

S-38.3184 Assignment: From Traffic Measurements to Conclusions

Boris Nechaev

boris.nechaev@hiit.fi

Helsinki Institute for Information Technology HIIT

May 17, 2009

1 Introduction

Network measurements and analysis of network activity is an important undertaking leading to valuable contributions both to the research community and to industrial sector. For instance ISPs are constantly seeking ways to optimize the network resources and perform efficient traffic engineering. However, network measurements is not only a paying-back undertaking, but unfortunately also a complicated one. In [8] authors describe the main pitfalls awaiting those who are trying to simulate the Internet and study its properties. Nevertheless, experienced researchers share their knowledge of how to pursue this uneasy task [14]. In this paper we pursue the basic analysis of two data sets aiming at investigating several simple aspects of the traffic found in them.

2 Data set 1: Flow data

In this section we investigate the data provided by the course staff. The data is in the `cr1_flow` [1] format which consists of two parts: (1) various metadata, e.g. trace start timestamp, trace duration, total number of packets, etc; (2) reconstructed IP flows data represented by 11 columns, e.g. source IP, destination IP, number of bytes in the flow, etc. The metadata is helpful to get a general understanding of the data in the trace, but it precludes the use of statistical packages (e.g. `R`). Specifically, `R` can't read the a file with non-uniform format. For this reason for some of our analyses we had to exclude all the metadata lines starting with '#' symbol as well as empty lines. This results in a uniform (11 columns) format of the data under analysis.

The data set consists of 168 files each with duration of 1 hour. Data capturing started 13 Apr 2008 at 22:00 GMT. We have assessed that all the files are directly following each other in time, i.e. there are no time gaps between files. Thus we can conclude that the data spans 168 hours or 7 full days. The total number of flows in the data set is 17,317,418.

2.1 Traffic volume

The first goal we have pursued is the analysis of traffic volume as a function of time. The fact that the files in the data set are contiguous allows us investigate patterns of traffic volume changes up to within one week. The number of bytes per hour can be readily found in the metadata.

Figure 1(a) plots the traffic volume as a function of time. Earlier work on methodology of traffic analysis strongly encourages the search for *invariants* one of which is diurnal patterns [8]. These can be easily observed in the plot which has vivid sinusoidal-type form. The spikes correspond to busy hours, while the falls are most probably night or lunch hours. In Figure 1(a) it's easy to see 7 rough spikes of activity that can be plausibly attributed to 7 days incorporated by the data set. We have also tried to validate if there is a week-scale pattern, which can exhibit smaller amount of traffic on weekends and bigger amounts during weekdays. Towards this goal we divided the whole data set into 7 equal contiguous parts (days) each 24 hours long. We calculated average number of bytes per day, but failed to see any easily distinguishable pattern. The number of bytes per day seemed to be randomly fluctuating.

Further we tried to scale down to traffic patterns within one day. For each of the 7 days we constructed plots similar to Figure 1(b) which represents day 1. Even though we don't know in which time zone the traces were recorded, we can nevertheless deduce that they start close to midnight. This hypothesis is supported by the steep slope which hits the bottom of almost no traffic in the left side of the plot. Presumably this hollow corresponds to

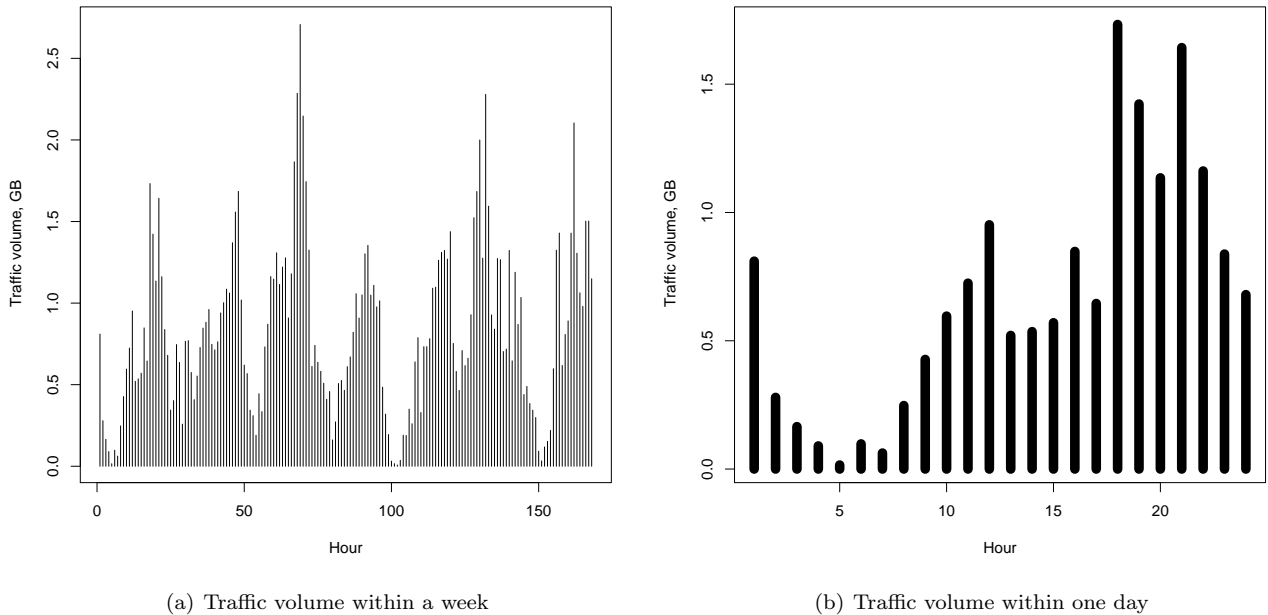


Figure 1: Traffic volume as a function of time.

night time. Another slight drop in the traffic volume found in the center of the plot might be caused by lunch time. Other day-long plots in general follow the same pattern.

2.2 Breakdown by port number

Studying applications and services used in a network is of particular interest to network operators. The easiest way to conduct this study is to consider transport layer port numbers. The two port numbers seen in TCP and UDP packets are a service port number and an ephemeral port. The service port is a destination port found in the TCP SYN packet (or first packet for UDP). Usually it's a well known port that can be used to identify application unambiguously. Ephemeral ports are chosen by the client randomly [3] and are therefore of little interest. Each flow in the data set has two ports - source and destination. Thus, there is a non-trivial problem of deciding which port in the flow is the service port. We have implemented a simple heuristic inspired by our previous experience in traffic analysis for choosing service ports. Firstly we check if the first port's value is less than 1024 or if it is in the list of well known ports and consider this to be the service port if either is true. Secondly, if neither is true, we check the same for the second port. Finally if none of the ports is less than 1024 nor in the list of well known ports, we choose the smaller one since there is a higher probability that it happens to be the service port. We borrowed a short list of well known ports from Bro [13]. Thus, after applying this heuristic to all the flows in the data set we get a list of service ports which we can use to assess applications and services mix in the data.

Using the service port numbers as input we construct frequency tables for them and study the top ones. Surprisingly, even after applying the above heuristic we see unusual, undoubtedly ephemeral, service ports (e.g. 17533 or 37156) having many occurrences in the data. We propose two theories for explaining the frequently seen non-well known ports. First is the wide spread of Internet worms, botnets and other malicious network programs that can use some of these ports for communication [12]. Second is the use of peer-to-peer file sharing applications that can initiate TCP connections with both ports being ephemeral. We exclude these confusing ports from further analysis since for them we can't deduce anything meaningful about the application in use.

