? It is known that volume based packet metrics, such as the number of bytes or the number of packets decrease proportional to the sampling rate [5]. Thus, anomaly detection using this type of metrics is hardly disturbed by sampling [1], but they usually do not expose an anomaly well. On the other hand, flow volume metrics are strongly biased. The number of flows sampled is not proportional to the packet sampling rate. Large flows consisting in many packets are less likely missed than shorter flows, as proven in [3]. The difficulty is to find metrics that expose an anomaly well but are also hardened against sampling. This means, an anomaly should clearly be visible even when packet sampling was applied before measuring the anomaly. In order to be able to detect anomalies in unsampled traffic, such as portscans, DDoS attacks and worms, [4] suggests using entropy-based summarisation of packet and flow metrics. It assesses an anomaly well, since changes in the entropy content indicate a massive network event. Fortunately entropy metrics are only disturbed in an acceptable manner by sampling [1]. Equation 2.1 shows the definition for packet entropy and equation 2.2 for flow entropy as we used them. They are derived from Shannon's information entropy formula.

$$H_{packet} = -\sum_i \log_2\left(\frac{packets_i}{total\ number\ of\ packets}\right) \cdot \frac{packets_i}{total\ number\ of\ packets} \tag{2.1}$$

$$H_{flow} = -\sum_i \log_2\left(\frac{flows_i}{total\ number\ of\ flows}\right) \cdot \frac{flows_i}{total\ number\ of\ flows} \tag{2.2}$$

Chapter 3

Methodology

In order to measure an anomaly we needed to reconstruct the unsampled packet traces. Logging traffic in flow records is a lossy process since not all information of the original packet flow is saved. So, we use for the packet size an average value, and the start time of a reconstructed packet is randomly chosen within the start and end time boundaries given in the flow record. Preserving in average the original throughput leads to a good approximation for large flows, while shorter flows are adequately approximated [6]. Measurement deviations from the original packet traces are only expected at the aggregation interval borders, where a flow will be divided up in two parts [1]. We focus now on the conceptual procedure steps. For a deeper technical insight see chapter 4.

1. Three of the four SWITCH routers are logged together in one dataset. We need to separate the wheat from the chaff and assign every flow record to the corresponding router. As a result we obtain a separate dataset for each device.

2. We introduced a new step for the purpose of concatenating unintentionally splitted flows. To simplify the mechanisms, sorting the flows in advance according to the start times is mandatory yet. Every flow record is tagged with a unique flow-ID, which allows to track a flow in all ongoing steps. If no chaining of splitted flows is desired, this step can be omitted.

3. To rebuild the packet traces we divide the number of bytes by the number of packets per flow and round it down to the next byte. Some packets get an extra byte to preserve the averaged throughput. The first packet of each flow is placed at the flow start time. All others obtain a randomly selected time within the flow duration interval. We abuse the Cisco NetFlow container and store packets instead of flows. Every packet gets its own Netflow header and record. That way, the same netflow framework can be used again for further steps, although the disk memory consumption greatly increases.

4. In a last step we measure several metrics over interval lengths of 15 minutes. We use a heuristic to recognize the traffic derived from an anomaly. This allows us to measure a so-called baseline without the anomaly and of course the total traffic. Although not practicable in real life, this approach has two advantages: We get the best case any other detection method would achieve. And it is independent of any blurring a detection method would cause.

   The introduced usage of flow-IDs allows us to count flows only once, that is in the first interval they appear. In addition we do not need any heuristic to distinguish between different flows and obtain exact results when calculating flow metrics. A table of measured values, including the new added ones, can be found in appendix D.

   To measure the impact of sampling we resample the packet stream at the rates of {1, 10, 100, 250, 1000} using a random generator. Packet sampling is a stochastic process and inevitably leads to a uncertainty in the obtained results. To estimate the induced error, we measure every result 10 times and calculate the mean and standard deviation of every measured value.

Chapter 4

Implementation

Beside many smaller improvements in the code of the DDoSVax NetFlow Toolset we introduce three new applications. The results of chapter 5 are based on the tool `fpdD` and `fid`. The latter concatenates unintentionally splitted flows. To analyze the Blaster Worm (chapter 6) we completely redesigned the existing code. The result is called `bl_sw`.

4.1 C-Code

4.1.1 fpdD

In chapter 5, we presented several cumulative distribution function plots. The **f**low-length, **p**acket-size and flow-**d**uration **D**istribution-tool collects the distribution of those three metrics in simple arrays. The output can directly be further processed by using the Perl script `graph_cdf02.pl` to obtain a cumulative distribution plot. UDP and TCP will be collected in separate files. This application replaces the former `flowpktsizedist` and `flduration`.

4.1.2 fid

Two objectives are handled with the **f**low-**ID** tool. It allows you to find (and concatenate) similar flows if a time segment between is smaller than a chosen period. This is done by collecting the 5-tuple and a timestamp of a flow in a hashtable. The second functionality is to mark every flow with a unique flow-ID. We overwrite the nexthop field in the netflow record header since we do not need the nexthop information later on. The tool in its current state has the disadvantage, that it needs sorted flows according to their start time (using the `flowtrace_decollator`). Please notice, *tagging the records with unique flow-IDs tremendously complicates further parallelization*. As long as no flow concatenation is needed, the `ptrace_builder` is also able to set flow-IDs. Then you can save two additional computing steps.

4.1.3 step_phen

The step_phen tool is a small and easy understandable application which we used to study the step phenomenon in section 5.2.

4.1.4 bl_sw

The **B**ase **L**ine and **S**cale **W**orm tool consumed the most time of this thesis and can therefore be considered as one of the main outputs of this work. It is based on the code of `scale_Blaster` and `baselines` but was finally rebuilt from scratch. Analyzing the packet trace builds from `ptrace_builder` is a nontrivial task and reached the hardware limits of our cluster Scylla [11]. In order to get the RAM memory usage under control the tool needed to be radically optimized several times over again.
Two preprocessor directives can be set. Compiling with the flag `BLSW_DEBUG` causes the program to become more verbose on `stdout`. This is especially helpful when you run out of

RAM memory, because a statistic of approximated memory usage is printed for every interval. `ARNO_RANDOMIZER` gives the possibility to choose between two random generators, the one belonging to the standard library and the one from NetFlow toolset.

The usage is simple. In addition to packet trace builds, just specify a interval length over which the traffic will be aggregated, one or more sampling rates and a confidence factor. The latter will produce a defined number of statistically independent log files, which allows you to estimate the uncertainty caused by sampling. It is possible to declare a maximal flow duration. Choosing a value smaller than the real maximal flow duration will effect all flow statistics to become slightly inaccurate. But this will greatly save memory and speed up the process. 60 seconds (instead of the normally taken 15 min) are already enough for a really good approximation. As output you will get a human readable text file with 16 different metric statistics, which is optimized for further usage with `gnuplot`. For more usage details, run the tool with the `-h` flag.

Internal mode of operation: The code should be self-explanatory once you understand the main concept. In addition, every function starts with a descriptive comment.
For each sampling rate and every confidence factor an incoming packet is logged in one or more separate datasets defined by the `struct hts`. Such a struct contains three hashtables and six counters. The random generator decides if our packet is an elected one. If it is not dropped, the arrival time and flow-ID is logged in the hashtable named `fid` (see function `update_hts()`). This is necessary to be able to distinguish in which interval a flow has started so that we do not double-count old flows in a new interval. Afterwards we check if our packet matches the criteria of the monitored worm. If it does, we update the hashtable `worm`, otherwise the hashtable `base`. These two tables use the 4-tuple as key and a bucket includes a counter for the number of packets and one for the number of flows per 4-tuple.
After collecting data for the time of one interval length, the hashtables `base` and `worm` are evaluated in the function `collect_bltl()`. We collect for every metric a discrete distribution function in a smaller hashtable, which will be passed to the entropy calculation function. After writing out the result collected out of the baseline table, we reuse the small table and append the information from the worm table to obtain the result for the whole traffic. After printing all results `print_stats()`, the hashtables `base` and `worm` will be deleted and we perform a cleanup for the `fid` hashtable (see function `realloc_hts()`).
The proposed method is optimized for low memory consumption. Another Pareto optimality would be a faster execution time, but the more limiting factor is the available memory.

4.1.5 netflow_router_split_19991, netflow_PR_sorter, flow-trace_decollator, ptrace_builder

The basic concepts of these four tools remain unchanged. But tons of small improvements and speedups where added to the code. Especially the handling and naming of input and output files has become much more convenient. Like all other tools, there should now always be a suitable help available by using the flag `-h`.

4.1.6 netflow_iterator_template3_new.c

Since the previous version of the template produced some memory leaks and exposed other flaws, the code is now in a better consistency.

4.2 Perl Scripts

4.2.1 bl_sw02.pl, fid01.pl flduration.pl, flowpktsizedist.pl, flow-trace_decollator.pl, fpdD.pl, ptrace_builder01.pl, routersplit19991-01.pl

For every application presented in the previous section, there exists a control script written in Perl. These scripts allow you to run and distribute several instances of an application from one central place. In order to parallelize the computing process, these scripts allow calculation overlapping at the interval beginning and end. This is necessary if the result of an interval is dependent on its past.

4.2.2 graph_cdf02.pl, graph_bltl01.pl, graph_reldiff01.pl, graph_sampcmp01.pl

In order to visualize the output of the c-tools, the class of graph scripts has been created. The most important features can directly be specified by using the command line options. This allows you to easily fabricate many pictures without loosing a lot of time and effort. All scripts follow the same strategy. After reading and arranging the input, a gnuplot command script is generated (ending with `gnp`) and if necessary the data points are written into a further file. `graph_cdf02.pl` parses the output of `fpdD`, all other scripts handle the output files of `bl_sw`.

4.3 Shell Scripts

`do_on_all_nodes.sh`, `freespace01.sh`, `kill_on_allnodes.sh`, `load01.sh` and `see_running_procs_on_nodes.sh` are shell scripts that help to manage and supervise the cluster. Application name and usage are rather self-explanatory.

4.4 Tool Chains

We used the following two tool chains for our findings.

- `netflow_router_split_19991` → `fpdD` → `graph_cdf01.pl`, `graph_cdf02.pl`, `step_phen`.

  This chain is needed to plot the (cumulative) distribution functions.

- `netflow_router_split_19991` → `netflow_PR_sorter` → `fid` → `ptrace_builder` (calls `netflow_PR_sorter`) → `bl_sw` → `graph_bltl01.pl`, `graph_sampcmp01.pl`, `graph_reldiff01.pl`.

  Step two and three can be omitted as mentioned in chapter 5. No flows will then be concatenated.

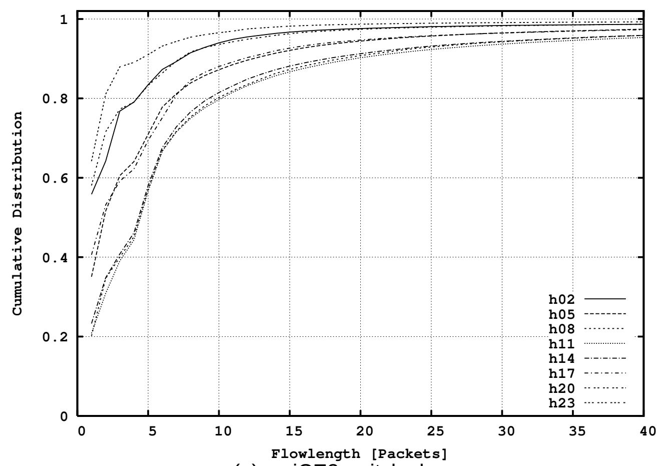
Chapter 5

Traffic Analysis

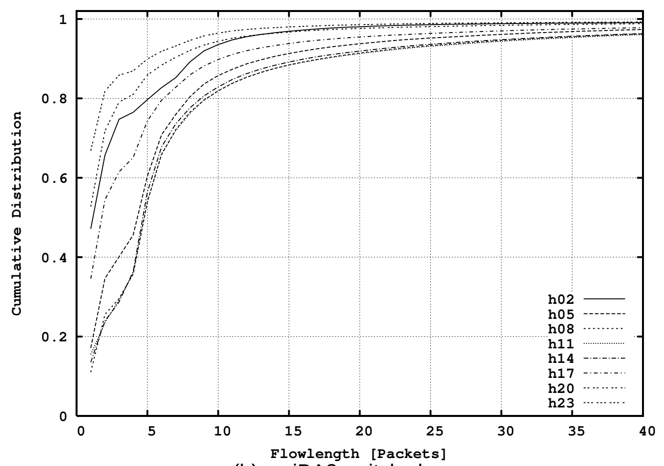
A preliminary task of this thesis was to do some basic traffic analysis first, to become familiar with the available NetFlow toolset. All discussions are based on a dataset recorded between August 1, and August 24, 2003. We investigate in this chapter the influence of a special router timeout, which is triggered when a connection is idle for more than 4 seconds if not more than two packets are sent.

5.1 Cumulative Distribution of Flow Length

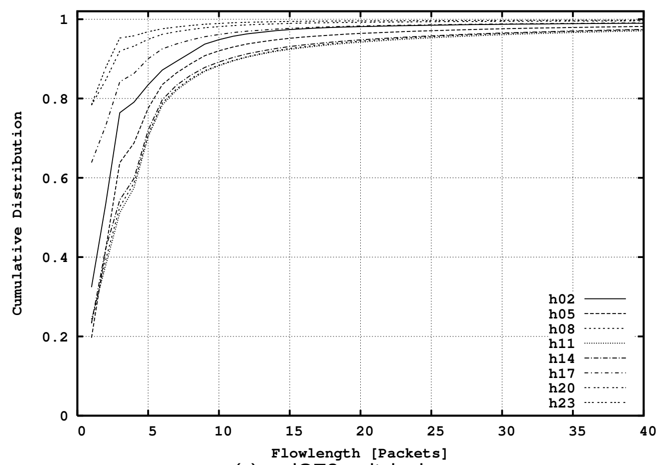
In this section we are interested in this question: how many packets are sent within one flow. To avoid ambiguities in advance, in our context, flow is defined as *unidirectional* connection uniquely given by a 5-tuple (source IP, destination IP, source port, destination port, transport layer protocol). Figure 5.1 shows the cumulative flow length distribution of all four border gateway routers belonging to the SWITCH [8] network. Or to state it otherwise: it shows how many percent of TCP flows consist in fewer packets than the value mentioned on the abscissa. Each of the eight curves represents another hour on August 8, 2003.
Let us emphasize two crucial points. First, the dominant amount of TCP connections is based on very short flows. In the presented examples more than 80% of all TCP connections are shorter than 10 packages. Secondly, the lines of 5 p.m., 8 p.m. and 23 p.m. denote an increased number of short flows. In addition the standard deviation to a daily averaged graph is larger compared to the plots of one week before (August 4th, figure not shown). The explanation is easy if we know that the w32_Blaster outbreak was on August 8, 2003 around 4 o'clock. Worm infections are usually accompanied by port scans and connection attempts which result in an increased amount of short flows.
For the highest loaded router, `swiIX1.switch.ch.`, there is an auxiliary plot showing the cumulative distribution of UDP flows (Fig. 5.2). Again compared to the plots of the week before, the parameter UDP flow length seems not to follow strong hourly variations within a day. Please notice that the ordinate starts at a probability of 0.7. Somehow surprising is the fact that 95% of all flows are shorter than 5 packets. This is one of the reasons why we determine in chapter 5.3 if it is possible to concatenate shorter flows to longer ones as well.



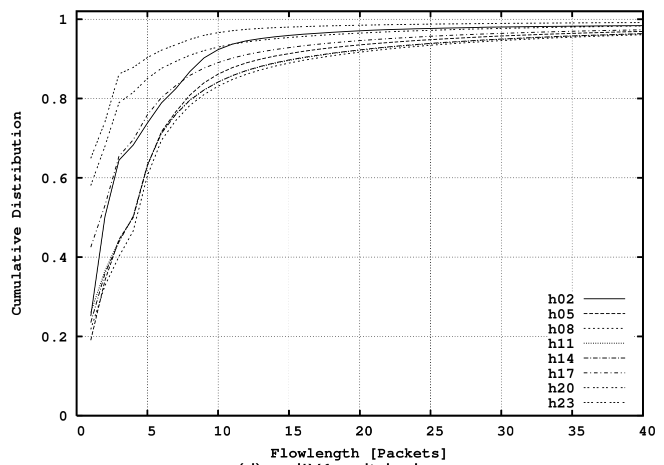
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.



(c) swiCE3.switch.ch.



(d) swiIX1.switch.ch.

Figure 5.1: Number of TCP Segments per Flow, 11.08.2003

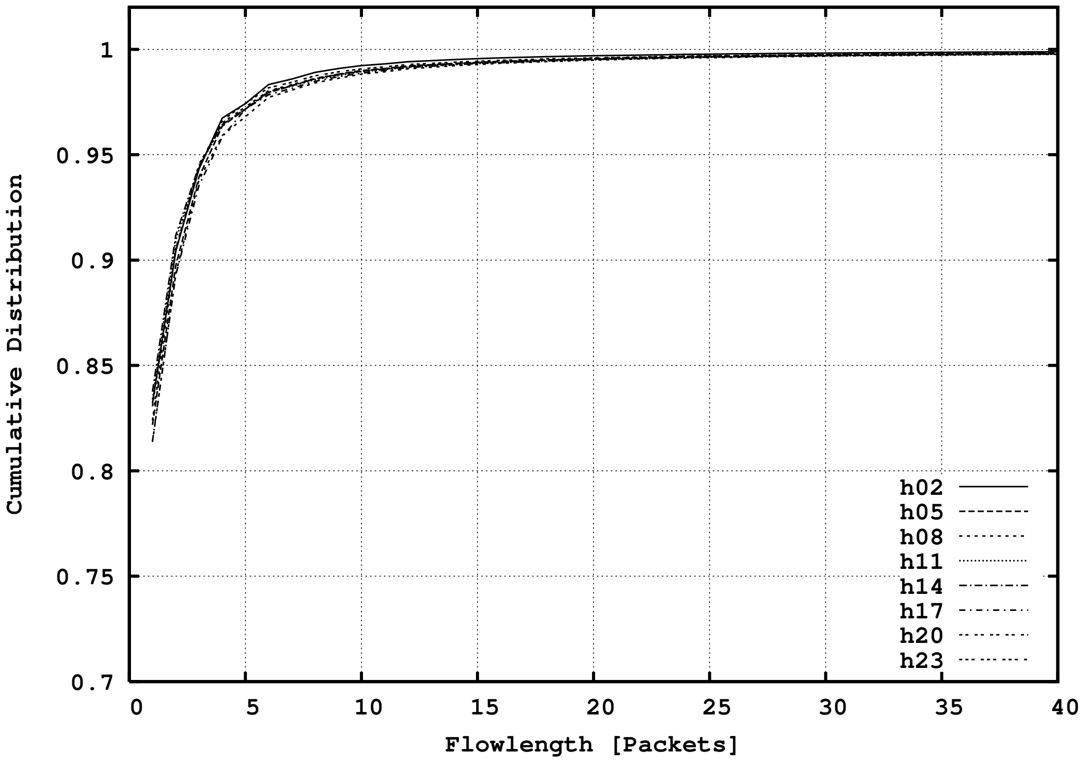


Figure 5.2: Number of UDP Datagrams per Flow, 11.08.2003, swiIX1.switch.ch.

5.2 Cumulative Distribution of Flow Duration

A similar observation can be obtained by investigating the durations of flows. Around 90 percent of all flows last no longer than 10 seconds. The x-scale in figure 5.3 stops at 20s, since the interesting part is below this limit. The upper bound for flow duration in the dataset is at 15 minutes, as it is configured on the border routers (see chapter 5.3). An interesting phenomenon are the on all routers observed steps at 2950 ms and 8950 ms. Up to 30 percent of the flows lie in between one of the two intervals of just 150 ms width. Let us focus on the 2 a.m. line in figure 5.3(c). The data collected in one hour between 2 a.m. and 3 a.m. provide the following facts:

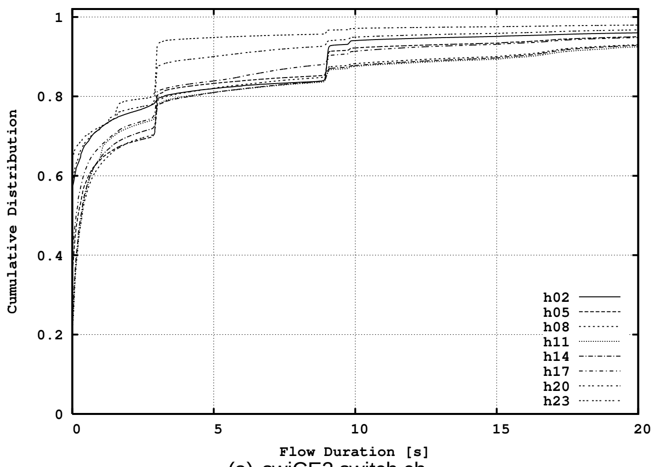
$$\frac{2packetflows\ between\ [2900\text{ms}..3000\text{ms}]}{all\ flows\ between\ [2900\text{ms}..3000\text{ms}]} = \frac{241840}{245428} = 0.9854 \tag{5.1}$$

$$\frac{2packetflows\ between\ [2900\text{ms}..3000\text{ms}]}{all\ 2packet\ flows} = \frac{241840}{527127} = 0.4588 \tag{5.2}$$

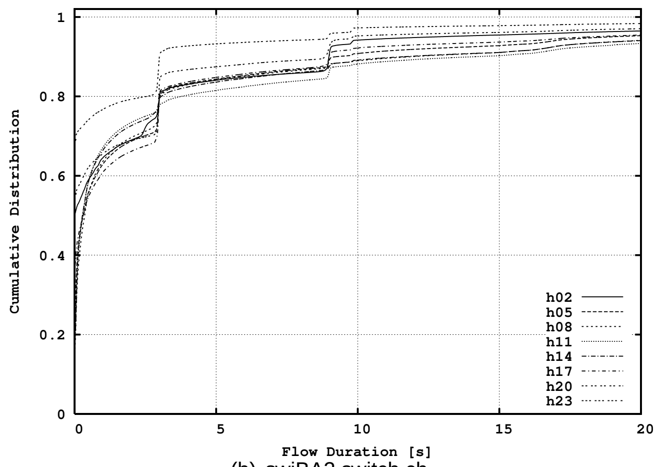
$$\frac{3packetflows\ between\ [8900\text{ms}..9000\text{ms}]}{all\ flows\ between\ [8900\text{ms}..9000\text{ms}]} = \frac{212040}{213495} = 0.9932 \tag{5.3}$$

$$\frac{3packetflows\ between\ [8900\text{ms}..9000\text{ms}]}{all\ 3packet\ flows} = \frac{212040}{580267} = 0.3654 \tag{5.4}$$

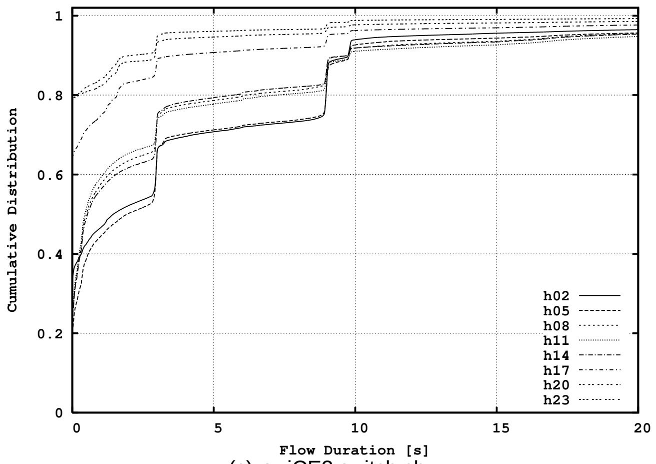
It is quite evident, the first step is mainly caused by 2-packet-flows and the second one by 3-packet-flows. We checked whether at least the 2-packet-step can be explained by reason of one special router setting. It triggers an export of flows after 2 packets, if there is a pause of more than 4 seconds afterwards (see also chapter 5.3). This rule does not seem to cause a significant change, since in the current circumstances only 20582 2-packet TCP flows would vanish. At best, when all vanished flows belonged to the interval [2900 ms .. 3000 ms] before, the first ratio (5.1) reduces to $\frac{241840-20582}{245428} = 0.9015$ and (5.2) to $\frac{241840-20582}{527127} = 0.4197$.



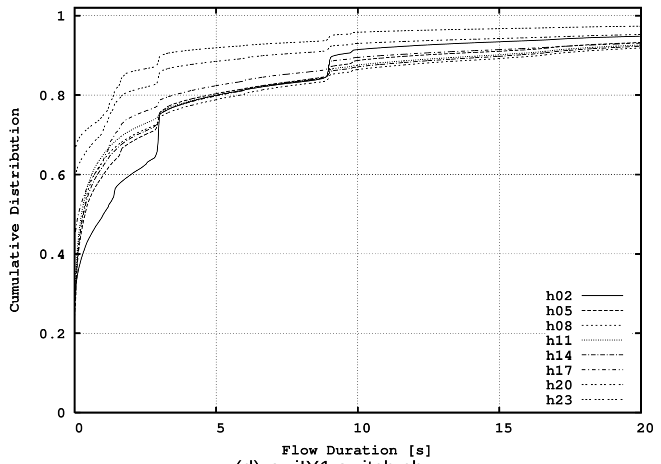
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.



(c) swiCE3.switch.ch.



(d) swiIX1.switch.ch.

Figure 5.3: Duration of a TCP Flow, 11.08.2003

Figure 5.4 shows the cumulative distribution function of the UDP flow durations. Notice again, the ordinate starts with a probability of 0.8. Again, we find two principal steps. One at 3000 ms and the second one is now at 6100 ms instead of 8900 ms as we have seen in the TCP chart (Fig. 5.3).

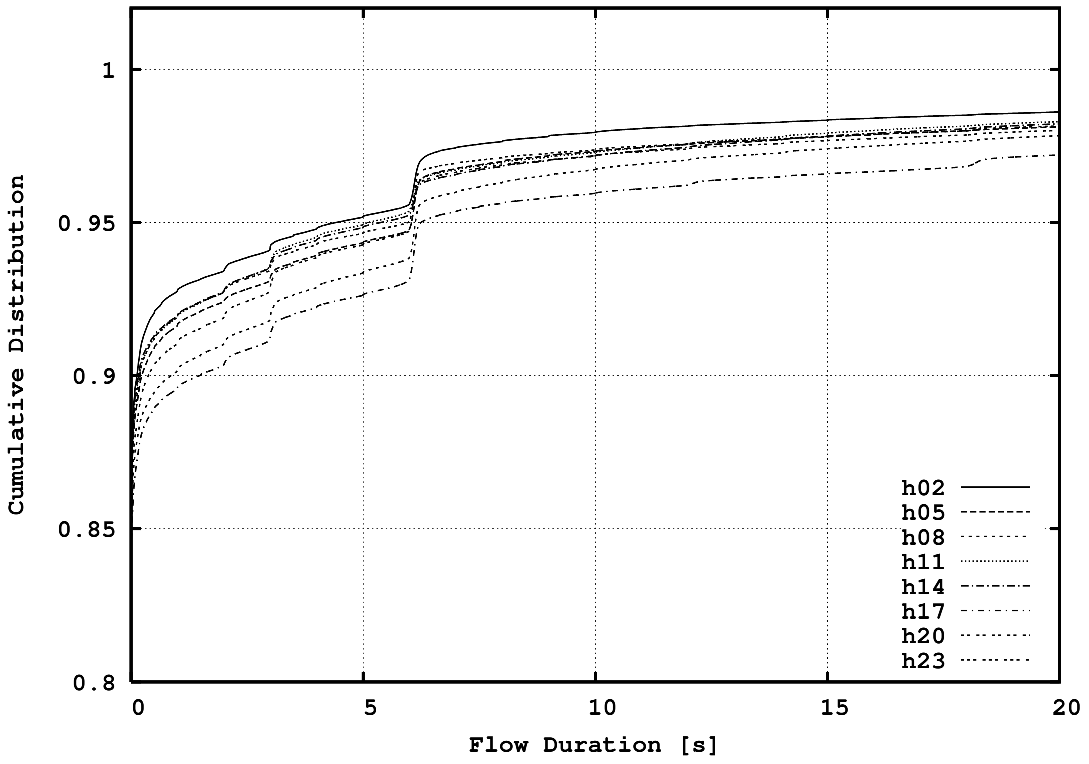


Figure 5.4: Duration of a UDP Flow, 11.08.2003, swiIX1.switch.ch.

