Statistical Analysis of Network Sensor Data

Robin Pfeiffer Sai Krishna

July 2023

Abstract

When building a network sensor there are many arbitrary numbers that must be decided on. One of these is the space that you allocate towards the hash table where the active flows are stored. This paper aims to give statistics-backed methodology for deciding how much space to allocate it. Presented in a tutorial style, it is easy to follow and explains the reasoning behind each step.

1 Introduction

Packet capture devices are an essential part of today's cyber security landscape. They work by taking a copy of every packet that they see, and extracting all of the information it has access to. This usually includes the source and destination IP addresses, source and destination ports, its internet transfer protocol, and its IP protocol. One can then use this information to set up alerts, or store it to have a history of internet traffic.

Many flows of data consist of more than one packet being sent. To prevent having to store similar packets separately, they are usually grouped together by flows.

In order to group the packets together, we must keep track of which flows are actively sending packets. We call these flows active. This is done in a hash table. The hash table keeps a list of all the active flows. When a new packet comes along, it will look through the hash table to see if it belongs to one of the active flows, and if not, it will generate a new flow. Flows are evicted from the hash table after a certain amount of time with no packets coming through.

The hash table has a fixed amount of space allocated to it. If there is not enough space allocated to it, then the hash table will have to drop active flows, and the sensor will be dropping information. Alternatively, if we allocate too much space to the hash table, it is a waste of space and money. And so, we want to find the ideal amount of space to allocate to the hash table.

This paper will act as a tutorial style methodology for determining the ideal amount of space, using data collected beforehand of how many active flows are in the hash table.

The paper will follow the following layout. First, we present the methodologies for collecting data, with the following steps:

- Flow Definition
- Collection of Flow Records
- Merging Flow Records

Each step includes a detailed description of how we performed it, and describes how it can be repeated.

Next, we describe the data that we used for this procedure. We cover where we sourced it from, when we collected it and how we cleaned it. In this section we also describe in detail and explain the analysis that will be performed on our data. This includes extracting the necessary statistics from the data, and how to fit these statistics to a model. It then explains how to interpret the results.

Finally, we perform the procedure on the data that we have collected. This provides a real world example of how this procedure is performed. It also shows how actual data can be interpreted. Finally, this section provides a baseline to see what results can be expected.

Key Words
Network flow, statistics, hash table, network traffic

2 Literature Review/Related Works

The article [2] presents a tutorial-style approach to analyze traffic data, focusing on the methodology employed. The authors use data from sources like NetFlow and construct a model to estimate flow length and size. They utilize a month's worth of data from a University campus, consisting of approximately 4 billion flows. While their data collection and preprocessing method slightly differs from the one employed in this paper (which uses data from the Rapid7 sensor), an essential distinction is that the authors of [2] do not draw any conclusions using the models they create.

The authors in [4] also calculate flow statistics based on NetFlow data. However, their focus is limited to specific transport layer ports associated with applications such as peer-to-peer (P2P), web, and TCP-big. Because of this, these models are not as accurate as [2]

[1] analyses traffic across 10 different data centres, including university, enterprise, and cloud data centers. The authors highlight the variations in traffic encountered by these three types of data centers, thereby reinforcing our assertion that the process outlined in this paper should be implemented separately for each sensor.

Papers [6] and [5] are also widely cited; both provide graphical representations of flow size, duration and rate distributions, but lack numerical data. In contrast, in [3] the authors focused solely on the flow duration.

3 Methodologies

This section outlines the steps involved in the collection and preparation of ipv4 flow data.

3.1 Flow Definition

The definition of a flow depends on the protocol layers that you are interested in. Most commonly, these will be layer 3 (transport layer) and layer 4 (application layer). A layer 3 flow is the collection of packets with matching source and destination IP addresses, and the same transport layer protocol (eg. TCP/UDP). This collection of Protocol, Source IP & Destination IP is commonly called the 3-tuple key. A layer 4 flow goes one layer deeper than the layer 3 flow. A layer 4 flow is the collection of packets with matching source and destination IP addresses, and the same transport layer protocol, and the same source and destination ports. Protocol, Source IP, Destination IP, Source Port & Destination Port is commonly called the 5-tuple key.

The definition used throughout this paper is the layer 4 flow. All of the fields in the 5-tuple are easily accessible in the packet, and are never encrypted, which means that we can reliably determine the packets that belong to the same flow.

3.2 Fixed vs Dynamic Hash Table

A hash table is fixed if it has a fixed amount of space allocated to it. A hash table is dynamic if the amount of space allocated to it can change. There are advantages and disadvantages associated with both. A fixed hash table is simpler to implement. However, there can be large peaks in network traffic, or the traffic may be increasing spontaneously over time. This would lead to the sensor dropping packets.