Another weirdness in the top of port frequency table is ports 0, 1 and 3. Normally one wouldn't expect to see these in the Internet. Indeed, our manual inspection of the traces showed that these ports are appearing in flows that have IP protocol value equal to 1 which corresponds to ICMP. We do not know why `crl_flow` outputs port numbers for ICMP packets and thus exclude these ports from our analysis.

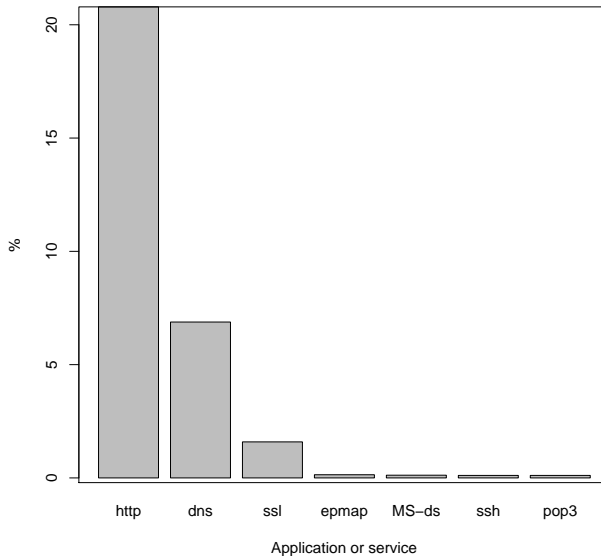


Figure 2: Distribution of port numbers across all flows.

Finally for the remaining top port numbers we provide a barplot (see Figure 2). For mapping ports to applications and services we used IANA’s port assignment catalog [2]. As expected, the seven most used application level protocols are HTTP and DNS with SSL/TLS coming third. Epmap corresponds to port number 135 and MS-ds (microsoft-ds) to port 445. We were surprised to see that SSH traffic which is used mostly by IT professionals is not very rear in the traces. In Section 2.6 we combine this finding with another observation to make a guess on the capturing vantage point location.

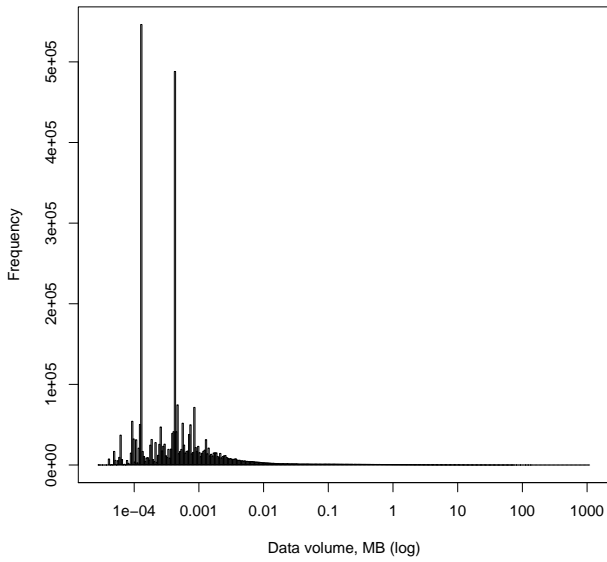
2.3 Origin-destination pairs analysis

In this section we study the amount of data flowing between pairs of IP addresses (origin-destination). As a first step for each origin-destination pair we computed the number of flows and the sum of bytes transferred. Note, that origin and destination IPs can flip their positions from flow to flow. E.g. first we observe a connection initiated by IP_1 and therefore the $IP_1 \rightarrow IP_2$ flow and later a connection initiated in the opposite direction resulting in the $IP_2 \rightarrow IP_1$ flow. Our script takes this into account and counts both directions as a single origin-destination pair. More specifically, we use a hashtable to store the number of flows for each pair. For the flow under consideration we check if $hash(IP_1, IP_2)$ is in the hashtable, and if it is not, we also check $hash(IP_2, IP_1)$. To sum up, our script outputs the number of flows for each origin-destination pair and the total number of bytes transferred between the two hosts. The total number of unique origin-destination pairs found in the data set is 2,933,241.

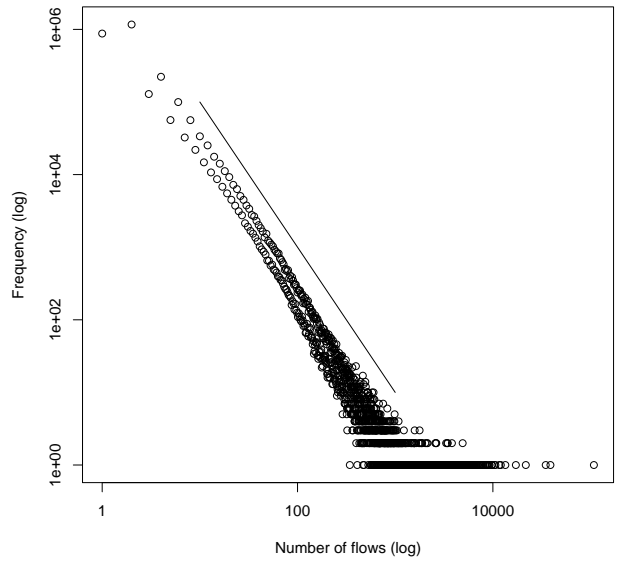
For the total number of bytes transferred between each set of IPs, we plot a histogram with x-axis in log-scale (see Figure 3(a)). The distribution has a heavy tail which is common for network traffic [16, 18]. For the number of flows per origin-destination pair we built a so called Zipf-type plot (see Figure 3(b)). These kind of plots depict the dependency between number of occurrences (x axis in our case) of a variable and its rank in the frequency table (y axis). The Zipf’s law basically says that for each following rank the number of occurrences is β times less than for the previous rank. Due to the high variability of data Zipf-type plots are produced with both axes in logarithmic scale. The fitting line in our plot corresponds to $y \sim \frac{1}{x^2}$. The Zipf’s law is also very widespread in network measurements [6, 4].

2.4 User-aggregated data volume

Further we step down in our analysis from origin-destination pairs to separate users. In our notion a user is equivalent to a single IP seen in the traces, however this assumption not always holds in real life because of NATs, multihomed