5.3 Concatenation of Short Flows

The SWITCH border routers are configured to export a flow according to the following policies:

- The control flag FIN or RST in a segment is set.
- No packet has been received for 30 seconds.
- A flow lasts longer than 15 minutes. In this case the flow will be divided up in multiple records.
- The router runs out of memory.
- Just one or two packets are sent and followed by a pause longer than 4 second.

As we have seen in section 5.1 and 5.2 the dominant fraction of flows is very short. To find out if this is a consequence of the router configuration or usual traffic behavior, we tried to concatenate short flows.
Especially the last mentioned router policy raises suspicion to cause a lot of unintentional splitted flows. It was introduced to harden the routers against the flood of record information induced by DDoS attacks. The first analysis works as follows. Whenever a 1- or 2-packet flow ends, the next 15 seconds are scanned for the same 5-tuple. Two matching flows are concatenated, as well as further occurrences. Chart 5.1 shows exemplarily the outcome for the router `swiCE3.switch.ch`. The other routers yield the same order of magnitude.
The overall change is only a small percentage. So we decided not to reassemble flows in the ongoing work, since the gained accuracy is small compared to the additional required computing time.

date	records	conc fls[a]	% of all fls[b]	% of 2pkt fls[c]	conc tcp[d]	conc udp[e]
2003.08.01	49514949	1003779	2.027224%	5.828461%	447156	425944
2003.08.02	45676354	952267	2.084814%	5.538596%	434180	357977
2003.08.03	44720970	842696	1.884342%	5.034930%	402025	369386
2003.08.04	62502148	1205923	1.929410%	5.980281%	590727	538472
2003.08.05	58803007	1096151	1.864107%	6.062501%	542136	483792
2003.08.06	60056786	1230544	2.048967%	6.527514%	663662	502973
2003.08.07	61122201	1166353	1.908231%	6.044600%	595650	465165
2003.08.08	54101168	1109516	2.050817%	6.960556%	643957	411771
2003.08.09	40391894	811717	2.009604%	5.603616%	465119	288631
2003.08.10	44545465	849250	1.906479%	4.734250%	534901	260540
2003.08.11	65116091	1168925	1.795140%	4.844957%	693700	418395
2003.08.12	158354339	903423	0.570507%	0.761515%	458059	387565

[a] number of concatenated flows per day
[b] ratio of (conc fls / records) * 100
[c] ratio of (conc fls / 1- and 2flow records) * 100
[d] number of reassembled TCP flows
[e] number of reassembled UDP flows

Table 5.1: Appended Flows to 1 and 2 Packet Flows on swiCE3.switch.ch

The same method can also be applied to the router policy, which causes an export of a flow if it is idle for 30 seconds. Of course, we are not bound to the 2-packet threshold any more. We increased it to 1000, hence far more than 95% of all flows will be considered. For a maximum allowed gap of 31 seconds about 4.5% records would vanish. If we go further and enlarge the gap up to 5 minutes around 20% flows are concatenated. Although this amount is noticeable, we do not pay regard to it, because all router settings except the last mentioned one are widespread. As long as we keep in mind that a flow does not have to correspond exactly to a real TCP/UDP connection we are not obliged to recover the reality.

5.4 Packet Sizes and Damaged Flow Records

No packets greater than 1500 Bytes can be monitored, although theoretically 65536 Bytes would be allowed on IP layer. However, this is not surprising, since fragmentation is not desired and most Internet attached networks are based on Ethernet. And on the MAC layer of Ethernet the maximum payload threshold is at 1500 bytes. Figure 5.5 and 5.6 also show, that most packets sizes are close to the upper or lower observed limits.
We also discovered some flow records with impossible entries. Even when the flow records only offer a field for the accumulated bytes per flow, the arithmetic average of one layer 3 packet cannot be below 40 bytes for TCP segments. Tabular 5.2 lists the damaged packages collected within 24 hours on August, 11. They are not seen on figure 5.5, because compared to the intact flow records a large magnification would be needed to see them.
Some flow records, a magnitude up to 10 flow records per hour, pretend to have a duration around 30 minutes. This contradicts the router settings, since all flows should be exported after 15 minutes.

TCP/IP Octets	Counted Occurrences
20	109
24	2
26	2
27	725
28	2438
30	17
32	1
36	14
37	1
39	7
40	509429161
41	70205480
42	63628986

Table 5.2: Damaged TCP/IP Packets, 11.08.2003, swiIX1.switch.ch.

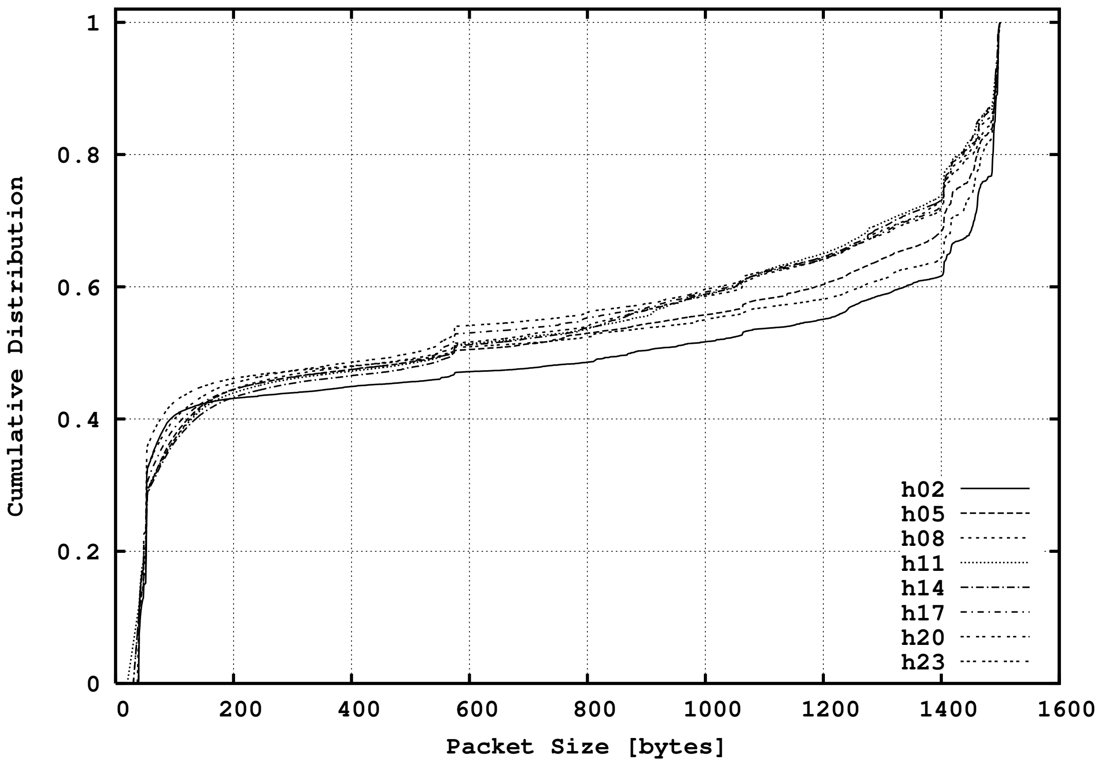


Figure 5.5: IP Packet Size, TCP, 11.08.2003, swiIX1.switch.ch.

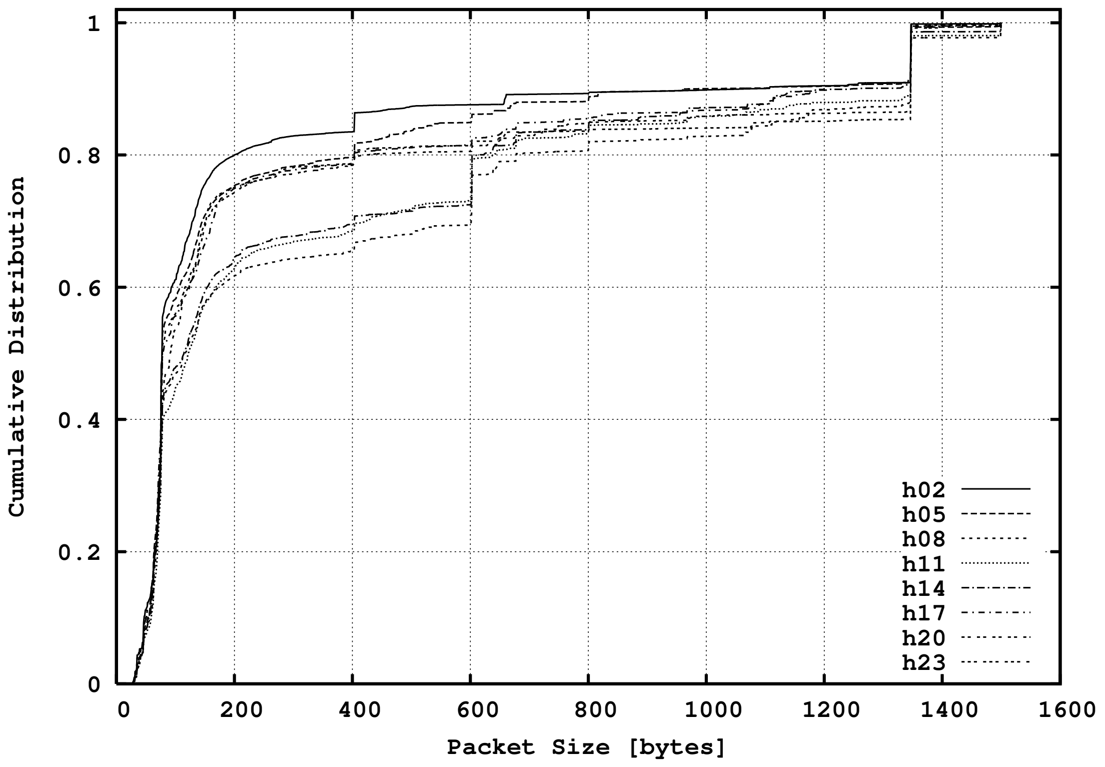


Figure 5.6: IP Packet Size, UDP, 11.08.2003, swiIX1.switch.ch.

Chapter 6

Blaster Worm Results

To verify the suggested measurement metrics we used a 245 GByte netflow_v5 dataset recorded between August 1, and August 24, 2003. On the SWITCH routers the start of the epidemic state of W32_Blaster was recorded at 16:00 UTC+1 on August 8, 2003 (e.g. see Fig. A.3(c)). Blaster, also known as Lovesan, uses a random scanning strategy to detect a vulnerable MS-Windows service running on port 135/TCP. The source port is randomly chosen, while the destination port of course is fixed. So to obtain the baseline we subtract all TCP packets with target port 135 and packet size of 40, 44, or 48 from the total considered traffic. We only regard all incoming TCP traffic as total traffic. That is to say, packets from outside targeted to hosts in the SWITCH network.

6.1 Visibility in Unsampled Data

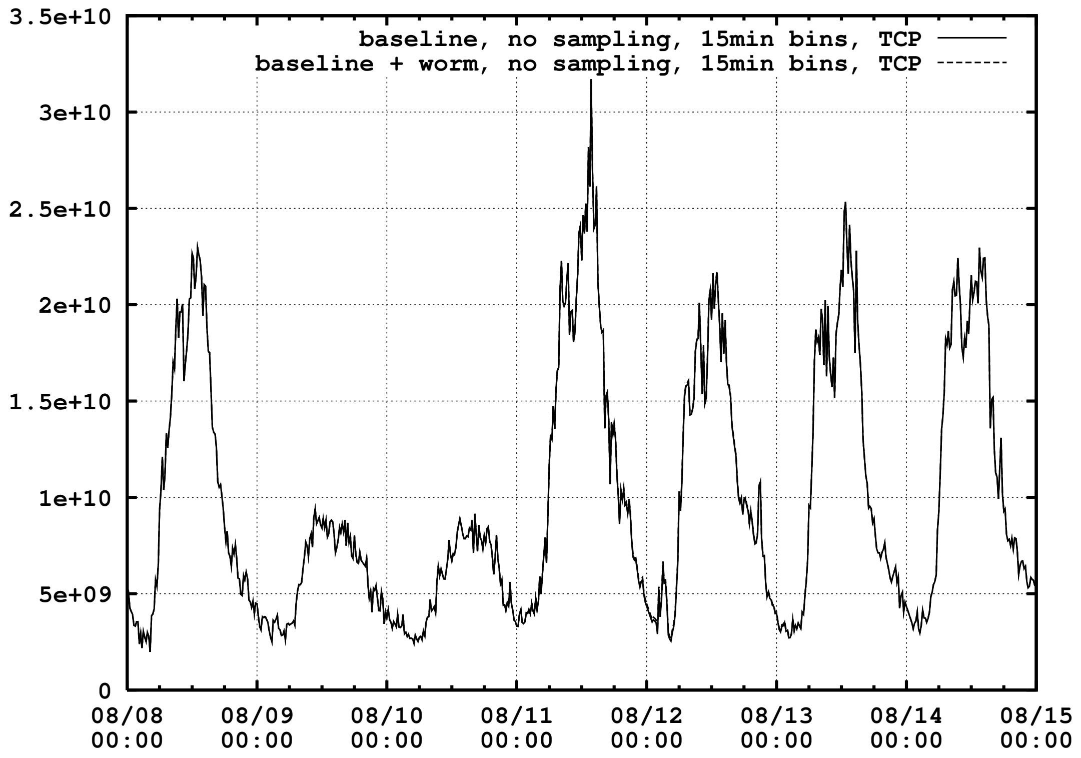


Figure 6.1: Baseline vs. Total Traffic, Counted Bytes, August 08-14, swiCE3.switch.ch.

Appendix A contains the plots we obtain, when we once apply our measurement methods to the total traffic and once to the baseline. That let us see how good the different metrics expose the worm in case of unsampled traffic. In general a small difference does not mean a measured

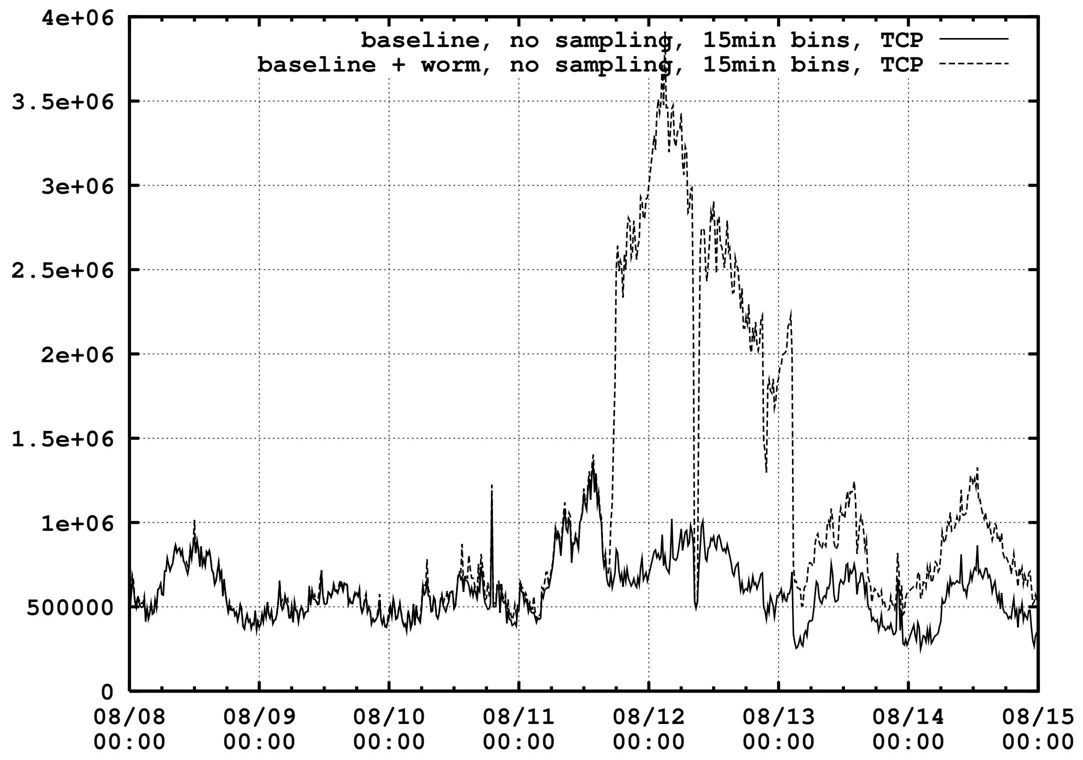


Figure 6.2: Baseline vs. Total Traffic, Counted Flow IDs, August 08-14, swiCE3.switch.ch.

value is useless. For example, looking at the counted bytes (Fig 6.1 or A.1) provides no cause for alarm whereas the counted flows (Fig. 6.2 or A.3) expose a massive network event. But if the anomaly would be a DDoS attack tuned to produce lots of large packets to take down a link, the counted bytes would indicate the event best. So a single metric cannot handle all types of anomaly. The design goal is to find a preferably small number of metrics which are independent of each other but cover a large set of anomalies.

The total traffic measured with entropy metrics can also become smaller than the baseline. For an example see figure 6.3. After the outbreak the totalline drops below the baseline. As we already know, the Blaster scanning algorithm is targeted to port 135. This means, the degree of order is increasing and the sum of equation 2.2 reduces and the totalline falls below the baseline.

Another thing worth mentioning is noise. Small routers (swiBA2, swiCE2, swiCE3) tend towards noisier signals. swiIX1 transmits more traffic and has therefore a higher probability to average out distortions (e.g. see Fig. A.9). But ignoring the smaller routers is also not a good idea. They are affected by Blaster in different ways, and it is one of the smaller devices, swiCE3, that shows the anomaly best (see Fig. 6.2 or A.3). SwiCE3 is also the first router that indicates an alarming trend.

As figures 6.10 to 6.13 uncover, flow entropy always exposes an anomaly better than the packet entropy. That leads us to another conflict of design targets. The flow entropy exposes a anomaly better than packet entropy, but the noise ratio is also higher. E.g. compare the flow and packet entropy of destination IPs, Fig. 6.4 and Fig. 6.5. This raises the legitimated question, which is more important: a good signal to noise ratio or a better relative difference between baseline and totalline. An idea would be to introduce a value measuring the quality of a metric. The product $SNR * Relative\ Difference$ should therefore be maximized. If we carry on the idea, we could extend the formula by a "sampling stability" factor or we make the SNR dependent on the sampling factor.

We have seen that noisy signals make it difficult to extract an anomaly. How do we distinguish between local peaks caused by noise and peaks from an anomaly? One option would be low-pass filtering. But shorter anomalies are also peaks in the data and would be flattened out. In addition, it is known from signal processing, good lowpass filters imply a time delay in order to

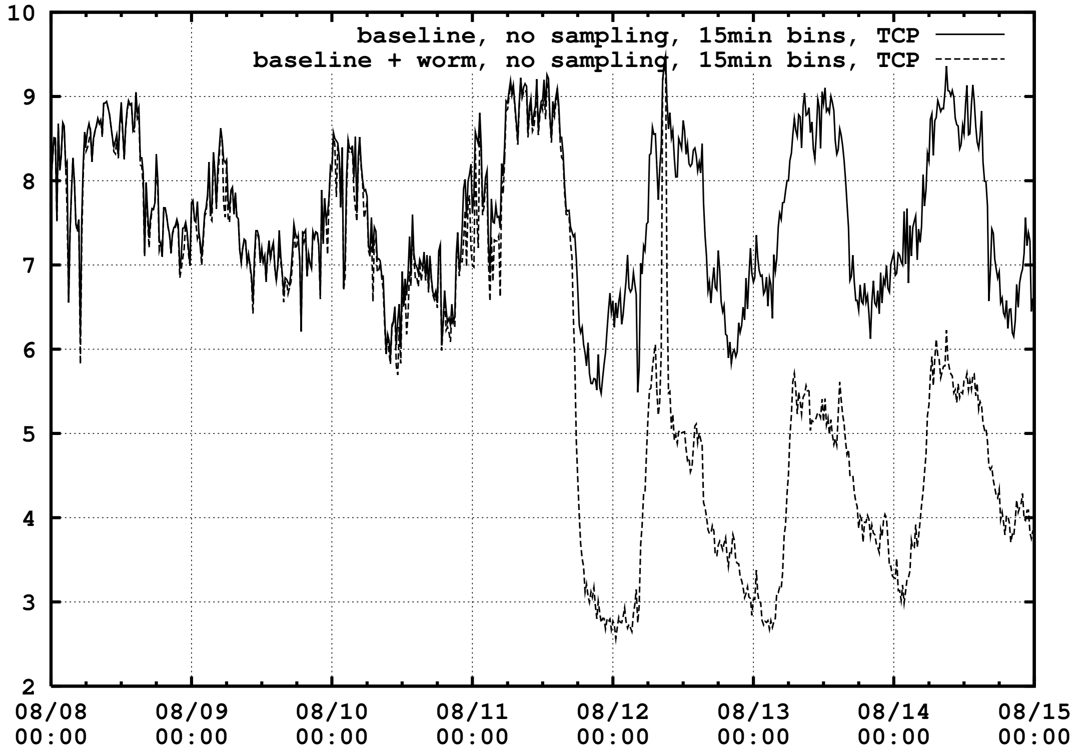


Figure 6.3: Baseline vs. Total Traffic, Flow Entropy of Destination Ports, August 08-14, swiIX1.switch.ch.

stay causal and therefore realizable. A time delay is undesirable when we try to measure near real-time. An option to filter just the noise could be wavelets.

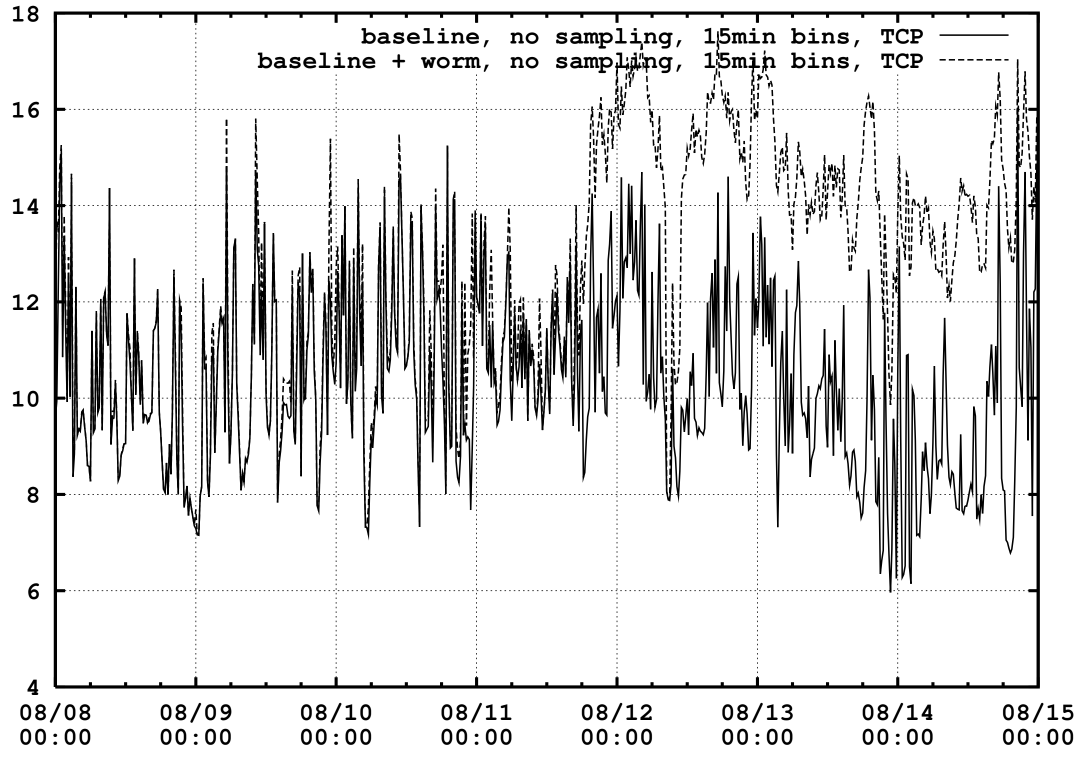


Figure 6.4: Baseline vs. Total Traffic, Flow Entropy of Destination IPs, August 08-14, swiCE2.switch.ch.

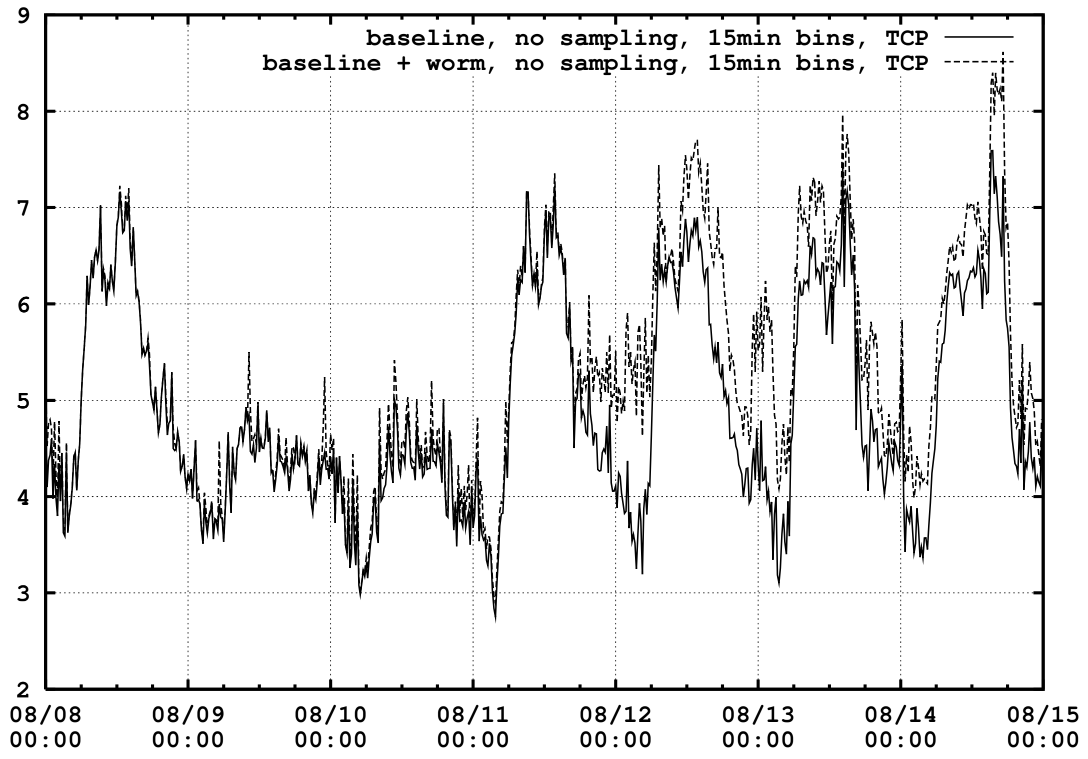


Figure 6.5: Baseline vs. Total Traffic, Packet Entropy of Destination IPs, August 08-14, swiCE2.switch.ch.