A dynamic hash table could fix this problem by allocating a suitable size to the hash table, which may grow / shrink over time.

3.3 Collection of Flow Records

The collection of flow records is done using a network sniffer. The sniffer is plugged into a switchboard, and it takes a copy of every packet that passes through it. We can then scrape any relevant information out of it. There are many choices of network sniffers out there. For our data collection we used a Rapid7 network sensor. However, there are many other options, and one could even build their own. For this procedure, the sensor needs to be able to group packets together that belong to the same flow. The simplest way of doing this is by grouping flows together that have the same 5-tuple key. From these flows, we only need to track at when these flows are actively in the hash table. To do this, we keep track of the timestamp when they begin, and their lifetime.

3.4 Merging Flow Records

Flows that were split up into separate flow records must be grouped together again. However, we must be careful here because two different flows could have the same 5-tuple. In order to differentiate between these flows, we set a hard time limit and we say that if two flows fall within this time of each other they are the same flow, otherwise they are different.

Once we have collected all of the flow records, we group them together by the 5-tuple key. We then group together the flows that occurred within a certain time of each other to recreate the flows. For UDP flows, the time limit was two minutes. For TCP flows, the time limit was ten minutes.

4 Data Used and Project Architecture

The data we used comes from the internal Rapid7 sensors. I used data from May 2023. However, there was a bit of cleanup that had to be done first. The amount of internet traffic reduces significantly on the weekends and on public holidays, because no one is in the office creating internet traffic. This will obviously reduce the number of active flows by a significant amount. Since we are only interested in the peaks of internet traffic, we remove weekends and public holidays from our data set. This is all of the data cleaning that is required.

Using just one day of data at a time, we calculated the number of active flows at every second throughout the day. Rather predictably, there are many more flows between 9am and 5pm than during the rest of the day.

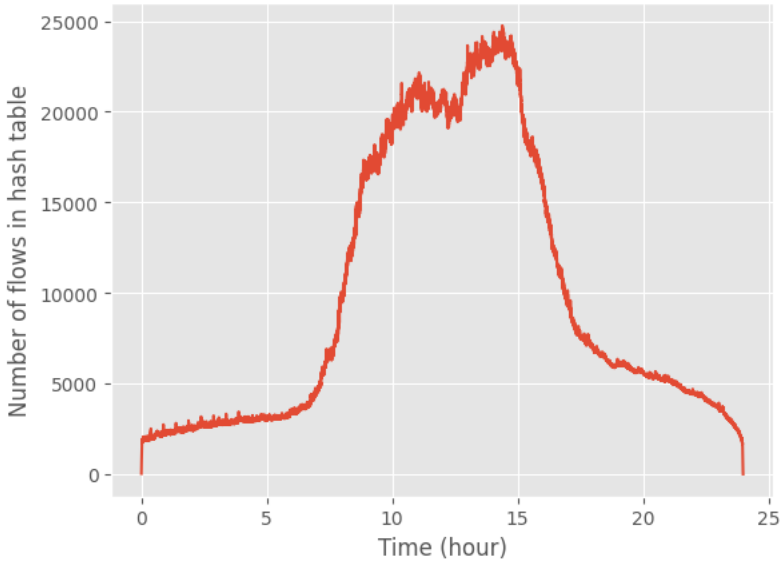


Figure 1: Number of flows active on May 4th

On this day, the peak occurs at approximately 2pm, with 25000 flows active in the hash table.

If we repeat this process on every day of data that we have collected, it will look like the maxes are normally distributed. However, there are three necessary conditions that we must assert before we assuming that it is actually normally distributed.

We firstly need our samples (the number of flows throughout the day) to be independent of each other. This is a reasonable assumption because what happens on one day will not necessarily affect what happens the next day by a significant amount.

Secondly, the samples must be drawn from an identical underlying distribution.

It was for this reason that we removed the weekends and public holidays from the data set. Since each day of data comes from a regular day at the office, we can say that this holds true.

Finally, each sample must have a finite variance. This also holds true.

Since all three of the necessary conditions hold true, we can confidently say that the maxes will follow a normal distribution.

5 Analysis and Results

Performing the above procedure on a month of data yielded the following results.