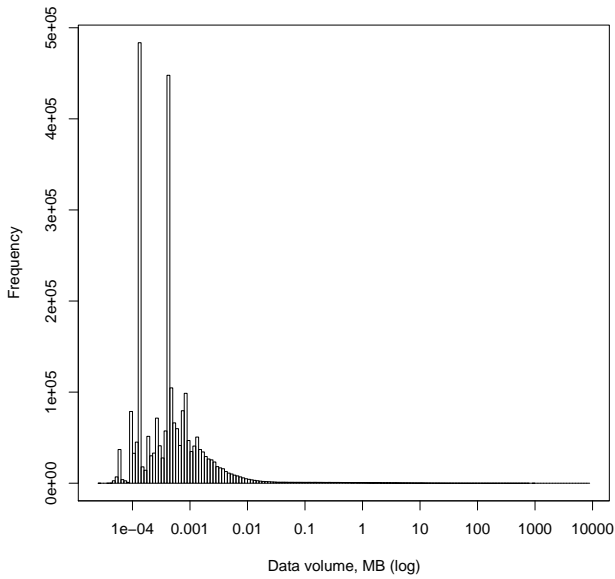
(a) Histogram of data volumes



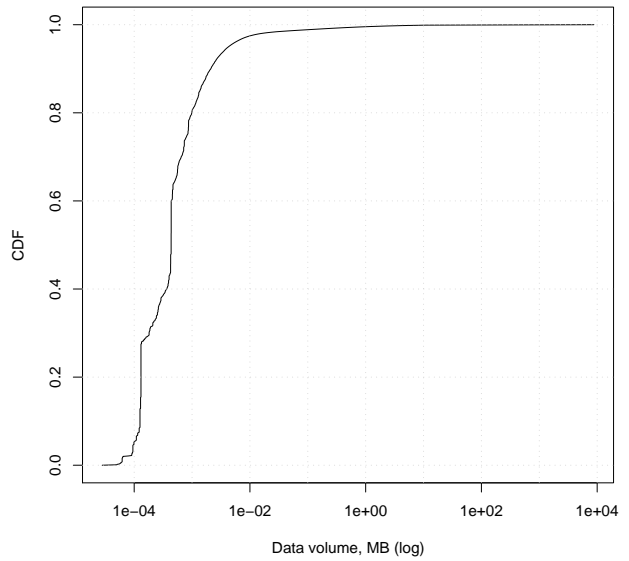
(b) Zipf-type plot for number of flows with fitting line

Figure 3: Origin-destination pairs analysis.

hosts and a single user possessing several machines. Note, that each flow has two distinct IP addresses, thus involving two users in our notion. Hence when calculating the total number of bytes per user, we attribute the amount of bytes in the flow to both of the IP addresses.



(a) Histogram of data volumes



(b) CDF of data volumes

Figure 4: Data volume per user.

Figure 4 shows the distribution and CDF of user-aggregated data volumes. Not surprisingly the Figures 3(a) and 4(a)

look very similar in shape and value ranges.

2.5 Flow length distribution

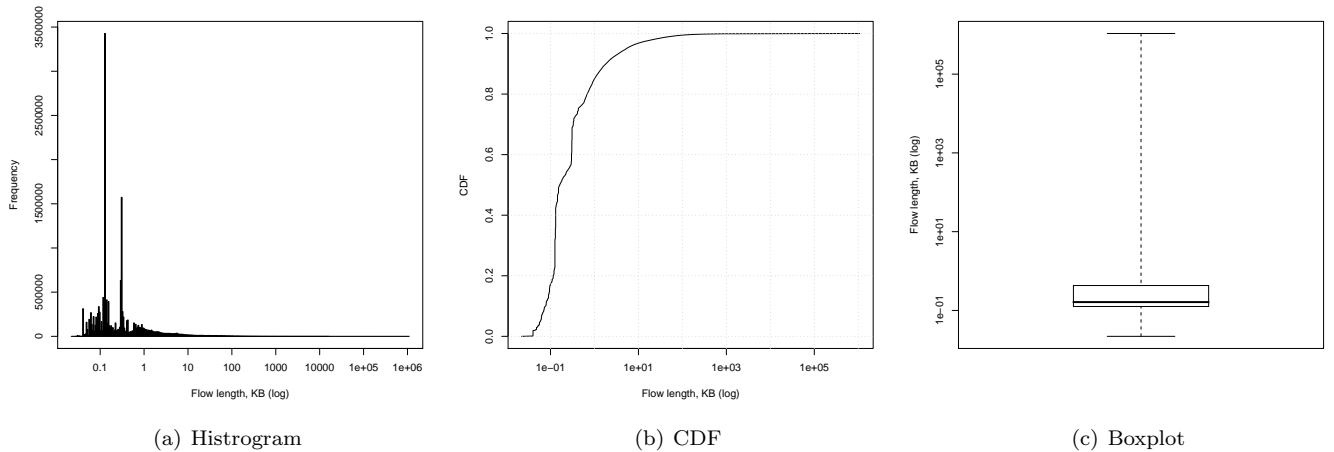


Figure 5: Flow length distribution.

In this section we investigate the flow length distribution and try to fit a theoretical probability distribution to it. The plots showing the flow length distribution, its empirical cumulative distribution function and key summary statistics in the form of a boxplot are shown in Figure 5. We have tried to fit a theoretical distribution to the flow lengths data employing **R** builtin functions for that purpose. The distributions we tried were exponential, gamma, log-normal and weibull. The **R** package estimated exponential distribution to fit with rate $1.23e - 04$, gamma with shape 3.32 and rate 0.01, log-normal with meanlog 5.646 and sdlog 1.4396, weibull with shape $4.664e - 01$ and scale $6.39e + 02$. We have tried to validate the above fitted distributions by plotting their probability distribution functions and respective qq-plots. We found that neither of the distributions fit the flow length data till any satisfactory degree. Presumably this happens because of the two outlying bins having high frequencies at flow lengths of 150 and 500 bytes (see Figure 5(a)). We could either try to remove these outlier bins or cut the heavy tail of the distribution and fit a theoretical distribution to the remaining data, but we believe that there is no much value in such a handicapped fitting. We conclude, that in analyzing network measurements data it is usually hard to find a properly fitting distribution.

2.6 Flow rates

As a last step in analyzing the data set we study flow rates seen in the data. Our definition of a flow rate is straightforward - flow size divided by flow duration. However, single packet flows have a zero duration by definition which makes the flow rate undefined for them. To avoid this complication and also to get rid of the too short flows that most probably didn't have a chance to accommodate large enough bulk data transfers, we consider only the flows with duration $> 1sec$. The distribution of flow rates in log scale can be found in Figure 6(a). We found that the median flow rate is 48 MByte/s, however an obvious spike at about 15 MByte/s must be the real throughput of the link where the capturing apparatus was installed. A bandwidth of about 120 Mb/s is usually considered too big for a private household, but at the same time too small for an ISP peering link. Such a bandwidth best matches a medium-size enterprise or department. Recalling that we earlier observed the use of SSH in the traces, we make a guess that these data were captured in a university department or laboratory.

In [5] authors found that the rate distributions are well modeled as log-normal. Thus we also tried to fit the logarithm of rates with a normal distribution. Even though the qq-plot in Figure 6(c) doesn't provide a strictly straight line, we can nevertheless conclude that the rate distribution observed in our data agrees with that of the authors.

