FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Network anomaly detection based on traceroute data
Student:	Bc. Jan Peřina
Supervisor:	doc. Ing. Kamil Dedecius, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

Abstract: The CERN LHC experiment runs a large network of devices that are interconnected over the global Internet. These devices regularly or irregularly exchange vast amounts of data, which imposes significant requirements on the communication paths. In particular, the stability and efficiency of the routes are required. In the CERN practice, this is facilitated by regular monitoring of the routes by means of the traceroute protocol. The aim of the thesis is to propose a new tool for the detection and analysis of routing anomalies in the network.

Goals:
1) Analyse the state of the art in the domain.
2) Select or propose an alternative method (or methods) for detecting and analyzing routing anomalies. Develop its (or their) experimental implementation using convenient statistical and/or machine learning tools.
3) Validate the results from the previous point on available CERN data.
4) Discuss obtained results, limitations of the method, and the approach in general, and propose possible improvements and future work.

References:
Wu, R. and Keogh, E.J. "Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress." 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, 2022.
Malkin, G. "Traceroute Using an IP Option", RFC 1393, January 1993.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 14 June 2023 in Prague.

Marchetta, P. et al. "How and how much traceroute confuses our understanding of network paths". 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN): 1-7, 2016.

Master's thesis

NETWORK ANOMALY DETECTION BASED ON TRACEROUTE DATA

Bc. Jan Peřina

Faculty of Information Technology
Department of applied mathematics
Supervisor: doc. Ing. Kamil Dedecius, Ph.D.
Co-Supervisors: RNDr. Dagmar Adamová, CSc.[1],
Dr. Shawn McKee[2]
January 11, 2024

[1]Nuclear Physics Institute of the Czech Academy of Sciences
[2]Research Scientist - The University of Michigan, Physics Department

Contents

List of Figures

List of Tables

List of code listings

Acknowledgements

I express my sincere gratitude for the support and guidance I received during the completion of this master's thesis and throughout my entire course of study.

Special thanks go to my supervisor, doc. Ing. Kamil Dedecius Ph.D. for his guidance, encouragement, and patience throughout the scope of this thesis.

I also thank RNDr. Dagmar Adamová CSc. for her help in arranging this exciting cooperation and for reviewing my work. Also, thanks to the whole Nuclear Physics Institute of the Czech Academy of Sciences for making this cooperation possible.

Great thanks goes to Dr. Shawn McKee, the whole network-discuss group of University in Michigan, to members of the IRIS-HEP/OSG-LHC/NRP project, and people from perfSonar team for providing me the data which were crucial for this work. Also, a great thanks goes to Dr. Shawn McKee, Dr. Marian Babik, and Petya Vasileva for discussing my work with me.

Furthermore, I would like to extend my gratitude to my family, friends, and, least, but not last, to my partner, BcA. Anežka Krézková, for their love, patience, support, and understanding during the completion of this work.

Declaration

Abstract

Network anomaly detection is crucial for identifying issues in computer networks, with particular significance in large-scale global networks like the Worldwide LHC Computing Grid, that is used for processing data from CERN experiments.

This thesis aims to enhance or replace the existing heuristic network anomaly detection algorithm, primarily designed to identify routing changes based on traceroute measurements between global datacenters, generated through the perfSonar software.

The work demonstrates the informational richness of traceroute measurements and proposes several Bayesian inference-based models to improve anomaly detection in traceroute data.

Keywords network anomaly detection, large-scale computer networks, traceroute, Bayesian inference, anomaly detection models

Abstrakt

Detekce anomálií v síti je klíčová pro identifikaci problémů v počítačových sítích, zvláště pro počítačové sítě globálního rozsahu, jako je Worldwide LHC Computing Grid, který slouží k zpracování dat z experimentů prováděných v CERNu.

Tato práce si klade za cíl zdokonalit nebo nahradit stávající heuristický algoritmus pro detekci anomálií v síti, který byl původně navržen pro identifikaci změn v síťovém směrování na základě traceroute měření mezi centry po celém světě, která jsou generována prostřednictvím softwaru perfSonar.

Práce demonstruje informační bohatost traceroute měření a navrhuje několik modelů založených na Bayesovském učení, které mají za cíl zlepšit detekci anomálií na základě traceroute dat.