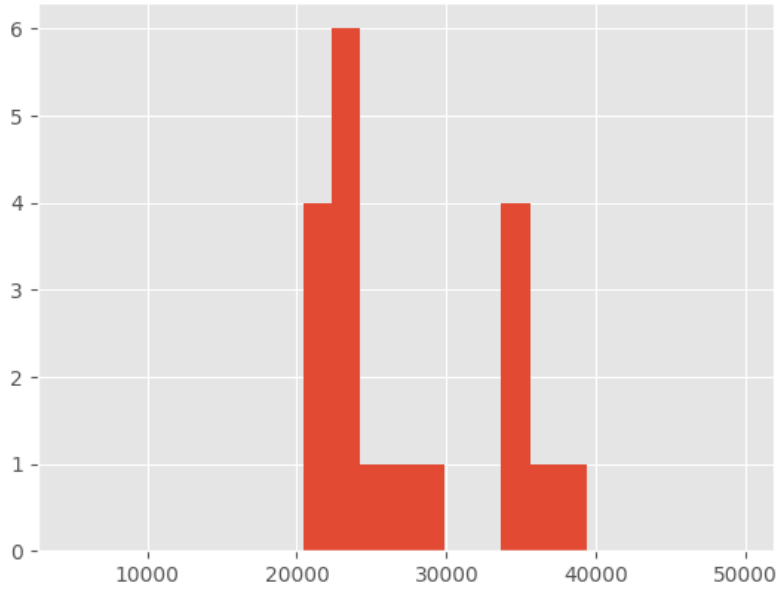


Figure 2: Histogram of maximum number of flows encountered throughout the day

Since we used a month of data, but had to remove multiple days, we ended up with just nineteen points of data. Thirty points of data are required to fit a normal distribution to a data set. Instead, we must use a Student's t-distribution with eighteen degrees of freedom.

Given $x =$ list of maxes, with $|x| = n$, we can define the random variable X as the maximum flow count observed in a day. Then,