Finally, we assess correlations between duration, rate and size of flows. We are forced to use logarithms of the above values because of high variability of the data. Table 1 presents the three possible combinations. We found

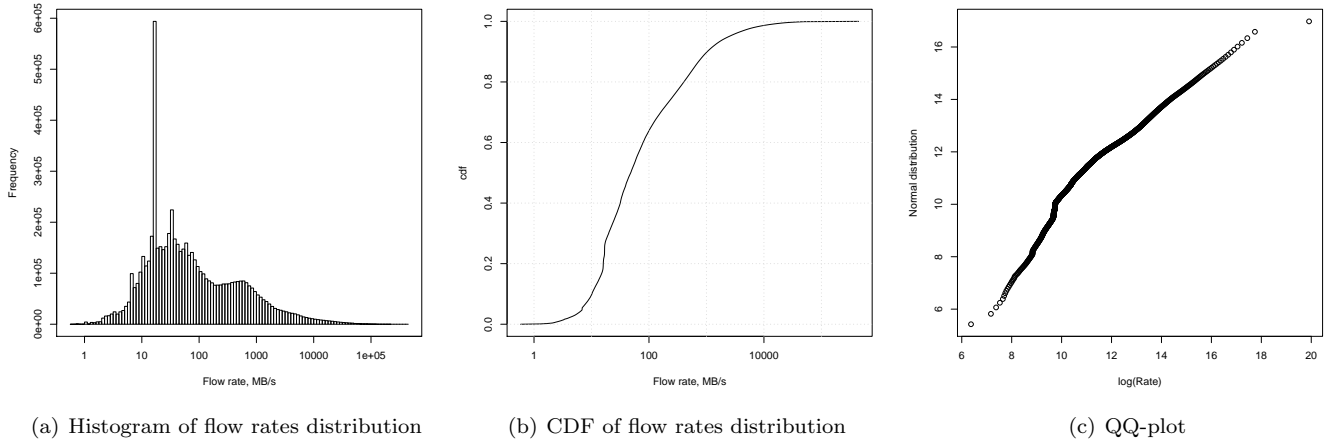


Figure 6: Flow rates.

Pair	Corr. coef.
$\log(\text{Dur}), \log(\text{Rate})$	-0.458
$\log(\text{Size}), \log(\text{Rate})$	0.739
$\log(\text{Dur}), \log(\text{Size})$	0.259

Table 1: Correlations between flow duration, rate and size.

that our results are in nearly-perfect agreement with the ones discovered in previous studies. In [19] authors also observed an utterly striking phenomena of high positive correlation between flow rate and size. We agree with the authors hypothesis that this is due to the fact that users tend to choose the size of the performed transfers based on the available bandwidth, thus being conservative if the bandwidth is low and willing to send big blobs of data if the available connection speed is high.

3 Data set 2: Captured data

In this section we analyze the data collected from a private laptop connected to a home wireless network. The data was captured under Linux operating system using the well known **tcpdump** tool which stored first 96 bytes of each packet. In total, we have collected three trace files. To preserve privacy we further anonymize traces with the **crl_to_pcap** tool. However, after applying the anonymization procedure to each separate trace file, we have found that this yields inconsistent set of anonymized IPs in different traces. For instance in the first anonymized trace the IP of the host where monitoring was performed happened to be 60.87.159.189, while in another trace - 153.105.243.189. Such a mismatch would severely skew our further analysis since it would seem that three distinct hosts were monitored. Thus, to preserve consistency, we merged all three traces and applied the anonymization procedure to the resulting file. For merging and printing the start and end timestamps we used **tcpslice** [7] which can be found in most Linux and FreeBSD repositories. Note, that during capturing we didn't specify any **tcpdump** flags, thus all packets seen to the host have been captured. However, **crl_to_pcap** is capable of traversing only IP packets, truncating all non-IP ones. Hence for all the further analyses we use only anonymized IP packets. The general information about our traces is summarized in Table 2.

Even though merging traces proved to be a beneficial technique during anonymization process, it on the other hand complicates some of the other analyses. For instance, there is a two-days gap between the first and the second trace, which will appear as zero activity area in the traffic volume as a function of time plot. Thus using **tcpslice** we split the merged anonimized trace file into the three files contemporary with the original (non-anonymized) ones. In our analysis we use both merged and separate traces.

Further we pursue the analysis of the above data from three different perspectives, basically meaning three different preprocessing steps and granularity levels — *packets*, *flows* and *TCP connections*.

Trace	Date	Start time	Duration	# IP packets	# !IP packets
1	07 May 2009	08:37	2h 38m	62,045	64
2	09 May 2009	12:08	12h 14m	63,394	1,286
3	10 May 2009	09:39	6h 25m	85,970	627
Merged	07-10 May 2009	08:37	3d 7h 27m	211,409	1,977

Table 2: Captured traces summary.

3.1 Packet data

The first perspective is the per-packet characteristics of the data. First we preprocess the data using our custom application written in C. The application uses the well known **libpcap** library and is thus capable of working directly with the traces recorded by **tcpdump**. For each packet our application outputs the following 4-tuple: unix timestamp, source port, destination port, packet length. For transport layer protocols other than TCP and UDP (e.g. ICMP) we output "-" in the ports columns. The packet length is taken from the IP header, meaning that 14 bytes of Ethernet header don't contribute to it.

3.1.1 Packet distribution by port

Similarly to Section 2.2 we make a breakdown by ports found in the traces. We again use the same service port identification technique which helps us to map individual packets to the services and applications used. The seven most frequent application layer protocols and ports are shown in Figure 7.

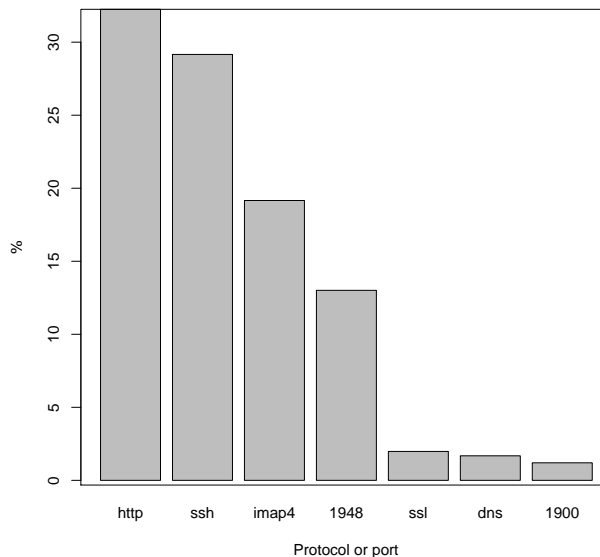


Figure 7: Top ports seen in the traces.

We indeed can confirm that during the packet capture web browsing, sending e-mails and use of SSH took place. However, we are surprised by the presence of ports 1948 and 1900 at the top of the frequency table. Investigating what processes on the monitored laptop use these port is a matter of future work.

3.1.2 Traffic volume

In this section we study the traffic volume as a function of time. We choose 5 minutes as a resolution bin. To obtain the number of bytes transferred within the chosen interval, we use a simple script that traverses the packets in the

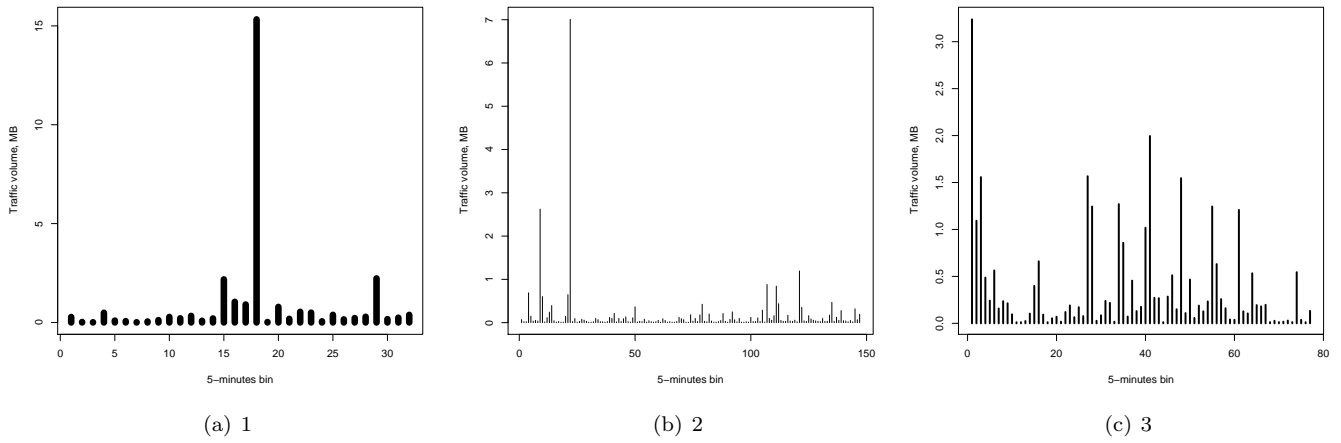


Figure 8: Traffic volume per 5 minutes.