Klíčová slova detekce anomálií v síťovém provozu, rozsáhlé počítačové sítě, traceroute, Bayesovské učení, modely pro detekci anomálií

Glossary

AI Artificial Intelligence. 18, 65

AS Autonomous System. vi, vii, 6, 7, 15, 16, 45–47, 50–53, 58, 60, 62, 63, 75

BGP Border Gateway Protocol. 7

CERN European Organization for Nuclear Research. 1, 5, 60, 65

CLT Central Limit Theorem. 36, 44

CTA CERN Tape Archive. 1

DAG Directed Acyclic Graph. 22

DARPA Defense Advanced Research Projects Agency. 3

DDoS Distributed Denial-of-Service. 2, 7

HMM Hidden Markov Models. 2

ICMP Internet Control Message Protocol. 3, 7, 8, 31, 44

IP Internet Protocol. v–vii, 4, 6, 7, 11, 12, 27, 29–31, 33, 36, 38, 39, 42, 45–47, 50–53, 55, 58, 59, 62, 63, 67, 76

LHC Large Hadron Collider. 1

MAP Maximum A Posteriori. 20, 26, 35

ML Machine Learning. 13, 17, 18, 63

RNN Recurrent Neural Network. 2, 43

RTT Round Trip Time. v–vii, 3, 4, 6–13, 17, 36, 38–45, 60, 62, 63, 65, 67, 70, 71, 78

SHA-1 Secure Hash Algorithm 1. 7, 46

SNMP Simple Network Management Protocol. 2

TTL Time To Live. v–vii, 3, 4, 6–9, 11–13, 16, 31, 37, 39–45, 62, 63, 65, 70, 71, 79

WLCG Worldwide LHC Computing Grid. 1

Chapter 1
Introduction

In European Organization for Nuclear Research (CERN), scientists who gather from throughout the world are trying to deepen our knowledge in the field of High Energy Physics. CERN is probably mainly known to the general public for its research with particle colliders, which are located underground next to the CERN headquarters near the French-Swiss border in Meyrin, Switzerland.

The latest and also the largest of them is Large Hadron Collider (LHC)[1]. The LHC is 27 km long and is located in an underground tunnel. In this tunnel, beams of particles are accelerated in opposite directions in a vacuum using very powerful superconducting magnets. These magnets operate at temperatures lower than those in outer space. LHC was built with the goal of finding rare physics processes and particles predicted by particle physicists. The two beams of particles, rotating in opposite directions, cross at four points in the LHC. These crossing points are located in the centre of cavities where four large particle detectors are built: ALICE[2], ATLAS[3], CMS[4], and LHCb[5].

The purpose of these particle detectors is to monitor the results of collisions created by a large number of particles colliding at these crossing points. The results of these collisions are then registered, when passing through detectors, converted to analogue signals, preprocessed, and stored. The preprocessed data are then transferred to storage facilities in CERN, which involves regular disk storage and CERN Tape Archive (CTA) [6, 7, 8].

This process produces tens of PetaBytes of data per month. Processing them only using CERN computational resources would require enormous investments in computational power, infrastructure, and storage. This is why a global distributed computing infrastructure was built. Worldwide LHC Computing Grid (WLCG) is an ecosystem of computing datacenters from all over the world connected over the Internet. These datacenters, provided by academic institutions involved in the LHC programme, are used to store and process the data generated by LHC experiments. Currently, there are more than 170 datacenters, also commonly referred to as sites, in 42 countries around the world that are available for scientific computation using roughly 1.4 million computers [9, 10, 11].

Connecting the data centres allowed the processing of the data to be done in a distributed manner. The problem starts when data transmission is interrupted by network problems. This idles the computing power until the problem is solved and the data is sent again. This bottlenecks the whole process of analysing the data, which is why network monitoring is used to detect these problems.

1.1 Network monitoring, protocols & metrics

In order to be able to communicate with each other, the network devices must first understand each other. To understand each other, devices use communication protocols, which are sets of pre-defined messages and guidelines. These guidelines can then further specify what are the proper responses to messages coming from other devices.

Applications using these communication protocols can then generate substantial amounts of data, which are usually collected for analytical purposes. These data can, for example, contain metrics, status messages, or device identification numbers.

These data can be processed using data mining techniques to extract valuable information regarding the state of the network. These insights can help detect permanent or temporary problems in the network, which can be caused by faulty configuration, malfunctioning devices, or by malicious events such as Distributed Denial-of-Service (DDoS), or malware attacks.

Having a monitoring tool that can uncover these network issues would allow network administrators to remove the source of these problems faster. This would ultimately help improve the overall performance and reliability of the network infrastructure, leading to increased efficiency. The goal of this work is to find an alternative to the current heuristic algorithm, that detects problems in the network based on traceroute data.

1.2 Protocols

Network protocols are used to allow devices to communicate with each other in a certain way. Each protocol generates different types of data that can be used to uncover hidden network problems. Some of them are briefly mentioned in the following section.

1.2.1 SNMP

For example, Simple Network Management Protocol (SNMP) [12] was developed to allow a system administrator to collect information about devices by running SNMP agents on them, which can then periodically send information about memory and processor usage, up-time, temperature, or throughput to the SNMP manager, which is a device, usually a server, that collects these data.