$$X \sim t_{n-1}(\mu = mean(x), \sigma^2 = var(x))$$

Adding this to the histogram gives the following

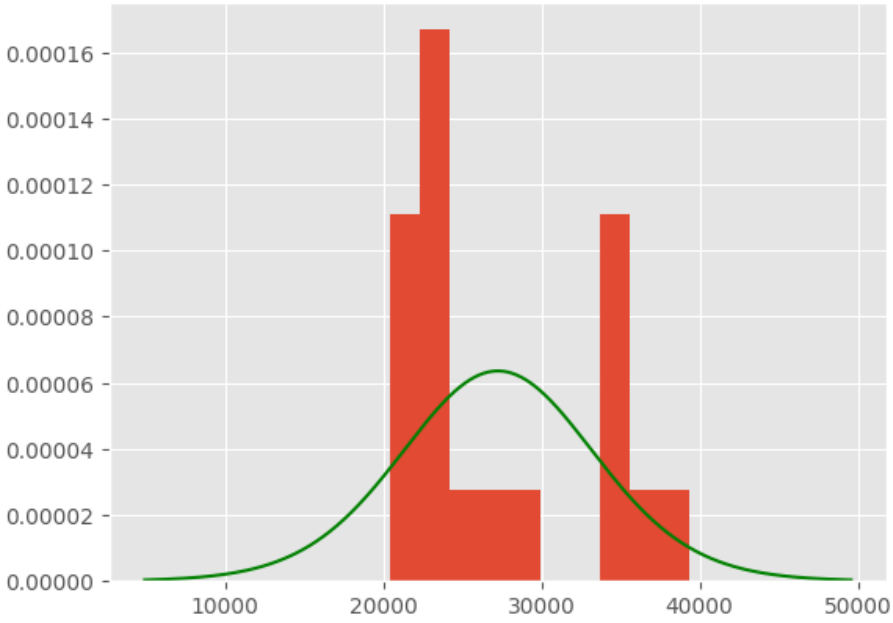


Figure 3: Comparing model to data

We can then find the 95th / 99th percentile of this distribution

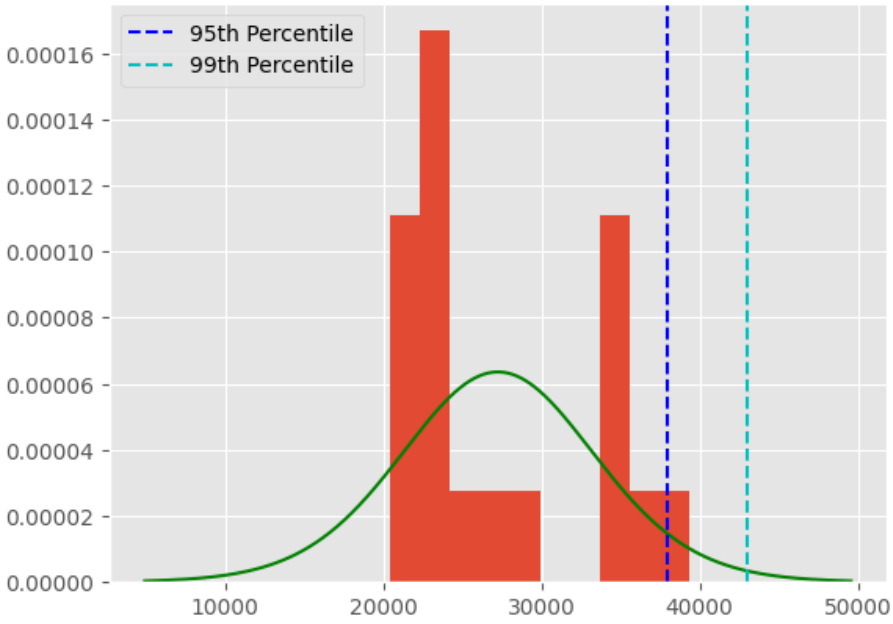


Figure 4: 95th and 99th percentiles

Results

95th Percentile	37971.55
99th Percentile	43033.53

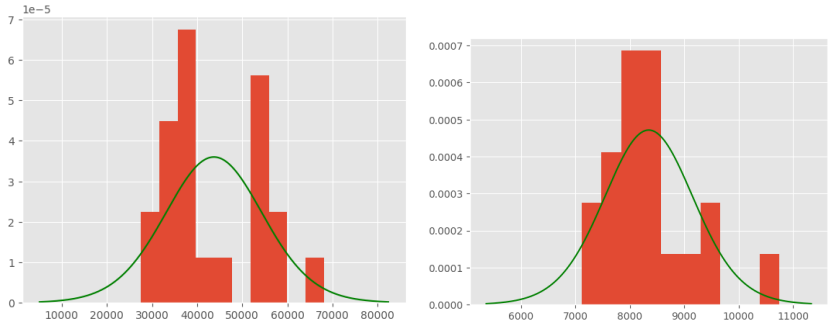
Interpretation

If our hash table has enough space for 37971.55 flows, then we can expect that for 95 out of 100 regular working days the hash table will be big enough.

If our hash table has enough space for 43033.53 flows, then we can expect that for 99 out of 100 regular working days the hash table will be big enough.

Repeating the Procedure on other Sensors

Running the procedure on different sensors yielded the following results. Sensor (a) was set up in a university, which used a big mixture of protocols. Sensor (b) was in large part made up of TLS flows. However, the protocols used did not have an effect on the outcome, and the results are very similar.



(a) A busier sensor (b) A less busy sensor

Figure 5: Different sensors

In sensor (a), the 99th percentile is 71289.47
In sensor (b), the 99th percentile is 10469.25

6 Conclusion and Further Work

This paper contributes in many ways. Firstly, it describes a process to determine the maximum number of active flows observed in a day, which as explained above, can be used to save space and money. It achieves this in a tutorial style methodology. While doing this, it also explains the reason for each step in the process, and proving its mathematical soundness when necessary.

Additionally, it gives a real world example of implementing the process on a company sensor. This also acts as a benchmark for companies of similar size.

The procedure described in this paper has the potential to be the source of a dynamic model, which can change the allocated space of the hash table based on the most up to date data. This would mean that, even if your internet use grows, you can be confident that you are dropping very few packets.

References

[1] Theophilus Benson, Aditya Akella, and David A Maltz. "Network traffic characteristics of data centers in the wild". In: *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. 2010, pp. 267–280.

[2] Piotr Jurkiewicz, Grzegorz Rzym, and Piotr Boryło. "Flow length and size distributions in campus Internet traffic". In: *Computer Communications* 167 (2021), pp. 15–30.

[3] DongJin Lee and Nevil Brownlee. "Passive measurement of one-way and two-way flow lifetimes". In: *ACM SIGCOMM Computer Communication Review* 37.3 (2007), pp. 17–28.

[4] Matevz Pustisek, Iztok Humar, and Janez Bester. "Empirical analysis and modeling of peer-to-peer traffic flows". In: *MELECON 2008-The 14th IEEE Mediterranean Electrotechnical Conference*. IEEE. 2008, pp. 169–175.

[5] Feng Qian et al. "TCP revisited: a fresh look at TCP in the wild". In: *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. 2009, pp. 76–89.

[6] Yin Zhang et al. "On the characteristics and origins of internet flow rates". In: *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 2002, pp. 309–322.