4-tuple format described above and depending on the packet's timestamp maps it to a bin. The total number of bytes per 5-minute bin is the sum of packets lengths in the bin. Note, that in this analysis we use separate traces which results in three plots (see Figure 8). The plots don't show any discernible patterns other than the fact that the laptop was used irregularly and with frequent breaks.

3.1.3 Packet length

Next we study the distribution of packet lengths. Figure 9(a) plots the histogram of the respective distribution. Unfortunately, we found it impossible to plot the distribution using bins of width 1 byte because in this case the number of bins becomes bigger than the width of figure plotting area in pixels. Obviously, one can't plot a fraction of a pixel. Nevertheless, to study the most commonly seen packet lengths we provide Figure 9(b) which shows top

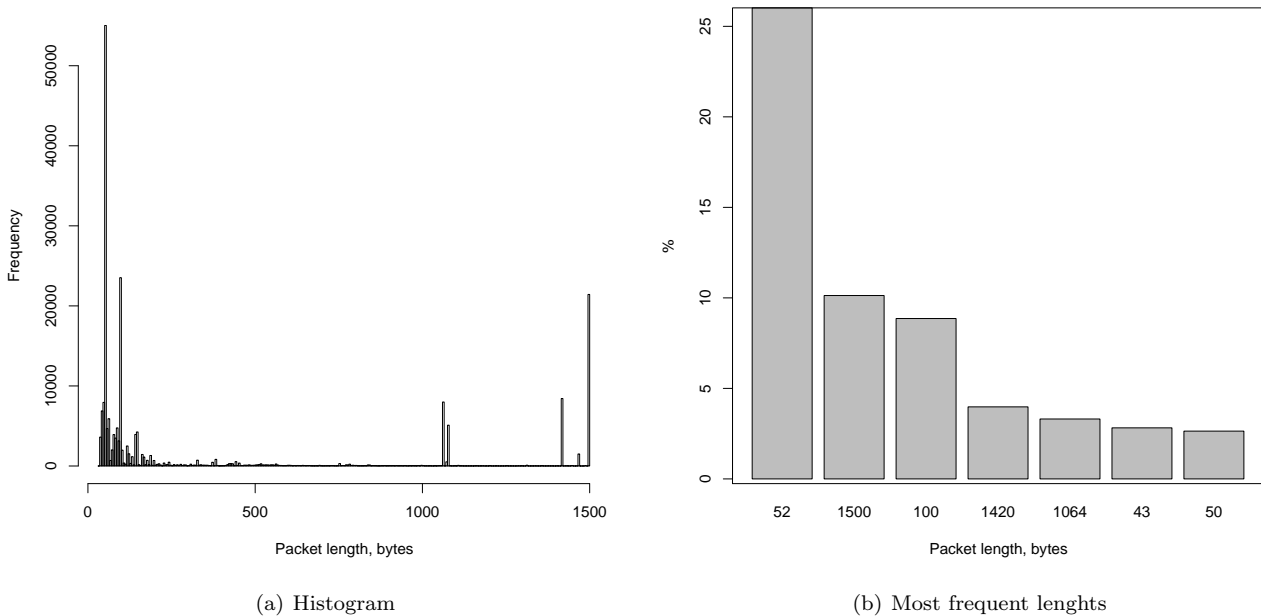


Figure 9: Packet length distribution.

seven most frequent values. The first value (52 bytes) undoubtedly refers to TCP pure ACKs¹ that are very frequent in any traces. The second value is also self-explaining - 1500 bytes is the maximum size of IP packet in Ethernet. We found that packets with the length of 100 bytes most commonly appear in SSH communication, presumably its some form of heartbeat or acknowledgment. 1420 bytes must be the MSS size chosen by some hosts the monitored laptop communicated to.

In addition to the above, we provide CDF and key summary statistics of the packet length distribution in Figure 10.

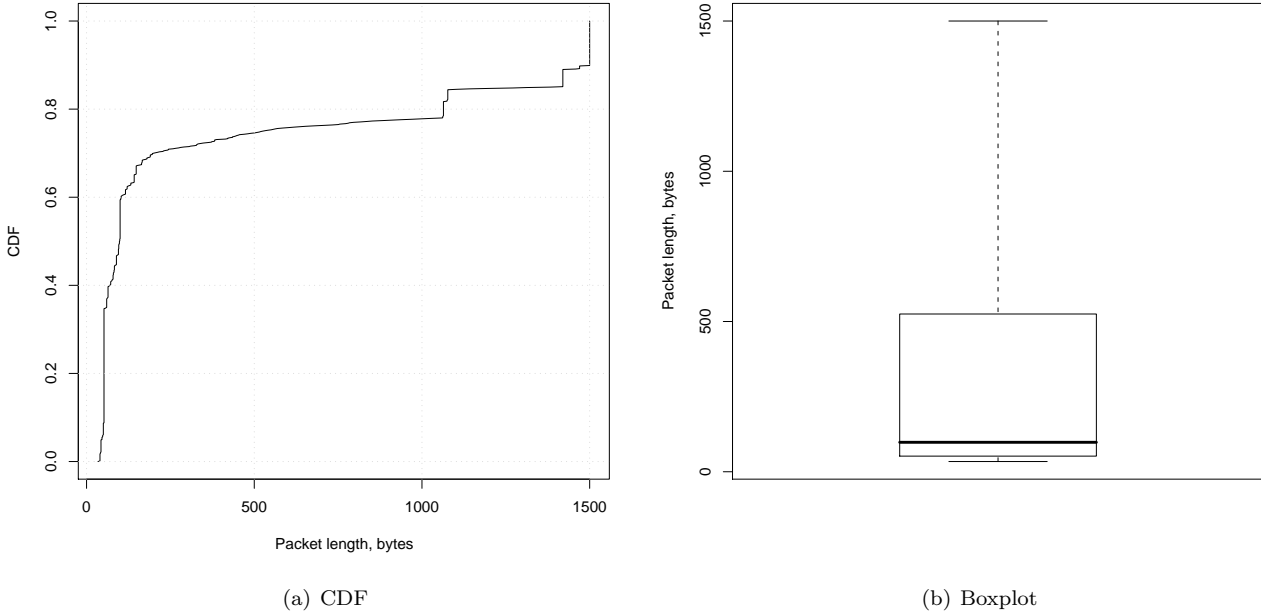


Figure 10: Packet length statistics.