6.2 Sampled Data versus Unsampled Data

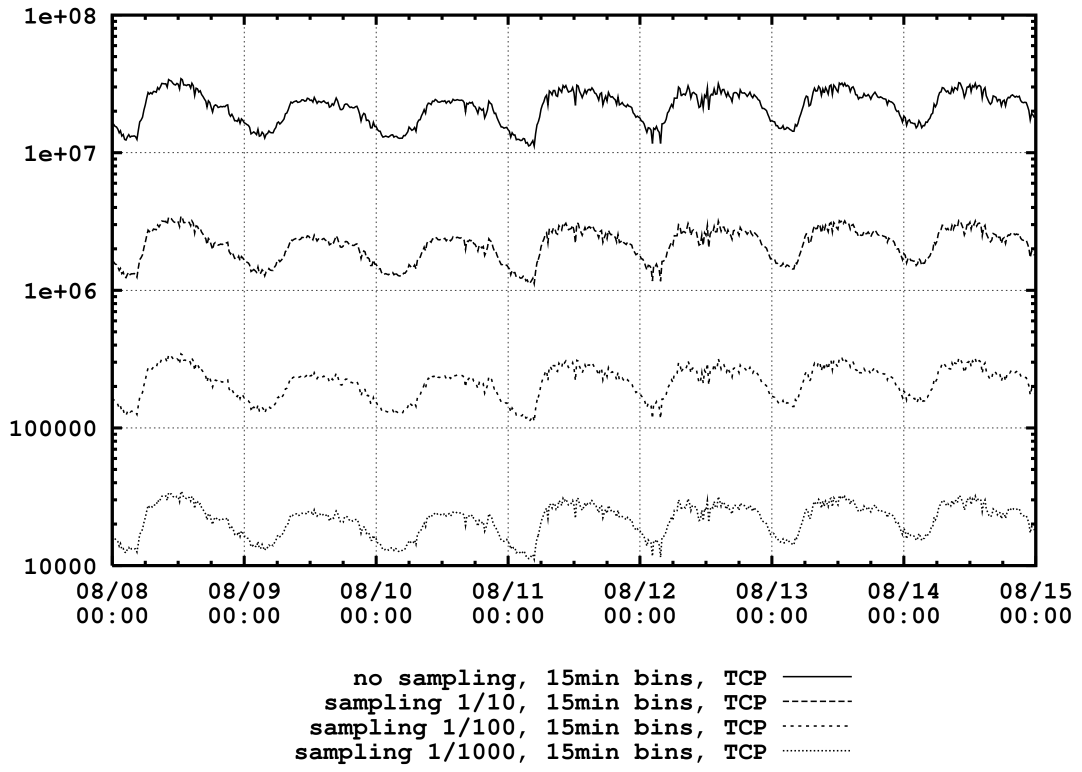


Figure 6.6: Impact of Sampling, Counted Packets, August 08-14, swiIX1.switch.ch.

The plots in appendix B show the impact of sampling for all metrics. They primarily do not show the influence of Blaster. We consider here mainly the falsification induced by sampling. Figure B.1 up to Figure B.7 use a logarithmic scale. So in the ideal case, when a metric is not affected by sampling, we have the same linear interspaces between the four curves. As already mentioned, the counted packets (Fig. 6.6 or B.2) and counted bytes (Fig.6.7 or B.1) are proven to be stable against sampling, and therefore are a good example. An example for a lager influenced value is the chart of counted source IPs (Fig. 6.8 or B.4).
On August 12, the supervisors of SWITCH tested countermeasures to block Blaster, which can be impressively seen in figure B.5. Between 7 a.m. and 11 a.m. there is a narrow valley. The interesting thing is, how the different metrics react to the intervention and if the expected behavior does acknowledge the theory. All flow entropies show off a strong reaction to the change (Fig. B.8 to Fig. B.11), whereas the event disappears in noise for most packet entropies (Fig. B.12 to Fig. B.15).

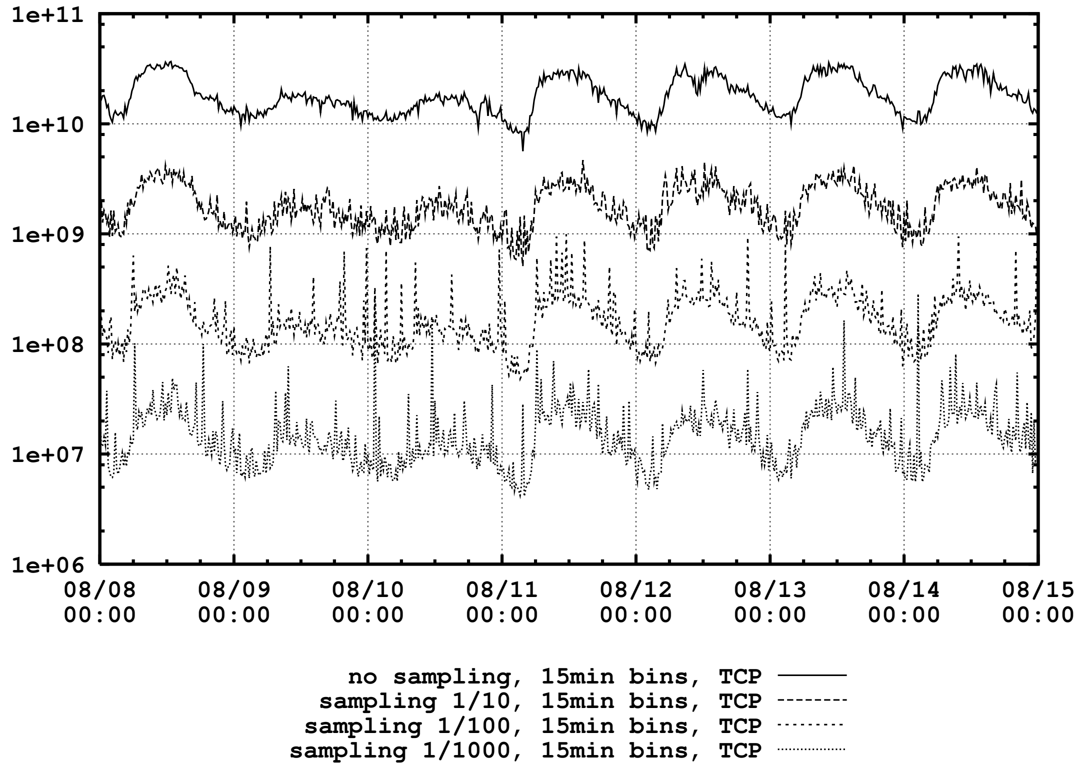


Figure 6.7: Impact of Sampling, Counted Bytes, August 08-14, swiIX1.switch.ch.

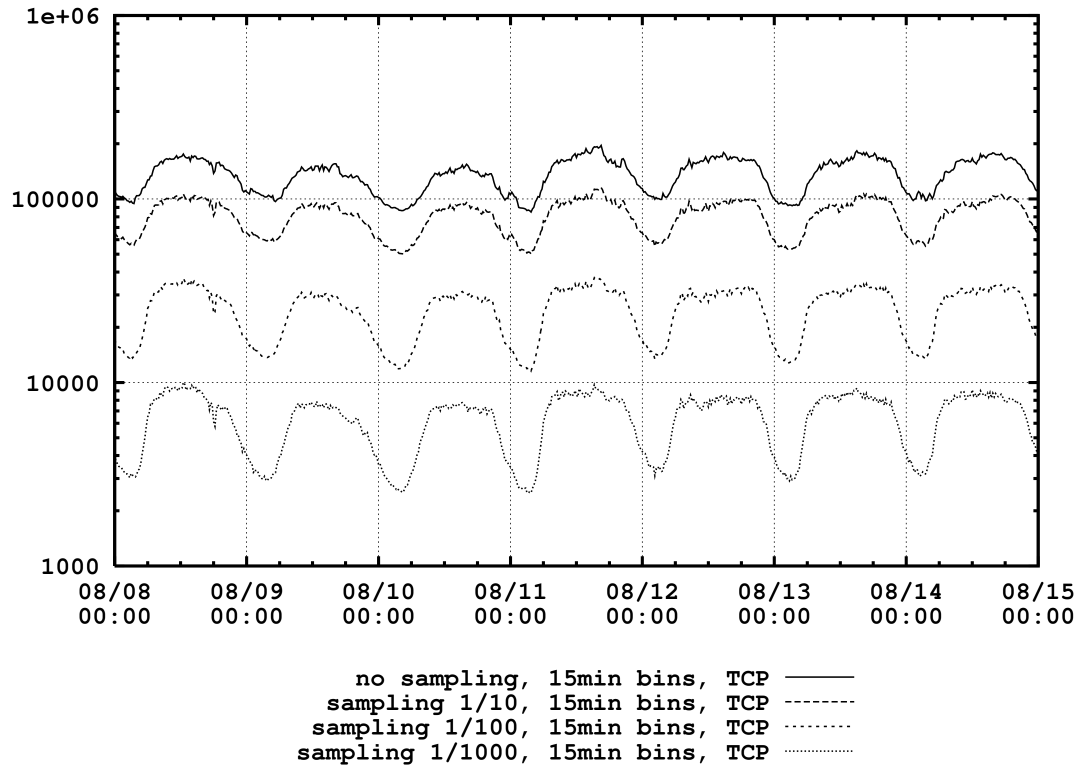


Figure 6.8: Impact of Sampling, Counted Source IPs, August 08-14, swiIX1.switch.ch.

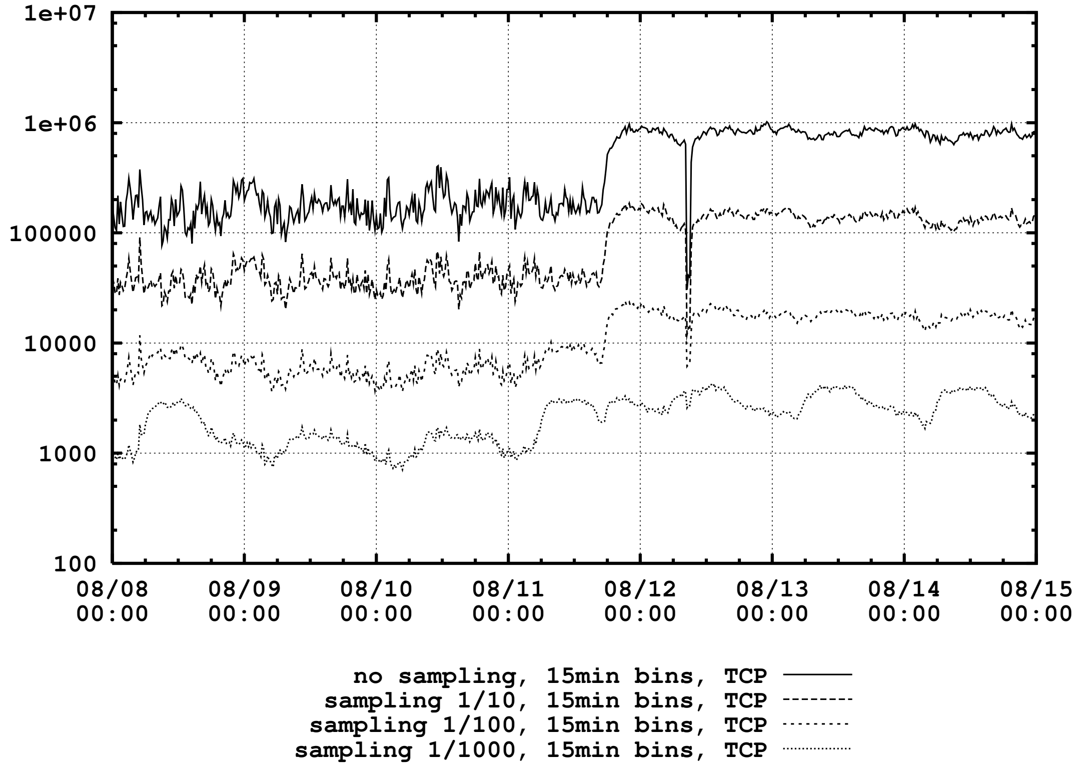


Figure 6.9: Impact of Sampling, Counted Destination IPs, August 08-14, swiIX1.switch.ch.

6.3 Outcome Comparison of all Analyzed Routers

A measure for the visibility of an anomaly is the relative difference between the baseline x_{bl} and totalline x_{tl} (Figs. 6.10 - 6.13 or Fig. C.1). We define the relative difference as $(x_{tl} - x_{bl})/x_{bl}$. The fraction becomes negative if the totalline metric drops below the baseline. We have seen such a case in section 6.1 (flow entropy of destination port). Please note that we consider now just one 15 minutes interval straight after the outbreak. For each sampling rate the relative difference is specified once. Flat curves denote a good sampling stability, higher values indicate a better visibility of the anomaly.
Sampling can also have a positive impact on certain metrics and emphasize the relative difference. The figure 6.12 for instance shows an initial ascent for the counted destination IPs (cnt_dip). DDoS attacks and port scans show either few hosts scanning a large range of hosts or many hosts attack few ones. The probability to sample an IP/port of an attacked or scanning host is much higher, since more connections end there. Thus, when we start to sample a slight increase results. The finding is consistent with [3]. For other intervals (not shown) the emphasis effect can also be observed for fe_sp, fe_dp, fe_sip, fe_dip, cnt_sp and cnt_dip.
For the counted source ports (cnt_sp) the explanation is similar but an additional phenomenon occurs. We know, that the infected hosts scan others using a randomly chosen source port. Since only $2^{16} - 2$ ports exist, Blaster is not visible as long as the port usage exceeds a saturation threshold. We observe a relative difference not until we thin out enough traffic. SwiIX1, our busy router, never undershoots the saturation threshold. Therefore an emphasis does not occur.
Considering figure 6.12 particularly states one thing. We cannot neglect the simple metrics, because two of them, cnt_dip and num_fid , expose the Blaster best. Our plots approve the results of [1], that flow and packet entropy withstand sampling better than any other metric. Nevertheless, the counted destination IPs and the number of counted flows point out a larger relative difference, even in the sampled case.
Let us raise one last question. What can we gain when we consider more than just one single router? We have exhibited that the intensity and starting time of an anomaly is different for the routers. But it would be desirable if we could combine the measured values of all routers to one single anomaly indicator. We have no method so far to close the topic, since it is a nontrivial task. For example, a simple weighted sum won't do it. The danger of averaging out the anomaly is tremendous. A possible approach could be to avoid false positives by using more than one router. When more than one device triggers the alarm bell the chance of an massive event is certainly higher. We must not rely on the combined alert exclusively, since an attack could be visible on a single router only.

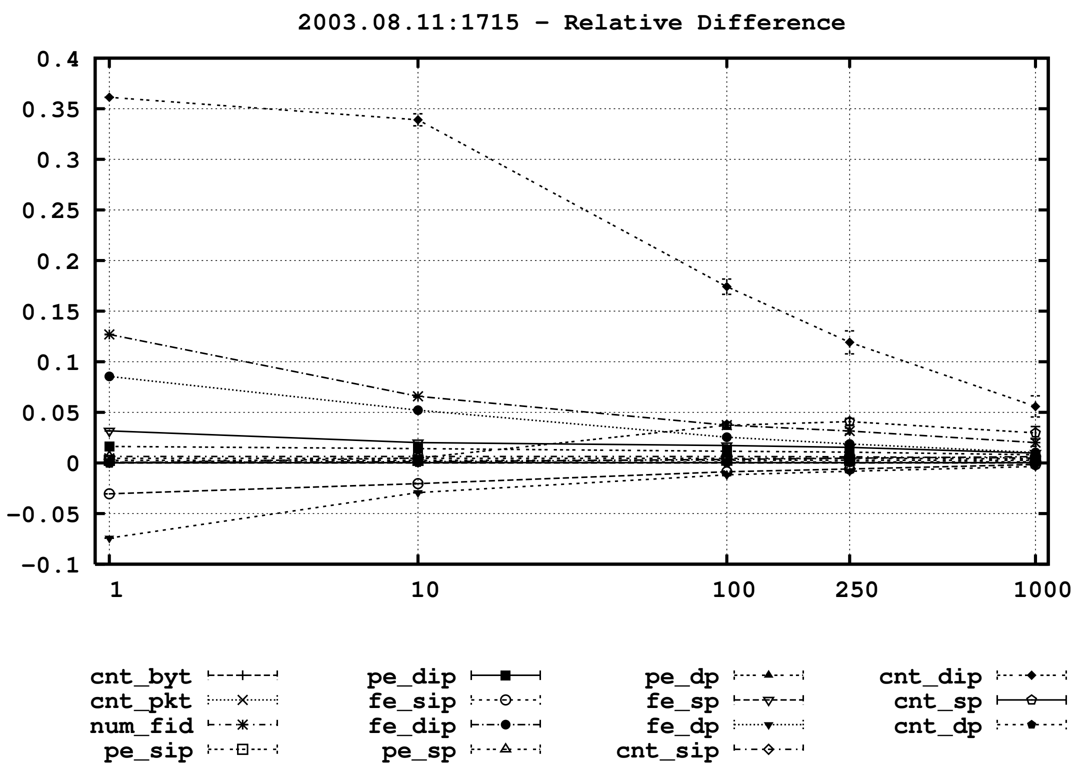


Figure 6.10: Relative Difference of Baseline and Total Traffic, swiCE2.switch.ch.

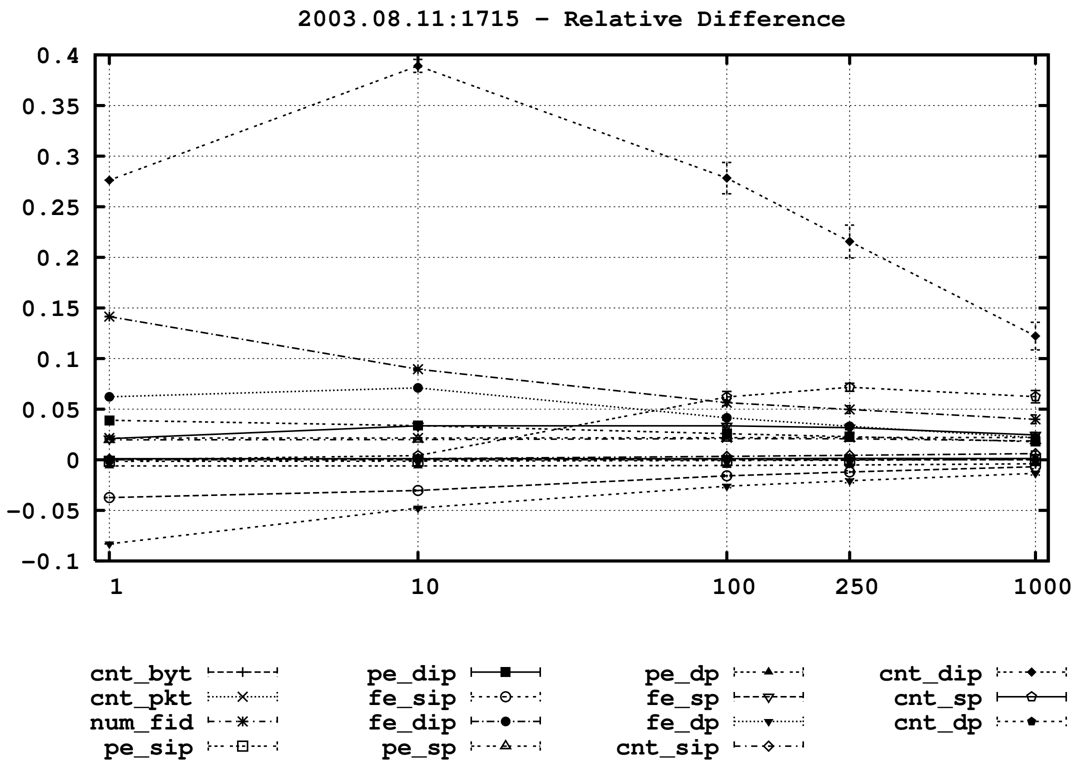


Figure 6.11: Relative Difference of Baseline and Total Traffic, swiBA2.switch.ch.

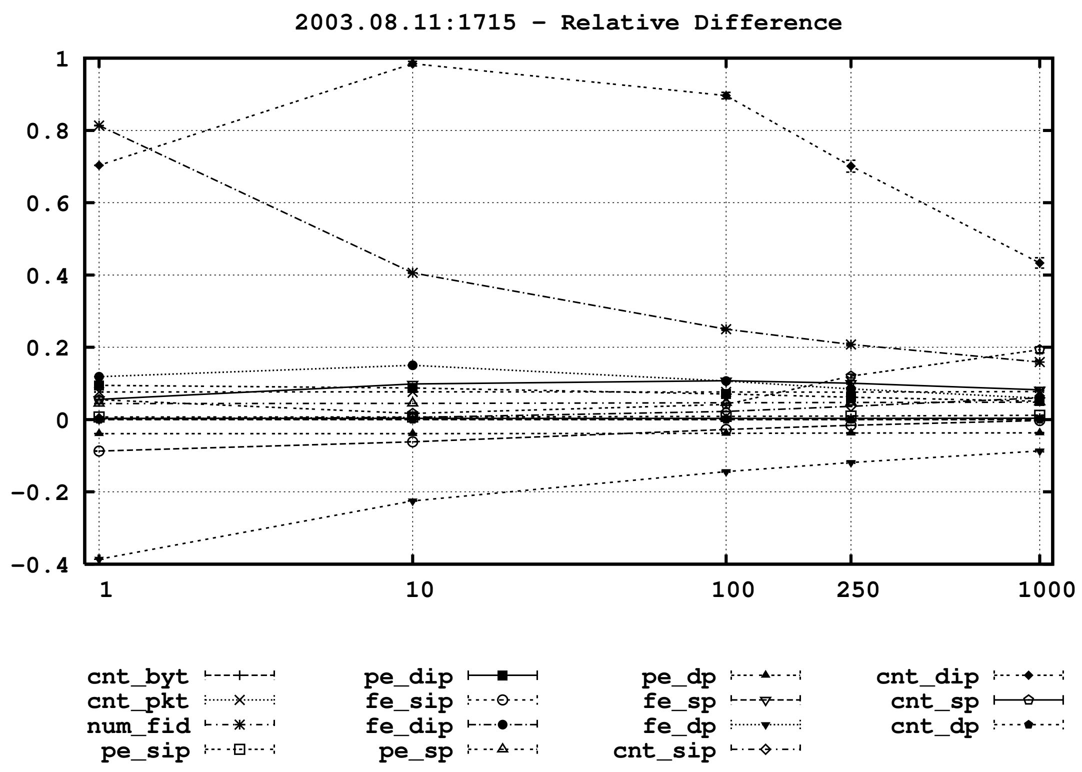


Figure 6.12: Relative Difference of Baseline and Total Traffic, swiCE3.switch.ch.

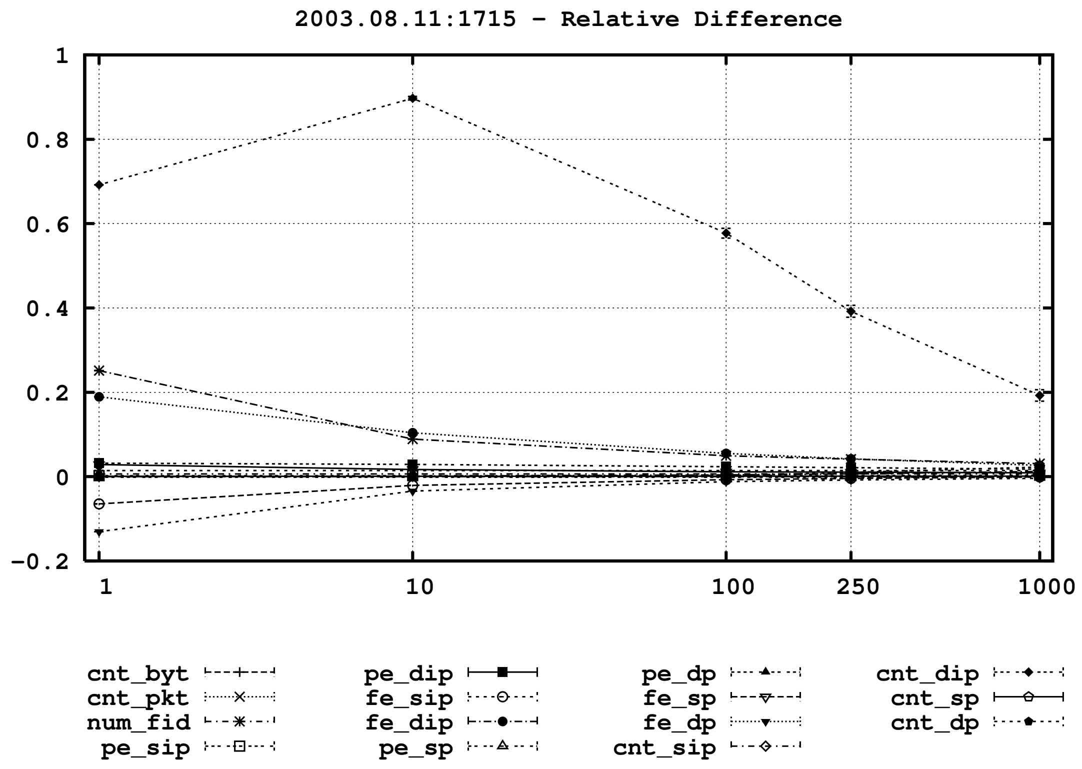


Figure 6.13: Relative Difference of Baseline and Total Traffic, swiIX1.switch.ch.

Chapter 7

Summary

In chapter 5 we presented several plots that demonstrate the properties of normal Internet traffic like flow duration or number of packets per flow. We discovered a larger percentage of 2-packet and 3-packet flows, that lasted about 3 and 9 seconds, respectively. We proved that the steps are not a problem caused by one SWITCH specific router setting. Therefore they should also be observable on many other devices.

Using the Blaster worm as an example, chapter 6 addresses the visibility and detectability of such an event. We approve the findings of [1], that entropy metrics are hardened against sampling. While especially packet entropy is hardly disturbed, flow entropy exposes an anomaly better. In addition, we showed that noisy signals complicate anomaly detection. Lowpass filtering is difficult to implement, since it usually introduces an unwanted time delay. Additional research is needed to find an optimal tradeoff between the three controverting parameters, information loss through sampling, low noise (stable signals), and good detectability (relative difference). This is a nontrivial task since optimizing one parameter usually downgrades the others in the opposite direction. We propose introducing a method to measure the quality of an anomaly metric that takes the three parameters into account.

Although routers with less traffic load show higher variations for all metrics they cannot be neglected, since the time and strength of an anomaly is different for the devices. Discovering an anomaly on more than one router is an indicator for a wide-ranging event and helps us avoiding false positives. The opposite direction does not hold for true. An anomaly can also affect just one traffic node.

In order to cover a large variety of anomalies, we are forced to measure a set of values which are independent of each other at best. Reducing the number of metrics is desirable. But especially simple count methods like the number of flows or counted IPs per interval should not be canceled, since they often expose an event better than any other inspected method.