These messages are used, for example, to detect anomalies based on the statistics using a Hidden Markov Models (HMM) based algorithms [13], or to find correlation with other data [14], which can allow a greater understanding of the behaviour of devices in the network and the network as a whole.

1.2.2 SYSLOG

Another important protocol is SYSLOG [15], which allows the sending of unstructured log messages over the network. These messages can be of various severity levels, from 0 to 8, which can be defined by the system administrators. These messages also contain textual messages about events that take place, which can be a source of additional information.

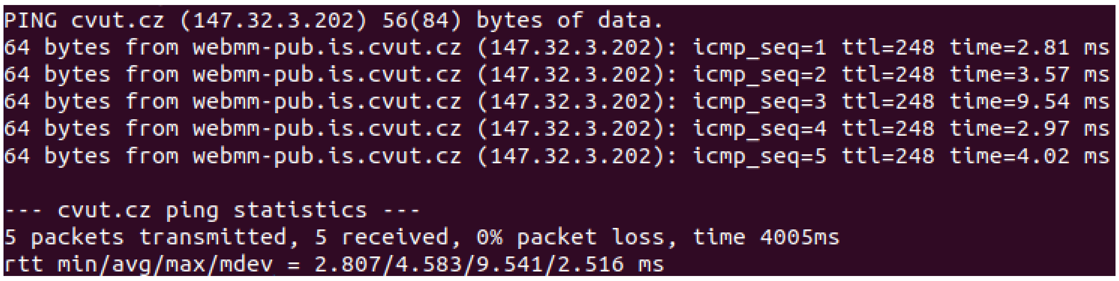
In [16] Recurrent Neural Network (RNN) are used to convert the unstructured to one of the predefined messages, which are then observed over time for anomalies.

1.2.3 ICMP

Internet Control Message Protocol (ICMP) [17], which was developed under Defense Advanced Research Projects Agency (DARPA) more than 40 years ago, allows devices to send control messages to each other. It operates at the level of IP addresses between the devices responsible for inter-network communication.

1.2.3.1 Ping

Ping is a network utility that uses ICMP messages to contact other devices connected to the Internet network. It starts by sending the ICMP echo request to the destination (ping), asking the destination device to respond with an ICMP response (pong). If any path is found over the Internet network and the destination device responds, the source device calculates the duration of the communication in milliseconds. If the destination device does not respond, either due to the device being busy, absence of any network path from source to destination, or its firewall blocking the ping, the ping will result in failure. Since the Internet network is densely connected, it could happen that the ICMP message travels through the Internet indefinitely, which could deadlock the Internet for a large number of these messages. Because of this, the TTL value, which controls the maximum number of devices the ICMP message can pass through, also known as hops. If the number of hops is greater than the maximum TTL, or the request time outs, the destination is marked as unreachable and the ping is counted as failure.



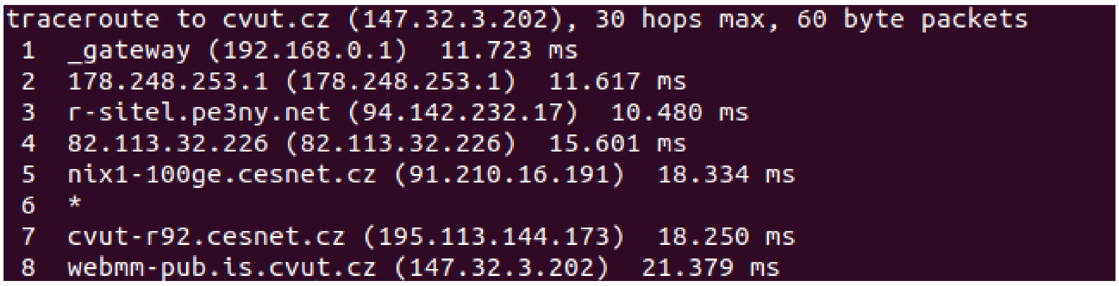
```
PING cvut.cz (147.32.3.202) 56(84) bytes of data.
64 bytes from webmm-pub.is.cvut.cz (147.32.3.202): icmp_seq=1 ttl=248 time=2.81 ms
64 bytes from webmm-pub.is.cvut.cz (147.32.3.202): icmp_seq=2 ttl=248 time=3.57 ms
64 bytes from webmm-pub.is.cvut.cz (147.32.3.202): icmp_seq=3 ttl=248 time=9.54 ms
64 bytes from webmm-pub.is.cvut.cz (147.32.3.202): icmp_seq=4 ttl=248 time=2.97 ms
64 bytes from webmm-pub.is.cvut.cz (147.32.3.202): icmp_seq=5 ttl=248 time=4.02 ms

--- cvut.cz ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 2.807/4.583/9.541/2.516 ms
```

Figure 1.1 Ping to cvut.cz showing average duration of 5ms based on five separate pings calls

Pings to the same destination are usually repeated multiple times, since the RTT value changes due to the dynamic nature of the Internet network. These values are then usually aggregated into more comprehensive statistics, such as total, min, max, and average RTT, together with the rate of failed requests, also known as packet loss. This can be seen in Figure 1.1.