3.2 Flow data

In this section we pursue the second approach in the captured data analysis. Now we aggregate separate packets into flows. To do so we use the `crl_flow` tool. The command that we used to create the flow-aggregated data (*.t2) is: `./crl_flow -o flows.t2 -Tf60 -Ci=0 -cl all.anon.pcap`. In all but the last analyses we use the 60 seconds timeout for inactive flows.

3.2.1 Breakdown by port number

This analysis is very similar to that of the Section 2.2. Here we also use only TCP and UDP flows (identified by the protocol numbers 6 and 17 respectively), because other protocols don't carry port numbers. The breakdown in the Figure 11 has five ports also appearing in the packet-level analysis (Figure 7). In addition we observe *bootps* (port 67), *bootpc* (port 68) and *netbios-dgm* (port 138). The two plots don't match exactly because each flow can contain arbitrary number of packets, thus it could happen so that for there were many packets involving the ports in the packet-level analysis, but these packets constituted only a small number of flows.

3.2.2 Distribution of flow interarrivals

Here we substitute the by-country analysis with the analysis of flow interarrivals. We do so for two reasons: (i) due to technical problems with the `crl_bycountry` tool we were meant to use for the analysis and (ii) we believe

¹We found that our TCP/IP stack implementation includes options with length of 12 bytes, resulting in the IP+TCP+options length equal to 52 bytes.

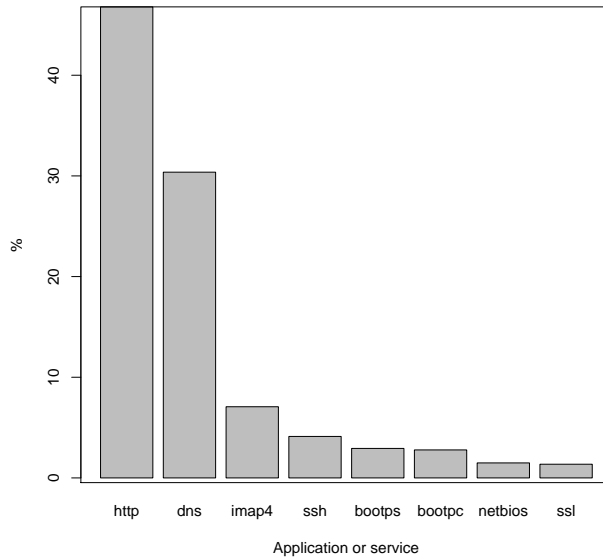


Figure 11: Top ports seen in the flows.

that the by-country analysis doesn't carry any practical sense since the IP addresses in the traces were anonymized and hence the results will not anyhow reflect the reality. On the other hand, interarrival analysis is an interesting and demanded topic in network measurements having a long history [11, 9, 10]. Being able to accurately model the interarrival process promises a lot of theoretical and practical benefits.

Quite often network interarrival events are modeled as a Poisson process because of its simplicity. However, many studies including the above cited and e.g. [15] found that the distribution of packet interarrivals is not in fact exponential as required by Poisson modeling. In this section we try to find out the distribution of the packet interarrivals found in our traces.

First of all we calculate time intervals between start timestamps for all flows beginning from the second one. Note, that since our three traces were collected on different days, we remove the two gigantic gaps between the traces start times. The resulting histogram of the flow interarrivals is shown in the Figure 12(a). Next we try to find out what theoretical distribution fits the observed data best.

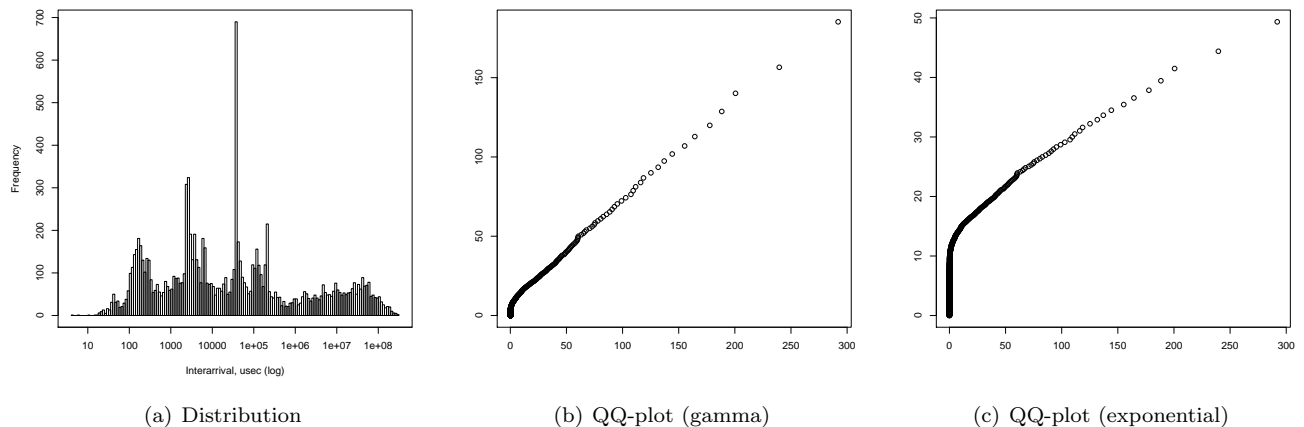


Figure 12: Flow interarrivals.

We have found that the gamma distribution with parameter $shape = 0.14$ and $rate = 0.02$ forms almost a straight line in the QQ plot (Figure 12(b)), while the exponential distribution with the estimated parameter $rate = 0.14$ fits badly (Figure 12(c)). We have also tried to investigate how the flow arrival rate is distributed. Unfortunately, we lack the data to do so. In order to see the distribution we were forced to use too small intervals (up to 10 minutes). However, we believe this is not enough for a representative picture (e.g. in [15] authors use one hour). Thus we can neither confirm nor disprove the normality of arrival rates suggested by Poisson processes.

3.2.3 Origin-destination pairs analysis

This analysis is very similar to the one presented in Section 2.3, here we used the same techniques as described earlier for aggregating data by origin-destination pair. In Figure 13 we provide the same two plots - distribution of data volume and a Zipf-type plot for the number of flows per origin-destination pair. The fitting line again corresponds to $y \sim \frac{1}{x^2}$. The Zipf-type plot looks very sparse because we lack proper amount of data to populate the plot.