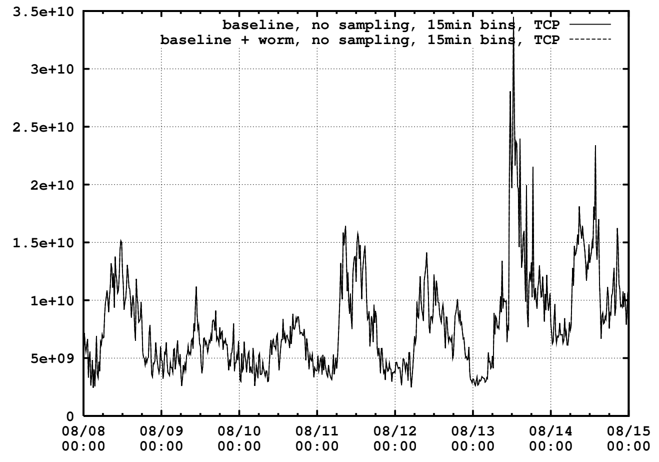
Chapter 8

Outlook

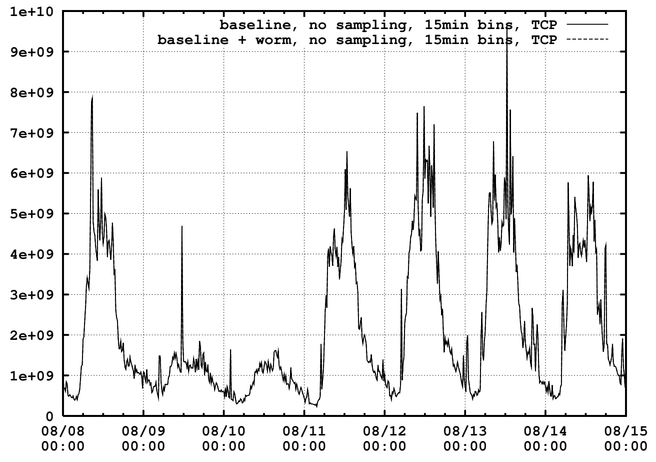
A lot of research time has been invested in finding sampling hardened metrics that still expose a anomaly well. Future work should also take noise into account, to check its relevance. Noise suppression methods, such as wavelets, may help to increase the signal to noise ratio.
As mentioned in the summary, we suggest introducing a function to measure the quality of an metric. Maximizing the output of the function that is dependent on SNR, sampling stability, and relative difference should lead us to an optimal solution. Of course, any other detection metric which is realizable using the widespread Cisco NetFlow Protocol is welcome.
Finding a method to combine all routers and all metrics into one single indicator would also be desirable. We disbelief a simple weighted sum will do it, since we risk to average out relevant information.
A focus just on ICMP flows would also be interesting. Random scanning algorithms and DDoS attacks typically are not able to avoid or even hide lots of ICMP error messages. For instance, our Blaster worm happening was escorted by a massive ICMP flooding. Unfortunately NetFlow v5 reveals only minimal information about the content of a ICMP message.

Appendix A

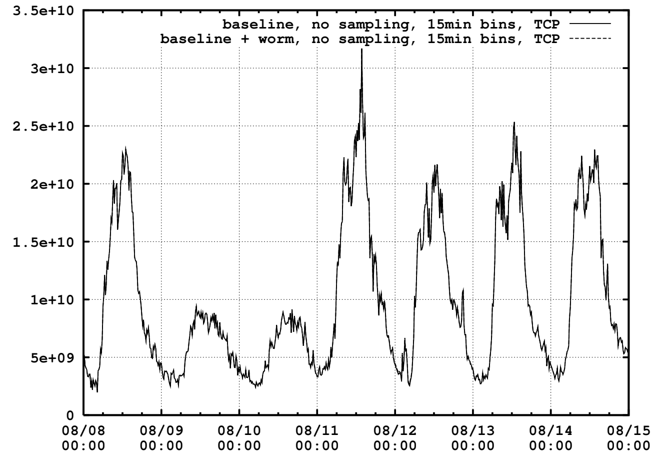
Baseline vs. Totalline, Plots



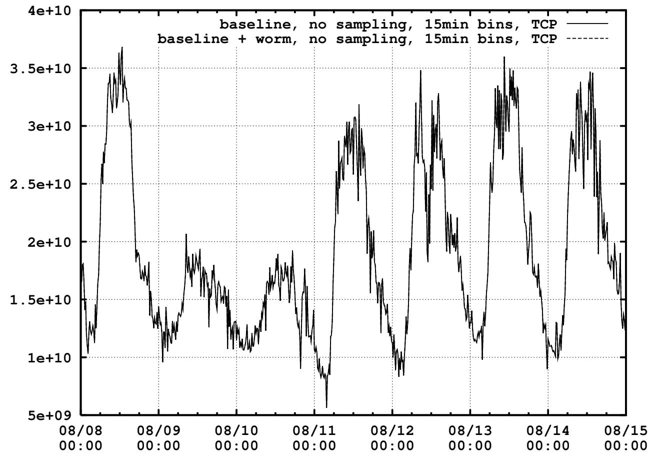
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

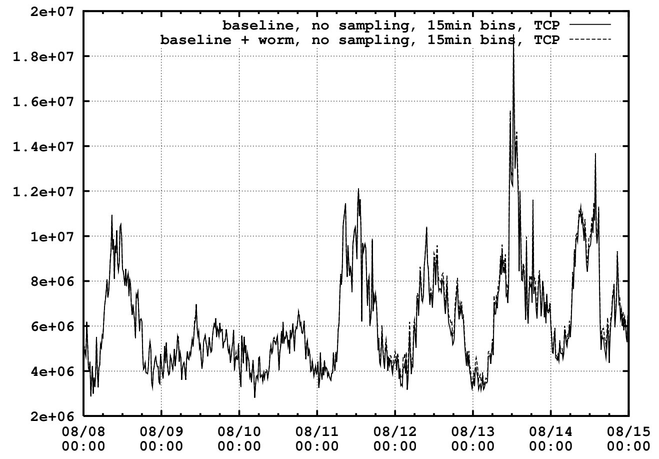


(c) swiCE3.switch.ch.

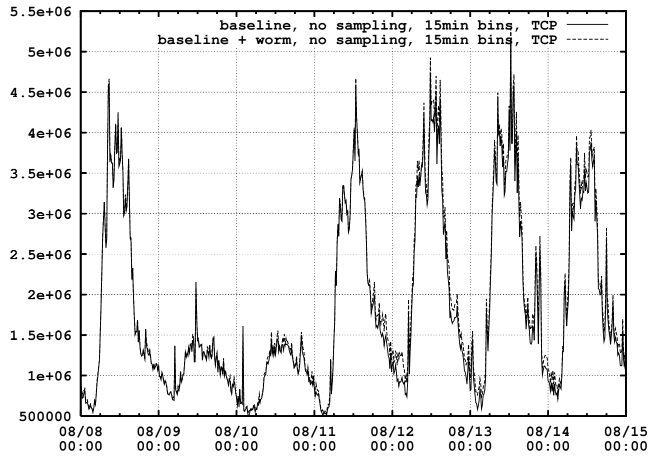


(d) swiIX1.switch.ch.

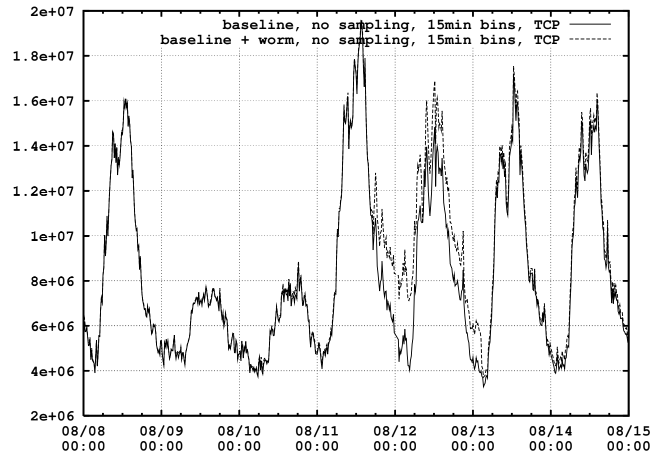
Figure A.1: Baseline vs. Total Traffic, Counted Bytes, August 08-14, 2003



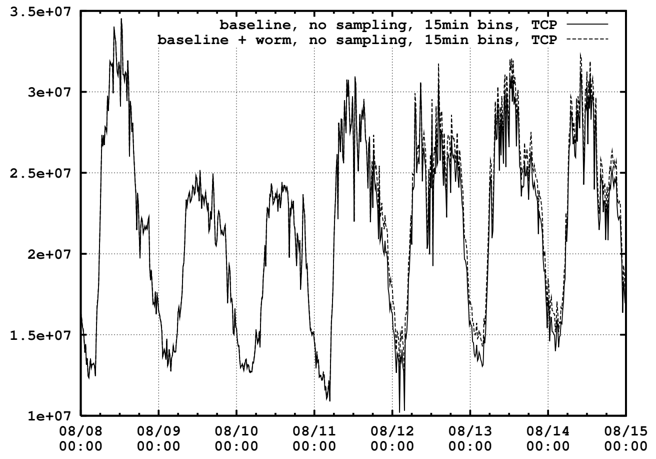
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

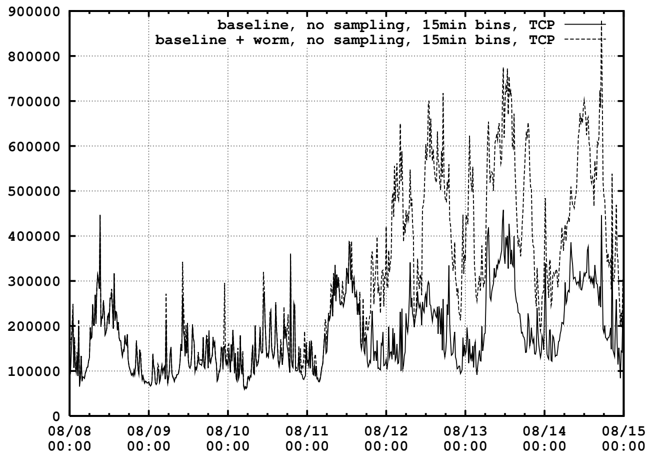


(c) swiCE3.switch.ch.

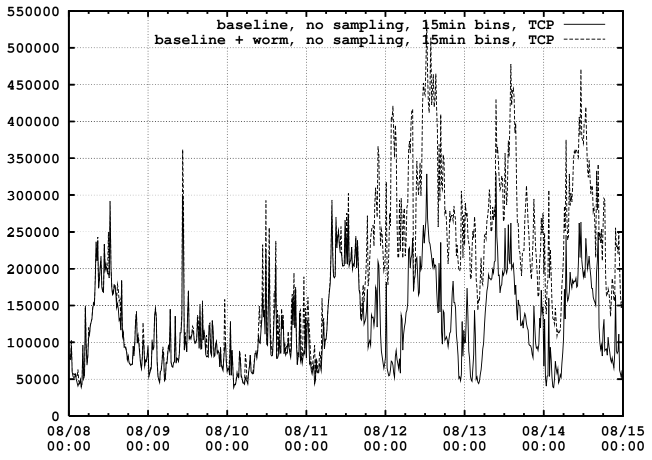


(d) swiIX1.switch.ch.

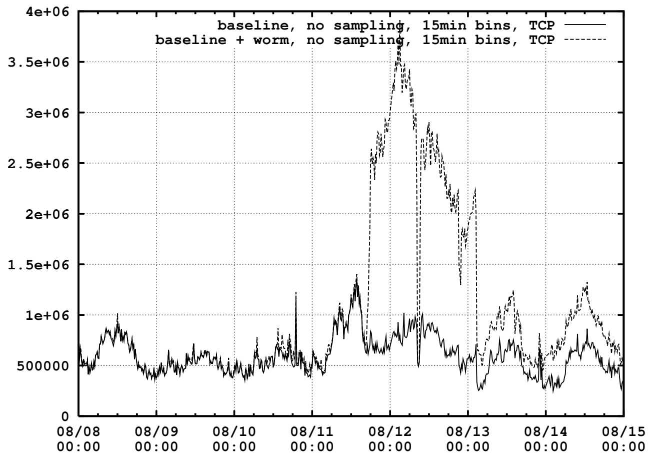
Figure A.2: Baseline vs. Total Traffic, Counted Packets, August 08-14, 2003



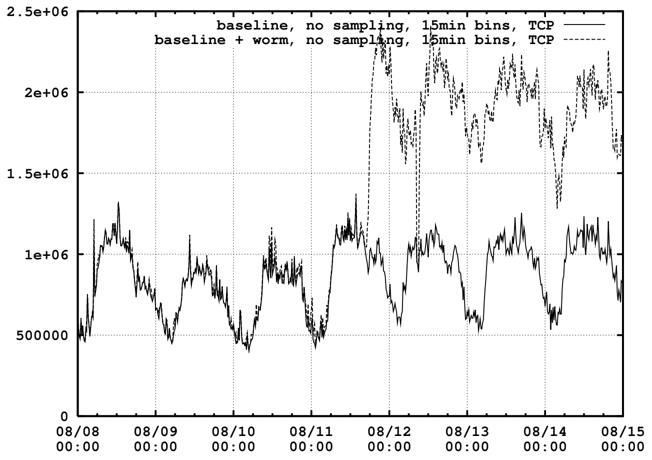
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

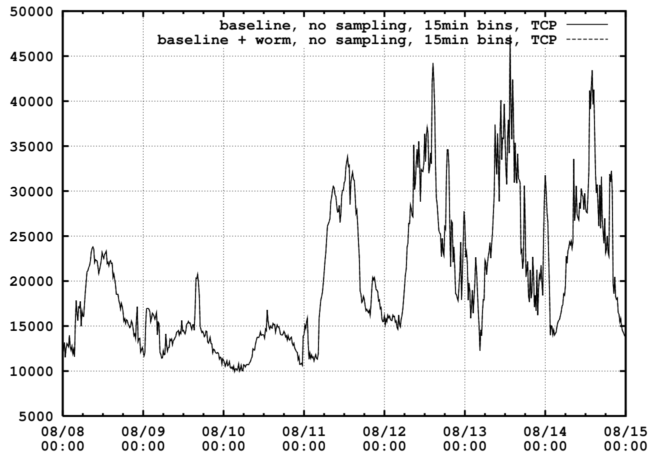


(c) swiCE3.switch.ch.

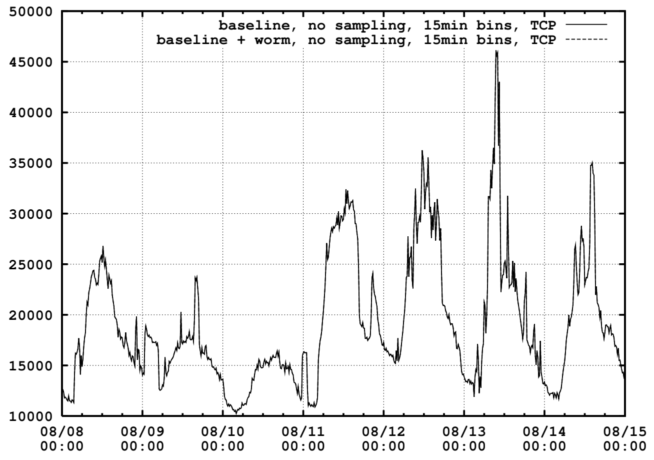


(d) swiIX1.switch.ch.

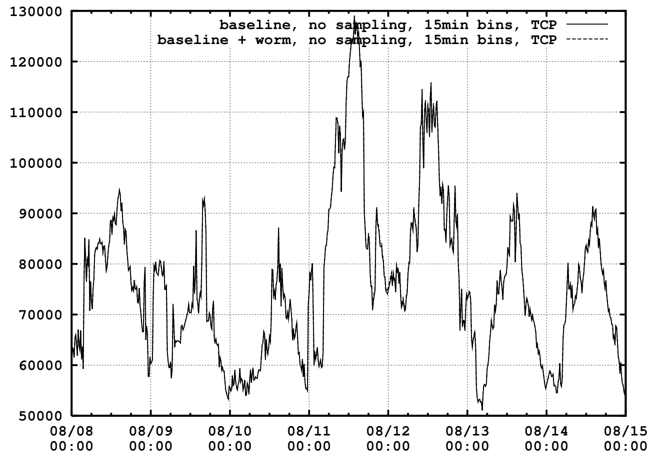
Figure A.3: Baseline vs. Total Traffic, Counted Flow IDs, August 08-14, 2003



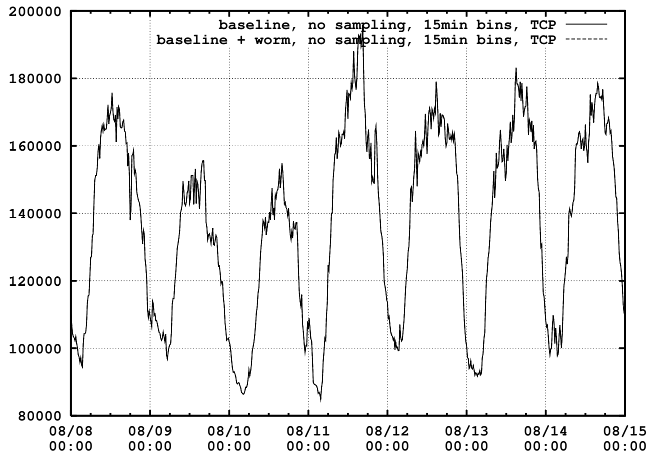
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

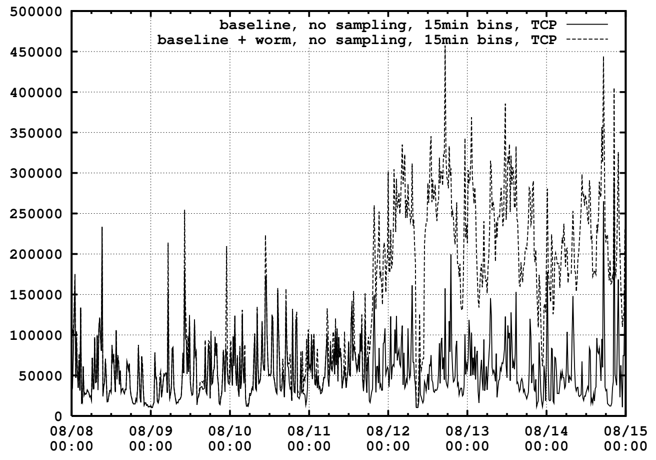


(c) swiCE3.switch.ch.

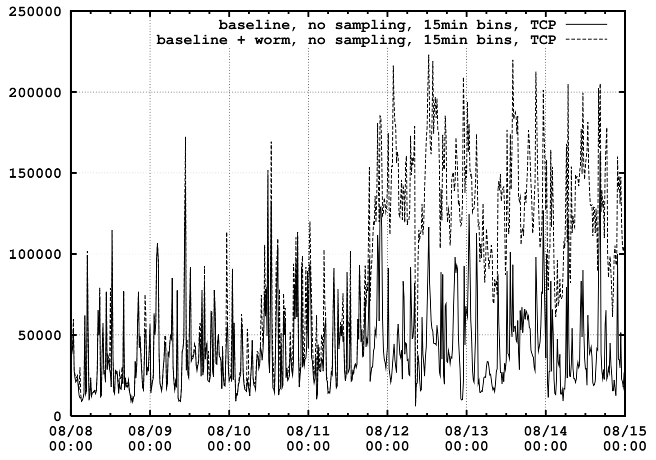


(d) swiIX1.switch.ch.

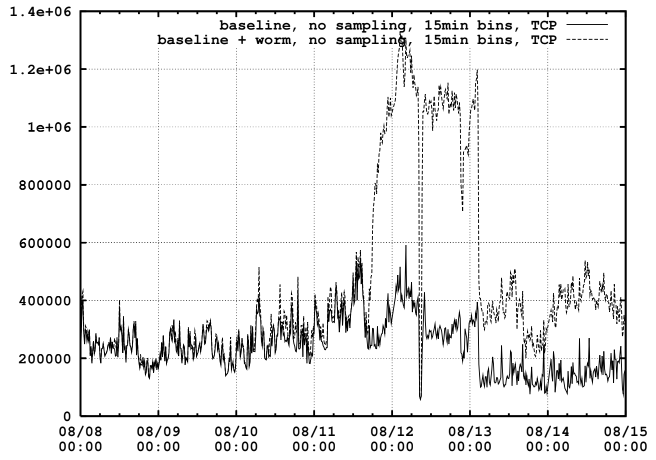
Figure A.4: Baseline vs. Total Traffic, Counted Source IPs, August 08-14, 2003



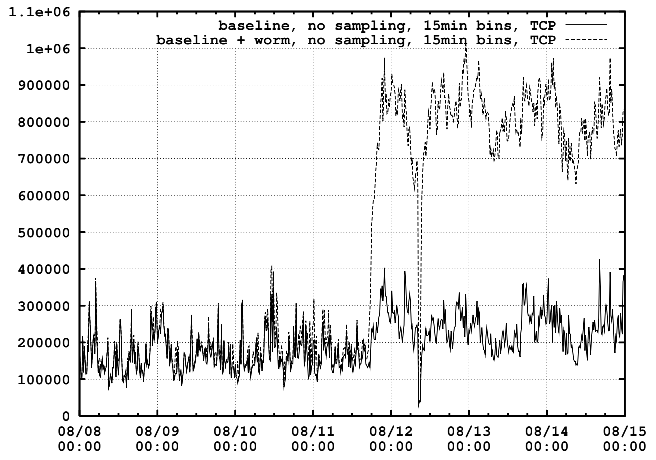
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

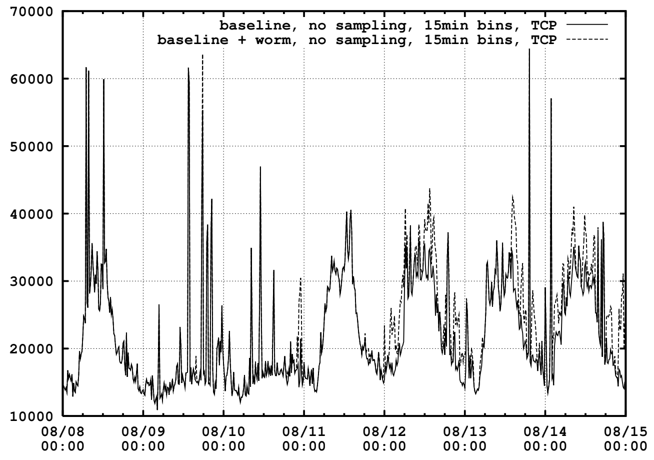


(c) swiCE3.switch.ch.

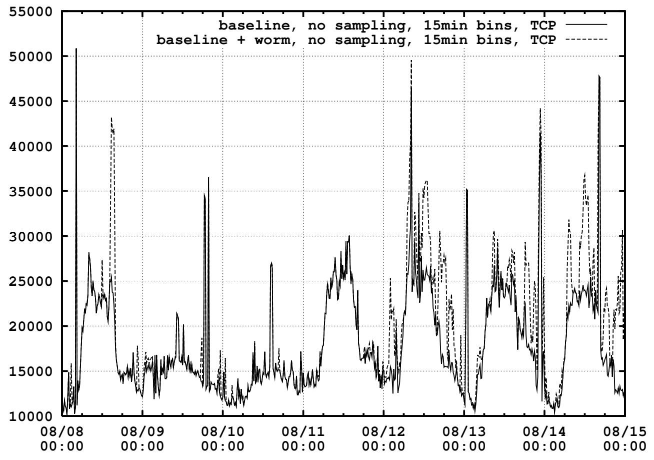


(d) swiIX1.switch.ch.

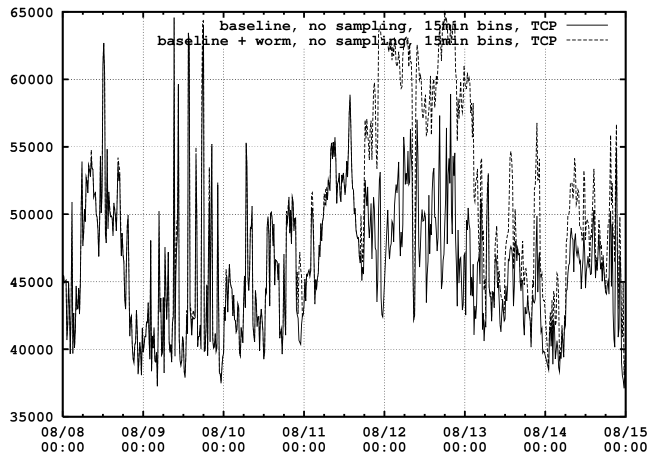
Figure A.5: Baseline vs. Total Traffic, Counted Destination IPs, August 08-14, 2003



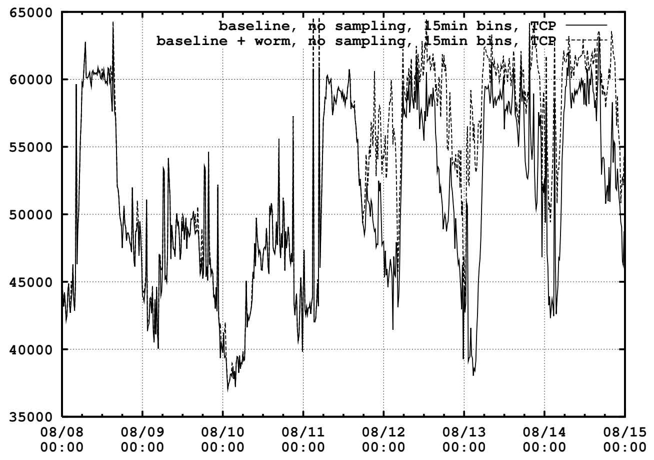
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

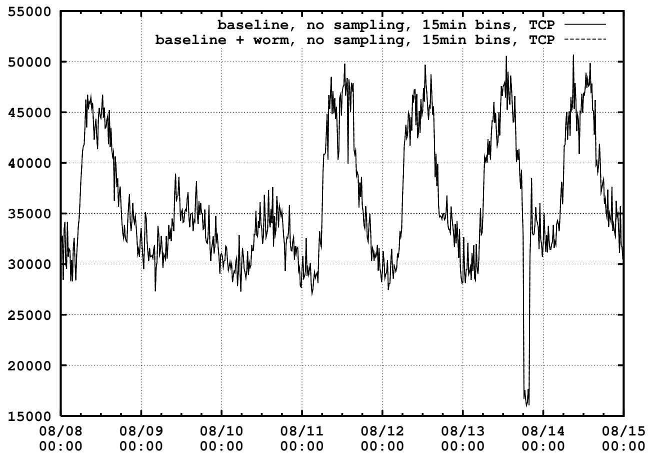


(c) swiCE3.switch.ch.

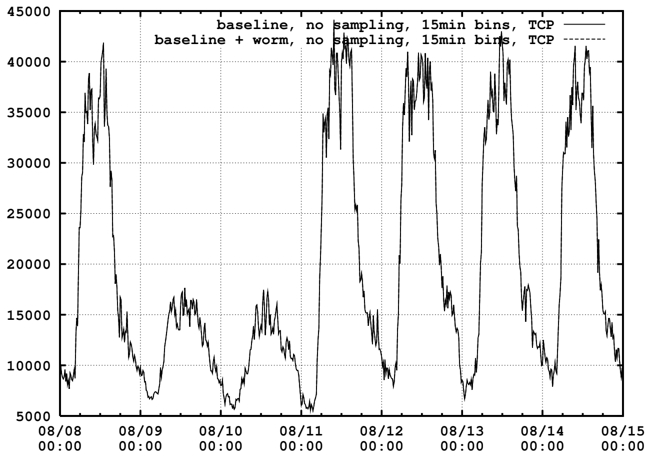


(d) swiIX1.switch.ch.