1.2.3.2 Traceroute

The traceroute utility is very similar to ping. It is also a network diagnostic tool that sends ICMP messages from the source device to the destination device in order to track the path over the computer network. However, since the ICMP messages cannot contain the whole path, the main difference from ping is that traceroute starts sending the ICMP message from the source device to the destination, but limits the message to travel only to the first device on the path, setting the TTL value to 1. If that device responds, the source device will increase the TTL value and send another message with the increased TTL value to the device that is one hop further. This is repeated either until the destination device is reached, or when the TTL value exceeds the maximum TTL value, which is usually 30, but can be changed prior to running the traceroute utility.



```
traceroute to cvut.cz (147.32.3.202), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  11.723 ms
 2  178.248.253.1 (178.248.253.1)  11.617 ms
 3  r-sitel.pe3ny.net (94.142.232.17)  10.480 ms
 4  82.113.32.226 (82.113.32.226)  15.601 ms
 5  nix1-100ge.cesnet.cz (91.210.16.191)  18.334 ms
 6  *
 7  cvut-r92.cesnet.cz (195.113.144.173)  18.250 ms
 8  webmm-pub.is.cvut.cz (147.32.3.202)  21.379 ms
```

■ **Figure 1.2** Traceroute to cvut.cz through 6 intermediate devices, where one did not reply

The result of the traceroute utility is a list of intermediate devices represented by their domain name and IP address, together with RTT and ordered by the TTL value. It can happen that some of the devices do not respond, and the request times-out. In that case, the device is marked as unreachable and the corresponding line is replaced with an asterisk symbol ($*$), which can be seen, for example, in Figure 1.2 for TTL 6. The fact that the host is unreachable does not necessarily indicate an issue with the device, unless it is systematically unreachable. The issue with that is that there is no way to compute any statistics, nor detect that the device is not working properly, unless there is at least some response from the device.

It is also worth mentioning that the output from the traceroute tool might not be deterministic even for a fixed source and destination, since there can be certain segments of the path, where multiple candidate devices are suitable for the path to a given destination. This can be, for example, due to dynamic routing, or due to load balancing of servers that handle the traffic in the Internet backbone, which can affect the result of the traceroute utility.

1.3 Network monitoring at CERN

In order to monitor such a large computational network, the network data are collected on the side of each site. For example, throughput, latency, number of retransmissions, packet loss, end-to-end delay, and traceroute data. This is managed by perfSonar software[18], a network monitoring software that collects these data for a pre-defined set of sites. These measurements are then uploaded to ElasticSearch [19], which is used as a database for network analytics.

The analytics platform then periodically runs various algorithms that use these data to detect network problems. If a network problem is detected, the algorithm creates an alert that is then propagated to a monitoring dashboard.

There is currently a heuristic algorithm which tries to detect network problems based on traceroutes, namely the change in path. But it is believed to be underperforming. The goal of this work is to either improve this heuristic algorithm or propose an alternative approach that would be able to detect network problems based on traceroute data.

Chapter 2

Data & Alerts Analysis

In the following sections, the data used in this thesis are described and analysed to find useful relationships in the data that could be used for the detection of anomalies. This is further complemented by the analysis of the alarms created by the alerting algorithm, which is currently being used, and also the analysis and discussion over the algorithm itself.

2.1 Dataset

The traceroute dataset contains measurements extracted from the results of the traceroute utility enriched with additional attributes that are computed before the traceroute is uploaded to the analytical platform. There are currently 81 unique source sites and 113 unique destination sites, indicating that not all source sites are also a destination site, which may reflect the importance of sites in CERN computing network architecture. For each site, there are multiple devices that send traceroute probes to a subset of destination sites every few minutes. This can also be clearly seen in Figure 2.2. The total number of records per day is presented in Figure 2.1. On average, sites can produce up to roughly three million records daily; however, the number of records may vary depending on the condition of the devices that collect the data within the sites.