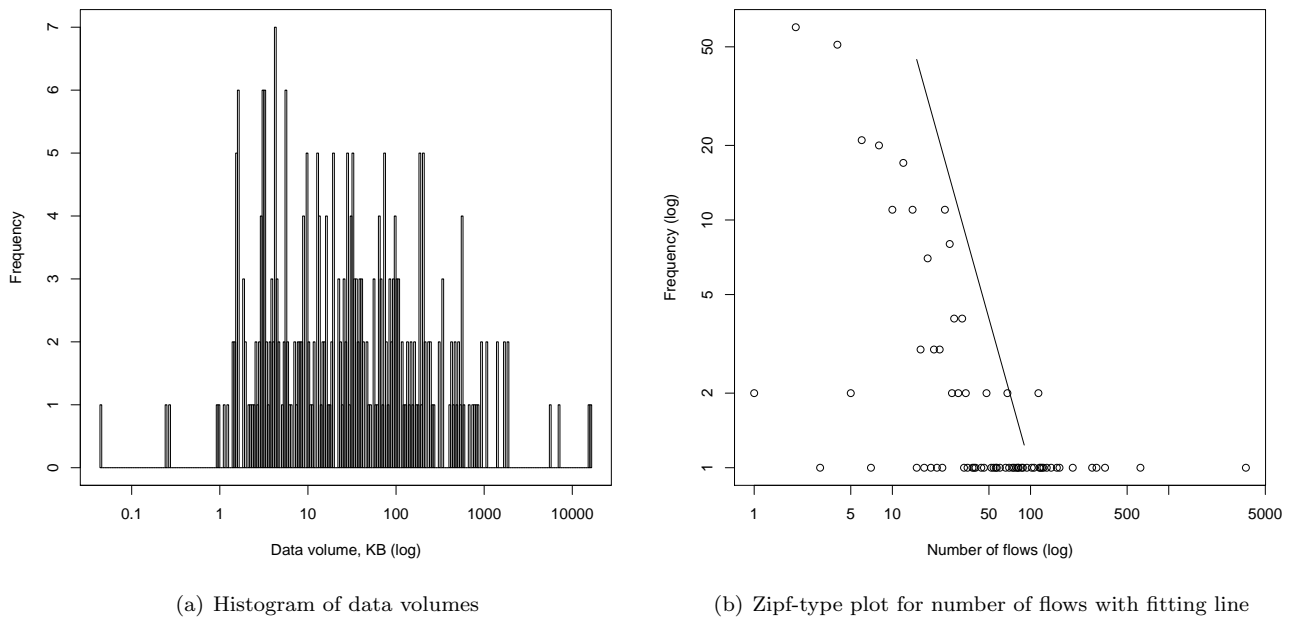


Figure 13: Origin-destination pairs analysis.

3.2.4 Flow length distribution

The Figure 14 shows the distribution, CDF and key summary statistics for the flow length values. Unfortunately we failed to fit any theoretical distribution to the flow length distribution. All log-normal, weibull, gamma and exponential distributions yielded lines far from straight in QQ plots.

3.2.5 Number of flows for varying timeouts

In this section we study how the number of flows varies depending on the timeout value specified to the `crl_flow` tool. Figure 15 is a scatter plot of number of flows versus timeout values (we varied the timeouts from 1 second to 30 minutes). In general, the bigger the timeout the less flows we see. The reason for that is that with longer timeouts one has higher chances to captures in a single flow either the continuation of an inactive flow or another connection with the same 5-tuple that will be attributed to the same existing flow as the previous similar connections.

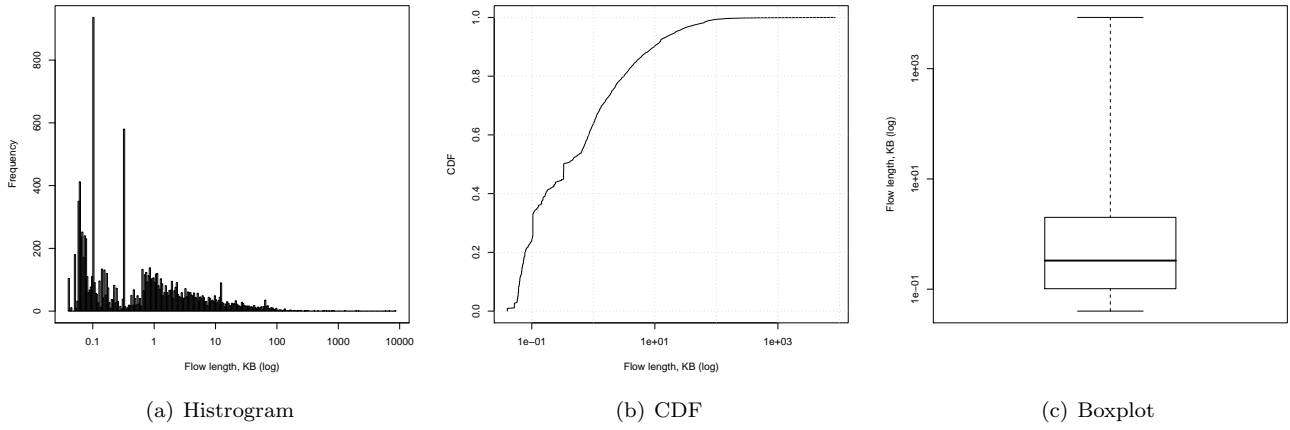


Figure 14: Flow length distribution.

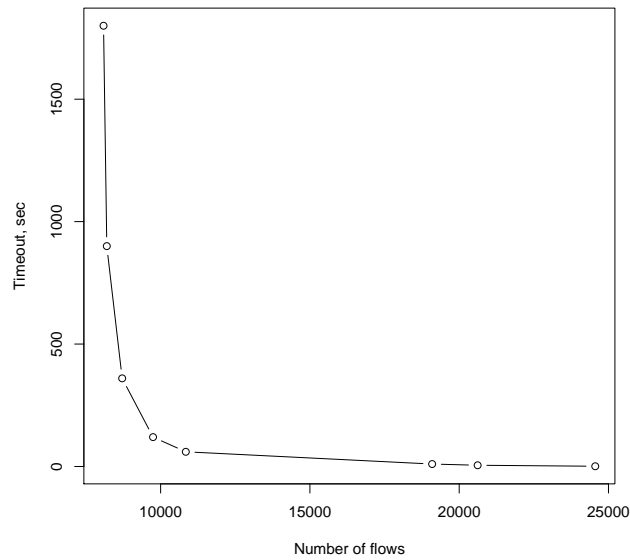


Figure 15: Number of flows for varying timeout values.

3.3 TCP connection data

Finally, we focus on the TCP-specific aspects of the captured traces. In particular we confine our analysis to retransmissions occurring during the capturing time. To aid with TCP connections reassembly we use the **tcptrace** tool which is capable of outputting the number of retransmitted data packets and average RTT value. Using the custom scripts we extract necessary columns from the **tcptrace** output to further analyze and plot them. In total the tool identified 2,631 TCP connections in our traces.

3.3.1 RTT

As a first goal we try to find out if there is any association between retransmissions and packets round-trip times. The tool calculates RTT values as a difference between sent data packets and received acks omitting ambiguous cases such as retransmissions. **Tcptrace** also outputs RTT values in both directions of the connection. However, finding the RTT value on the responder side is usually a difficult task [17]. We don't know how **tcptrace** does that, but

it appears to be employing the same approach as for the sent packets. This reasoning arises from the fact that the RTTs observed in both directions differ dramatically. For the *sender* scenario (sent data packets, received ACKs) the median RTT is 44.6 msec, while for the opposite *receiver* scenario (received data packets, sent ACKs) it is as small as 0.2 msec. This happens because in the latter case such estimation doesn't reflect the actual RTT, but is rather time the monitored host needed to process the data packet and send an ACK. As we said, estimating RTTs on the receiver side is a non-trivial task that requires more sophisticated techniques. For this reason we use only the sender-side estimated RTT as a representative of the RTT the connection experienced. Note, that similarly to RTT, the tool outputs two values for retransmissions. For simplicity we use the sum of these two fields as the number of retransmissions for the connection.

