Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Data-Driven Performance Correlation

by Jan-Philipp Schulze

Tutor: Maria Apostolaki
Co-Tutor: Dr. David Gugelmann
Supervisor: Prof. Dr. Laurent Vanbever

Semester Thesis SA-2018-16
March to April 2018

Abstract

In Internet Tomography, hidden network topologies are revealed by probing and correlating network traffic. We present an algorithm based on passive probing that allows to find clusters of destination prefixes which are affected by the same incident. It was designed in a flexible and modular way, without adding any traffic overhead to the network. Our algorithm observes the loss rate of the aggregated signal. By applying peak detection and Association Rule Learning, common clusters are found.

Acknowledgements

A special thank-you to my advisors, *Maria Apostolaki* and *Dr. David Gugelmann*. The fruitful discussions and their constant support have been highly motivating.
For putting his trust in this thesis, I like to thank *Prof. Dr. Laurent Vanbever*. It was an inspiring experience to work at his research group.
For their mental support, I like to thank *Franziska Sophie Guhr*, *Gamuret Friedewalt Hack*, *Maximilian Schütte* and *Nadja Müller-Seip*. With their help, even the time close to deadlines is enjoyable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet is a highly distributed, hiarachical system where each user only knows parts of the underlying structure. Wheras a Internet Service Provider (ISP) knows the connections and routing behaviour within the reach of its own infrastructure, the networks of other ISPs, enterprises or homes are unknown to it. This is a strong limitation to the set of solutions in case of network congestions or failures. Network problems could be shifted rather than solved if common paths between different network flows are not identified. Thus, insufficient knowledge about the true nature of problems can starkly degrade the Quality of Service (QoS) for customers and other network participants.

In the research field of Internet Tomography ([1]), statistical observations on the network are used to infer the inner topology. Two main approaches are used: active probing inserts specifically crafted packets, passive probing makes conclusions from the observed traffic only. In either method, statistical features are calculated, e.g. the throughput or in case of TCP traffic the loss rate and the delay. By virtue of an appropriate correlation algorithm, information about the network is revealed.

Our approach applies a correlation analysis to passively probed traffic recorded on a single network router. Using the loss rate of each network destination as our prime feature, we were able to cluster network destination. We show, these clusters are likely to share a common path as they respond similarly to the observed network congestions. With our solution, strong correlations in network paths can be revealed with a minimum of technical prerequisites.

1.1 Motivation

Over the last years, Internet traffic has been on a steady rise. In 2016, "average traffic grew at 32 percent". More importantly, at the same time "busy hour Internet traffic grew 51 percent" [2]. In that, traffic during peak times exceeds the average increase dramatically. Where the general traffic is more evenly distributed among destinations, the peak traffic is highly concentrated in time and origin. As studies showed, in 2016 more than 70% of all peak downstream traffic was due to "real-time entertainment" platforms like YouTube or Amazon Video [3]. Other destinations that share links with these services are prone to congestions if the common routing paths cannot handle the increase during prime times. The paramount importance of this issue is underlined by initiatives like Netflix's Open Connect network which provides a "proactive, directed caching solution" in cooperation with the ISPs [4]. Here, the company builds a Content Delivery Network (CDN) which allows to access the streaming data closer to the end-user, thus lowering the path length.

We argue that intelligent routing will be of crucial importance with the ever-increasing demands in the Internet. Being able to anticipate congested links and diversify the routing

	Passive	Active
Internet	[12], this work	[13], [9], [6], [7], [8]
Local	[12], [11], [10]	[5], [6], [7], [8]

Table 1.1: Overview about Related Work Sorted by Probing and Network Type

behaviour accordingly, lowers losses especially during peak times. Internet Tomography is a major building block in this goal as alternative paths are usually not commonly known; due to privacy concerns or competitive rivalry, the network topology is hidden from third parties. Our motivation was to design an algorithm that does not insert additional traffic into the network, but detects common paths shared with major sources of network traffic. In the following sections we will present the underlying theory and results of our analysis.

1.2 Related Work

Internet Tomography or in the case of local networks, Network Tomography, has been subject to active research. A quick overview about recent publications and how this thesis is related to them is shown in Table 1.1. The research content is mainly distinguished by the underlying probing type (active/passive) and the network topology it was designed for (Internet/local).

Especially active probing has been of ongoing research interest. Specifically crafted packets are inserted on the network paths such that features, e.g. based on the timing, can be calculated. For local networks as found in datacentres, well-known ping messages can be used to check the state of the servers. In [5], an infrastructure using pinging is proposed. A visualisation supports network administrators to find reasons of outages. For global networks like the Internet, active probing messages are adapted to increase the information content. In [6], a probing framework is designed optimising the Fisher information, i.e. how much new information is acquired for each probe. The paper concludes that uniform probing is not optimal for most networks. Other papers try to leverage characteristics of the features, e.g. the sparsity of loss data, to gain more information. Sparse Bayesian learning is proposed in [7]. For more efficiancy in the probing process, unicast probes are used in [8]. Nonetheless, for all active probing approaches it should be noted that the probing packets themselves influence the behaviour of the network. Too aggressive probing could cause losses for normal network traffic, altering the calculated features. An overview about the loss behaviour in networks is given in [9]. To avoid any interference, we used passive probing in our approach, i.e. the analysis of recorded network traffic.

So far, passively probed approaches are mostly used for local networks. Also here, monitoring datacentres is an important application. In [10], a system storing meta information of packets at each routing step was developped. With the proposed debugging framework, detailed information about congestion locations and failed links are revealed. For the same setting, [11] proposes methods to improve the accuracy of latency measurements. It shows the importance of exact data to lower thresholds for uncertainty.

Passively probed methods rely on large data sets. Due to the amount of measurement points, effective storage methods for the correlation data have been explored in [12]. Here the feature, delay sequences, is stored in tree-like structures. Using wavelets as preprocessing step reduces the dimensionality of the problem further more. As our approach leverages passive probing to the setting of Internet traffic, the use of sparse representations is important. We will propose the use of data around peaks as they bear important information about the overall network behaviour. With our algorithm, we aim to give a flexible, yet powerful analysis tool for network topologies without adding any overhead on the network paths.

1.3 Contributions

During this semester thesis, the theory of the clustering algorithm and the software implementation of it was developped and evaluated. The entire clustering task is covered, in that:

- Reading, filtering and estimating commonly used traffic recordings in the `.pcap` format
- Calculating the two main features for TCP traffic, i.e. throughput and loss rate
- Detecting peaks, correlating and rating the features
- Clustering of the correlation statistics using association rule learning
- Visualising the implications of parameter choices in a graphical interface for interactive data exploration

1.4 Overview

In chapter 2, the main theoretical background will be introduced.
Based on this knowledge, chapter 3 shows how the theory was implemented in our solution.
The performance of the software implementation is evaluated in chapter 4.
Finally, we will give an outlook about possible future work in chapter 5 and a summary of the thesis in chapter 6.

Chapter 2

Preliminaries

To fully grasp the content of the thesis, we will introduce the reader to the main theory. In 2.1, an overview about TCP's loss behaviour is given such that general trends can be understood. As the network traffic is a sparse data source, the density estimation techniques described in 2.2 are used to preprocess the data. Our correlation process consists of peak detection and calculations with the techniques shown in 2.3. Finally, we will give a short overview about association rule learning in 2.4.

2.1 Network Losses

In our traffic analysis, we look at general features of the underlying flows, primarily their loss rate. To obtain feedback on how lossy a connection is, we need to observe traffic that is handled by a form of transmission feedback. By far the most ubiquitous traffic fulfilling this requirement is TCP. We will solely concentrate on this protocol. Note that we would not gain any extra knowledge by incoportating protocols that do not use traffic control, e.g. UDP. Their impact is simply an offset added to the router queues which in turn impacts all other flows equally. Thus, the features of the TCP flows will still have the same relation to each other, only altered by a common constant at the respective instance of time.

2.1.1 TCP Congestion Control

To gain more insight on which loss behaviour we expect in our analysis, it is instructive to familiarise oneself with the congestion control used in TCP connections. We will concentrate on the TCP congestion control standard in [14] which is used in TCP Reno as described in [15]. It should be noted that TCP Reno is not the only TCP standard used in the Internet. In [16], the usage of the algorithms is compared, showing that enhanced versions of TCP Reno, e.g. NewReno or RenoPlus, are popular as well. The algorithm is subdivided into three components: slow start, congestion avoidance and fast recovery. Assuming a large enough receiver buffer, the sender's rate is governed by the congestion window, `cwnd`, which limits the amount of bytes that it may sent. The simplified finite-state diagram for better overview is shown in 2.1.

Slow Start

When the TCP flow is created, `cwnd` is initialised with 1 MSS (Maximum Segment Size). The MSS is set by "determining the lengths of the largest link-layer frame that can be sent by the local sending host". For each acknowledge packet, `cwnd` is increased by 1 MSS. Effectively, this doubles the sending window in each timestep so that a limit on the available bandwidth is found quickly.

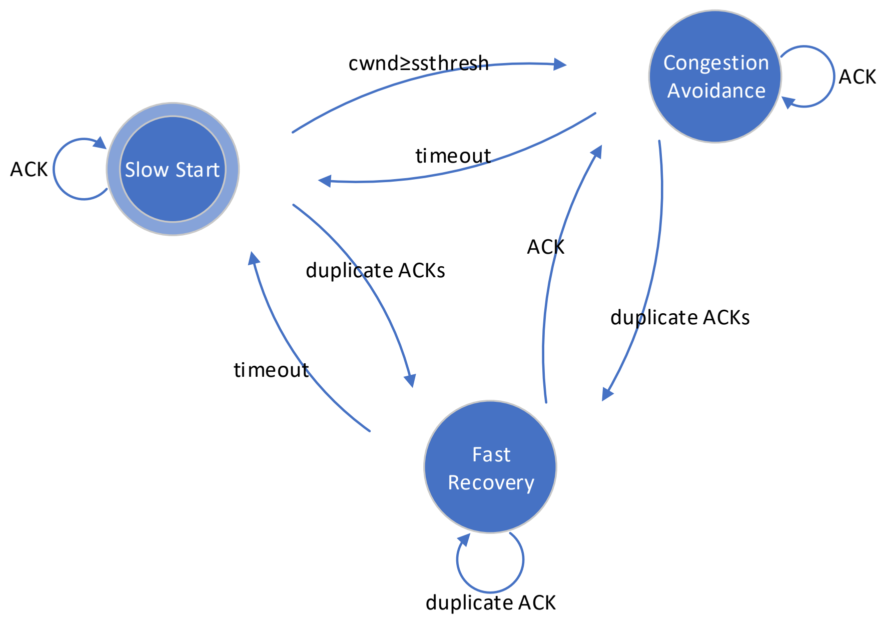


Figure 2.1: Simplified Finite-State Machine of the TCP Congestion Control Showing the Causes of State Changes

After the first detected timeout, the threshold for the slow start phase is defined to half the current congestion window, i.e. `ssthresh=cwnd/2`. The slow start phase starts again, but is replaced by the congestion avoidance as soon as reaching `ssthresh`. Alternatively, if three duplicate `ACK` are detected, the fast recovery phase is entered.

Congestion Avoidance

The congestion avoidance phase is entered when the throughput is close to the last known congestion limit. To avoid immediate packet loss, `cwnd` is increased more gently. In contrast to the slow start, 1 MSS is added to `cwnd` not for each but for all acknowledged packets in a respective time step. Thus, for each `ACK`, `MSS/cwnd` bytes are added to `cwnd`.

In case of a timeout, the transmission continues in the slow start phase with `ssthresh=cwnd/2` and `cwnd` set to 1 MSS. Alternatively, if three duplicate `ACK` are detected, the fast recovery phase is entered.