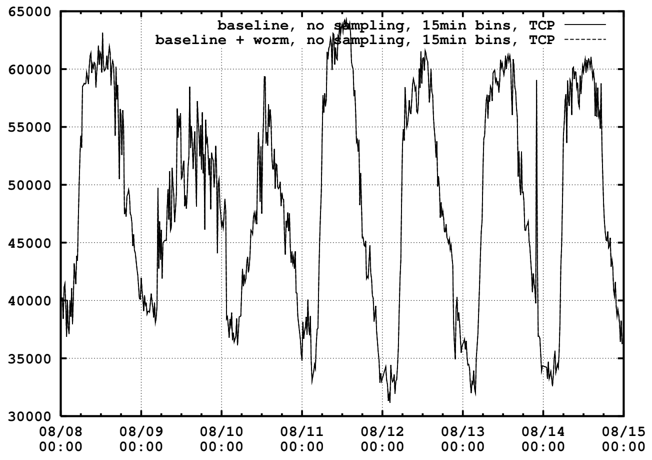
Figure A.6: Baseline vs. Total Traffic, Counted Source Ports, August 08-14, 2003



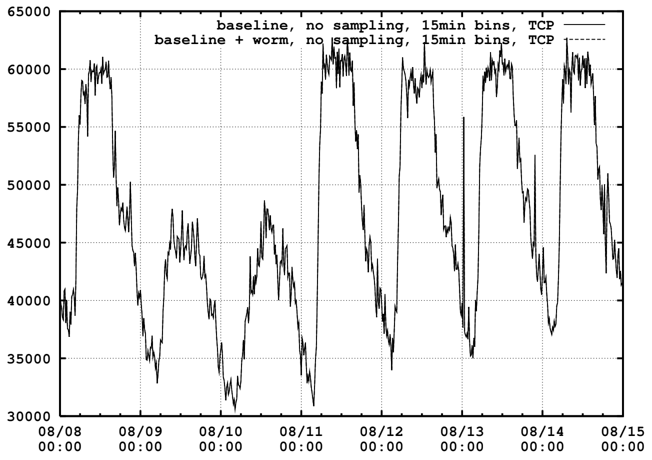
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

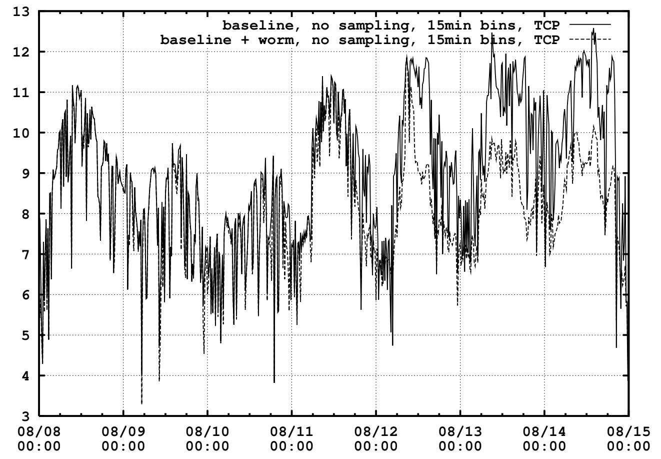


(c) swiCE3.switch.ch.

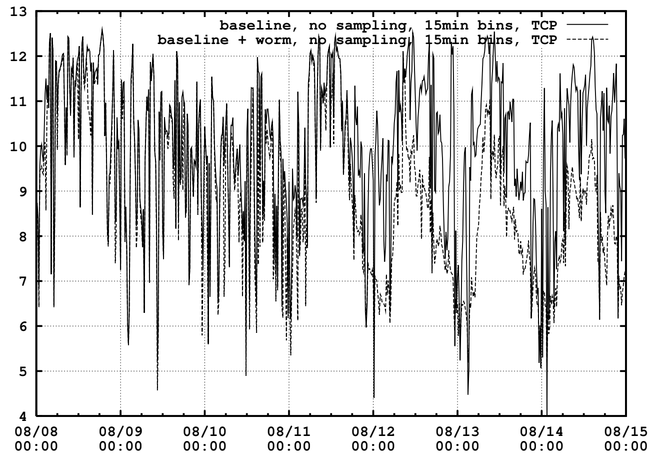


(d) swiIX1.switch.ch.

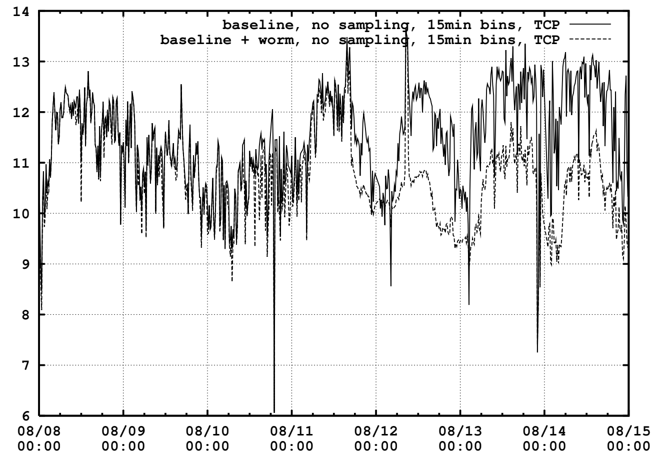
Figure A.7: Baseline vs. Total Traffic, Counted Destination Ports, August 08-14, 2003



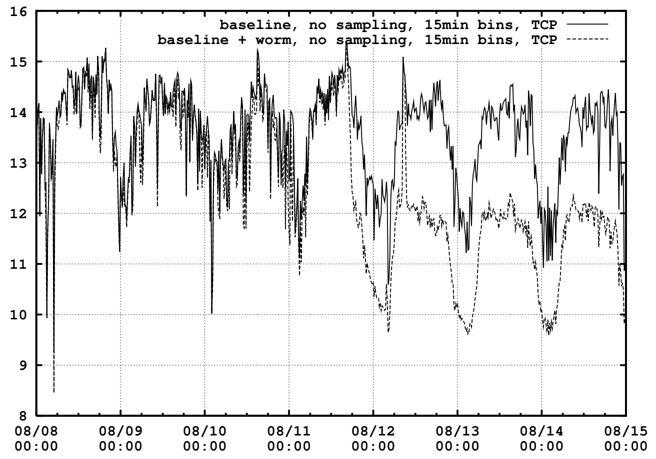
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

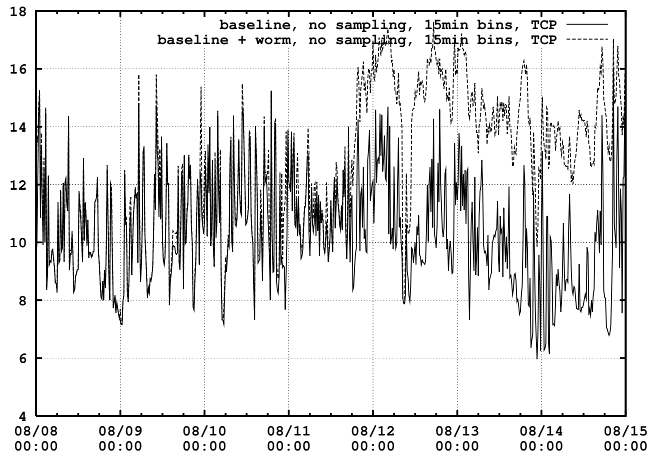


(c) swiCE3.switch.ch.

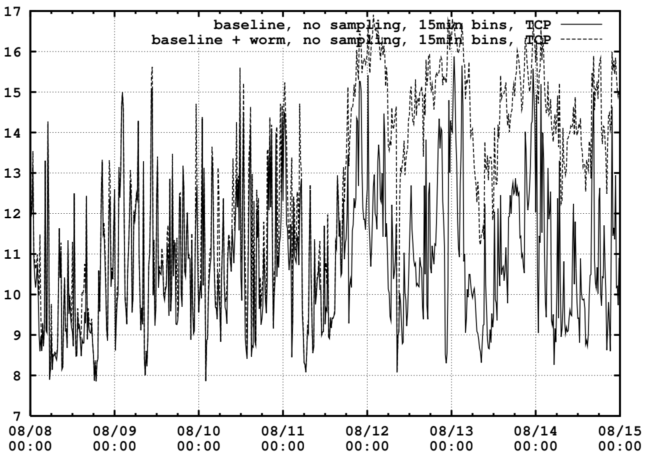


(d) swiIX1.switch.ch.

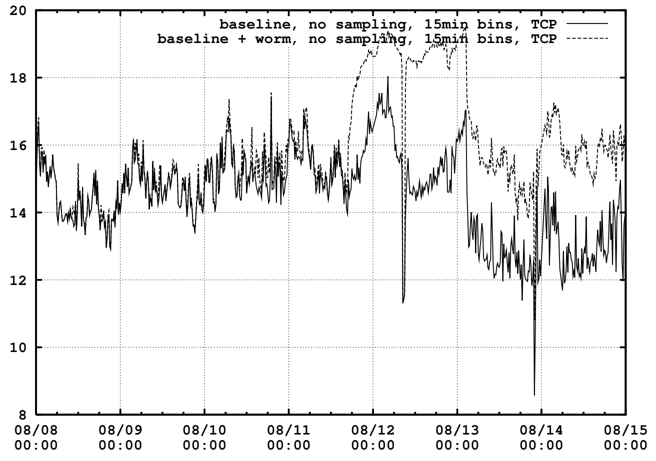
Figure A.8: Baseline vs. Total Traffic, Flow Entropy of Source IPs, August 08-14, 2003



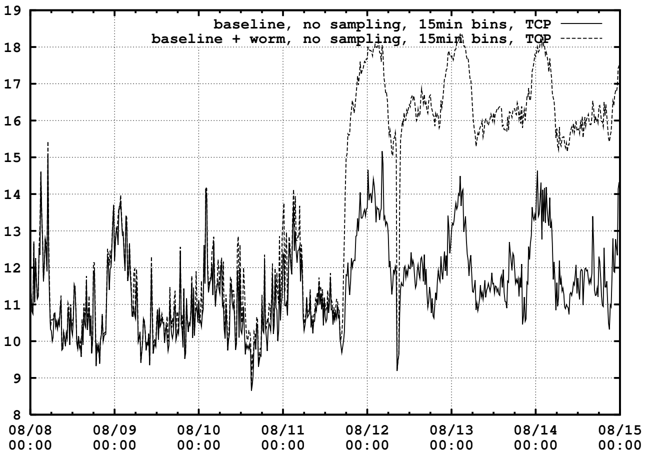
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

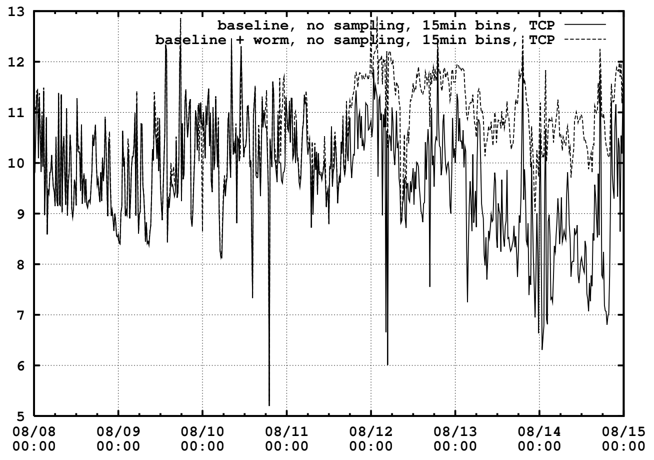


(c) swiCE3.switch.ch.

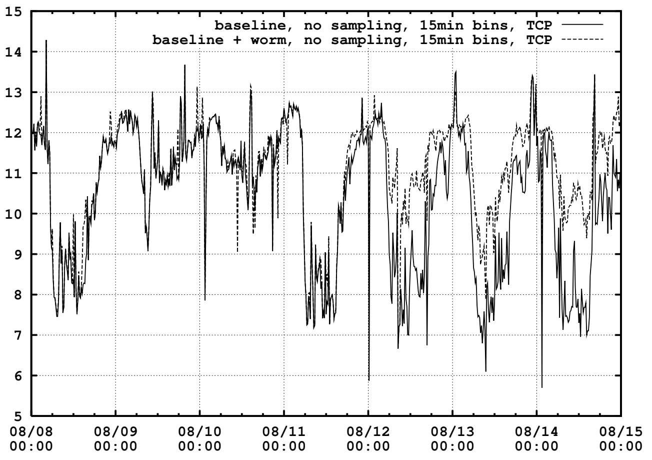


(d) swiIX1.switch.ch.

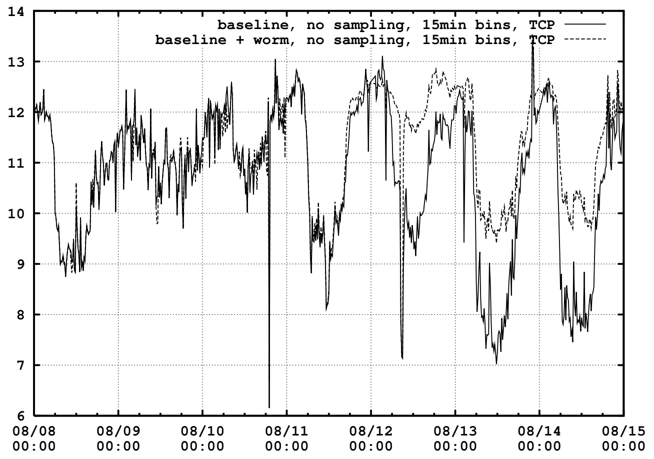
Figure A.9: Baseline vs. Total Traffic, Flow Entropy of Destination IPs, August 08-14, 2003



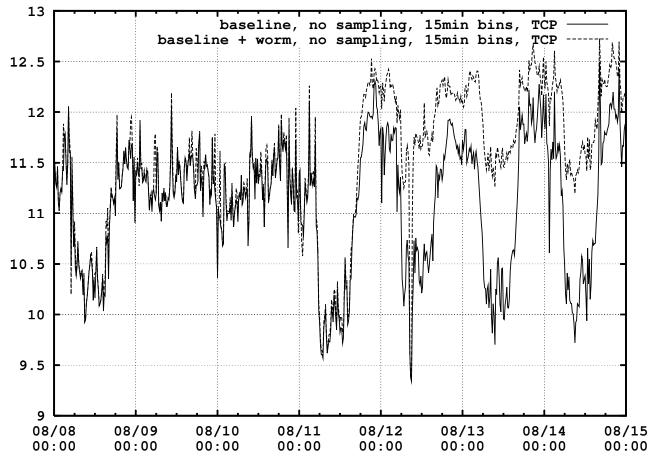
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

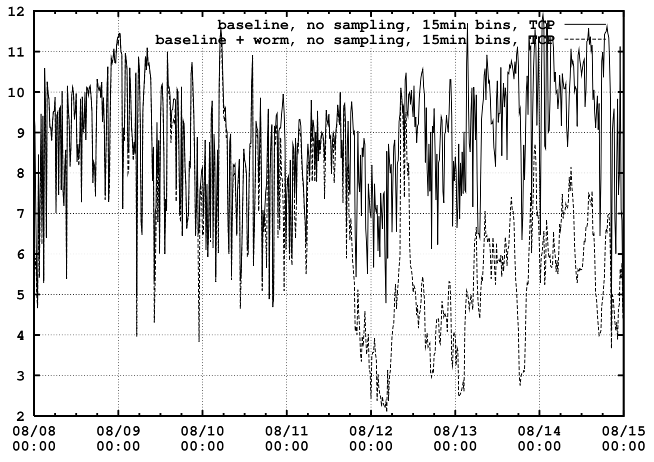


(c) swiCE3.switch.ch.

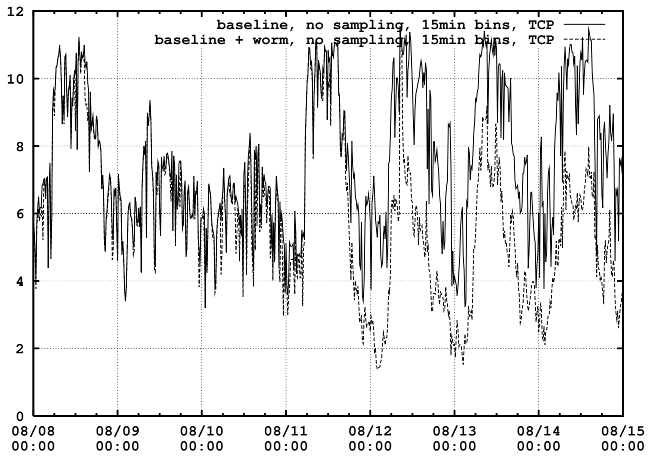


(d) swiIX1.switch.ch.

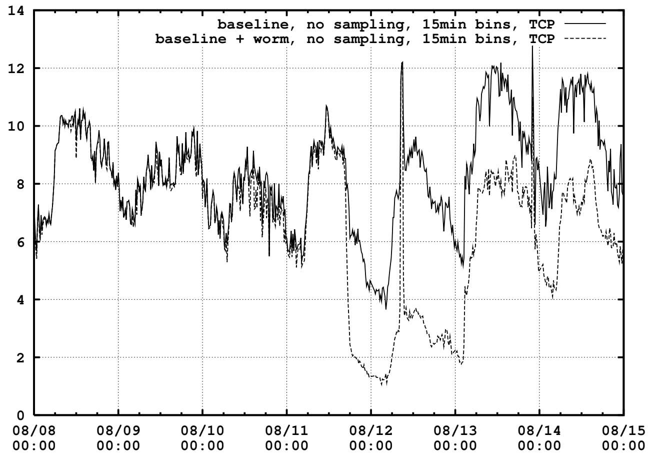
Figure A.10: Baseline vs. Total Traffic, Flow Entropy of Source Ports, August 08-14, 2003



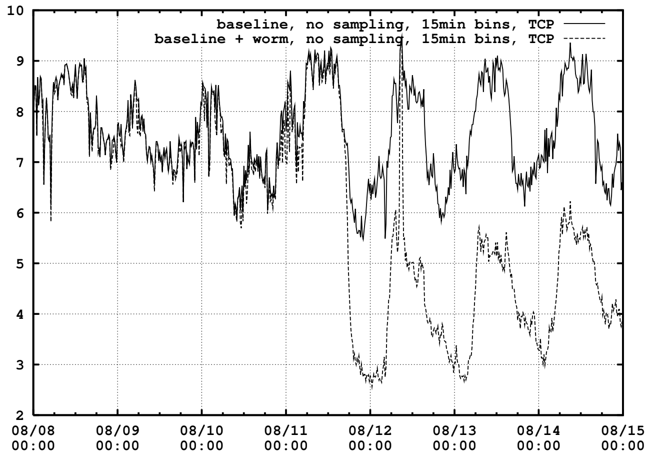
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

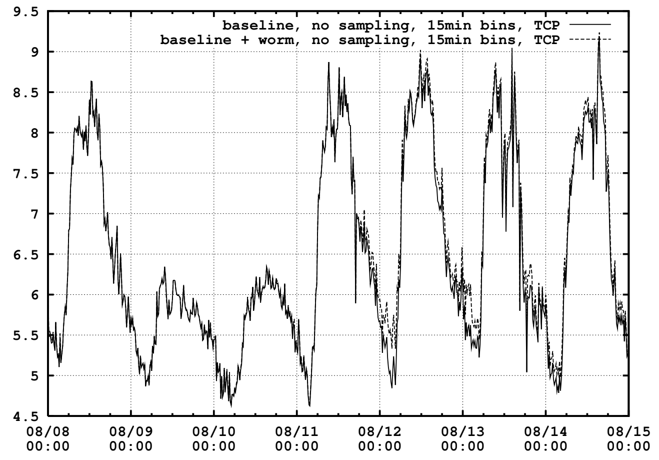


(c) swiCE3.switch.ch.

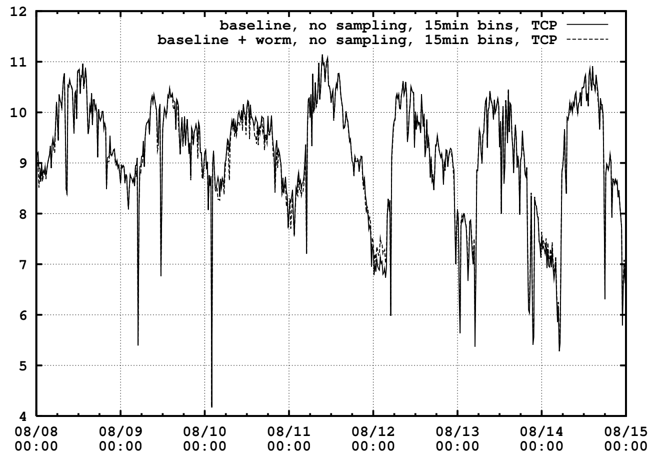


(d) swiIX1.switch.ch.

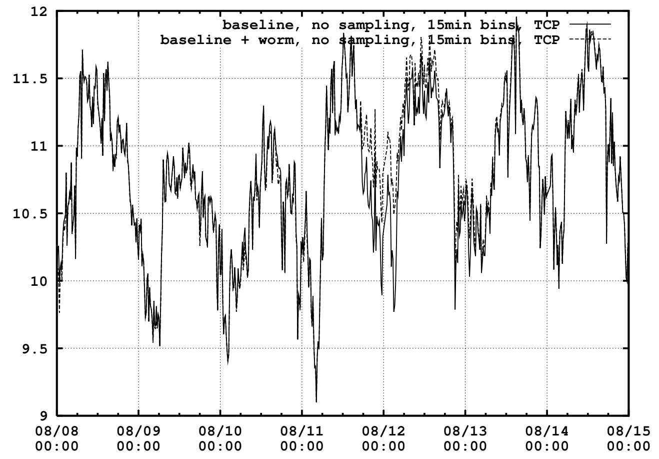
Figure A.11: Baseline vs. Total Traffic, Flow Entropy of Destination Ports, August 08-14, 2003



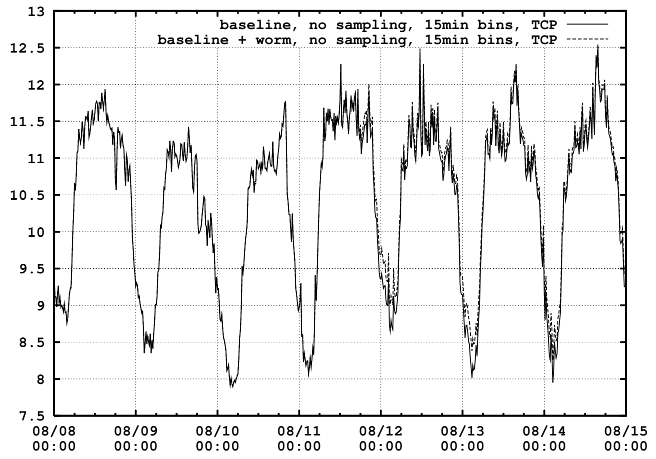
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

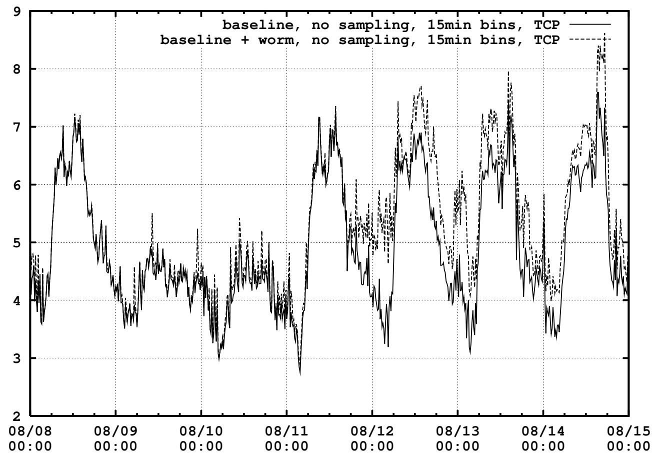


(c) swiCE3.switch.ch.

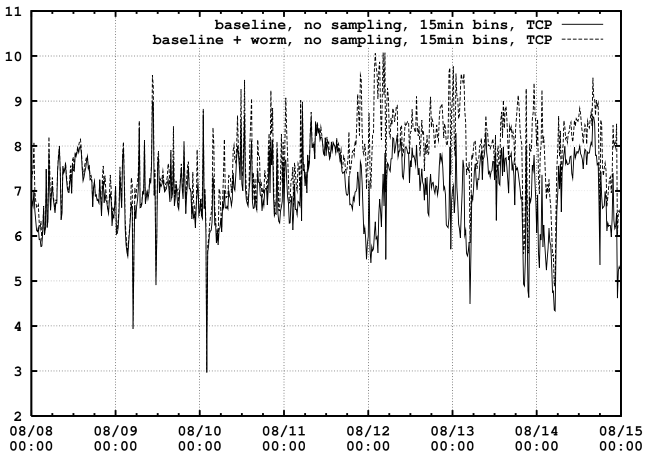


(d) swiIX1.switch.ch.

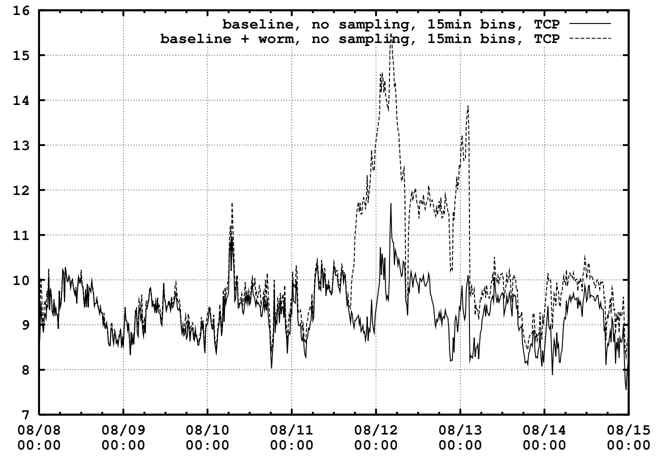
Figure A.12: Baseline vs. Total Traffic, Packet Entropy of Source IPs, August 08-14, 2003



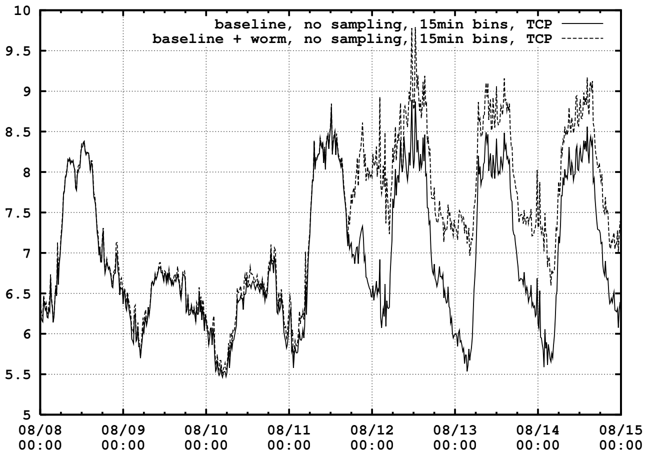
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

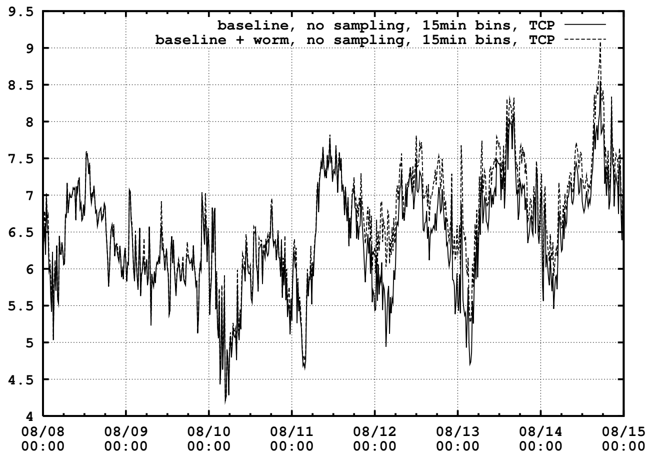


(c) swiCE3.switch.ch.