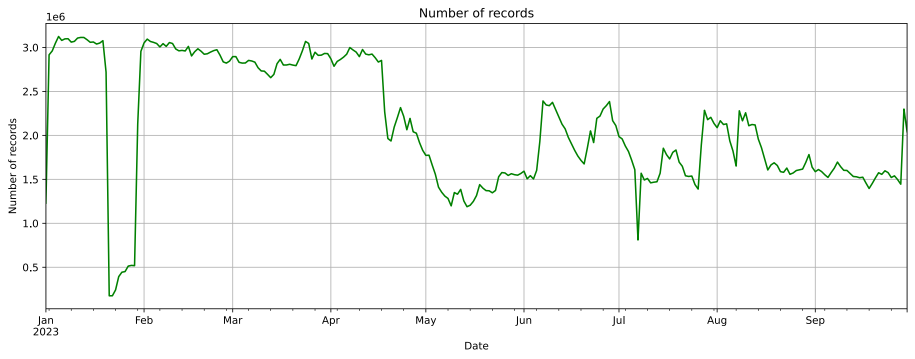
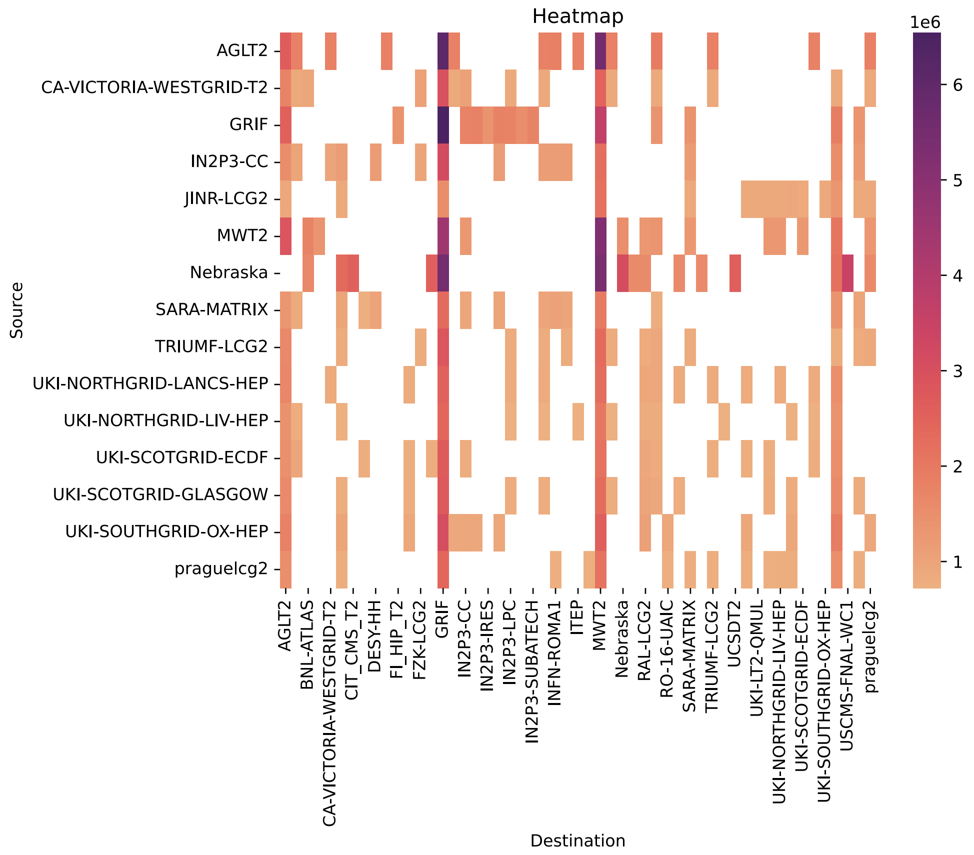


Figure 2.1 Daily number of records over time (in millions) showing sudden dip in late January and overall decrease of measurements since April



■ **Figure 2.2** Heatmap of traceroute tests done between the sites for 15 most frequent source sites

There are more than 3 million records in the first half of January being uploaded to an ElasticSearch, then in late January there is a significant decrease in the number of records being uploaded. After the initial decrease, the number of records began to gradually grow until February, which was followed by a small, but gradual, decrease in records per day. In mid-April, the average daily record count dropped again and began to approach 1.5 million, which became a sort of a new daily standard with some variations in June, July, and August. Additionally, there was also a slight increase in the number of records per day in late September.

From the raw traceroute result, the aligned sequences of TTL values, IP addresses, and RTT values are extracted. If there were any unreachable routers on the traceroute, they are simply skipped, but note that the information about the unreachable device is not lost due to the difference in values of two neighbouring TTL values. For example, for the traceroute from 1.2, the TTL sequence would look like the following: $[1, 2, 3, 4, \mathbf{5}, \mathbf{7}, 8]$.

For the sequence of IP addresses, a list of Autonomous System (AS) numbers is obtained using an AS translation table. The AS numbers contain additional information about the device hierarchy and could be useful for analysis and can work as a group identification for the IP addresses.

AS is a designation for a part of the computer network, where routing is predefined not only for devices within this section of the network by a given set of routing rules (intra-routing), but also for routing from one AS to another. This inter-AS routing is commonly maintained using the Border Gateway Protocol (BGP)[20] which allows more than one optimal path between the AS. The AS are usually managed by one or more Internet providers, and each AS is identified by a unique id from the range 0 - 65534, where some AS numbers are reserved. For example, the AS numbers from 64512 to 65534 are reserved for private use and 0 is reserved for unrouted AS [21].