Fast Recovery

As resetting `cwnd` to 1 MSS each time a network congestion is detected limits the overall throughput, the fast recovery phase is introduced instead. In case of three duplicate `ACK`, the `cwnd` is initialised to `ssthresh`+3 MSS and the fast recovery phase is entered. Here, `cwnd` increases for each duplicate `ACK` by 1 MSS. A main reason why the congestion window is increased although duplicate `ACK` are received is that these are mainly due to packets received out-of-order [17]. In case of a timeout, the slow start phase is entered again.

2.1.2 Loss Peak Correlation

Knowing how the throughput of TCP traffic behaves, we can evaluate which types of loss peaks are likely to be seen. Peaks will lead us to a compact time range where losses occur on several end-points potentially because of a common source. Here, we expect the losses from different, correlated flows to accumulate, giving us information to deduce the underlying network state.

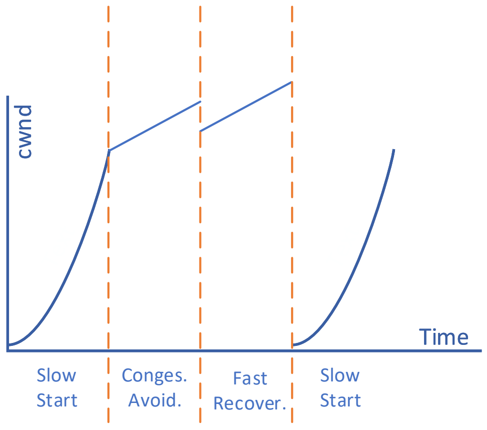


Figure 2.2: Possible Time Series of the TCP `cwnd` for TCP Reno

In Figure 2.2, a possible time series for the TCP `cwnd` is shown. After an exponential increase of the congestion window and thus the throughput, first duplicate `ACK`s are received. The figure ends with a timeout and the subsequent slow start phase. The loss rate behaves accordingly: before switching to the fast recovery phase, the losses increase; a peak in losses leads to the timeouts, leading to fewer losses. In the global view of multiple TCP flows, this behaviour shows as wave-like behaviour of the loss behaviour. Peaks of losses are followed by troughs of different magnitude. Please note, this does not mean that oscillations occur: the peaks and troughs do not form a regular pattern, but are governed by the amount of packets that are sent on the link. The behaviour of TCP in relation to the number of active flows has been studied in [18].

In general, losses will not only occur due to queue overflows in routers. Possible other sources of errors lie in the network hardware and the applications itself. Errors on the routers add a hard offset in losses to all connected endpoints. These events will thus be captured in the correlation process as the loss rate of all flows instantly increases. Application errors are bound to specific end-points, thus do not correlate to others. It is assumed that these errors are sporadic events, so that on average the loss rate will follow the behaviour introduced by the TCP congestion control mechanism.

Describing and quantifying how correlated extrema in the loss rate are, allows us to weight the information we gain from the analysed traffic. We will compare the individual flows to the aggregated total flow. Intuitively, paths with a congested shared link show similar loss behaviour as long as the common path is the main source of lost packets. In the following, we will introduce two important factors that influence the correlation; the selected estimation technique and the applied correlation coefficient.

2.2 Density Estimation

Each recorded packet can be identified by its timestamp and 5-tuple, i.e. the protocol as well as the IP address and port for the underlying source and destination. The correlation coefficient will be calculated for a selected set of points in time, an inherently continuous measurement. For a valid comparison, it must be decided which packets of one flow may be compared to the ones of another flow, i.e. a form of importance weighting. Fixing the point in time where the correlation should be calculated, estimation techniques allow to determine the influence of all surrounding packets. Intuitevely, a packet closer to the selected time should have more influence than a packet far away.

As traffic data is sparse, i.e. flows are unlikely active at the exact same point in time, we require the estimation algorithm to perform a form of aggregation. A simple idea is time binning where all packets of a given flow are summed if they fall in a given time range. This algorithm fulfills our requirements of weighting and aggregation, but has serious disadvantages for our use case. Binning is highly dependent on the selected bin size and position of the bins, resulting in different observable behaviour for the same underlying data. Too small bin sizes lead to separate peaks and de-correlates events with a common cause. In the limit, this leads to single events without any correlation in time. Too large bin sizes average out events such that peaks and troughs merge. The problem is illustrated in 2.3: two peaks merge to one if the binning parameters are selected differently. Thus, we will use Kernel Density Estimators, a more flexible estimator as described in the following.

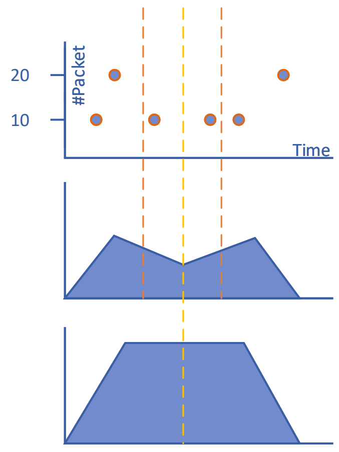


Figure 2.3: Influence of Different Bin Sizes on the Apparent Behaviour of Flows

2.2.1 Kernel Densitiy Estimation

Kernel Density Estimation (KDE) is a non-parametric, i.e. without assuming a specific distribution for the data, probability density estimator. In the case of network traffic, given that a packet occurs at times $t_i, i \in 1, 2, ..., N$ for N observations, the KDE estimates the probability distribution $p(t)$ for any time instance t of interest. The estimation is governed by two main parameters: the kernel function $k(t, t')$ and the kernel bandwidth h. For the estimation, the support weighted by the kernel function is accumulated based on the number of observations (as shown in [19]):

$$p(t) = \frac{1}{N \cdot h} \cdot \sum_{i=1}^{N} k\left(\frac{t - t_i}{h}\right)$$

Kernel Function and Width

A kernel function $k(t, t')$ is a symmetric, positive-definite function measuring the similarity between two points. We will focus on the Tophat, Gaussian and Epanechnikov kernel, each representing a different design choice. Wheras the Tophat and Epanechnikov kernel are compact functions related to h, the Gaussian kernel has infinite support. Using the Tophat kernel, all observations within the kernel width are weighted equally, wheras points away from the point of interest are weighted less important for the Gaussian and Epanechnikov kernel. For an instructive overview, they are plotted in Figure 2.4 for a fixed kernel width. Which kernel function to choose is a design choice based on the underlying data. We will thus postpone the discussion until section 3.2. Once the training data is fitted to the KDE, the likely response for a given set of nodes, i.e. in our case points in time, can be calculated.

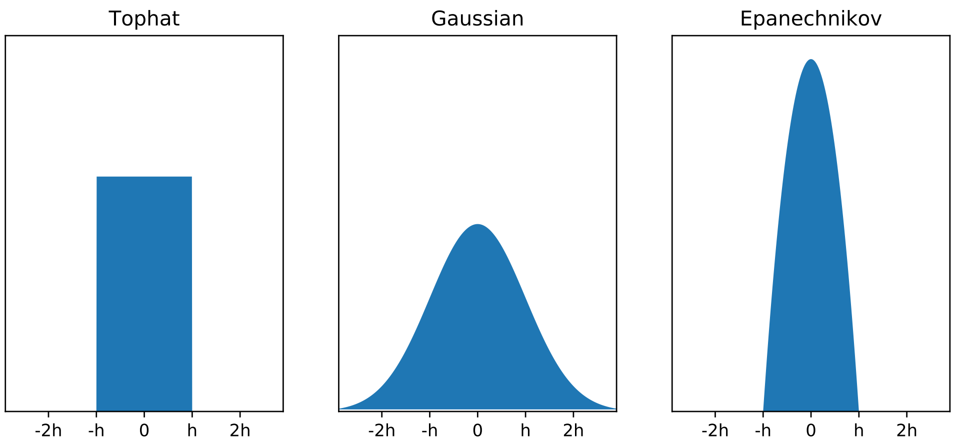


Figure 2.4: Different Kernel Functions for a Fixed Bandwidth h

2.3 Correlation

The correlation process quantifies the relationship between two variables, commonly referred to as the predictor X and criterion variable Y. By correlating different flows to the total loss rate, we capture how similar they behave. Intuitively, if the loss rate of a single flow is proportional to the overall loss rate, it is influenced by the same congestion as the majority of flows in this time region. Popular choices in literature are the Pearson product-moment and Spearman's rank-order correlation coefficient. We will apply the Spearman coefficient around the detected peaks in the loss rate. The used algorithms are presented in the following.

2.3.1 Peak Detection

Peaks in the loss total rate are of special interest in our analysis: here, more losses than usual occur which is likely a sign that multiple destinations are affected. A wide range of peak detection algorithms exist for, e.g. medical (electrocardiograms as in [20]) or chemical applications (mass spectrometry as in [21]). They mostly rely on an integrated preprocessing step, which in our case is already done by the KDE. Thus, the use of a simpler algorithm is valid. Using `argrelextrema` as implemented in [22], each element (in our case: discrete point in time t) is compared with the n elements to either side, referred as the *order*. For our application, in case it is the largest value among all others, the point is identified as a peak.

2.3.2 Spearman's Rank-Order Correlation Coefficient

The Spearman coefficient r_S measure the monotonic relationship between two variables. Its values range between $r_S \in [-1, 1]$, where $r_S = 1$ denotes absolute agreement in the same direction, $r_S = -1$ in the opposite. In chapter 3 we will argue that the Spearman coefficient is better suited for our application than the Pearson coefficient. This is mainly due to the assumptions necessary for applying the Pearson coefficient: "[t]he two variables have a bivariate normal distribution" and homoscedasticity, i.e. the relationship is equal across the whole data range. However, both measures are intererelated: the Spearman coefficient "is a special case of the Pearson [...] coefficient, if the latter measure is computed for two sets of ranks" [23]. It should be noted that the significane of the Spearman coefficient is governed by the correlated sample size. Only if enough data is available, the proability that a non-correlated pair of variables is interpretated as correlated, is low. This threshold is a function of the sample size an is listed in suitable tables. As we compare several correlations, the probability of errors is further lowered: we require multiple peaks to show the same relation before the variables are assumed to be correlated.

In case distinct integers are used as ranks, the coefficient can be calculated by using the rank difference $d_i = rank(X_i) - rank(Y_i)$ and the number of samples N:

$$r_S = 1 - \frac{6 \sum_i d_i^2}{N(N^2 - 1)}$$

2.4 Association Rule Learning

Association rule learning has its origins in analysing customer transaction data. Relations between products should be revealed by analysing which items are often bought together. Formally introduced in [24], each transaction t_i is identified by a binary vector where $t_i[k] = 1$ if item I_k from itemset $\mathcal{I} = I_1, ..., I_m$ was bought. For each subset $X \subseteq \mathcal{I}$ the association rule is the implication $X \Rightarrow I_j$, i.e. what other product is most likely to be needed in case the given other items are found in the basket.

Important quantities are the *confidence* and *support*. The confidence c describes in what percentage "of transactions in T that satisfy X also satisfy I_j". Intuitively, if more data backs the implication, it can be seen as a more reliable association. The support s describes the proportion of transactions $t_i \in T$ that contain X. A possible algorithm for association rule learning is the Apriori algorithm as presented in [25].

Chapter 3

Software Architecture

The software for the entire clustering process is separated in four main routines: data import, preprocessing, analysis, correlation and clustering. All parts are described with their respective sub-routines. Choices for the parameters introduced in chapter 2 are discussed with examples. During the thesis, graphical interfaces have been developped to see the influence of the respective parameter changes. These tools allowed us to get a detailed view of the data and select appropriate parameters for each software part; we will show descriptive plots at the appropriate places. For an easier overview of the entire software project, the structure is shown in figure 3.1.