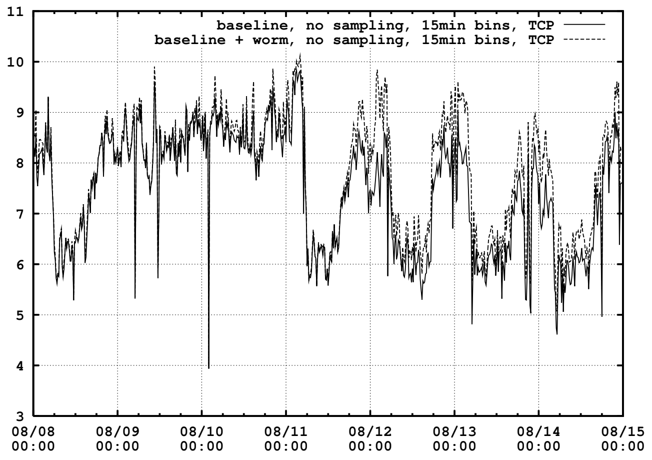


(d) swiIX1.switch.ch.

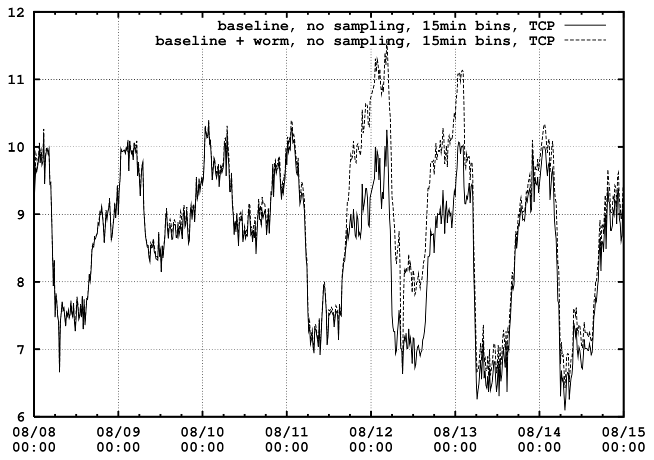
Figure A.13: Baseline vs. Total Traffic, Packet Entropy of Destination IPs, August 08-14, 2003



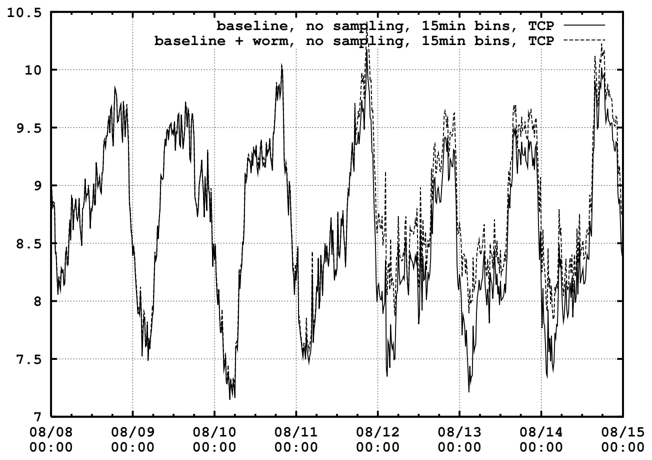
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

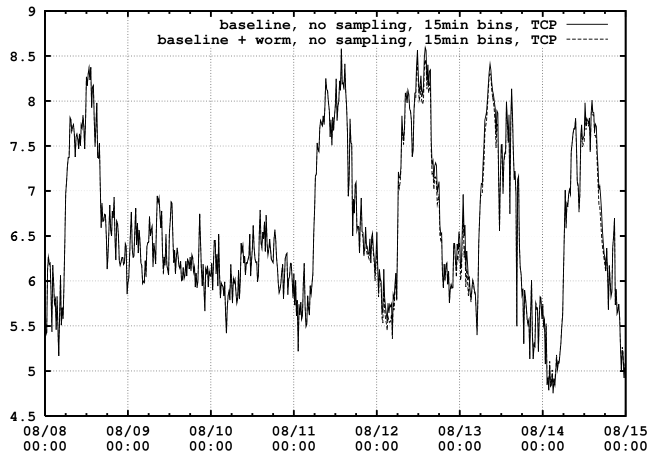


(c) swiCE3.switch.ch.

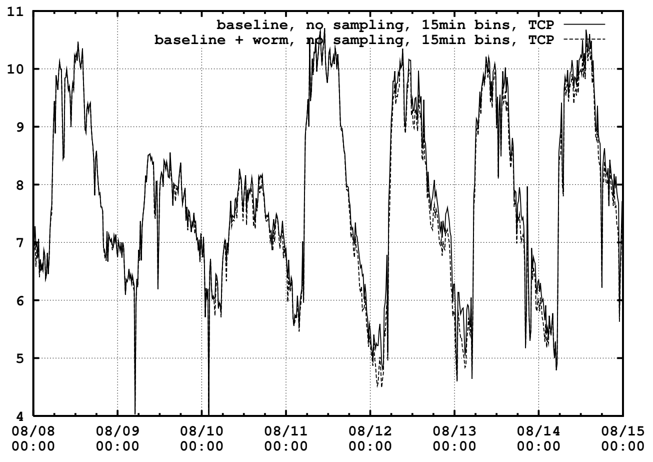


(d) swiIX1.switch.ch.

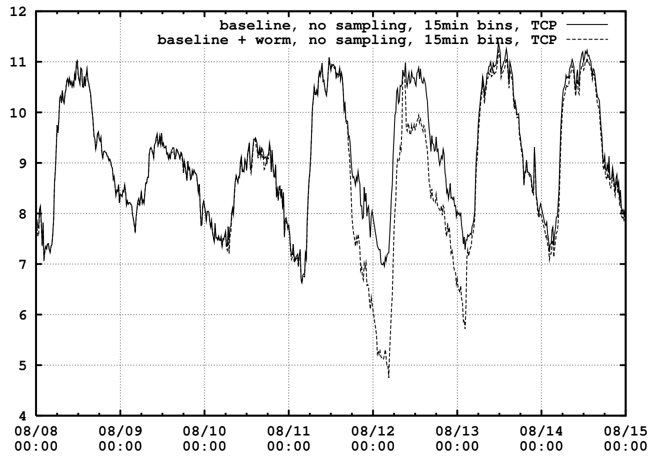
Figure A.14: Baseline vs. Total Traffic, Packet Entropy of Source Ports, August 08-14, 2003



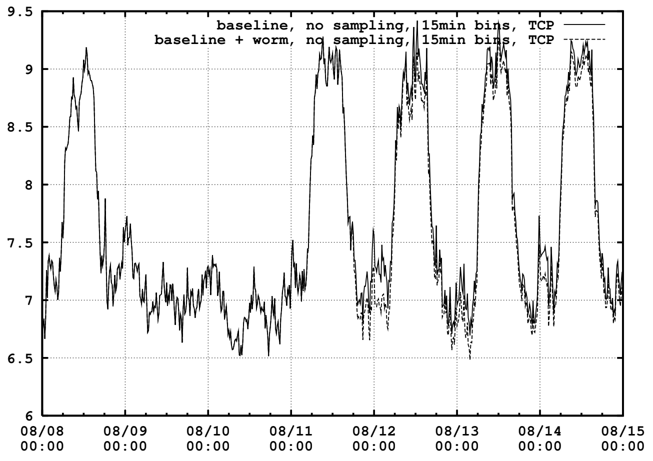
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.



(c) swiCE3.switch.ch.

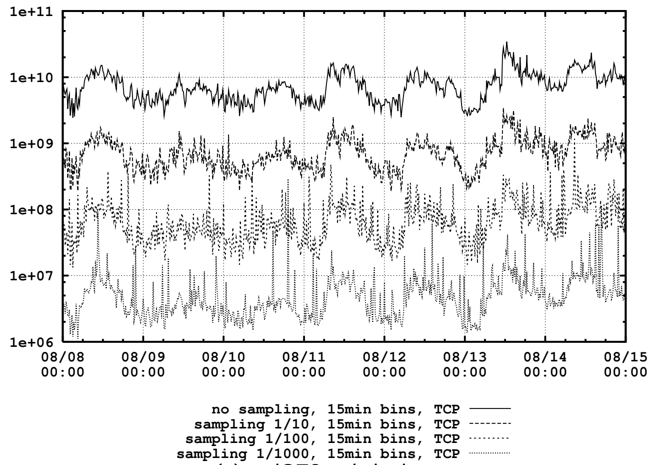


(d) swiIX1.switch.ch.

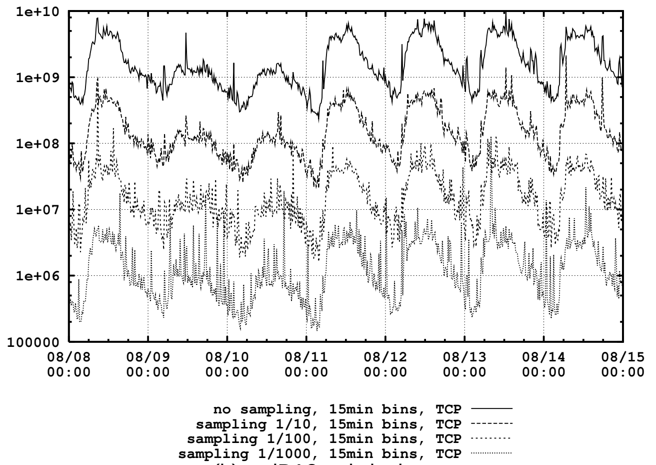
Figure A.15: Baseline vs. Total Traffic, Packet Entropy of Destination Ports, August 08-14, 2003

Appendix B

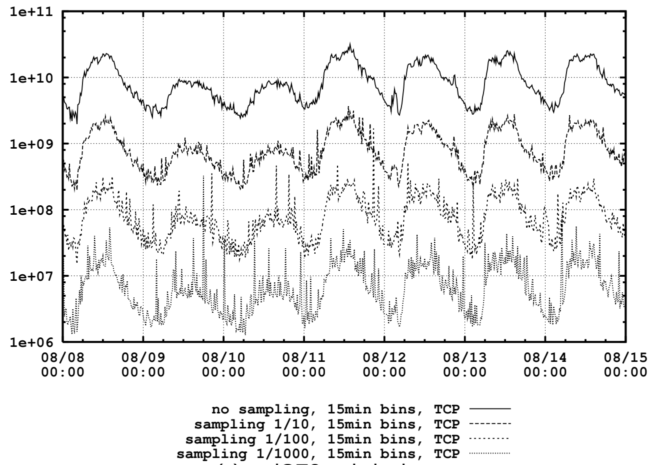
Impact of Sampling, Plots



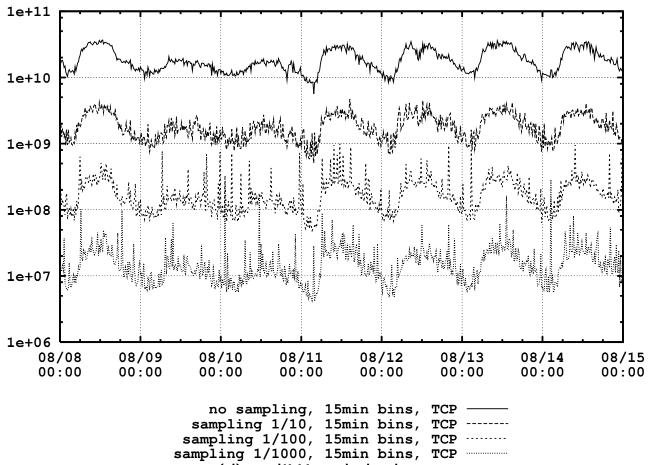
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

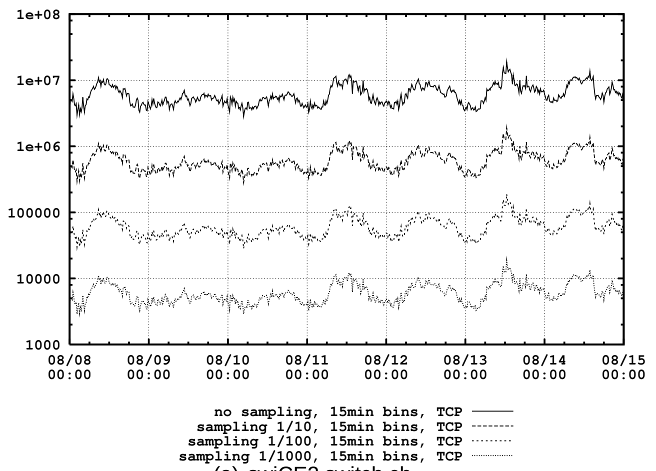


(c) swiCE3.switch.ch.

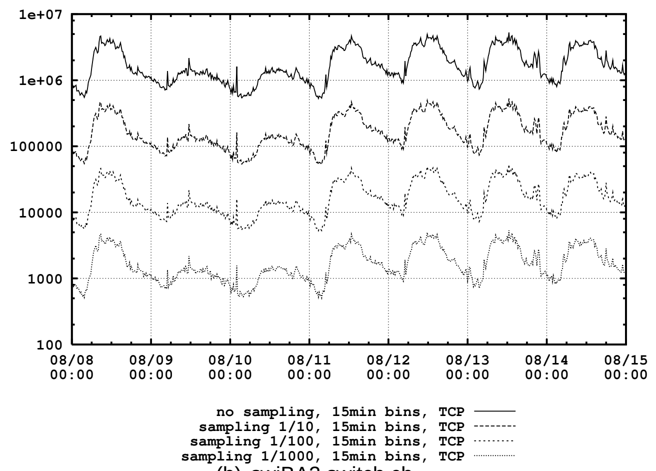


(d) swiIX1.switch.ch.

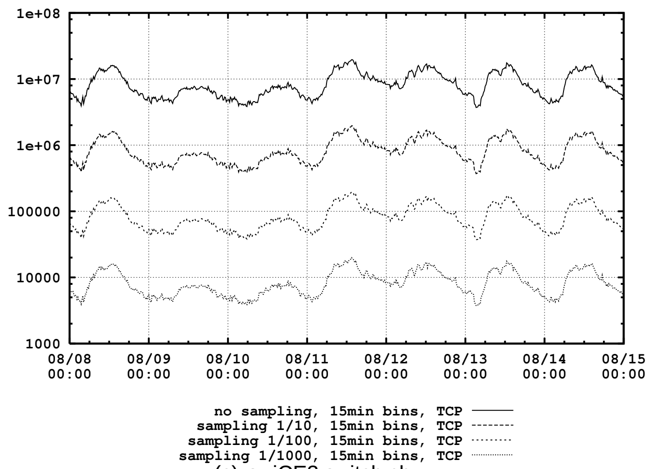
Figure B.1: Impact of Sampling, Counted Bytes, August 08-14, 2003



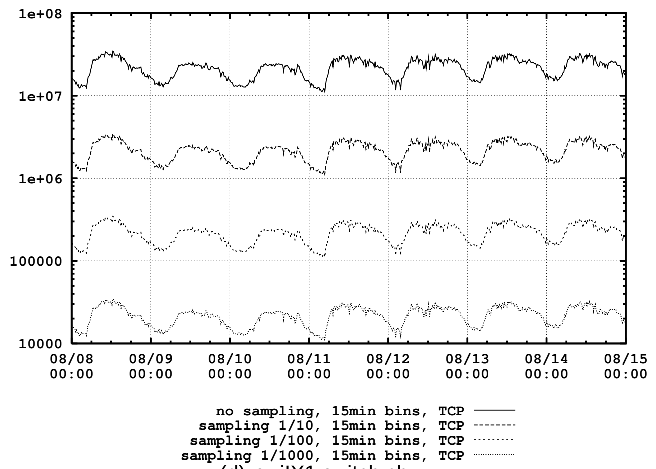
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

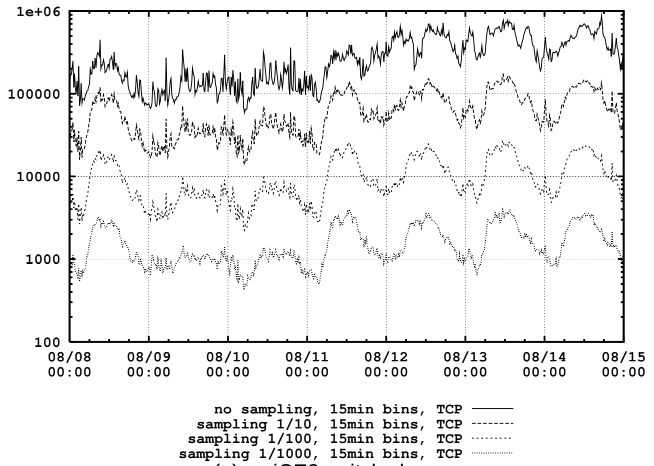


(c) swiCE3.switch.ch.

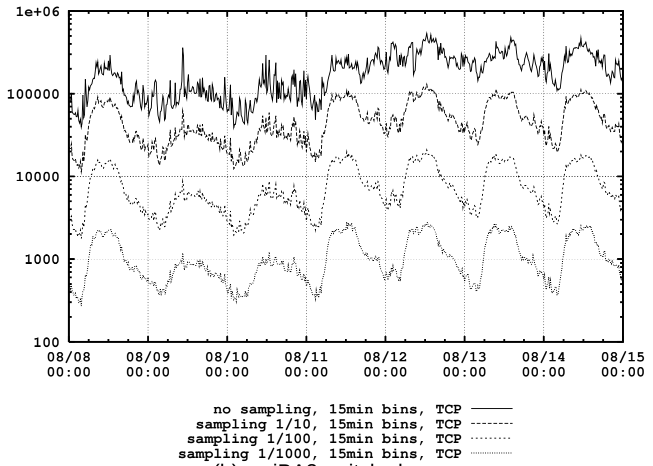


(d) swiIX1.switch.ch.

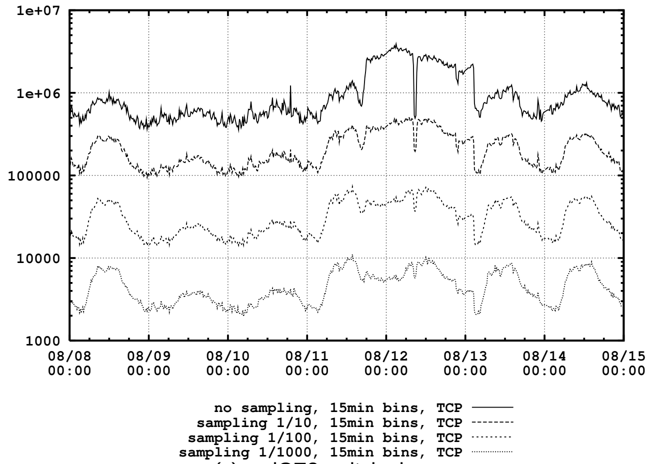
Figure B.2: Impact of Sampling, Counted Packets, August 08-14, 2003



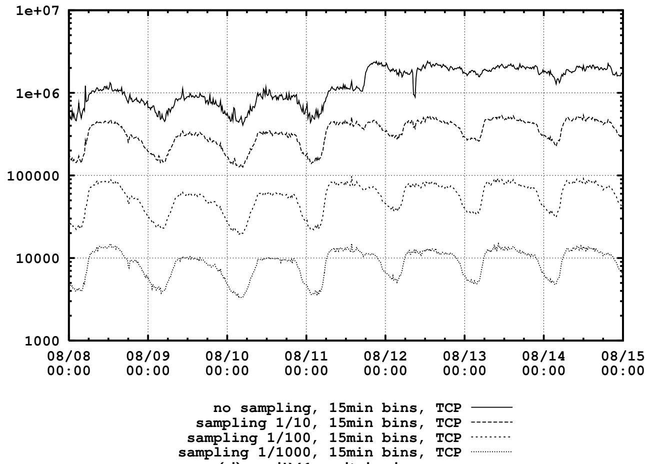
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

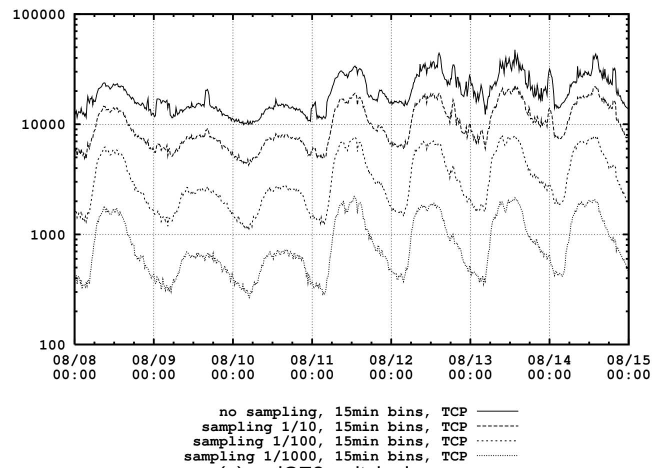


(c) swiCE3.switch.ch.

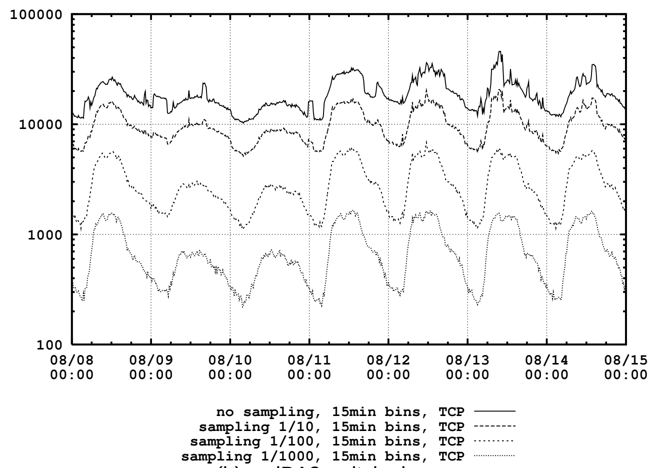


(d) swiIX1.switch.ch.

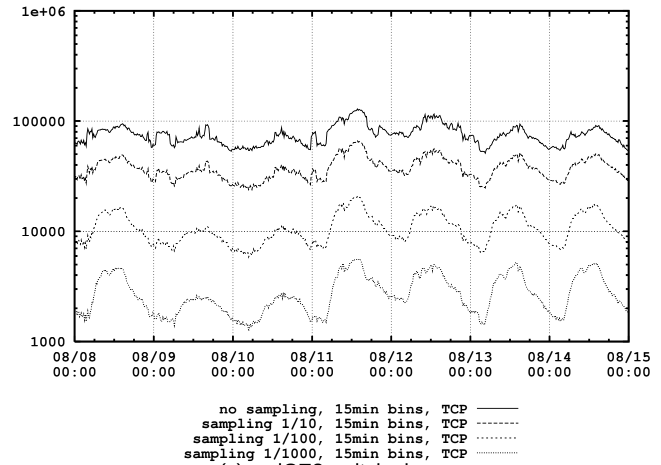
Figure B.3: Impact of Sampling, Counted Flow IDs, August 08-14, 2003



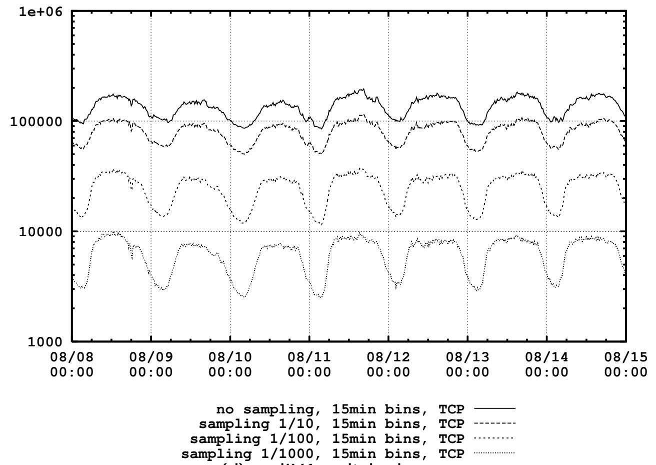
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

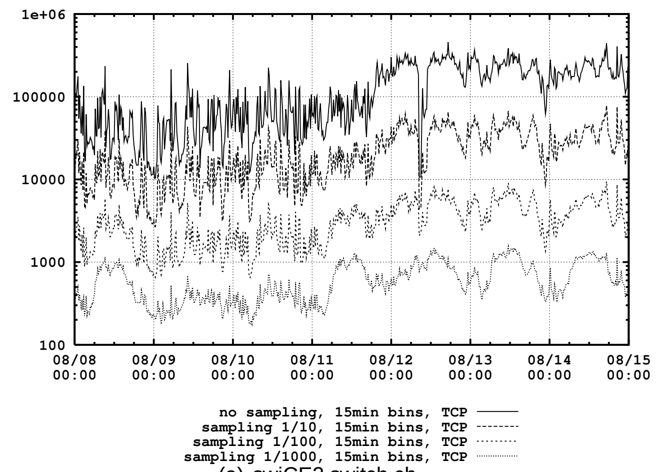


(c) swiCE3.switch.ch.

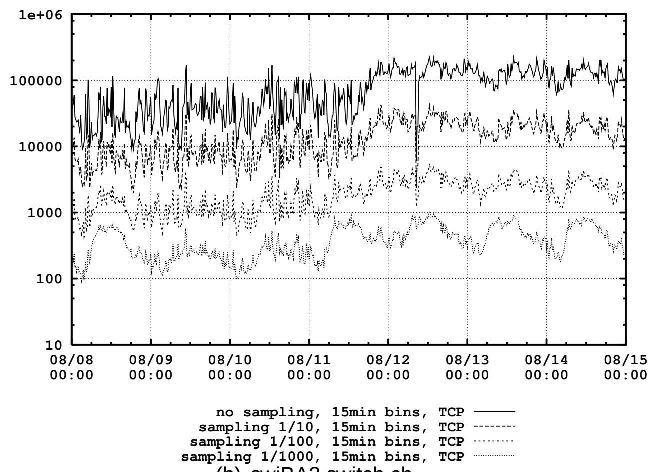


(d) swiIX1.switch.ch.