In addition, inside each record is a SHA-1 hash of the AS sequence , which is currently being used as an identification of the real route of the network, maximum RTT, and number of hops of the traceroute, together with the IP address, domain name, and the name of both the destination and the source.

The stored data also contain timestamps for when the measurement was created and uploaded to ElasticSearch. These fields themselves do not bring much value, but if combined with the statistics extracted from the rest of the fields, they can be used to get a better understanding and valuable insights about how the state of the network developed over time. When the dimension of time is present, time-series analysis and time-series methods can also be used, for example, to check for presence of trend or seasonality. Lastly, based on the list of the IP addresses, several Boolean fields are computed, which are described in the following subsections.

2.1.1 IPv6

As was previously mentioned, the traceroutes contain several Boolean variables, one of which provides information on whether or not the communication was carried out using the IP protocol version 6, or older version 4. Both versions are used for communication over the Internet, but there are several ways in which they are different. For example, the IPv4 used 32-bit addresses, but IPv6 uses 128-bit addresses. IPv6 comes with network security layer, which IPv4 does not have [22]. Due to the differences between the two versions, seemingly identical network operations for the user might in fact be implemented differently, which might result in different measurements for both versions.

Information about the version of the IP protocol can be used, for example, to test whether there is a significant difference between the behaviour of the traceroutes. For example, if IP version 6 were to indicate that traceroutes would be less likely to include any unreachable devices or have significantly lower RTT values, this could be a powerful argument to completely abandon IP version 4.

2.1.2 Destination Reached

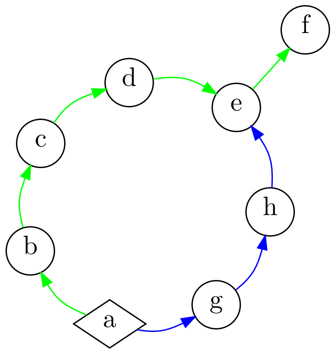
Another calculated field is based on the destination ip address, as the post-processing algorithm checks that the last IP address in the sequence is the same as the destination IP address, meaning that the traceroute probe successfully reached the destination. This can be useful to detect paths or specific devices that might prevent network communication from reaching its destination. However, if any device on the path blocks ICMP messages, for example to avoid potential network attacks (e.g. DDoS using the ping utility), the destination would never be reached, which would mean that the value might not be very useful in this case.

2.1.3 Path complete

This attribute is calculated using the sequence of TTL values by checking, whether or not the sequence contains all the values from 1 to the maximum TTL, meaning that none of the devices within the path was unreachable. Having the path complete maximises the chance of getting the most accurate insights about the state of the computer network devices.

2.1.4 Looping

Looping in this context means that, for a given route between the source and destination devices, a certain device occurs more than once. There can be several reasons for this to be happening; for example, it can be due to the wrong configuration of some routers along the path or due to the presence of multiple paths between the source and destination, through which the communication is routed dynamically.



(a) Network paths

TTL	Device	RTT(ms)
1	g	0.34
2	h	0.56
3	e	1.23
4	e	**5.56**
5	f	2.45

(b) Traceroute results

■ **Figure 2.3** Hypothetical example of a looping traceroute measurement

For example, consider a simple network represented by a graph in Figure 2.3. There are two distinct paths from node a to node f, coloured blue and green. Now, for a hypothetical traceroute measurement, the result could look like the one shown in Figure 2.3. The traceroute starts at device a, then goes through g, then h, and then e. However, after that some event forces the next ICMP message to go through the green path stopping again on the e device. Subsequently, the final round goes all the way to the destination device f.

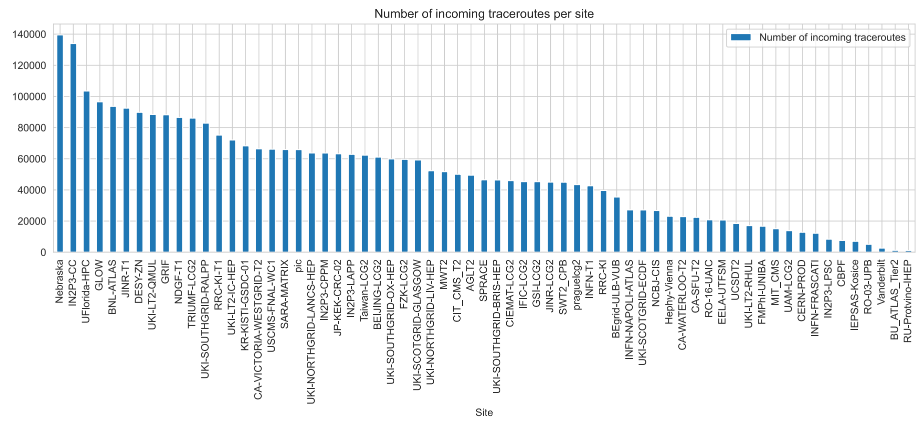
This shows that even when the blue path is shorter and is based on RTT probably faster, sometimes it can happen that due to routing, the ICMP request goes through the green path. Due to the longer green path, it was possible that the device e occurred twice in a row. This way the configured router, or whole network segment may create communication issues, but also can negatively influence the statistics from other network monitoring tools.