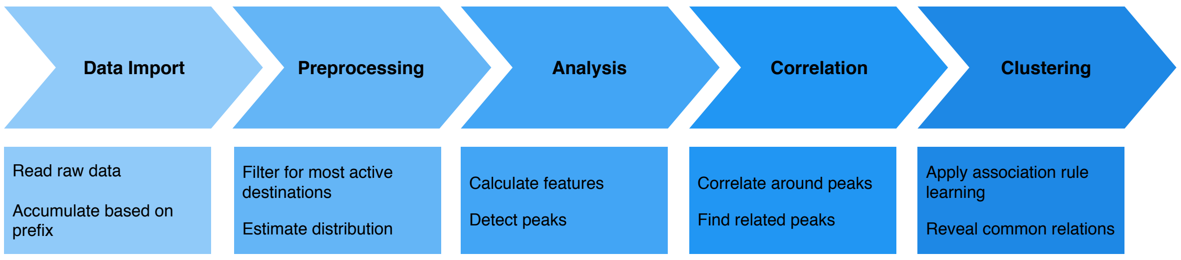


Figure 3.1: Overview about the Program Flow

3.1 Data Import

As a first step, the necessary information has to be extracted. For this purpose, `tshark`, a command line network protocol analyser from the `wireshark` project, was used. Due to the massive amount of data, only the necessary information was requested, ignoring other protocols than TCP. The 5-tuple describing the flow along with the timestamp and retransmission statistics are used. Retransmissions are identified by the `tcp.analysis.retransmission` flag which is true if "[t]he next expected sequence number is greater than the current sequence number" [26].

A major problem to solve is the inherent sparsity of the traffic data: flows are not active at all times, thus leaving time ranges with no values for the correlation process. A suitable solution would have been to only consider flows with enough data within the desired correlation window. However, in this case, less stable results are seen. Imagine the case of flow A, $B1$ and $B2$, where the latter two flows are known to derive from the same network. In case the respective feature of A correlates only to $B1$ at time instance t_0 and to $B2$ at t_1, then multiple results with the same cause are produced. Instead of accumulating after correlation, we decided to consider network destinations instead of single flows.

The detected flow is accumulated to a respective subnetwork based on a given prefix. According to the resulting destination network, the timestamp is categorised either as normal transmission or retransmission. All information is also added to a total count which we will compare the separate features to. It should be noted that we accumulate by the destination IP as this is where errors are detected. The traffic is recorded by a router in the middle of the routing path: only if packets pass the router, we are able to detect the retransmission. A simple example is shown in Figure 3.2, where four packets are sent: P1 and P3 are lost on the source and destination side respectively; P2 and P4 are their successful retransmission. Recording the traffic at point A, we cannot identify P2 as retransmission as P1 has not been seen; P4, however, is correctly identified as retransmission of P3. Thus, we cannot detect errors on the source path, but on the destination path.

Example: (tcp, 10.20.30.40, 50.60.70.80, 123, 456)[1] $\xrightarrow{/24}$ 50.60.70.0/24

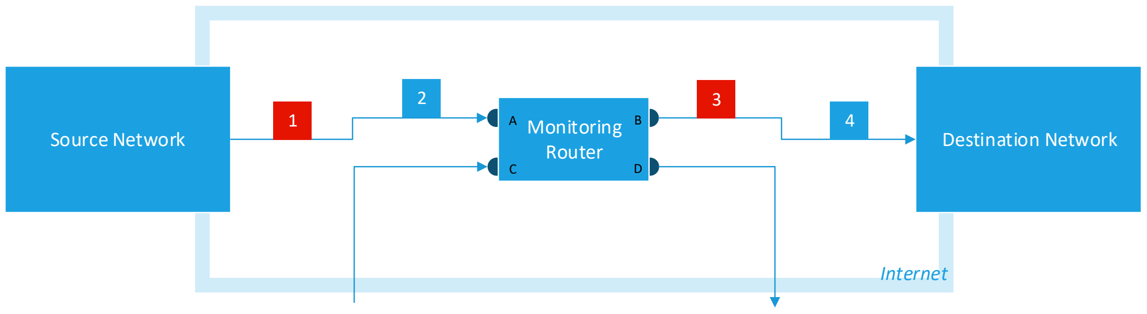


Figure 3.2: Simplified Overview about the Network Topology of the Traffic Records

3.1.1 Data Source

Throughout the thesis, the raw data is taken from the MAWILab database [27] and the Center for Applied Internet Data Analysis (CAIDA, [28]). Both provide anonymised Internet traces as `.pcap` files. For MAWI, the measurements are located at sample point F which represents a trans-Pacific link between Japan and the United States. For CAIDA, the `equinix-chicago` traces have been selected, which are located at a datacentre in Chicago. The recorded time ranges from multiple traces of 1 min to several days of 15 min each, which corresponds to roughly 100 million packets in total. Both datasets have been anonymised with a prefix-preserving algorithm. Table 3.1 gives an overview which data source has been used in which chapter.

Source	Date	Time	Chapter
CAIDA	19/02/2015	12:59:11	3.2
		13:00:00	4.1.1, 4.1.2
		13:01:00	4.1.1
		13:02:00	
MAWI	01/03/2018	14:00:00	4.2.1
	03/03/2018		
	05/03/2018		
	07/03/2018		

Table 3.1: Overview about the Used Data Sources

[1]Assuming the format (protocol, source IP, destination IP, source port, destination port)

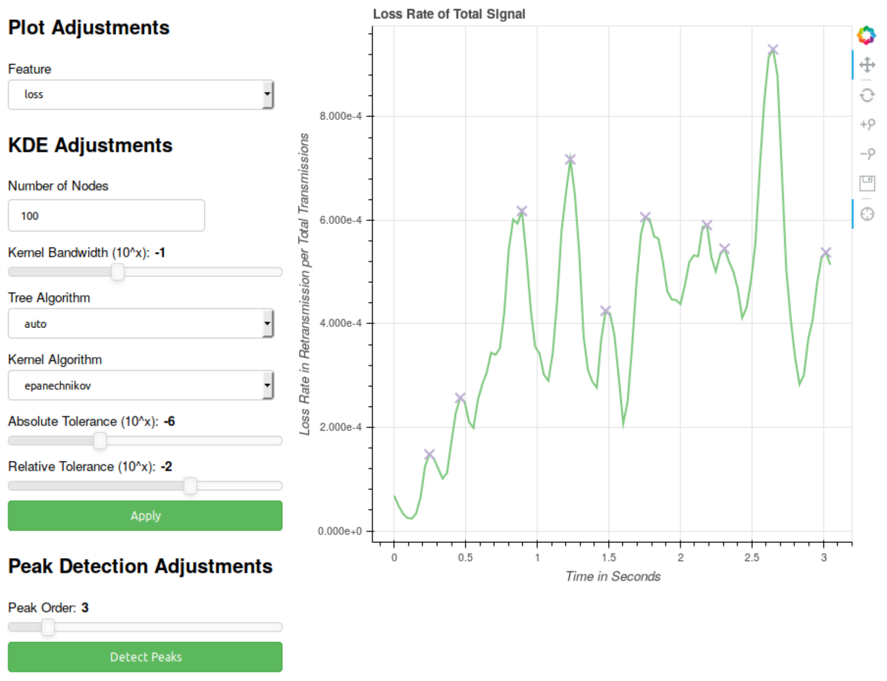
3.2 Preprocessing

Having an accumulated version of the raw data, further filtering is applied to increase the interpretability of the correlation process. The more data there is available per destination network, the more meaningful will be the correlation as finer dependencies can be revealed. Also, the computational burden lowers signficantly if fewer destinations are considered. We thus filter for destinations that make up a given percentage of the total amount of transmitted packets. This is done by sorting the destinations based on the amount of packets transmitted. From the top, destinations are added to the filtered list until reaching the desired threshold from the original total count. These destinations make the new total. In our analysis, we decided on a threshold of 80% as we are mainly interested in results backed by strong evidence. Please note, effectively less than 80% of the packets will be available after filtering as the algorithm stops as soon as the next destination would increase the packet count above the threshold.

The recorded transmission and retransmission timestamps are used to estimate a distribution for each by means of the KDE. Scikit-learn [29] has a suitable implementation in `sklearn.neighbors.KernelDensity`. We will optimise the necessary parameters for performance and interpretability of the data. As shown in [30], the runtime of the KDE is not only dependent on the amount of data, but also on the given kernel and bandwidth. Intuitively, compact kernels with less variation (e.g. the tophat kernel) reduce the runtime. However, because of the hard edges of these kernels, they produce a noisy outcome. Additionally, two other parameters arise in this special implementation of the KDE: the absolute and the relative error. Both influence the underlying tree-based data format where the fitted model is saved. We decided on a visual comparison of suitable alternatives to find the best match for our application. For this, we will plot the total loss rate which will be available after the analysis step. In the following, the first one million packets (corresponding to roughly 3 s) of `equinix-chicago` from the 19th of February, 2015, at 12:59:11 are taken. A graphical tool that was developped for this purpose during the thesis, was of great help for the comparison. Its user interface is shown in Figure 3.3.

In Figure 3.4 the KDE for the chosen parameters is plotted along with the detected peaks. All of the following plots base on estimations for 100 nodes and a peak order of 3 if not stated otherwise. We decided on an Epanechnikov kernel which provides a good trade-off between locality and preservation of information. These advantages become evident when looking at alternative kernels, Gaussian and Tophat, as shown in figure 3.5. For the same bandwidth, the Tophat kernel is much noisier causing an undesirable amount of peaks. In contrast, due to the smoothness and infinite support of the Gaussian kernel, potentially interesting peaks are erased. Although without a ground truth this choice is subjective, clear signs of over- and respectively underfitting can be seen. Measureing the runtime of `KernelDensity`'s `fit` routine for different kernels under the same other parameters, Tophat and Epanechnikov took 24 377 ms and 23 973 ms respectively, the Gaussian kernel 28 233 ms. This difference is likely to increase for a higher number of observations. For a detailed performance evaluation, please refer to [30]. In our case, the Epanechnikov kernel is a desirable choice between performance (compact kernel) and interpretability (preserves peaks).

Having decided on a suitable kernel function, the kernel bandwidth must be chosen accordingly. As shown in Figure 3.6, two extrema are visible. A small kernel size preserves the sparsity of the data; the result is merely a set of peaks where data points are available (in case of the loss rate: retransmissions). In contrast, a large kernel combines the underlying data to a indistinguishable function. Here, we argue that a bandwidth of 0.1 s suits the given data best. For the additional parameters, the estimation became noisy if taken larger than 10^{-6} for the absolute and 10^{-2} for the relative error. Hence, we took the stable values.