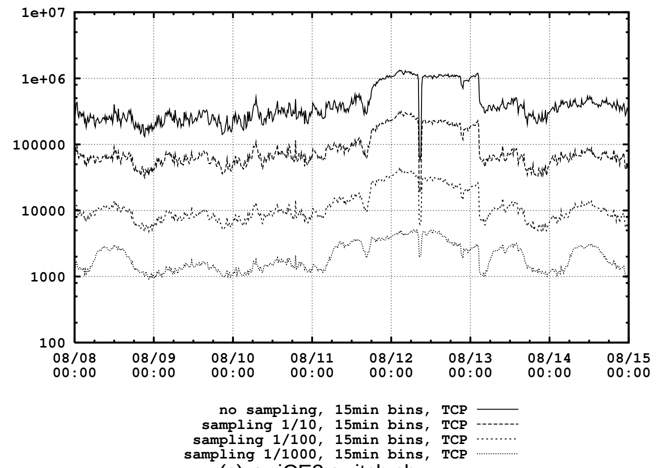
Figure B.4: Impact of Sampling, Counted Source IPs, August 08-14, 2003



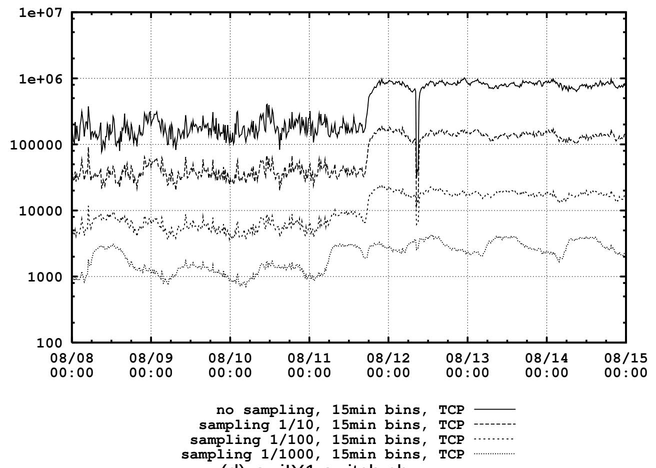
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

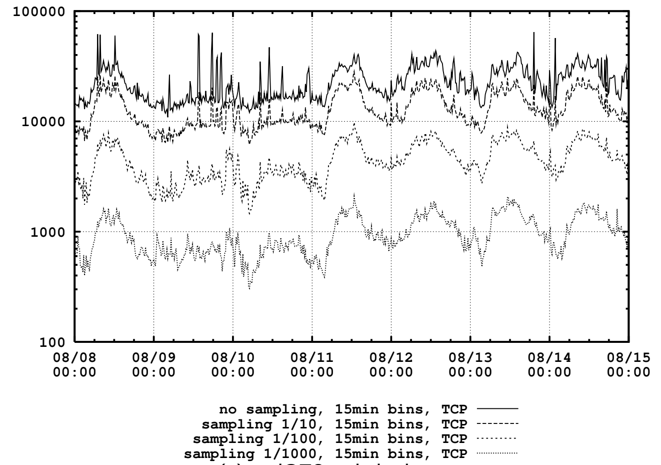


(c) swiCE3.switch.ch.

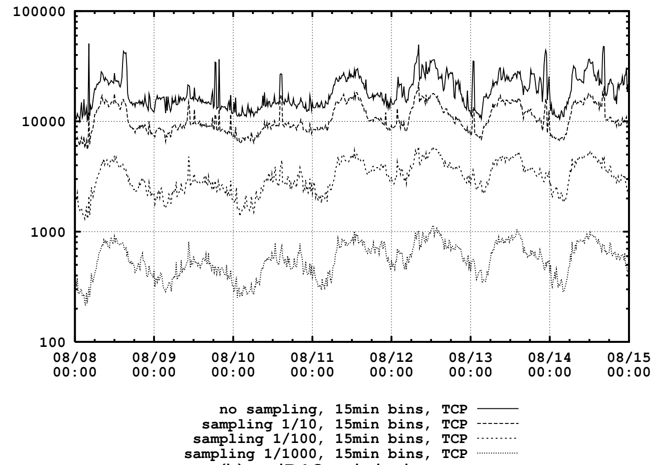


(d) swiIX1.switch.ch.

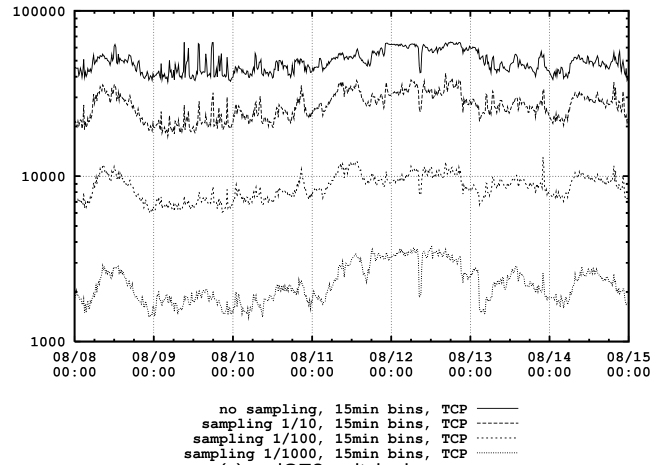
Figure B.5: Impact of Sampling, Counted Destination IPs, August 08-14, 2003



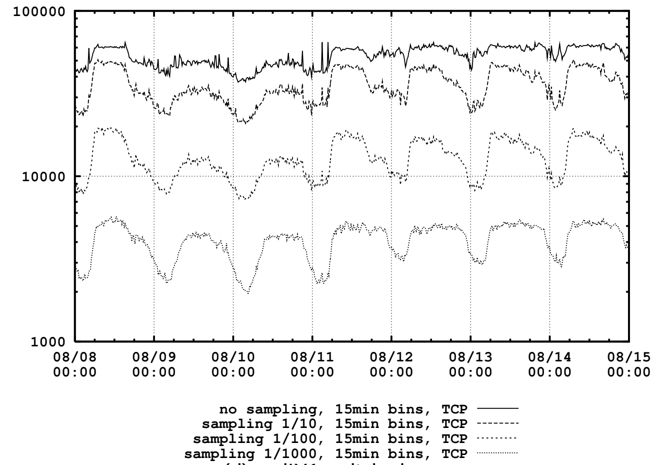
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

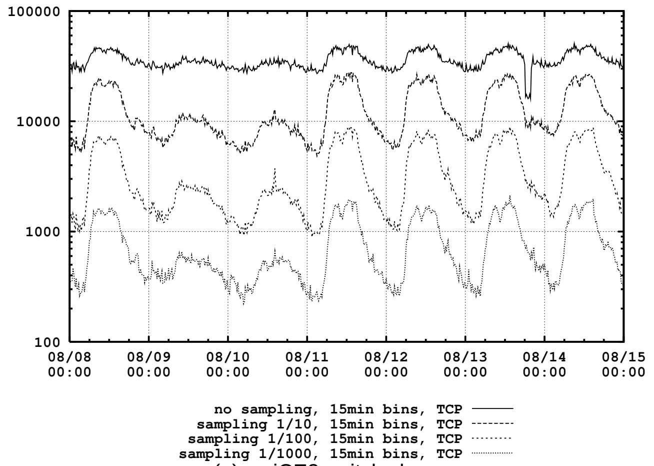


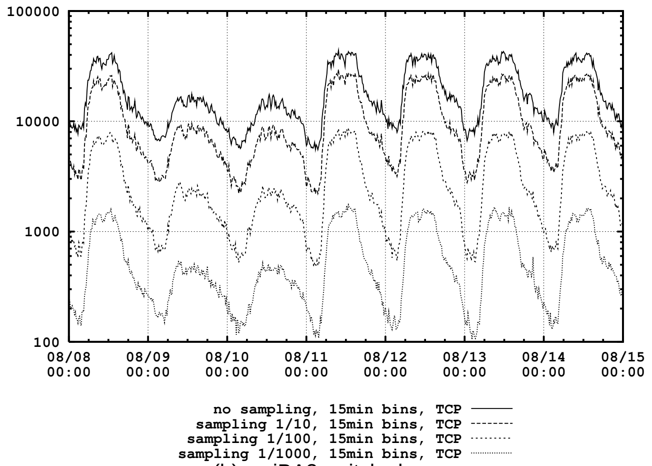
(c) swiCE3.switch.ch.

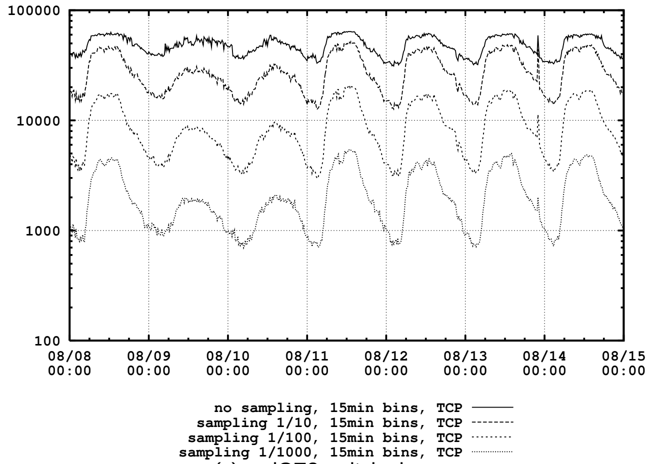


(d) swiIX1.switch.ch.

Figure B.6: Impact of Sampling, Counted Source Ports, August 08-14, 2003


(a) swiCE2.switch.ch.


(b) swiBA2.switch.ch.


(c) swiCE3.switch.ch.

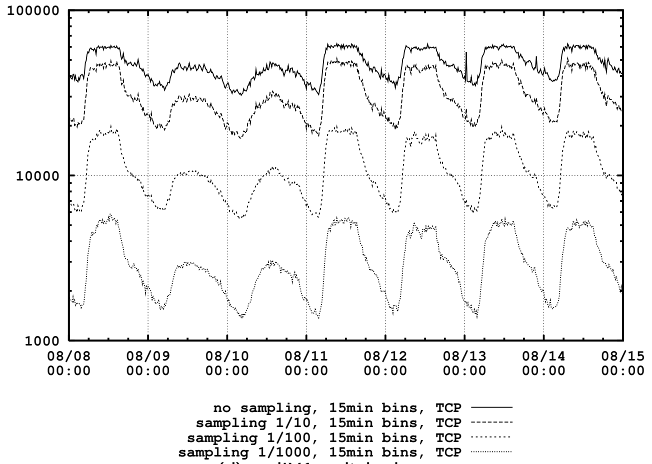
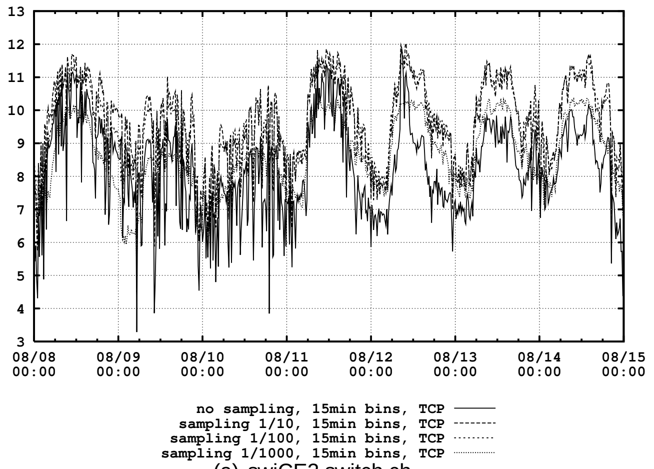
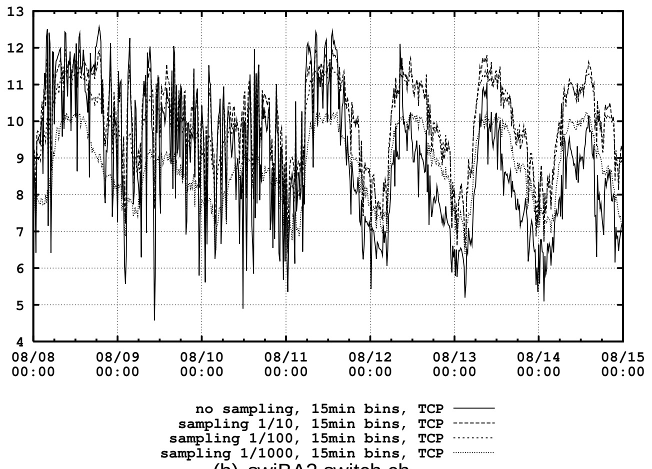

(d) swiIX1.switch.ch.

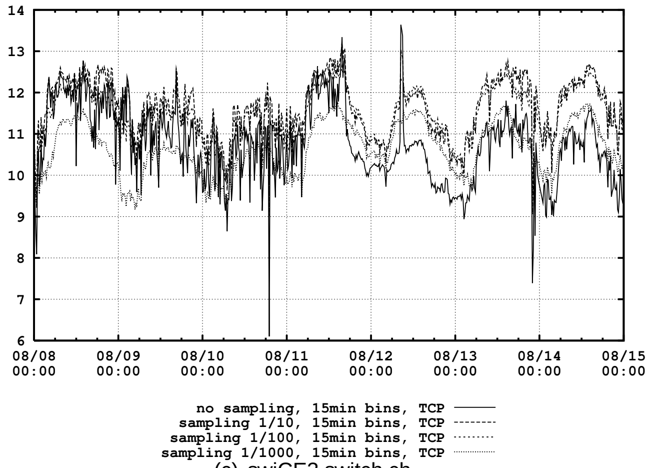
Figure B.7: Impact of Sampling, Counted Destination Ports, August 08-14, 2003



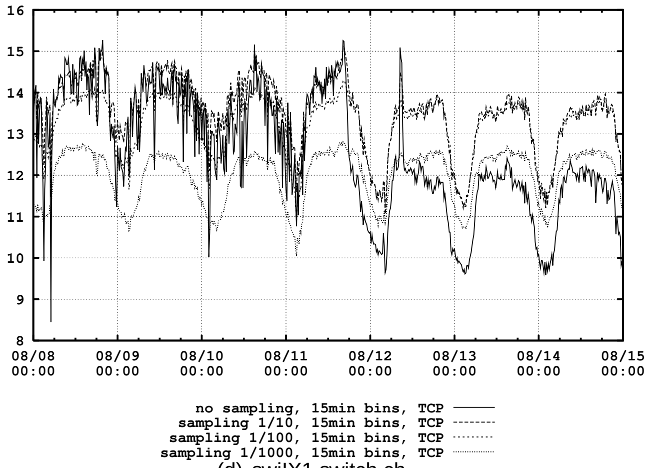
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

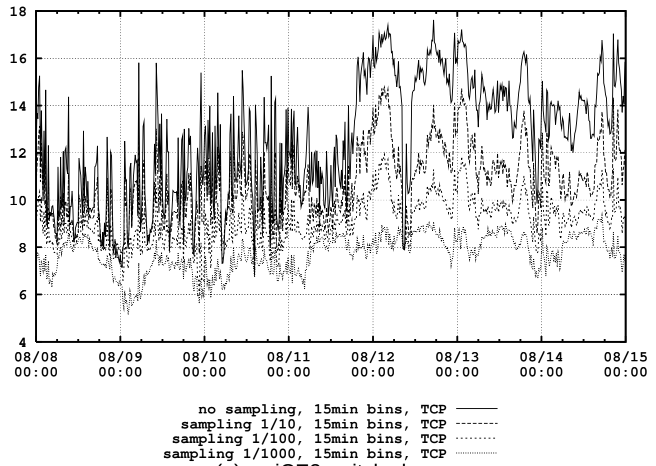


(c) swiCE3.switch.ch.

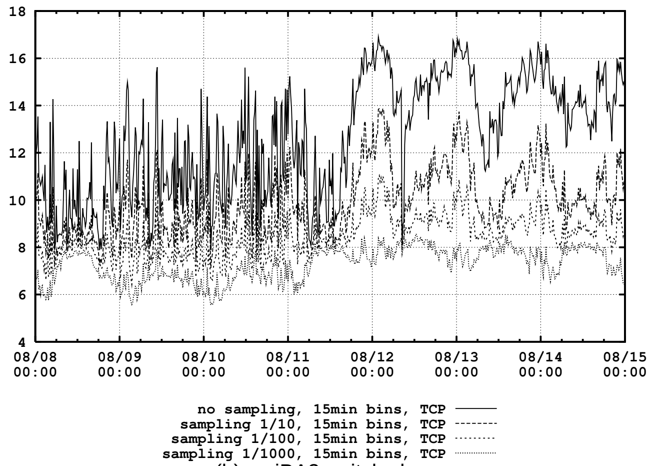


(d) swiIX1.switch.ch.

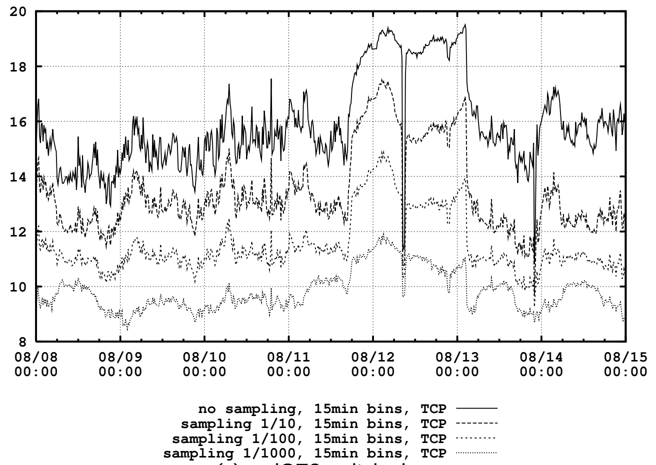
Figure B.8: Impact of Sampling, Flow Entropy of Source IPs, August 08-14, 2003



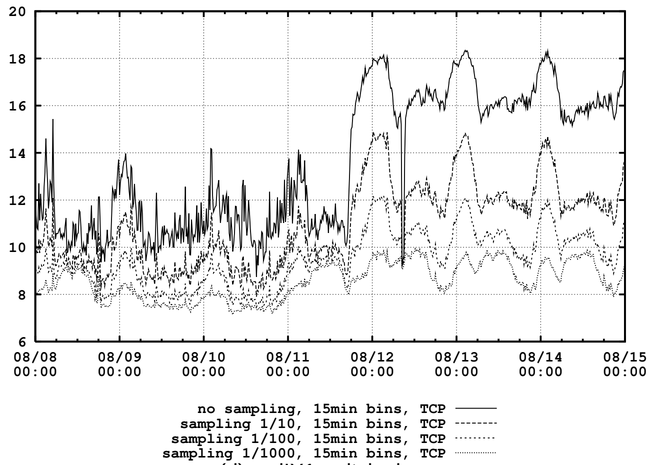
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

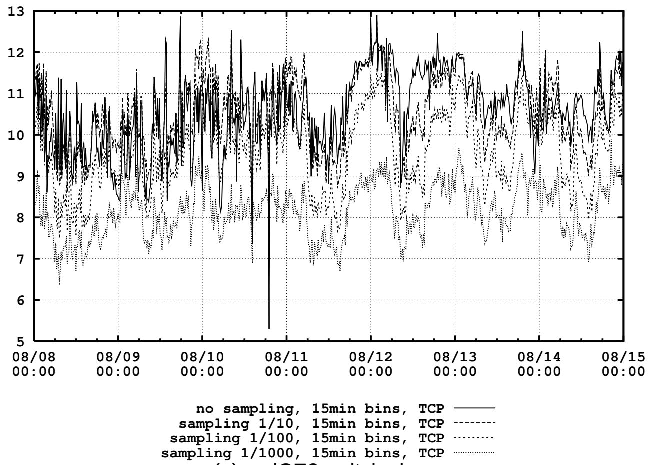


(c) swiCE3.switch.ch.

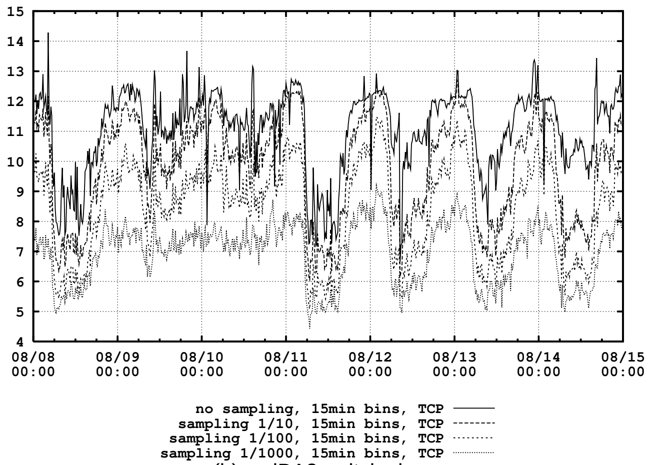


(d) swiIX1.switch.ch.

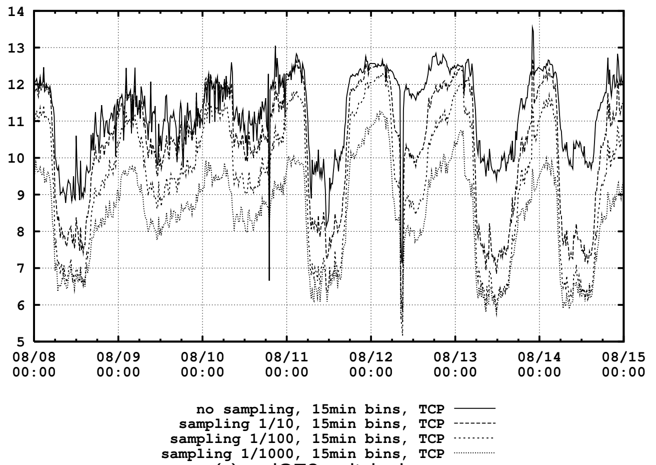
Figure B.9: Impact of Sampling, Flow Entropy of Destination IPs, August 08-14, 2003



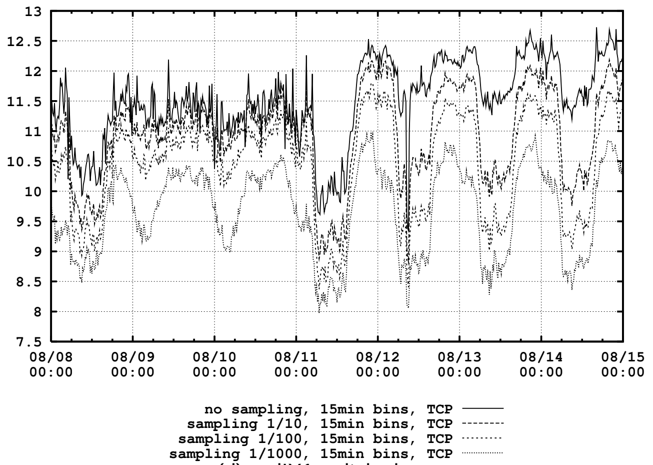
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

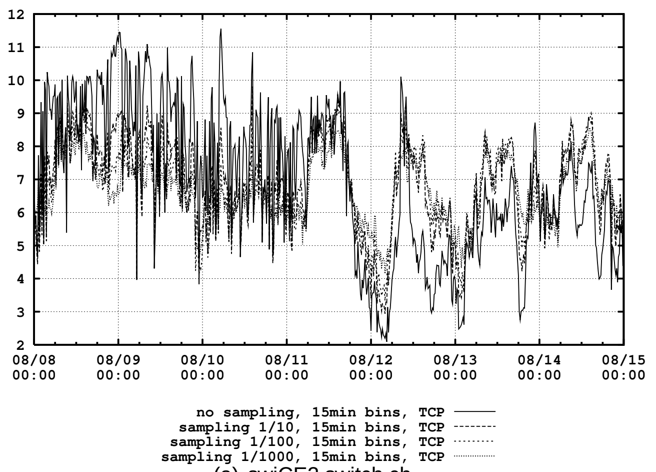


(c) swiCE3.switch.ch.

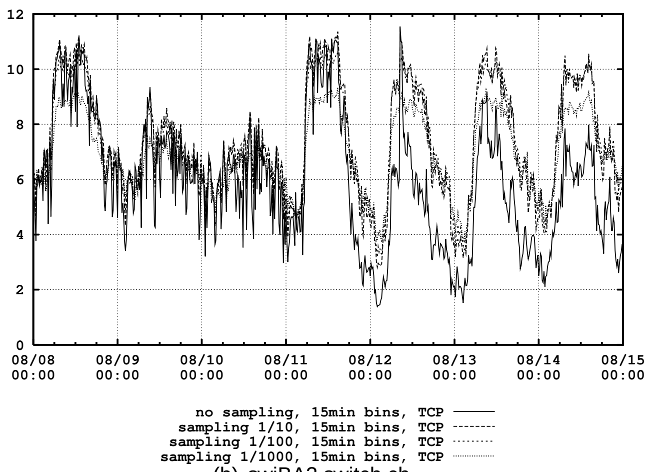


(d) swiIX1.switch.ch.

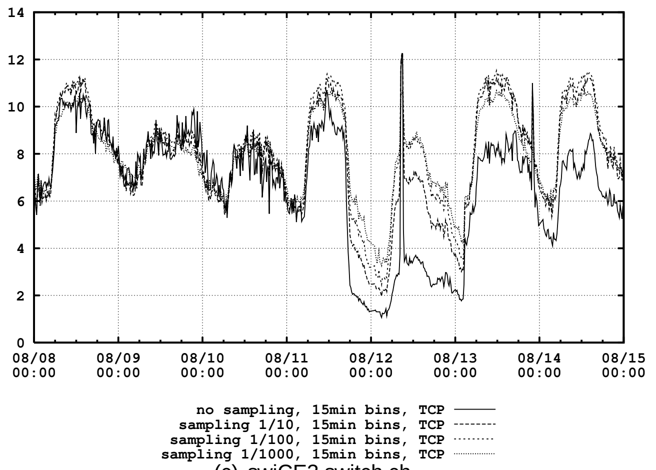
Figure B.10: Impact of Sampling, Flow Entropy of Source Ports, August 08-14, 2003



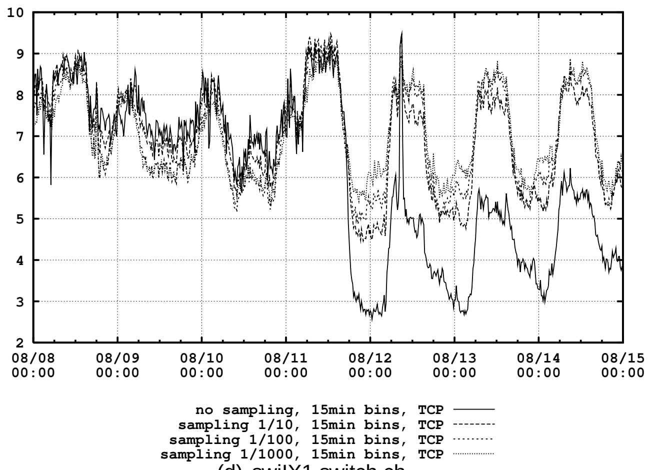
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

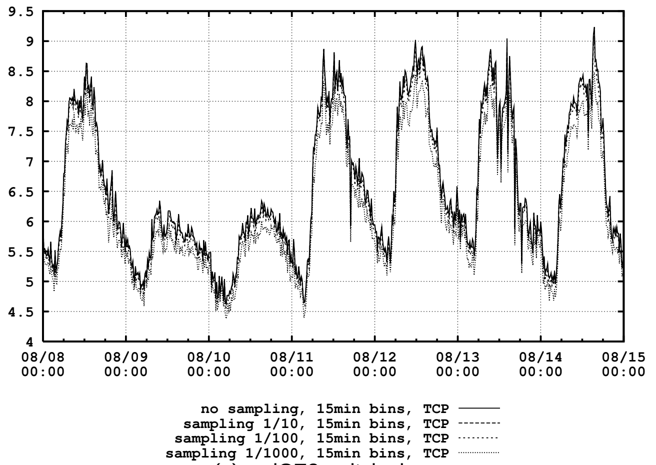


(c) swiCE3.switch.ch.