2.2 Analysis of the data

As mentioned in the previous section, the volume of the data is quite large. To facilitate manual analysis, a decision was made to select only a subset of the time frame. Additionally, a site with sufficient traffic was chosen, and only traceroute requests with this site as the destination were the focus of the primary analysis.

This was done to reduce the volume of data, as analysing all of the sites at once would require huge computational resources. Since the selected site contains traceroutes originating from all over the world, it should be a sufficient source of information for analysis. Reducing the number of sites in the analysis should also reduce the amount of noise that would be present when analysing all sites at the same time, as the distributions of RTT and TTL would likely be different. Furthermore, this implies that the candidate model should be capable of learning site-specific relationships or a unique model for each site would be required.

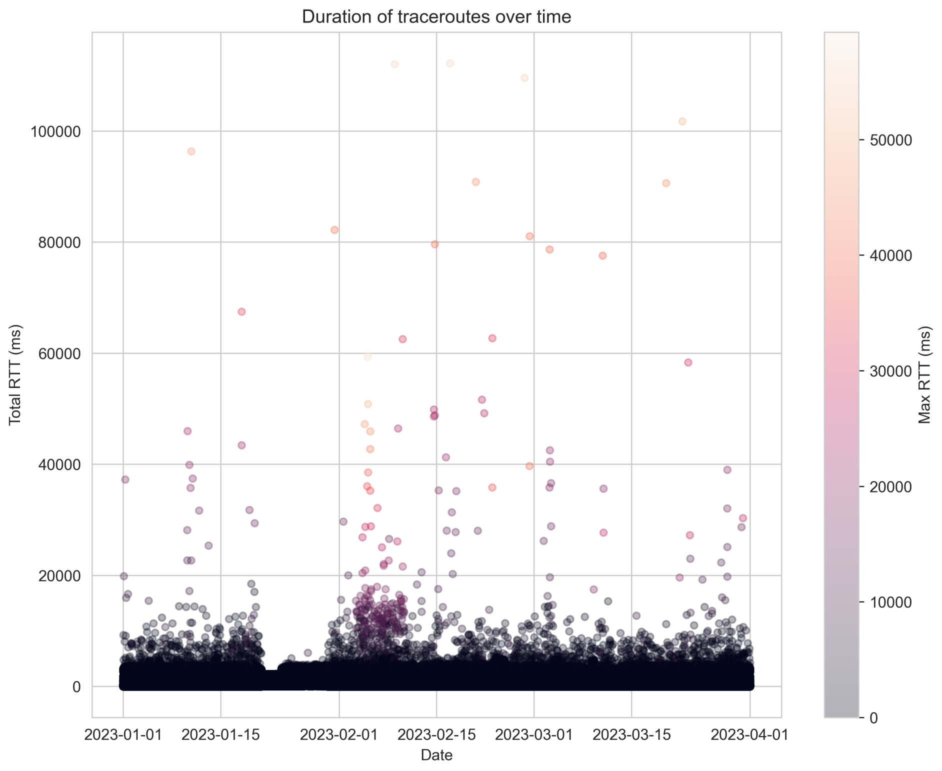
The final time period selected for analysis spans from the first of January 2023 to the thirty-first of March 2023. The selected site for analysis is "INFN-T1," which is owned and managed by the Italian National Institute for Nuclear Physics.



■ **Figure 2.4** Number of incoming traceroutes from other sites to INFN-T1

In this dataset, there are more than 3.2 million traceroute measurements from 65 different source sites, each of them sending a different number of traceroute probes to the INFN-T1 site. From the scatter plot of the average RTT over the entire time period (see Figure 2.5), it can be observed that the total RTT for the vast majority of measurements is less than five seconds. Throughout the observed period, there is a high variance in the total RTT, except for the late January period, where the traceroutes appear to take less total time in general. This is likely due to the smaller amount of data mentioned in a previous section. Except for this time period, the majority of the traceroutes take around five seconds in total. There is also a noticeable number of records with a total RTT between five and twenty seconds. Additionally, there are traceroutes that took more than twenty seconds, although there appears to be very few of them.