Plot Adjustments

Feature: loss

KDE Adjustments

Number of Nodes: 100

Kernel Bandwidth (10^x): -1

Tree Algorithm: auto

Kernel Algorithm: epanechnikov

Absolute Tolerance (10^x): -6

Relative Tolerance (10^x): -2

Peak Detection Adjustments

Peak Order: 3

Figure 3.3: Graphical Frontend for KDE Parameters

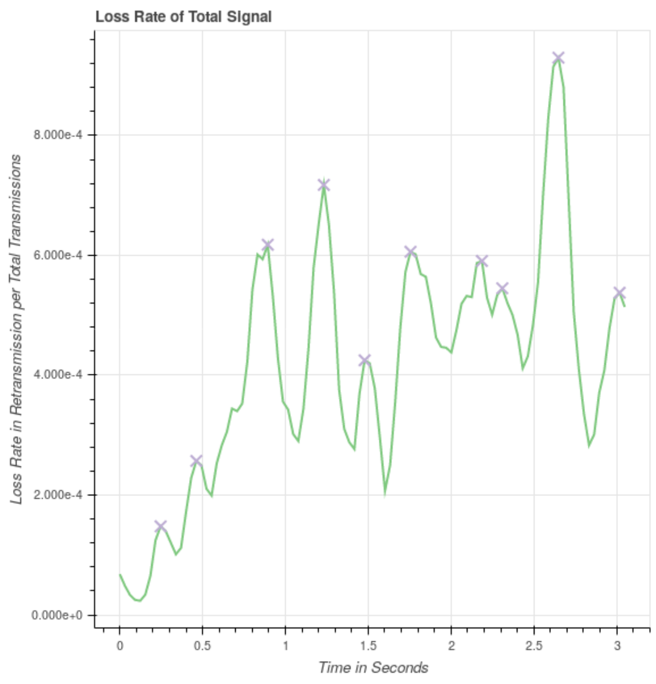
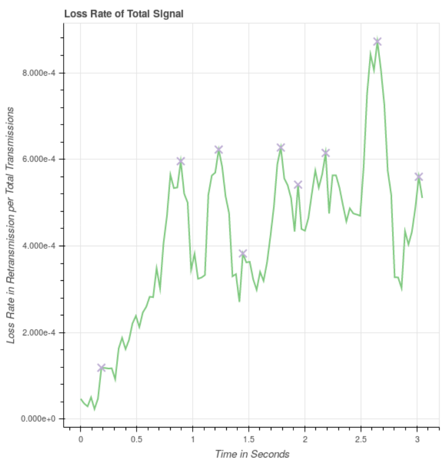
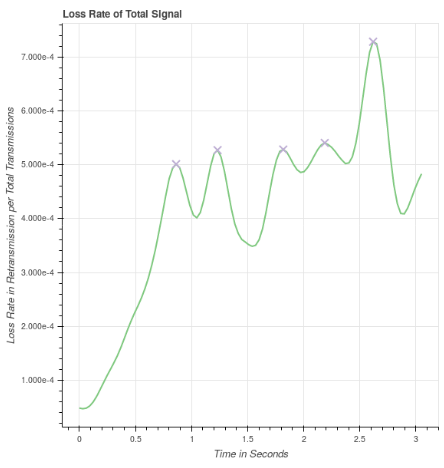


Figure 3.4: Chosen KDE Parameters: Epanechnikov Kernel with a Bandwidth of 0.1 s

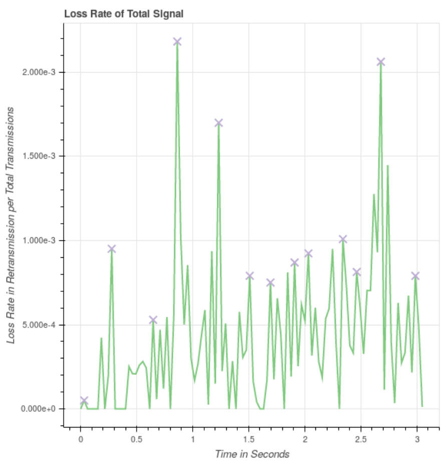


(a) Tophat Kernel

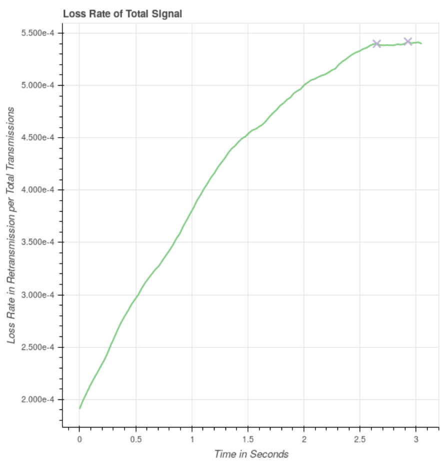


(b) Gaussian Kernel

Figure 3.5: Alternative Kernels for a Bandwidth of 0.1 s



(a) Bandwidth of 0.01 s



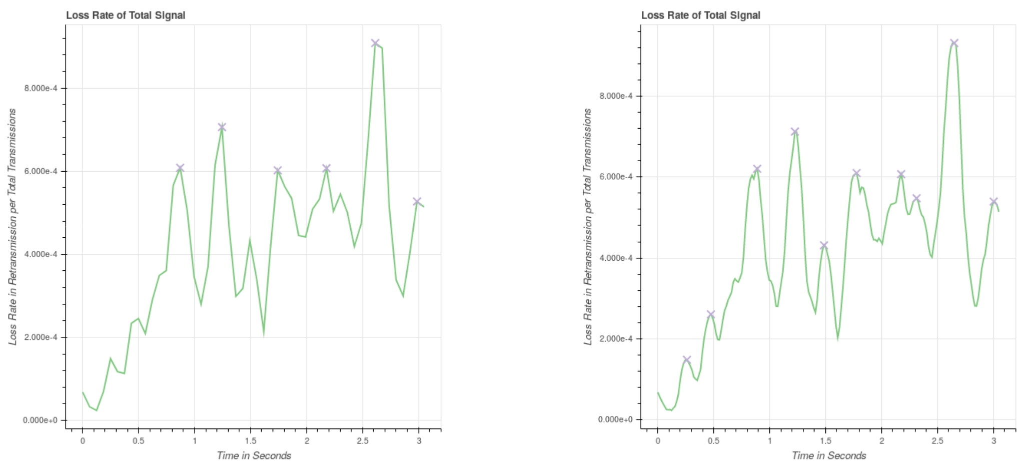
(b) Bandwidth of 1 s

Figure 3.6: Epanechnikov Kernel for Different Bandwidths

3.3 Analysis

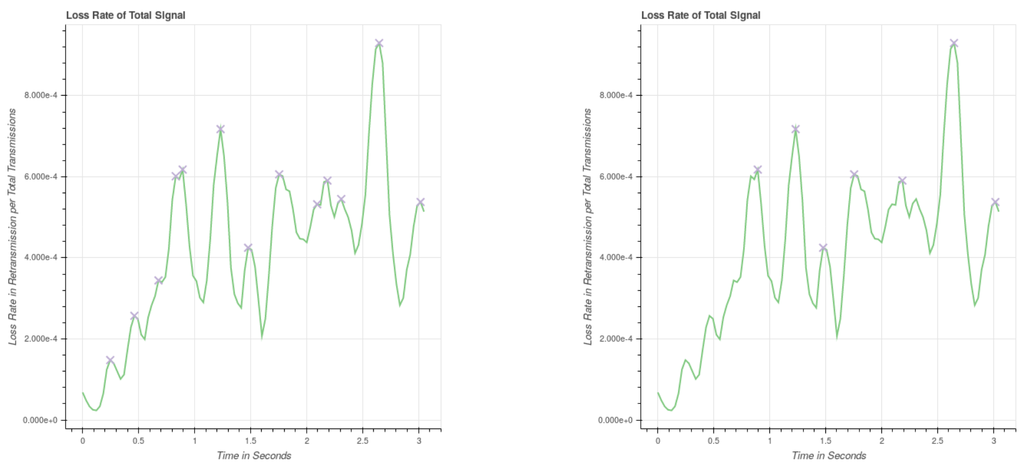
Thanks to the preprocessing step, a continuous estimator is available for destinations that have the most influence on the total signal. For further calculations, the estimation is calculated at the appropriate nodes, i.e. the desired time instances. The nodes further influence the peak detection as the estimate is further discretised if a long time interval lies between each node. We decided on 2000 nodes per analysed minute which corresponds to a gap of 30 ms between the nodes. In Figure 3.7, it can be seen that half the amount of nodes leads to loss in information, especially in the peaks; double the amount, however, does not add to any new information. Please note, as the data in this example has been limited to 3 s, 2000 nodes per minute correspond to $2000/60 * 3 = 100$ nodes for the given time interval. For this value, we have seen minor changes in the peak detection at adequate performance.

Most notably, the peak detection is influence by the selected order. During the initial analysis of the data, we noticed that not only the most prominent peaks bear valuable information for the correlation, but also smaller side peaks. For this reason, we decided on a low order which, however, does not trigger redundant detected peaks. This trade-off lies around 3. As shown in Figure 3.8, a smaller peak order separates a common peak into two, a higher one ignores side peaks.



(a) 50 Nodes (b) 200 Nodes

Figure 3.7: Estimation for Different Amount of Nodes



(a) Order of 1 (b) Order of 5

Figure 3.8: Peak Detection for Different Orders

3.4 Correlation

The main logic for the clustering phase is prepared during the correlation phase. Based on the detected peaks, every available destination is compared to the total signal. If a destination shows a similar loss pattern as the total count, we conclude that it is affected by a common reason. We decided on the Spearman coefficient as a suitable measurement of correlation.

As the loss behaviour is influenced by the amount and the timing at which packets arrive on a congested link, we do not expect linear dependency as measured by the Pearson coefficient. A destination where fewer packets arrive shows a higher percentage of retransmissions although the absolute retransmission count is low. Also, even though a destination is affected by the same congested link, the packets might have arrived just early enough such that they are not lost due to filled buffers, for example. Nonetheless, a similar pattern will evolve in each case where the loss rate rises to the maximum where the back-off mechanisms work. This relation is best described by a ranked correlation coefficient as the Spearman coefficient: we expect the shape to be similar, but not in a linear fashion. The interpretability of the Spearman coefficent is governed by the size of the sampled data: more correlated data will give the result more significane. For a suitable low error, but still accounting for relatively sparse data, we decided to correlate 10 time instances before and after each peak with each other. This corresponds to a total correlation window of $30\,\text{ms} \cdot 20 = 600\,\text{ms}$. Here, all correlation coefficients above 0.391 [23] show a probability of below 10% that the respective destinations is not correlated to the total flow. As mentioned in 2.3.2, this error is drastically reduced in our algorithm as we require multiple peaks to show the same correlation statistics before the peaks are assumed to be related.

For each peak, the correlation coefficient of each destination prefix to the aggregated signal is calculated. After sorting by the correlation coefficient, we will refer to it as the peak's top list. With the overview about which destinations correlate the most to each peak, the relation between the peaks can be measured. At first, each peak's top list is filtered based on the entry with the highest correlation coefficent. Each peak considered for the top list must have a minimal correlation coefficient corresponding to a fixed percentage of the top entry. We refer to this percentage as the *minimal top percentage* (MTP). Peaks are then labelled related if their respective top list overlaps by a fixed percentage. This percentage is know as the *minimal top overlap* (MTO). Intuetively, we expect a peak to have more related peaks if a smaller number of overlapping destinations is needed, i.e. the MTO is low. For a low MTO more related peaks will show if more destinations are in the top list, i.e. the MTP is also low. These two parameters will be the main variables throughout the results section in chapter 4.

3.5 Clustering

Having a list of top correlated destinations for each peak, association rule learning can be leveraged easily in the setting of Internet Tomography. Modelling the peaks as transactions t_i and the top destinations as items I_k, we try to find the underlying implications. We are interested in items that often appear together in the loss analysis, i.e. that likely are affected by a common congestion and thus share a common link. Converting the found peaks in a binary list of vectors $t_i[k]$ as defined in chapter 2.4, we are able to use the Apriori implementation of the Python toolset `mlxtend` [31]. In our analysis, we will request the top associations sorted by their support.

Chapter 4

Results

In the following, the performance of the proposed software architecture is evaluated on publicly available data. A main limitation with this data source is the missing ground truth because of the unknown network topology and a strict anonymisation. We will show that our algorithm is able to find correlations even given these obstacles. As visualisation, a cumulative distribution function (CDF) over the number of related peaks is chosen. This powerful representations allows to conclude what percentage of peaks is related to how many other peaks. Also, the results from the association rule learning are given.