Further we try to correlate TCP retransmissions with RTTs within each connection. In Figure 16 for each TCP connection we plot its RTT value in milliseconds and the respective number of retransmissions. Note, that we had to scale the number of retransmissions along Y-axis to fit RTT values. More specifically, the Y coordinates for retransmissions are calculated as $retransmit \times \frac{\max(retransmit)}{\max(rtt)}$. The plot suggests that there is no significant correlation between the magnitude of retransmission and the RTT value for the connection. If there was such correlation this would be seen as circles coinciding with RTT peaks. However, we see the opposite - the a circle might be located high indicating intensive retransmission, while the respective RTT value is within the "normal waist" not exceeding 200 msec in the plot. The opposite case is also present — for the RTT peaks we don't see any significant retransmission. The estimated correlation coefficient between the two vectors we found to be equal to 0.03, also supports our finding that retransmission and RTT are not correlated in our case.

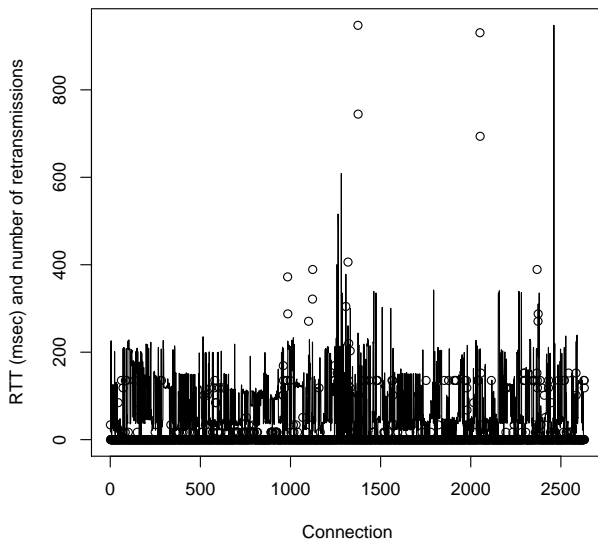


Figure 16: RTT values and scaled number of retransmissions.

3.3.2 Traffic volume

TCP retransmissions are usually caused by packets loss. And loss in turn usually occurs when the network load is high. Therefore it's natural to assume that TCP retransmission is likely to occur during network load bursts. In this section we study the correlation of connection retransmission amount with the total number of bytes transferred within a connection. However note, that this approach has a fundamental limitation that makes it inaccurate. As we stated above, a retransmission is likely to take place when the network load is high. But such periods usually don't last long. In our analysis we operate over retransmissions and network load for the whole connection. We do so because **tcptrace** outputs only aggregate retransmission summary and thus we lack the timestamps of individual events that we could correlate with the network load at that moment to get more accurate estimation.

Figure 17 is similar to the one in the previous section — it plots scaled number of retransmission events over the network load within the connection. The latter value is taken from the per-packet analysis — for each TCP connection we sum up lengths of all (not only belonging to this connection) packets that fall between the connection’s start and end timestamps. The plot doesn’t reveal any significant correlation between the two vectors under consideration. We believe that this is due to the reason that TCP retransmission analysis needs a more fine-grained approach that should focus on individual retransmission events instead of aggregated per-connection values.

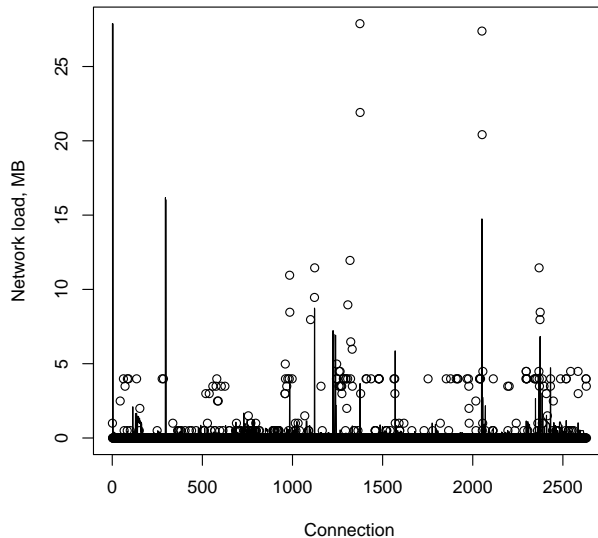


Figure 17: Connection size and scaled number of retransmissions.

4 Summary

In this paper we have pursued a some-what basic analysis of the two data sets, one provided by the course staff and the other one captured from the private laptop. For the both data sets we investigated traffic volume temporal patterns and most common applications in use. The course-provided data set spans a significantly larger time interval and thus we were able to discern some activity cycles in it. Not surprisingly both data sets revealed HTTP as the most frequently used application layer protocol. In our study we actively employed aggregation techniques, e.g. aggregation by origin-destination pairs and by user. The second data set was analyzed from three different perspectives: on the per-packet, per-flow and per-connection levels. During the study we have confirmed several well-known facts about the general nature of network traffic: *(i)* Zipf-type nature of distribution of number of flows; *(ii)* heavy-tailness of flow sizes; *(iii)* strong correlation between flow rate and flow size; *(iv)* non-Poisson nature of flow interarrival times. For some of our analyses we tried to derive and evaluate analytical models that would fit the data best.

References

- [1] tcpslice. <ftp://ftp.ee.lbl.gov/tcpslice.tar.gz>, 2000.
- [2] Iana assigned port numbers. <http://www.iana.org/assignments/port-numbers>, 2009.
- [3] M. Allman. Comments on selecting ephemeral ports. *SIGCOMM Comput. Commun. Rev.*, 39(2):13–19, 2009.
- [4] C. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing reference locality in the www. *Parallel and Distributed Information Systems, International Conference*, 0:0092, 1996.

- [5] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing stability in wide-area network performance. In *SIGMETRICS '97: Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 2–12, New York, NY, USA, 1997. ACM.
- [6] L. Breslau, P. Cue, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM Conference*, pages 126–134, 1999.
- [7] CAIDA. Coralreef. <http://www.caida.org/tools/measurement/coralreef/>, 2008.
- [8] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Trans. Netw.*, 9(4):392–403, 2001.
- [9] H. J. Fowler and W. E. Leland. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE Journal of Selected Areas in Communications*, 9(7):1139–1149, 1991.
- [10] R. Gusella. A measurement study of diskless workstation traffic on an Ethernet. *IEEE Transactions on Communications*, 38(9), Sept. 1990.
- [11] R. Jain and S. A. Routhier. Packet trains: Measurements and a new model for computer network traffic. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 4:986–995, 1986.
- [12] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 27–40, New York, NY, USA, 2004. ACM.
- [13] V. Paxson. Bro: a system for detecting network intruders in real-time. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium*, pages 3–3, Berkeley, CA, USA, 1998. USENIX Association.
- [14] V. Paxson. Strategies for sound internet measurement. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 263–271, New York, NY, USA, 2004. ACM.
- [15] V. Paxson and S. Floyd. Wide-area traffic: the failure of poisson modeling. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 257–268, New York, NY, USA, 1994. ACM.
- [16] V. Paxson and S. Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3:226–244, 1995.
- [17] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and D. Collange. A root cause analysis toolkit for tcp. *Computer Networks*, 52(9):1846–1858, 2008.
- [18] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. *SIGCOMM Comput. Commun. Rev.*, 25(4):100–113, 1995.
- [19] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of internet flow rates. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 309–322, New York, NY, USA, 2002. ACM.