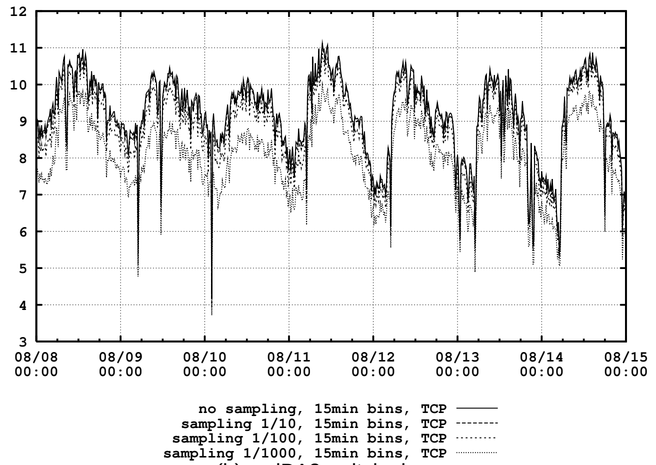


(d) swiIX1.switch.ch.

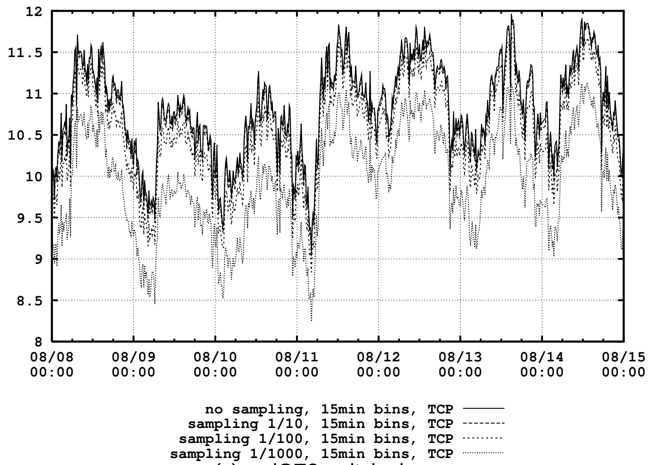
Figure B.11: Impact of Sampling, Flow Entropy of Destination Ports, August 08-14, 2003



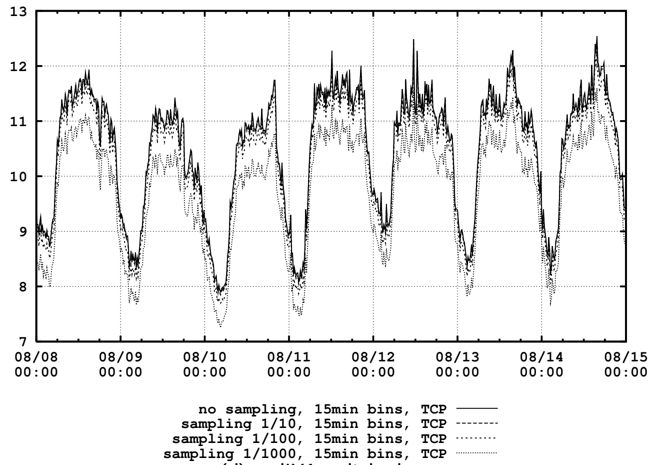
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

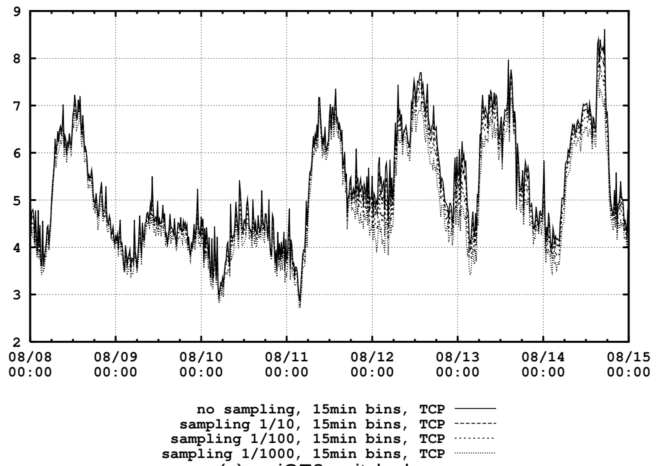


(c) swiCE3.switch.ch.

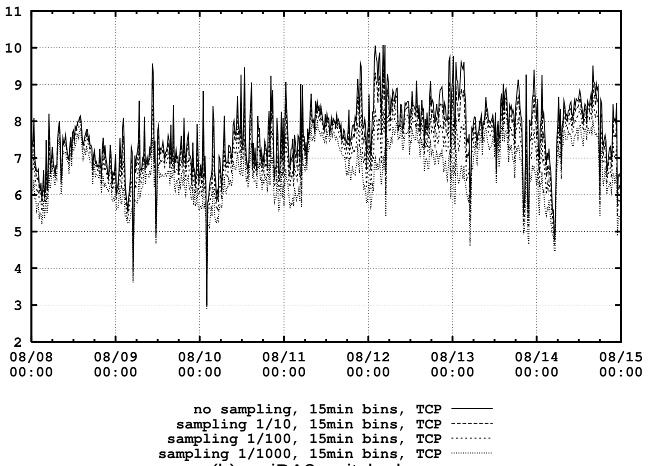


(d) swiIX1.switch.ch.

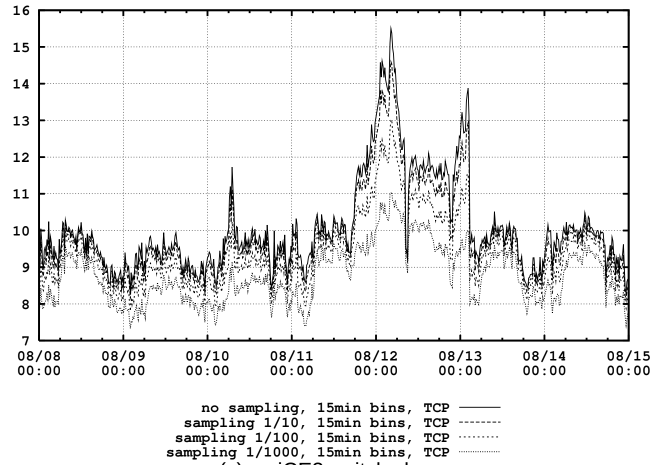
Figure B.12: Impact of Sampling, Packet Entropy of Source IPs, August 08-14, 2003



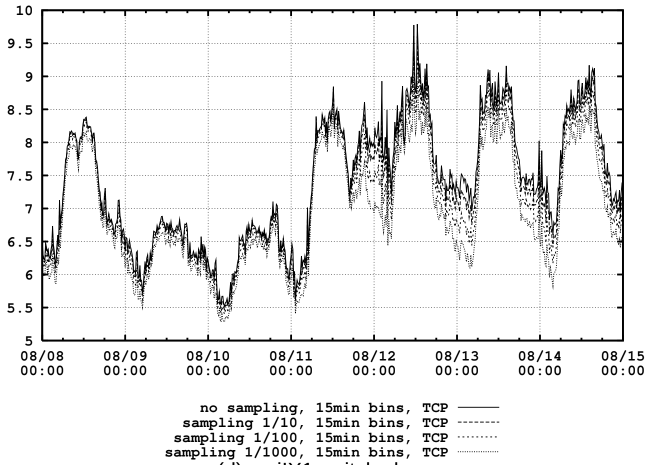
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

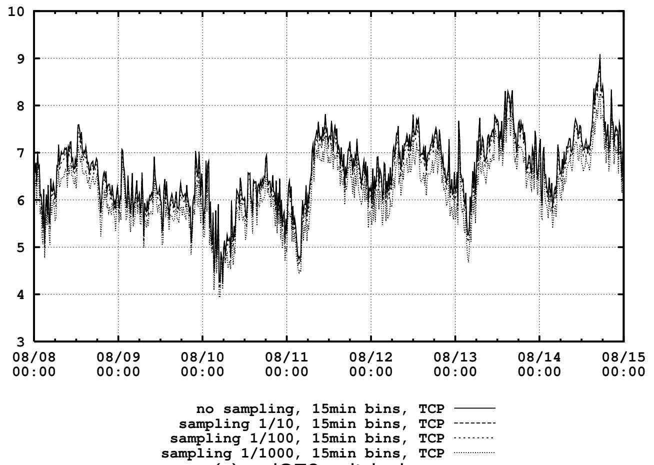


(c) swiCE3.switch.ch.

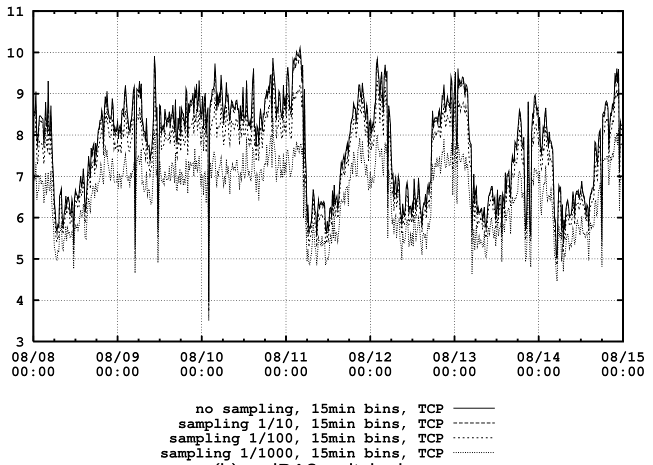


(d) swiIX1.switch.ch.

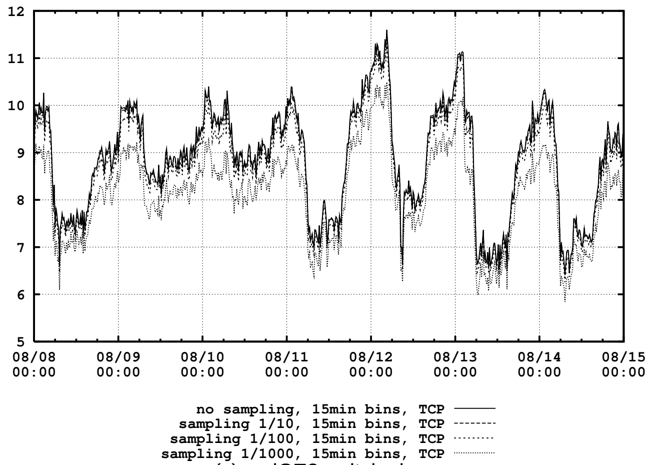
Figure B.13: Impact of Sampling, Packet Entropy of Destination IPs, August 08-14, 2003



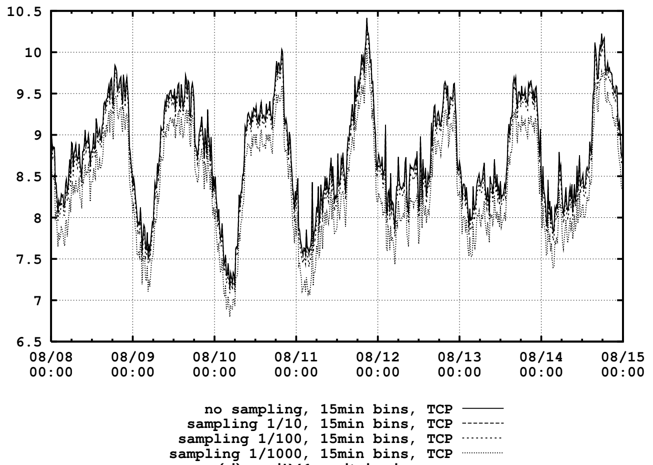
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.

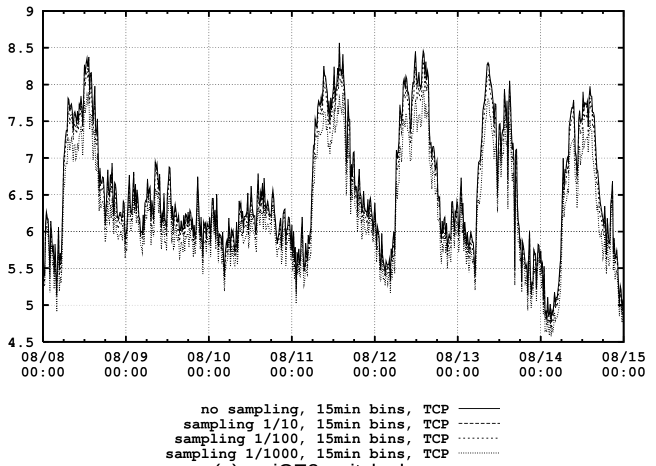


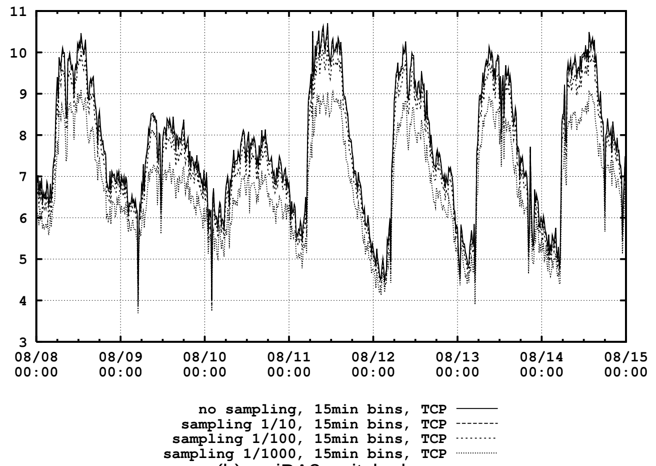
(c) swiCE3.switch.ch.

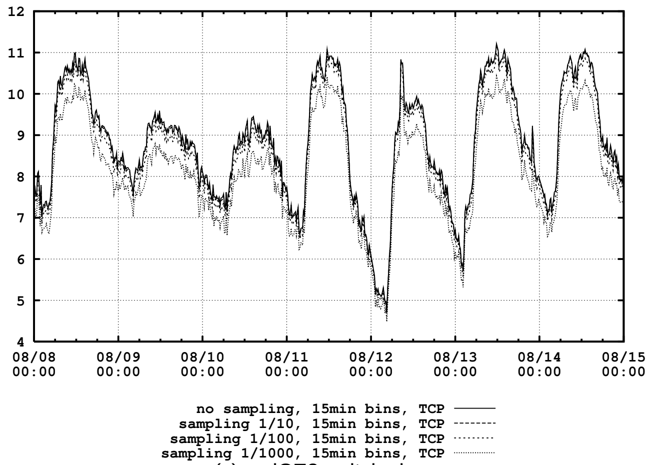


(d) swiIX1.switch.ch.

Figure B.14: Impact of Sampling, Packet Entropy of Source Ports, August 08-14, 2003


(a) swiCE2.switch.ch.


(b) swiBA2.switch.ch.


(c) swiCE3.switch.ch.

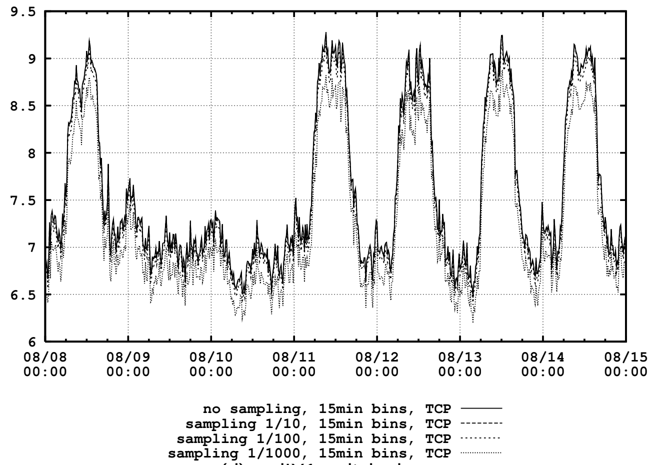
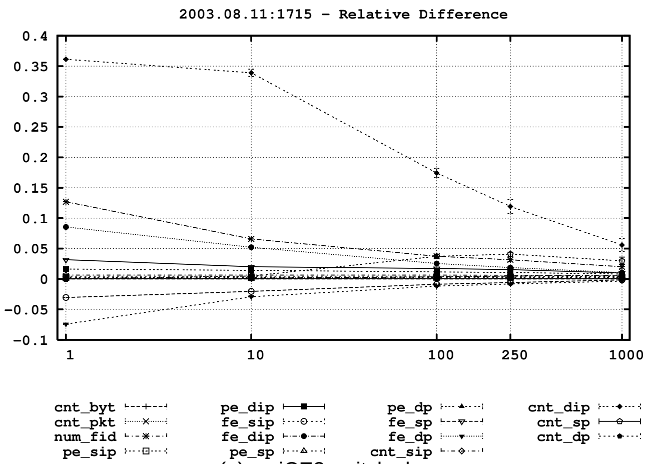

(d) swiIX1.switch.ch.

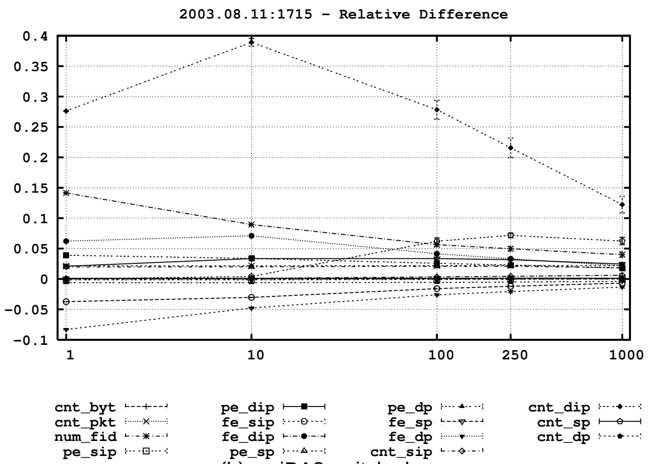
Figure B.15: Impact of Sampling, Packet Entropy of Destination Ports, August 08-14, 2003

Appendix C

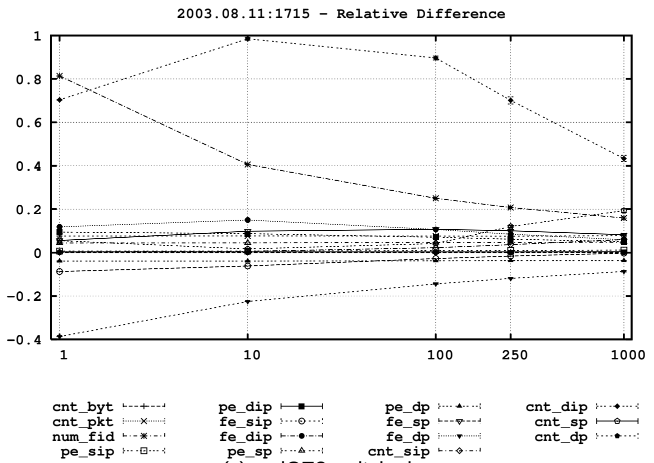
Relative Difference, Plots



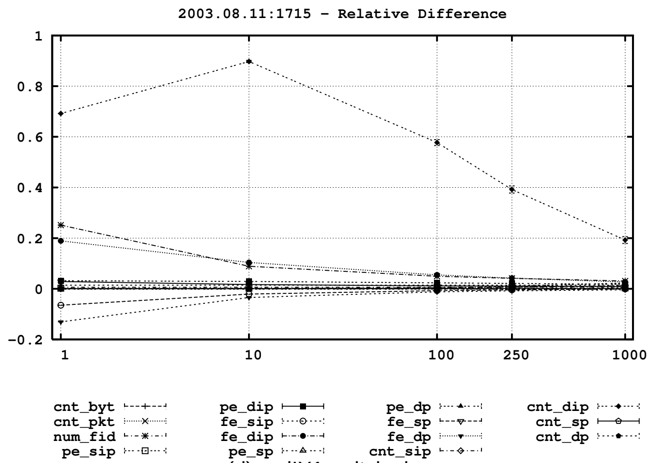
(a) swiCE2.switch.ch.



(b) swiBA2.switch.ch.



(c) swiCE3.switch.ch.



(d) swiIX1.switch.ch.

Figure C.1: Relative Difference of Baseline and Total Traffic

Appendix D

Abbreviations for the Measured Metrics

cnt_byt	Bytes per Interval
cnt_dip	Destination IP Count
cnt_dp	Destination Port Count
cnt_pkt	Packet Count
cnt_sip	Source IP Count
cnt_sp	Source Port Count
fe_dip	Flow Entropy of Destination IPs
fe_dp	Flow Entropy of Destination Ports
fe_sip	Flow Entropy of Source IPs
fe_sp	Flow Entropy of Source Ports
num_fid	Number of Flow-IDs
pe_dip	Packet Entropy of Destination IPs
pe_dp	Packet Entropy of Destination Ports
pe_sip	Packet Entropy of Source IPs
pe_sp	Packet Entropy of Source Ports

Appendix E

Schedule

Week	Calendar Week	Primary Task	Deliverables
01	44	Reading of Papers	
02	45	Cluster and code study	Schedule
03	46	Get the code running	
04	47	Logfile generation of one router for W32Blaster	
05	48	W32Blaster logfile analysis of the first router	
06	49	W32Blaster logfile analysis of the remaining routers	Table of contents
07	50	Interpretation of Results, graphical preparation	
08	51	Reserves	informational presentation
	52	Christmas holyday	
09	01	Preparation and evaluation of another anomaly	
10	02	Logfile generation of anomaly	
11	03	Logfile analysis	
12	04	Logfile analysis	
13	05	Interpretation of Results	
14	06	Reserves	Presentation / Report

Appendix F

Problem Statement

F.1 Introduction

In large backbone networks where high link speeds are common, security analyses (e.g. anomaly detection) are usually executed on flow data instead of packet data. Flow data (e.g. Cisco Netflow [7]) is faster to process than packet data since information is aggregated. But at the same time, aggregation means loss of information. Consequently, an information for speed trade-off is made here.

Very often, traffic data also needs to be *sampled* since routers cannot cope with the high data volume. Sampling is a selection process that captures only part of all data. The most commonly employed form of sampling in todays routers is random packet sampling, i.e., each packet arriving at a router's interface is captured with a certain probability or *sampling rate*. Common sampling rates are as high as 1 in 1000, or even higher for very fast backbone links.

This sampled backbone data is used in a variety different applications such as accounting, capacity planning, or anomaly detection [2, 4]. The question that arises here is whether sampling significantly impacts the result of these applications, e.g., the success and failure rates of anomaly detection systems. Today the common opinion towards sampling is that it considerably dwarfs anomaly signals [3] and should thus be avoided whenever possible. However, in a previous work [1] we have shown that anomaly visibility (i.e., the strength of the anomaly signal) in some detection metrics, such as feature distributions, is more resilient to sampling than others (e.g., packet counts).

F.2 The Task

This thesis is conducted at ETH Zurich. The task of this Semester Thesis is to analyse the visibility of anomaly signals in sampled traffic. The main focus is here to extend our developed methodology to include the view from multiple backbone routers of the SWITCH network, and to analyse and evaluate the information gain or loss that is associated with this extended view. Moreover, this extended visibility analysis should be applied to at least one other network anomaly. Additionally, if time permits, the detectability of anomalies in sampled traffic will be studied.

Blaster visibility analysis with data from multiple routers

The focus of this task is to compare and analyse the visibility of the Blaster anomaly in sampled traffic from multiple routers. In particular, we are interested in whether, why and how this extended view improves or disimproves the anomaly visibility in certain detection metrics. Metrics to be considered are packet/byte/flow counts, IPaddr/port entropy, IPaddr/port counts, and others if identified as interesting.

Visibility analysis for at least one more network anomaly

In this task, the same multi-router analysis should be carried out for other network anomalies. There are plenty of other network worms in the SWITCH data from which at least one is to be chosen and analysed. In order to use the same scripts as in the previous task, a modular software design is advisable.

Optional: Detectability analysis for one anomaly detection algorithm

If time permits, the student will try to analyse the detectability of a previously studied anomaly by implementing a simple anomaly detection mechanism, running experiments with different sampling rates and metrics and evaluating the obtained results.

F.3 Deliverables

The following results are expected:

- *Blaster visibility analysis across multiple routers.* A detailed analysis which compares the Blaster visibility in different detection metrics, in traffic from single routers and across multiple backbone routers, has to be conducted.
- *Visibility analysis of at least one more network anomaly.* A detailed, multi-router visibility analysis of at least one additional anomaly has to be conducted.
- *Documentation.* A concise description of the work conducted in this thesis (task, related work, problem statement, implementation, results and outlook). The analysis methodology for multiple routers as well as extensions for other anomalies, are part of this main documentation. The abstract of the documentation has to be written in both English and German. The original task description is to be put in the appendix of the documentation. One sample of the documentation needs to be delivered at TIK. The whole documentation, as well as the source code, slides of the talk etc., needs to be archived in a printable, respectively executable version on a CDROM, which is to be attached to the printed documentation.

F.4 Organizational Aspects

Documentation and Presentation

A documentation that states the steps conducted, lessons learnt, major results and an outlook on future work and unsolved problems has to be written. The code should be documented well enough such that it can be extended by another developer within reasonable time. At the end of the thesis, a presentation will have to be given at TIK that states the core tasks and results of this thesis. If important new research results are found, a paper might be written as an extract of the thesis and submitted to a computer network and security conference.

Dates

This Semester thesis starts on October 30th 2006 and is finished on February 9th 2007. It lasts 14 weeks in total. At the end of the second week Jonathan has to provide a schedule for the thesis. It will be discussed with the supervisors.
After six weeks Jonathan should provide a draft of the table of contents (ToC) of the thesis. The ToC suggests that the documentation is being written as the work progresses.
An intermediate informal presentation for Prof. Plattner and all supervisors will be scheduled 8 weeks into this thesis.
A final presentation at TIK will be scheduled close to the completion date of the thesis. The presentation consists of a 10 minute talk and reserves 5 minutes for questions. Informal meetings with the supervisors will be announced and organized on demand.

Supervisors

Daniela Brauckhoff, brauckhoff@tik.ee.ethz.ch, +41 44 632 70 50, ETZ G97
Bernhard Tellenbach, tellenbach@tik.ee.ethz.ch, +41 1 632 70 06, ETZ G97

Bibliography

[1] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Anukool Lakhina, and Martin May. Impact of Packet Sampling on Anomaly Detection Metrics. In IMC 2006: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, 2006.

[2] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining Anomalies Using Traffic Feature Distributions. In Proceedings of ACM SIGCOMM 2005, August 2005.

[3] Jianning Mai, Ashwin Sridharan, Chen-Nee Chuah, Hui Zang, and Tao Ye. Impact of packet sampling on portscan detection. 2006.

[4] Arno Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast IP networks. In WET ICE 2005, Linköping, Sweden, 2005.

[5] Choi, B.-Y., Park, J., and Zhang, Z.-L. Adaptive random sampling for total load estimation. In IEEE International Conference on Communications 2003.

[6] Wallerich, J., Dreger, H., Feldmann, A., Krishnamurthy, B., and Willinger, W. A methodology for studying persistency aspects of internet flows. SIGCOMM Comput. Commun. Rev. 35, 2005.

[7] Cisco Systems Inc. NetFlow White Papers. `http://www.cisco.com/en/US/products/ps6601/prod_white_papers_list.html`, 2007

[8] SWITCH. Swiss academic and research network. `http://www.switch.ch`, 2007

[9] `http://www.csg.ethz.ch/`, 2007

[10] `http://www.ethz.ch/`, 2007

[11] `http://www.tik.ee.ethz.ch/~ddosvax/cluster/`, 2007