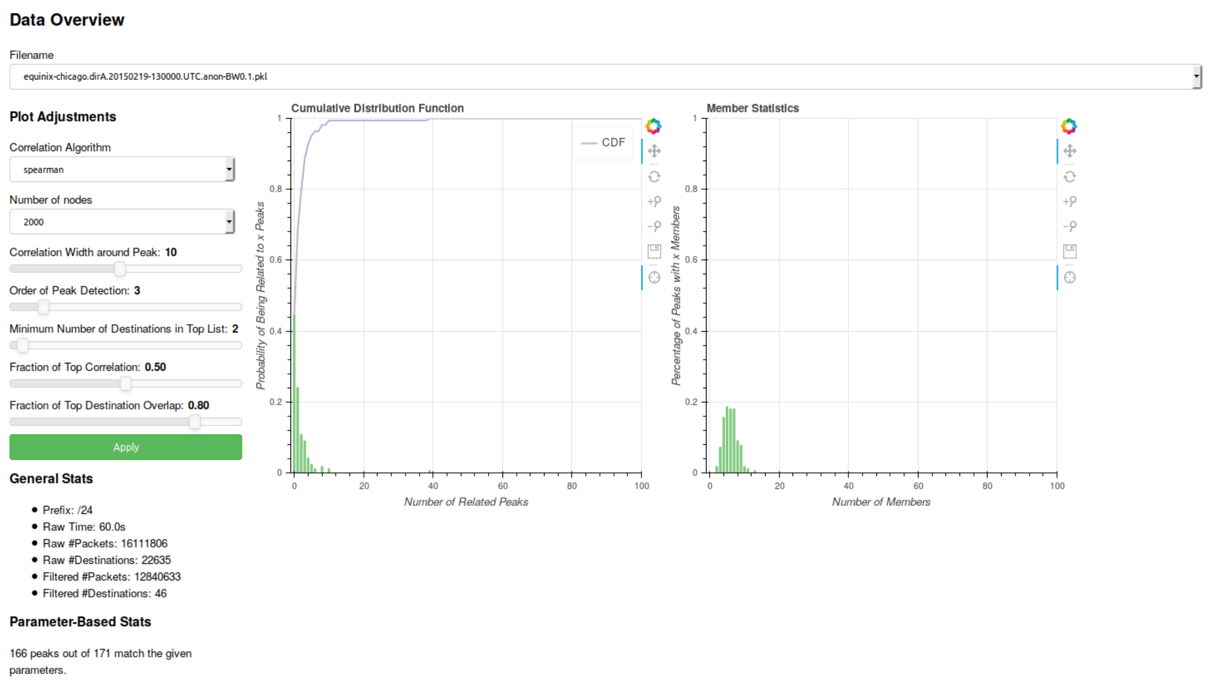
To visually inspect the impact of the parameters, an interactive graphical tool was developped during the thesis. A screenshot is shown in Figure 4.1. Throughout the analysis, we will apply two configurations, one with tight and one with loose parameters to the measures of relation. In numbers, in the tight case $(\mathrm{MTP}, \mathrm{MTO})_{tight} = (0.5, 0.8)$ wheras $(\mathrm{MTP}, \mathrm{MTO})_{loose} = (0.25, 0.6)$. From our intuition, we expect more related peaks under loose parameters as more destinations are available for the overlap and also less overlapping destinations are needed. To visualise the loss traces of each destination, we have further developped the tool shown in Figure 4.6. Here, a peak can be selected, showing all related ones including their top list.

4.1 Validation

To evaluate the validity of our correlation algorithm even on unknown data, we will first show that the expected behaviour is met. Two tests were designed for this purpose. The first uses an increasing time interval for the analysis under the assumption of an unchanged network topology. Here, the amount of related peaks should increase for more available peaks. We want to validate whether the algorithm does not show random correlation behaviour such that a strong basis for the clustering task is given. The second test uses a longer prefix during the data import to reveal correlations that are implied by the network characteristics. Here, known clusters should be revealed. By this, we validate that the algorithm performs the clustering task as expected.

4.1.1 Increasing Time Inverval

Throughout the first validation, an unchanged network topology during the time interval of the analysis is assumed. In this case, the same destinations should be correlated with each other when a congestion appears on the same network link. In other words, the longer the analysed time interval, the more peaks with a higher count of related peaks should show up: here the same congested links appears multiple times in the data. To fulfill the assumption, we will analyse three consecutive minutes of traffic data. Changes in the routing rules are done manually, such that the routing behaviour is well assumed to be constant during



Data Overview

Filename

equinix-chicago.dirA.20150219-130000.UTC.anon-BW0.1.pkl

Plot Adjustments

Correlation Algorithm: spearman

Number of nodes: 2000

Correlation Width around Peak: **10**

Order of Peak Detection: **3**

Minimum Number of Destinations in Top List: **2**

Fraction of Top Correlation: **0.50**

Fraction of Top Destination Overlap: **0.80**

General Stats

- Prefix: /24
- Raw Time: 60.0s
- Raw #Packets: 16111806
- Raw #Destinations: 22635
- Filtered #Packets: 12840633
- Filtered #Destinations: 46

Parameter-Based Stats

166 peaks out of 171 match the given parameters.

Apply
Cumulative Distribution Function
CDF
Probability of Being Related to x Peaks
Number of Related Peaks
Member Statistics
Percentage of Peaks with x Members
Number of Members

Figure 4.1: Graphical Frontend for CDF Parameters

this compact time span. Our analysis is based on CAIDA's `equinox-chicago` data in direction A for the 19th of February, 2015 during 13:00 and 13:03. The data was provided as three seperate files each containing one minute of traffic data. An overview about the analysed data before and after the preprocessing step is given in table 4.1.

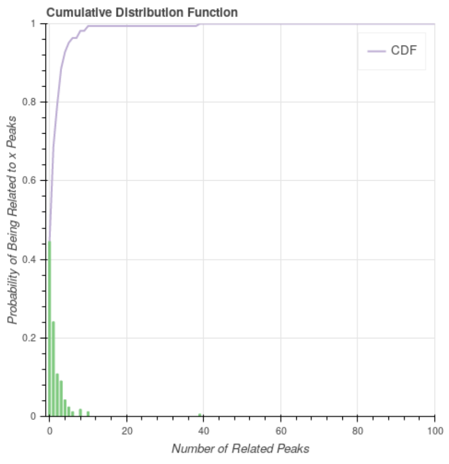
	Raw		Filtered		
Minute	#Packets	#Destinations	#Packets	#Destination	#Peaks[1]
0-1	16,111,806	22,635	12,840,633	46	166
1-2	16,655,946	22,561	13,265,038	45	180
2-3	17,758,364	22,838	14,194,720	43	178
0-2	32,767,752	30,107	26,116,635	46	346
0-3	50,526,116	35,939	40,312,751	45	528

Table 4.1: Time Series Data Statistics for CAIDA on the 19th of February, 2015, at 13:00

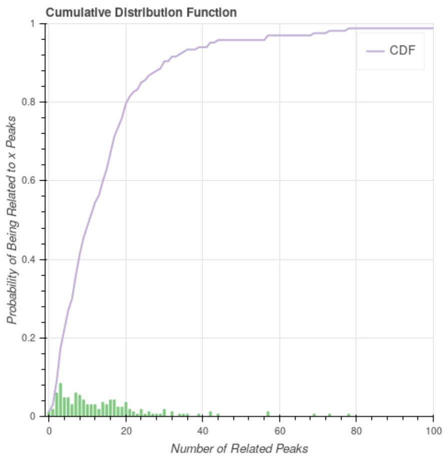
First, the three files are analysed separately. The first minute is shown in Figure 4.2 for tight and loose parameters; the other two minutes are shown in Figure 4.3 for tight relationship parameters. It can be seen that the CDF shows a similar behaviour in each analysis for the same setting. During the short time frame, 80% of the peaks are related to 2 or less related findings. There is little evidence of a higher number of related peaks. These are caused by short top destinations lists: here, the 80% MTO translates in two elements overlap due to rounding operations. Please note, that we require the top destinations lists to have at least two elements after setting the MTP. The loose setting increases the number of related peaks noticeably. Here, 80% of the flows are related to 20 or less peaks. Only 10% of the flows are related to 2 or less peaks: a stark contrast to 80% under the tight settings.

For the combined analysis, the data sets are merged to two and three minutes of consecutive data, see Figure 4.2. The CDF has moved to the right, showing that there are

[1]Number of usable peaks, i.e. where correlation window is not outside the signal

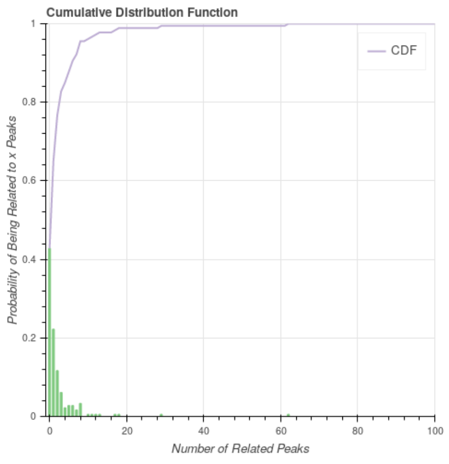


(a) Tight Settings

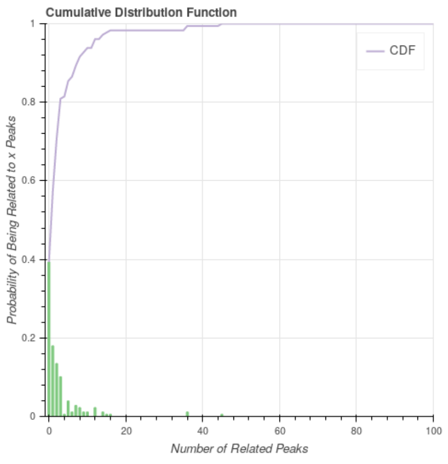


(b) Loose Settings

Figure 4.2: CDF for Time Series with Separated Data at Minutes 0 to 1

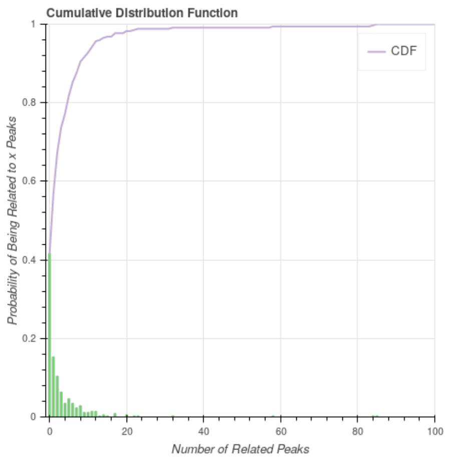


(a) Minutes 1 to 2

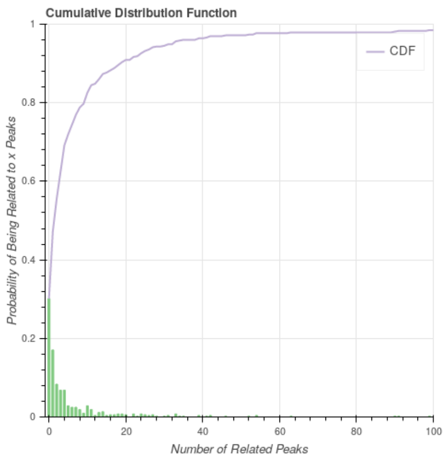


(b) Minutes 2 to 3

Figure 4.3: CDF for Time Series with Separated Data under Tight Settings

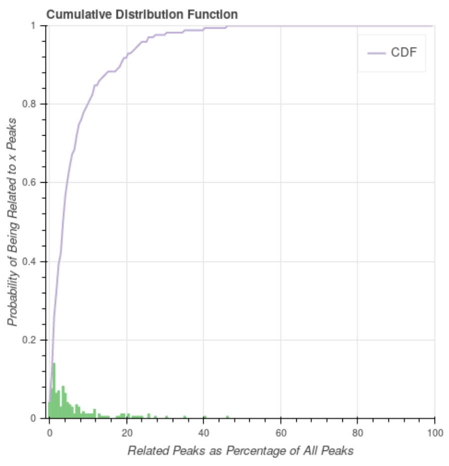


(a) Minutes 0 to 2

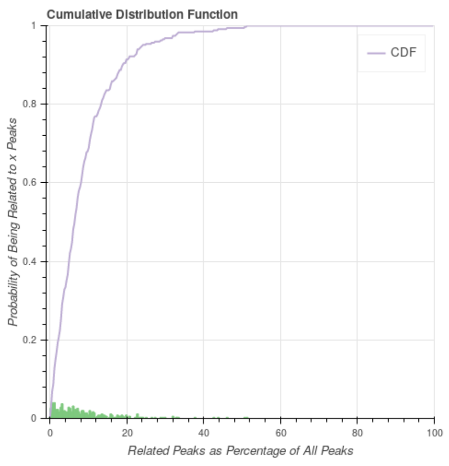


(b) Minutes 0 to 3

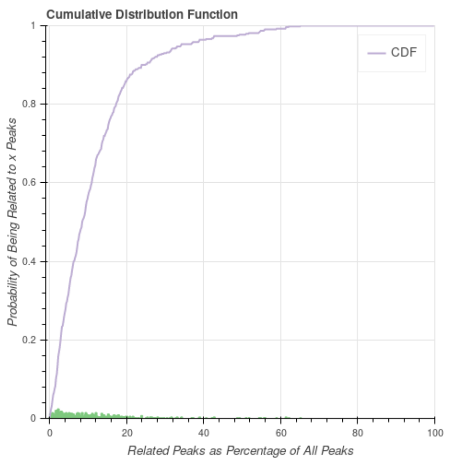
Figure 4.4: CDF for Time Series with Combined Data under Tight Settings



(a) Minutes 0 to 1



(b) Minutes 0 to 2



(c) Minutes 0 to 3

Figure 4.5: CDF for Time Series under Loose Settings, Percentages

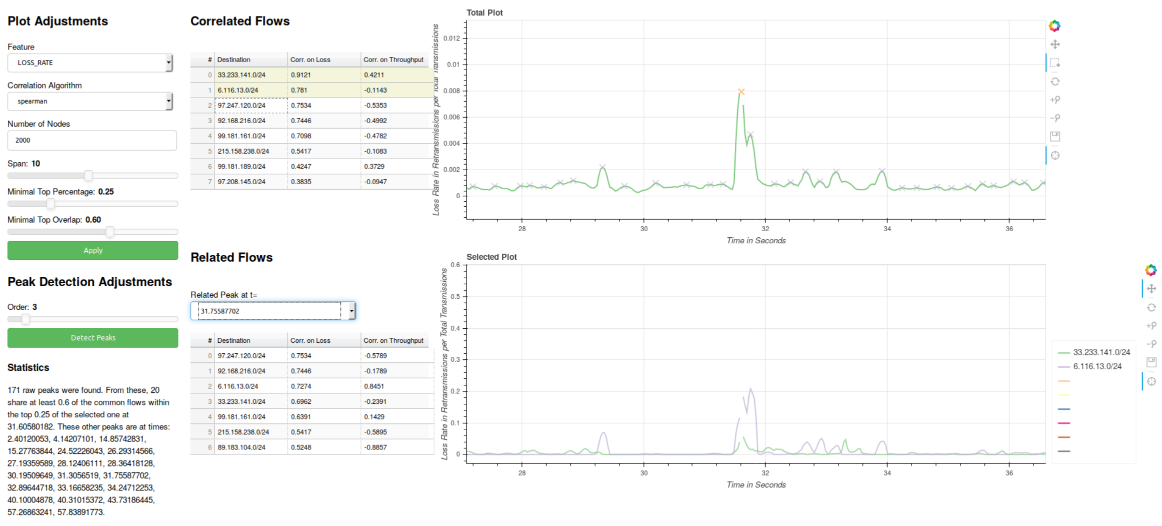
more peaks with a higher count of related peaks. Now, 80% of the peaks have 9 related peaks or less. This nicely supports the hypothesis: the correlation algorithm also reveals that commonly congested links are present throughout the whole time span.

It is also instructive to show the percentage of related peaks from all found ones instead of absolute numbers. In Figure 4.5 the combined data is plotted where the x-axis shows the percentage of related peaks to the number of overall peaks. As comparison to the previous plots, the loose settings have been used. It can be seen that with increasing time interval, the number of related peaks spreads more evenly. For short time intervals, there are many peaks that are not related to any other peaks. This behaviour is expected, as in a larger time span, there are more peaks that may be related to each other. As a result, the CDF smoothens, but keeps its shape.

In summary, the correlation algorithm behaves as expected in increasing time intervals. It is able to find correlations at several instances thus supporting the initial correlation by more examples. The parameters, MTP and MTO, influence the detection considerably and allow to finetune the correlation process to the desired scenario. We may conclude that our algorithm is able to reliably and flexibly detect common sources for the overall loss behaviour.

4.1.2 Longer Prefix

Both data sets, MAWI and CAIDA, use prefix-preserving anonymisation algorithms. In the Internet, announcements of prefixes longer than /24 are usually not propagated [32], such that we can approximately assume that longer prefixes belong to the same network. This said, all underlying flows should experience the same congestions as long as these happen outside the home networks. Consequently, in another point of validation the prefix prior to aggregation is extended to /26. We expect the algorithm to reveal correlations between networks that are known to derive from a common location. As data source, we use the first trace from the previous evaluation, i.e. CAIDA on the 19th of February, 2015, during 13:00 and 13:01. Another graphical tool that was developped as part of this thesis supports the analysis. It allows to read and compare the top destinations for a selected peak on the opened loss trace. Each top destination can be plotted separately, thus giving more information about the correlation decision. A screenshot is given in Figure 4.6.



Plot Adjustments

Feature: LOSS_RATE

Correlation Algorithm: spearman

Number of Nodes: 2000

Span: **10**

Minimal Top Percentage: **0.25**

Minimal Top Overlap: **0.60**

Peak Detection Adjustments

Order: **3**

Statistics

171 raw peaks were found. From these, 20 share at least 0.6 of the common flows within the top 0.25 of the selected one at 31.60580182. These other peaks are at times: 2.40120053, 4.14207101, 14.85742831, 15.27763844, 24.52226043, 26.29314566, 27.19359589, 28.12406111, 28.36418128, 30.19509649, 31.3056519, 31.75587702, 32.89644718, 33.16658235, 34.24712253, 40.10004878, 40.31015372, 43.73186445, 57.26863241, 57.83891773.

Correlated Flows

#	Destination	Corr. on Loss	Corr. on Throughput
0	33.233.141.0/24	0.9121	0.4211
1	6.116.13.0/24	0.781	-0.1143
2	97.247.120.0/24	0.7534	-0.5353
3	92.168.216.0/24	0.7446	-0.4992
4	99.181.161.0/24	0.7098	-0.4782
5	215.158.238.0/24	0.5417	-0.1083
6	99.181.189.0/24	0.4247	0.3729
7	97.208.145.0/24	0.3835	-0.0947

Related Flows

Related Peak at t= 31.75587702

#	Destination	Corr. on Loss	Corr. on Throughput
0	97.247.120.0/24	0.7534	-0.5789
1	92.168.216.0/24	0.7446	-0.1789
2	6.116.13.0/24	0.7274	0.8451
3	33.233.141.0/24	0.6962	-0.2391
4	99.181.161.0/24	0.6391	0.1429
5	215.158.238.0/24	0.5417	-0.5895
6	89.183.104.0/24	0.5248	-0.8857

Figure 4.6: Graphical Frontend for Related Peaks

Thanks to the tool, we were quickly able to find a peak where two /26 subnets belonging to the same /24 subnet are among the top correlations. The total loss and the separate loss rates are shown in Figure 4.8. Within the top list, 50.224.90.128/26 was rated with 0.7985 on the third, 50.224.192/26 with 0.7323 on the fifth place. The top entry had a correlation coefficient of 0.8099. All entries are shown in table 4.2a. In the separated plot, it can be nicely seen that the trace indeed follows the shape around the selected peak, but not at other time instances. This peak is an example for high, closely grouped correlation coefficients among the destinations. Ideally, this is a sign of an underlying cluster.

Applying the association rule learning algorithm to the overall data, the result as shown in Table 4.2b is derived. These are the top association rules with at least two members sorted by their support. It is a reaffirming result, that the two aforementioned subnets are clustered in around 15% of all peaks. The second entry shows that 50.224.192/26 was associated with 99.181.161.128/26 in more than 22% of all peaks, also in the one shown in Table 4.2a.

Summarising the validation, we see strong evidence that the correlation algorithm behaves as expected. Although no data with a known ground truth was available, we have shown that correlations are found among different traces and moreover, among subnetworks that are assumed to share a common path. We therefore conclude that the algorithm is suitable to detect clusters of destinations in passively probed network traffic.

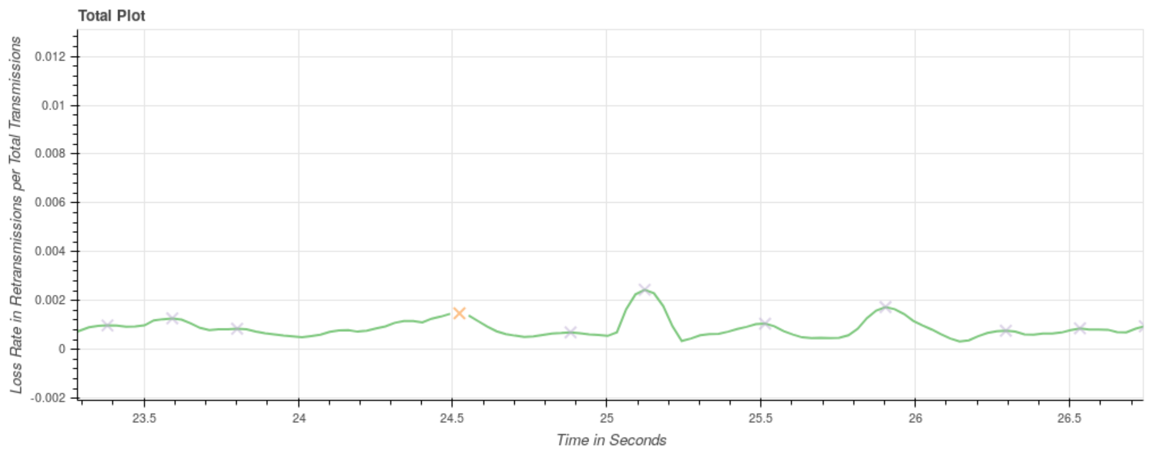


Figure 4.7: Total Loss Rate around the Selected Peak (orange, $t \approx 24.5\,\mathrm{s}$)

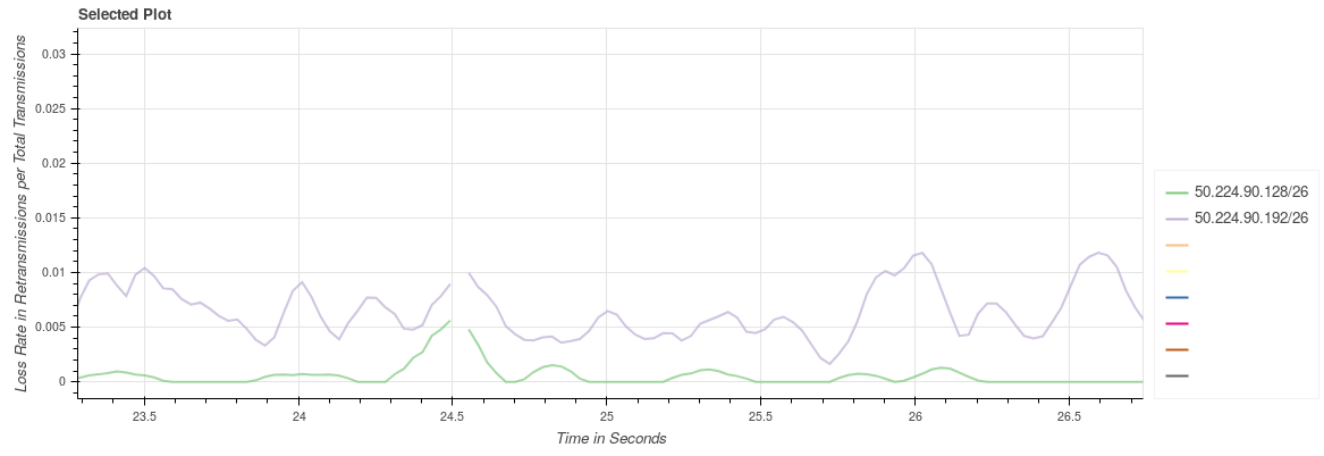
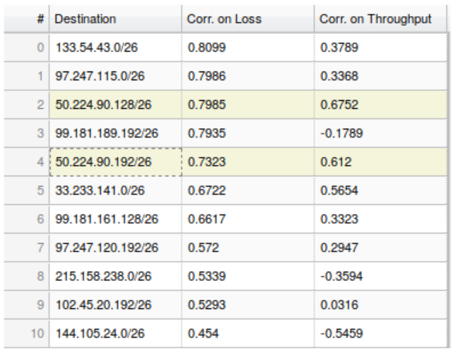
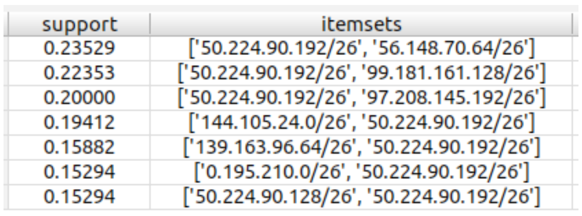


Figure 4.8: Selected Loss Rate of Two Related /26 Subnets



#	Destination	Corr. on Loss	Corr. on Throughput
0	133.54.43.0/26	0.8099	0.3789
1	97.247.115.0/26	0.7986	0.3368
2	50.224.90.128/26	0.7985	0.6752
3	99.181.189.192/26	0.7935	-0.1789
4	50.224.90.192/26	0.7323	0.612
5	33.233.141.0/26	0.6722	0.5654
6	99.181.161.128/26	0.6617	0.3323
7	97.247.120.192/26	0.572	0.2947
8	215.158.238.0/26	0.5339	-0.3594
9	102.45.20.192/26	0.5293	0.0316
10	144.105.24.0/26	0.454	-0.5459

(a) Top Destinations from the Above Peak



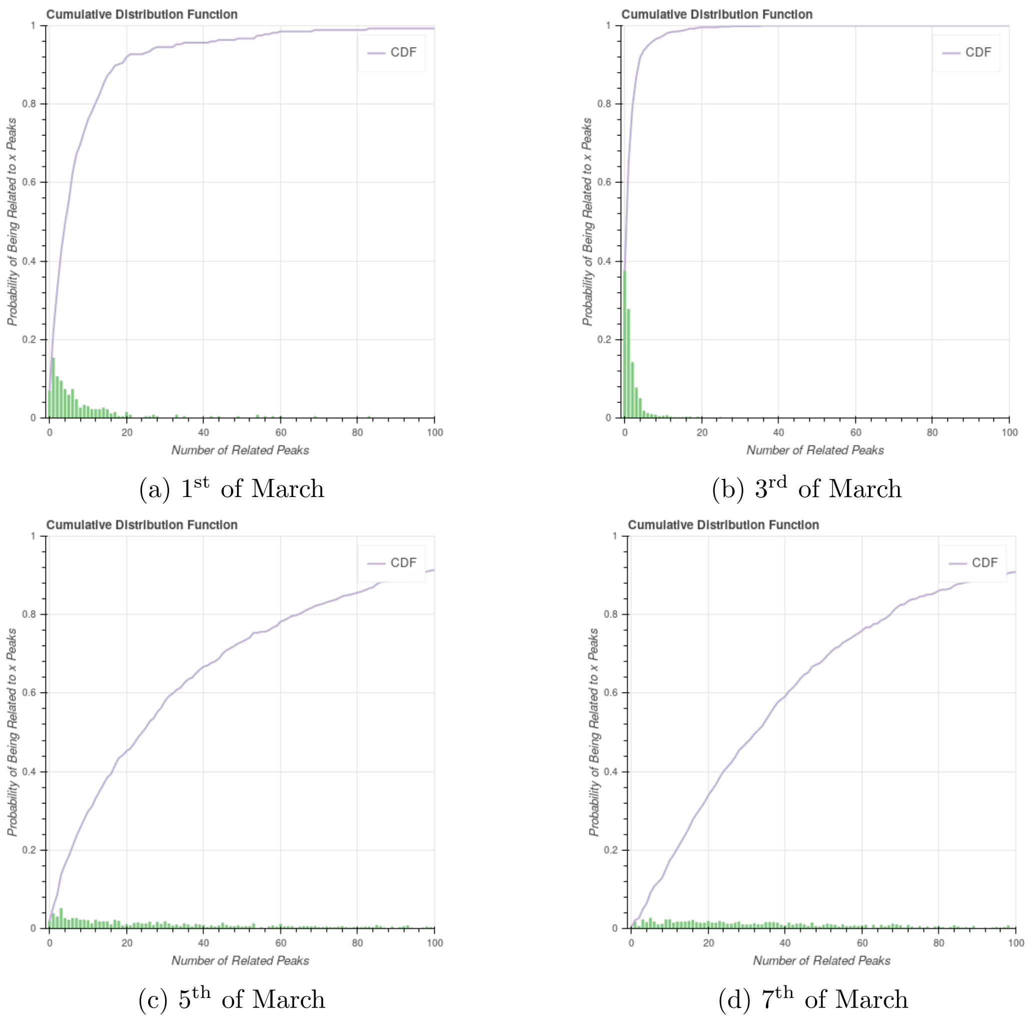
support	itemsets
0.23529	['50.224.90.192/26', '56.148.70.64/26']
0.22353	['50.224.90.192/26', '99.181.161.128/26']
0.20000	['50.224.90.192/26', '97.208.145.192/26']
0.19412	['144.105.24.0/26', '50.224.90.192/26']
0.15882	['139.163.96.64/26', '50.224.90.192/26']
0.15294	['0.195.210.0/26', '50.224.90.192/26']
0.15294	['50.224.90.128/26', '50.224.90.192/26']

(b) Top Overall Association Rules with Minimum Support of 15%

Table 4.2: Top Destinations in the Selected Peak and the Overall Association Rule Learning Response

4.2 Additional Findings

Knowing that the algorithm performs as expected, we would like to give additional interesting findings. These have not been made under any assumption, but reflect the information obtained by the correlation algorithm.



(a) 1st of March (b) 3rd of March (c) 5th of March (d) 7th of March

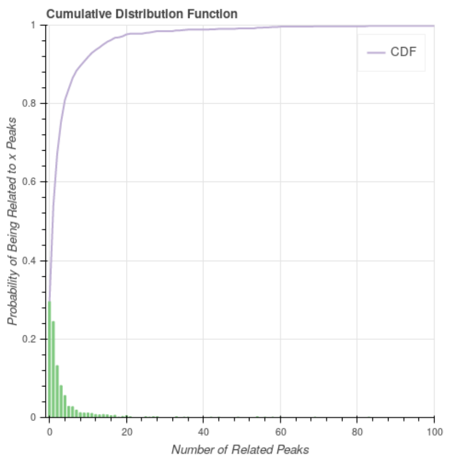
Figure 4.9: CDF of Separate Days under Loose Settings for March 2018

4.2.1 Series of Days

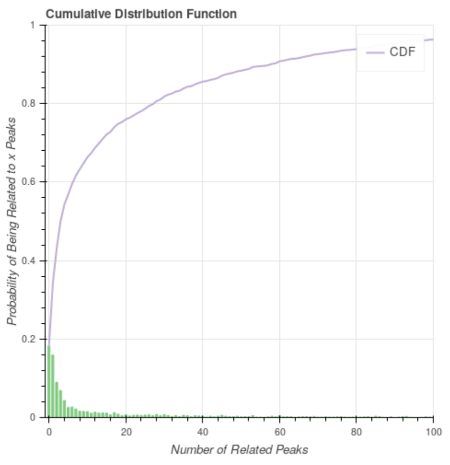
As we have analysed the performance for a time series in the range of minutes, an interesting next step is the analysis of several days. Here, we expect the assumption of unchanged network topologies to be violated. We might not detect an overlap in the most common destinations. Possible implications might be less related peaks among the different day. For the separate days, a similar distribution as in the previous cases should be seen.

In the following, four days in March 2018 are analysed: the 1st (Thursday), 3rd (Saturday), 5th (Monday) and 7th (Wednesday) of the month. These days have been chosen to capture the traffic of an entire week, which might give interesting results whether the results differ and correlate differently based on the week day. The data has been taken from the MAWI database and has been analysed for the first 25 million packets which corresponds to roughly 4 min of data. As we take 2000 nodes per minute throughout the analysis, we thus require 8000 nodes here. Loose parameters have been taken for the analysis such that relations are revealed even under the changed environment. An overview about the separate days is given in Figure 4.9.

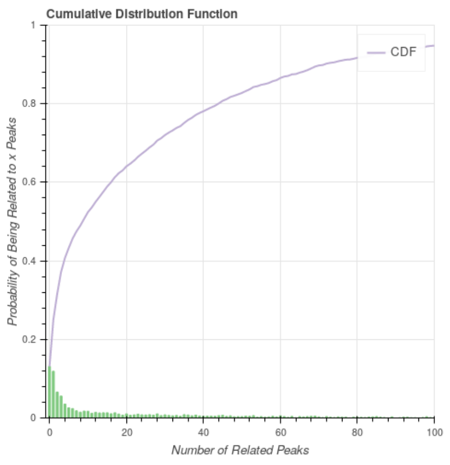
It can be seen that the days considerably differ in their CDF. Most notably, in the last two days much more peaks with a high number of related peaks are present. For more insights, the top 5 results of the association rule learning are shown in Table 4.4, sorted by their support requiring at least 3 entries. There are no matching subnetworks within these



(a) March, the 1st and 3rd



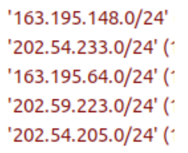
(b) March, the 1st, 3rd and 5th



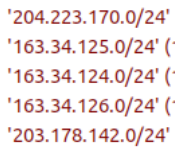
(c) March, the 1st, 3rd, 5th and 7th

Figure 4.10: CDF of Combined Days under Loose Settings for March 2018

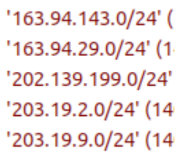
four sets. To further analyse this problem, the top 5 destination prefixes based on their transmission packet count are shown in Table 4.3. Intuetively, there should be overlaps in the most common destinations as these probably belong to popular services; however, also here no overlaps can be found. Nonetheless, the structure of the associated itemsets is very similar in that two networks starting with 163... are grouped with one network in the 202... range on all days but the 3rd. Without interpretating too much in the data, this could be a sign that a different random seed for the anonymisation process is used at separate days; although the apparent names change, the same structures are revealed. Another, simpler explanation can still be that not the same destinations are active during the lapse of different days.


'163.195.148.0/24'
'202.54.233.0/24' (
'163.195.64.0/24' (
'202.59.223.0/24' (
'202.54.205.0/24' (

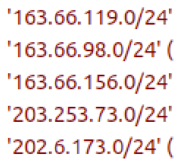
(a) 1st of March


'204.223.170.0/24'
'163.34.125.0/24' (
'163.34.124.0/24' (
'163.34.126.0/24' (
'203.178.142.0/24'

(b) the 3rd of March

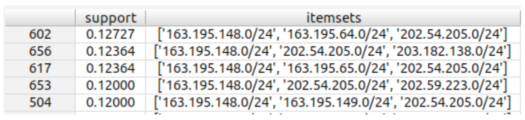

'163.94.143.0/24' (
'163.94.29.0/24' (1
'202.139.199.0/24'
'203.19.2.0/24' (14
'203.19.9.0/24' (14

(c) 5th of March


'163.66.119.0/24'
'163.66.98.0/24' (
'163.66.156.0/24'
'203.253.73.0/24'
'202.6.173.0/24' (

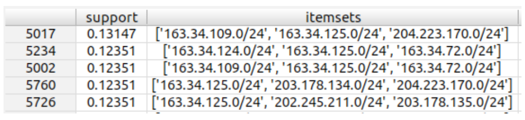
(d) 7th of March

Table 4.3: Top 5 Destination Prefixes Based on their Transmission Packet Count for March 2018



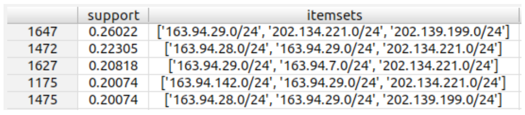
	support	itemsets
602	0.12727	['163.195.148.0/24', '163.195.64.0/24', '202.54.205.0/24']
656	0.12364	['163.195.148.0/24', '202.54.205.0/24', '203.182.138.0/24']
617	0.12364	['163.195.148.0/24', '163.195.65.0/24', '202.54.205.0/24']
653	0.12000	['163.195.148.0/24', '202.54.205.0/24', '202.59.223.0/24']
504	0.12000	['163.195.148.0/24', '163.195.149.0/24', '202.54.205.0/24']

(a) 1st of March



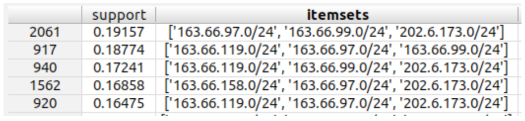
	support	itemsets
5017	0.13147	['163.34.109.0/24', '163.34.125.0/24', '204.223.170.0/24']
5234	0.12351	['163.34.124.0/24', '163.34.125.0/24', '163.34.72.0/24']
5002	0.12351	['163.34.109.0/24', '163.34.125.0/24', '163.34.72.0/24']
5760	0.12351	['163.34.125.0/24', '203.178.134.0/24', '204.223.170.0/24']
5726	0.12351	['163.34.125.0/24', '202.245.211.0/24', '203.178.135.0/24']

(b) the 3rd of March



	support	itemsets
1647	0.26022	['163.94.29.0/24', '202.134.221.0/24', '202.139.199.0/24']
1472	0.22305	['163.94.28.0/24', '163.94.29.0/24', '202.134.221.0/24']
1627	0.20818	['163.94.29.0/24', '163.94.7.0/24', '202.134.221.0/24']
1175	0.20074	['163.94.142.0/24', '163.94.29.0/24', '202.134.221.0/24']
1475	0.20074	['163.94.28.0/24', '163.94.29.0/24', '202.139.199.0/24']

(c) 5th of March



	support	itemsets
2061	0.19157	['163.66.97.0/24', '163.66.99.0/24', '202.6.173.0/24']
917	0.18774	['163.66.119.0/24', '163.66.97.0/24', '163.66.99.0/24']
940	0.17241	['163.66.119.0/24', '163.66.99.0/24', '202.6.173.0/24']
1562	0.16858	['163.66.158.0/24', '163.66.97.0/24', '202.6.173.0/24']
920	0.16475	['163.66.119.0/24', '163.66.97.0/24', '202.6.173.0/24']

(d) 7th of March

Table 4.4: Top 5 Association Rules with at least 3 Elements of Separate Days under Loose Settings for March 2018

These findings influence the possible interpretations of the combined view. The plots in Figure 4.10 show the CDF of the combined days. We observe, the first combined plot resembles the shape of the 3rd of March, and the last two combined plots the ones of the 5th and 7th of March. Also there is merely a change between the last two combined plots. Overall, it is likely that these plots seem rather to combine the data seen in the separate plots, not to show new related peaks among them.

We may summarise that we likely have not found relations between data of separate days. As the algorithm has passed the validation steps, we rather assume that the explanation is found in the raw data, potentially in the used anonymisation algorithm. This result is included nonetheless, as instructive example for this thesis.[2]

4.2.2 Comparison to Throughput

The analysis is based on the loss rate and the correlation coefficient on this feature. As comparison, we have calculated the Spearman coefficient on the throughput for the same peaks as found in the aggregated loss rate. For each top list, we have generated the top list of destinations based on the correlation regarding the loss rate; additionally, we have noted the equivalent correlation on the throughput. We have not seen any relation between the top entries based on the loss rate and the top entries based on the throughput. For the top entries of the former, the latter shows correlation coefficients in the whole range of values. We thus conclude that features based on the loss rate and throughput have to be considered separately.

[2]On the website of the MAWI traffic archive ([33]) is mentioned: “Traffic traces are made by tcpdump, and then, IP addresses in the traces are scrambled by a modified version of tcpdpriv”. The source code of the modified version of `tcpdpriv` is available. In its `main()` function, a call to `rand_start()` is done. This function’s description is: “at startup, generate a seed for the random number generator”. If each traffic trace is modified by a individual process of `tcpdpriv`, the IP addresses are indeed anonymised in different ways.

Chapter 5

Outlook

Having seen the performance of the algorithm, we are eager to further streamline the clustering process. Based on the outcome of the association rule learning algorithm, we envision further data mining steps for even richer results. The proposed algorithm could be the base for big data projects that analyse longer time intervals. From these, future work may develop interfaces where related networks are suggested to an IP address input; or even the visualisation of whole clusters up to the deciphered parts of the network topology.

To further increase the confidence in the clustering process and optimise the available parameters, the algorithm should be evaluated on other data sources, also artificial ones. Knowing the ground truth of the underlying network, would allow us to optimise the performance even further. We imagine a complete grid search over the parameters with a rating function based on the known network topology. This is left as future work, the scope of this semester thesis was the design and validation of the algorithm.

Thanks to the modularity of the algorithm, it can be easily adapted to other network scenarios. For example, other types of features may suit network types that are not limited by congestions or losses. Future work may evaluate the performance of the separate parts of the algorithm for given network topologies. Moreover, it would be interesting to implement the methods suggested in the related work. Thus, we are confident that the proposed algorithm can be used as a basis for future work. Its flexibility will allow to see further performance increases in the clustering process.

Chapter 6

Summary

In this thesis, a flexible, yet powerful algorithm for Internet Tomography has been designed and evaluated. The validation has shown that related peaks are correlated and appropriate clustering decisions are taken. We were able to reveal links that are affected by the same incident. A pair of easy-to-grasp parameters allows to finetune the behaviour to the desired setting. Thanks to passive probing, it does not add any overhead to the network links.

The evaluation process has revealed clusters of networks in publicly available data. We were able to apply the theory behind association rule learning to a network setting, to show related network links. As the algorithm was built with modularity in mind, it can be adapted for other network types. Thus, a strong basis for the analysis of network topologies has been developped.

Even though the underlying ground truth was not available for the used data set, we were able to show that:

- the peaks in the loss data reveal correlations between network destinations,
- the correlations repeat over time when analysing more data,
- the clustering based on the peak data reveals related destinations of the network topolgy.

During this thesis, several graphical tools have been developped, allowing to adapt and evaluate the parameters of the clustering algorithm. The explored theory has been implemented as easy-to-integrate libraries written in Python. We are confident that the contributions of this thesis will help future work in revealing correlations in network traffic.

Appendix A

Appendix

A.1 Abbreviations

Abbr.	**Meaning**
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
ISP	Internet Service Provider
KDE	Kernel Density Estimation
MSS	Maximum Segment Size
MTO	Minimal Top Overlap
MTP	Minimal Top Percentage
QoS	Quality of Service

Bibliography

[1] M. Coates, A. O. Hero, R. Nowak, and B. Yu, “Internet tomography,” *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[2] Cisco, “The Zettabyte Era: Trends and Analysis,” 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

[3] Sandvine, “2016 - Global Internet Phenomena - Latin America & North America,” 2016. [Online]. Available: https://www.sandvine.com/hubfs/downloads/archive/2016-global-internet-phenomena-report-latin-america-and-north-america.pdf

[4] Netflix Inc., “Open Connect Overview,” 2016. [Online]. Available: https://openconnect.netflix.com/en/

[5] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien, “Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 139–152, 2015.

[6] T. He, C. Liu, A. Swami, D. Towsley, T. Salonidis, A. I. Bejan, and P. Yu, “Fisher Information-based Experiment Design for Network Tomography,” *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems - SIGMETRICS ’15*, pp. 389–402, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2745844.2745862

[7] X. B. Fan and X. Li, “Network tomography via sparse Bayesian learning,” *IEEE Communications Letters*, vol. 21, no. 4, pp. 781–784, 2017.

[8] F. Lo Presti, N. G. Duffield, J. Horowitz, and D. Towsley, “Multicast-based inference of network-internal delay distributions,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 761–775, 2002.

[9] K. Lan and J. Heidemann, “On the correlation of internet flow characteristics,” pp. 1–12, 2003. [Online]. Available: ftp://info.isi.edu/isi-pubs/tr-574.pdf

[10] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, “I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks,” *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2014)*, pp. 71–85, 2014. [Online]. Available: http://blogs.usenix.org/conference/nsdi14/technical-sessions/presentation/handigol

[11] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, “Accurate Latency-based Congestion Feedback for Datacenters,” *Atc*, 2015. [Online]. Available: https://www.usenix.org/conference/atc15/technical-session/presentation/lee-changhyun

[12] M. S. Kim, T. Kim, Y. J. Shin, S. S. Lam, and E. J. Powers, “Scalable clustering of Internet paths by shared congestion,” *Proceedings - IEEE INFOCOM*, 2006.

[13] K. Harfoush, A. Bestavros, and J. Byers, “Robust Identification of Shared Losses Using End-to-End Unicast Probes,” Boston University, Tech. Rep., 2000.

[14] M. Allman and V. Paxson, “RFC5681: TCP Congestion Control,” pp. 1–18, 2009.

[15] J. F. Kurose and K. W. Ross, “TCP Congestion Control,” in *Computer Networking: A Top-Down-Approach*, 6th ed. Pearson, 2013, pp. 295–308.

[16] J. Pahdye and S. Floyd, “On inferring TCP behavior,” *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 287–298, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?doid=964723.383083

[17] W. Putthividhya and C. Papadopoulos, “About Fast-Recovery Algorithm,” 2000. [Online]. Available: https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_WANIDA/DR/JavisInActionFastRecoveryFrame.html

[18] R. Morris, “TCP behavior with many flows,” *Proceedings 1997 International Conference on Network Protocols*, no. October, pp. 205–211, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/643715/

[19] C. M. Bishop, *Pattern recognition and machine learning.* New York, NY: Springer, 2007.

[20] M. S. Manikandan and K. P. Soman, “A novel method for detecting R-peaks in electrocardiogram (ECG) signal,” *Biomedical Signal Processing and Control*, vol. 7, no. 2, pp. 118–128, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.bspc.2011.03.004

[21] P. Du, W. A. Kibbe, and S. M. Lin, “Improved peak detection in mass spectrum by incorporating continuous wavelet transform-based pattern matching,” *Bioinformatics*, vol. 22, no. 17, pp. 2059–2065, 2006.

[22] E. Jones, T. Oliphant, P. Peterson, and Others, “SciPy: Open source scientific tools for Python,” 2011. [Online]. Available: http://www.scipy.org/

[23] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, 5th ed. Boca Raton: CRC Press, 2011.

[24] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association in Large Databases,” *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD ’93*, pp. 207–216, 1993.

[25] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases,” *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB ’94)*, pp. 487–499, 1994.

[26] Wireshark, “TCP Analysis,” p. 3. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html

[27] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, “MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking,” *Proceedings of the 6th International Conference on - Co-NEXT ’10*, pp. 1–12, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1921168.1921179

[28] CAIDA, "The CAIDA UCSD Anonymized Internet Traces 2015," 2015. [Online]. Available: http://www.caida.org/data/passive/passive_2015_dataset.xml

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[30] J. VanderPlas, "Kernel Density Estimation in Python," 2013. [Online]. Available: https://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation/

[31] S. Raschka, "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, apr 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[32] E. Aben and C. Petrie, "Propagation of Longer-than- / 24 IPv4 Prefixes," pp. 1–6, 2014. [Online]. Available: https://labs.ripe.net/Members/emileaben/propagation-of-longer-than-24-ipv4-prefixes

[33] "MAWI Working Group Traffic Archive." [Online]. Available: http://mawi.nezu.wide.ad.jp/mawi/