The scatter plot is coloured according to the maximum RTT of each measurement. It is clear that for most of the traceroutes with a total RTT greater than 20 seconds, there are some devices responsible for the majority of the total time, indicating a delay in their response. There is also one particular period between the second of February and the eleventh of February where the same pattern occurs for a cluster of traceroutes with a total time less than 20 seconds, which might indicate a network issue.



■ **Figure 2.5** Scatter-plot showing sum of RTT over time with a period with large RTT in early January

In most cases, the traceroutes have from seven up to eighteen hops (see Figure 2.6). However, there are traceroutes with even more hops, going up to thirty hops, but these occurrences are less likely. It is possible that the data with fewer than five hops might indicate some problems in the network, as there appears to be a significant gap in frequencies. Additional examination has revealed that there are actually two distinct varieties of traceroute data. The first kind consists of traceroutes from sites that are indeed distant, up to five hops (for example, other INFN sites). The second kind consists of traceroutes that finished prematurely. Since the opposite extreme, where the route was much longer than usual, was also occurring for some sites, it might be worth trying to model the number of hops in order to be able to detect these anomalous traces.

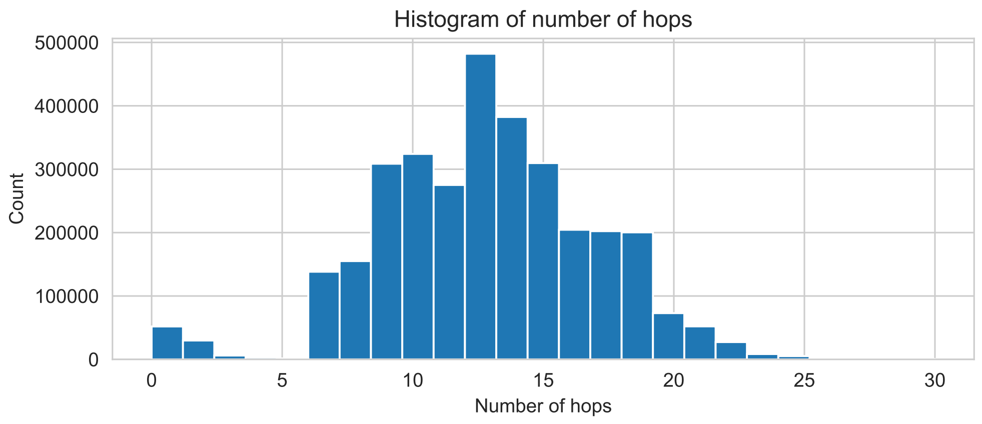


Figure 2.6 Histogram of number of hops of traceroutes showing potential outliers near 0 and 1 hops

2.2.1 Boolean fields

Based on these visualisations, it seems that both RTT and TTL provide useful information on the condition of the network. However, it is not yet clear whether there is any relation between RTT and TTL. In Figure 2.7, there are four subplots with maximum TTL values on the X-axis and maximum RTT values on the Y-axis, and for each of them, the traceroutes are coloured based on one of the Boolean fields.

For the RTT and TTL, there appears to be no visual indication of a strong linear relation between TTL and RTT. It seems that each TTL value has its own distribution of RTT values. However, for the TTL values between 12 and 20, there is a significant increase in the maximum RTT value.

Based on the first graph, it appears that traceroutes with fewer than seven hops usually do not reach their destination. However, further investigation has revealed that this is not entirely the case. Several sites send traceroute requests over long distances, and the average length of these traceroutes is roughly 15 hops. However, there are some traceroutes that are much shorter (4-5 hops), which means that they have not been successful, but ended prematurely.

For ”Path Complete”, it seems that most of the complete paths have fewer than seven hops. However, this information is distorted by the way this field is calculated. It is true that the longer the traceroute is, the more likely it is to contain a device that is unreachable. However, if the traceroute fails before any device becomes unreachable, it is still counted as a complete path. Combining the ”Destination Reached” and ”Path Complete” fields might yield valuable information.

The third scatter plot is coloured according to the looping field. It appears that both looping and non-looping traceroutes are present across the TTL values. There is a noticeable increase in the concentration of looping traceroutes between 13 and 20 hops, which could suggest that looping paths might affect the maximum RTT. However, the majority of traceroutes with a high maximum RTT are not looping. There is also a set of TTL values (between 6 and 13) where there seems to be very few looping traceroutes.

The last sub-plot is coloured based on the version of IP used for the traceroute. It seems that both versions of IP span across the TTL values. There appears to be no visible relation between the IP version and increased RTT.

Based on these visualisations, it seems that there are no visible patterns in the relation between the RTT, TTL, and the Boolean fields. However, they might still be useful for analysis. For example, they can be used to sort the traceroutes into groups for further processing or to monitor any changes in these values over time.

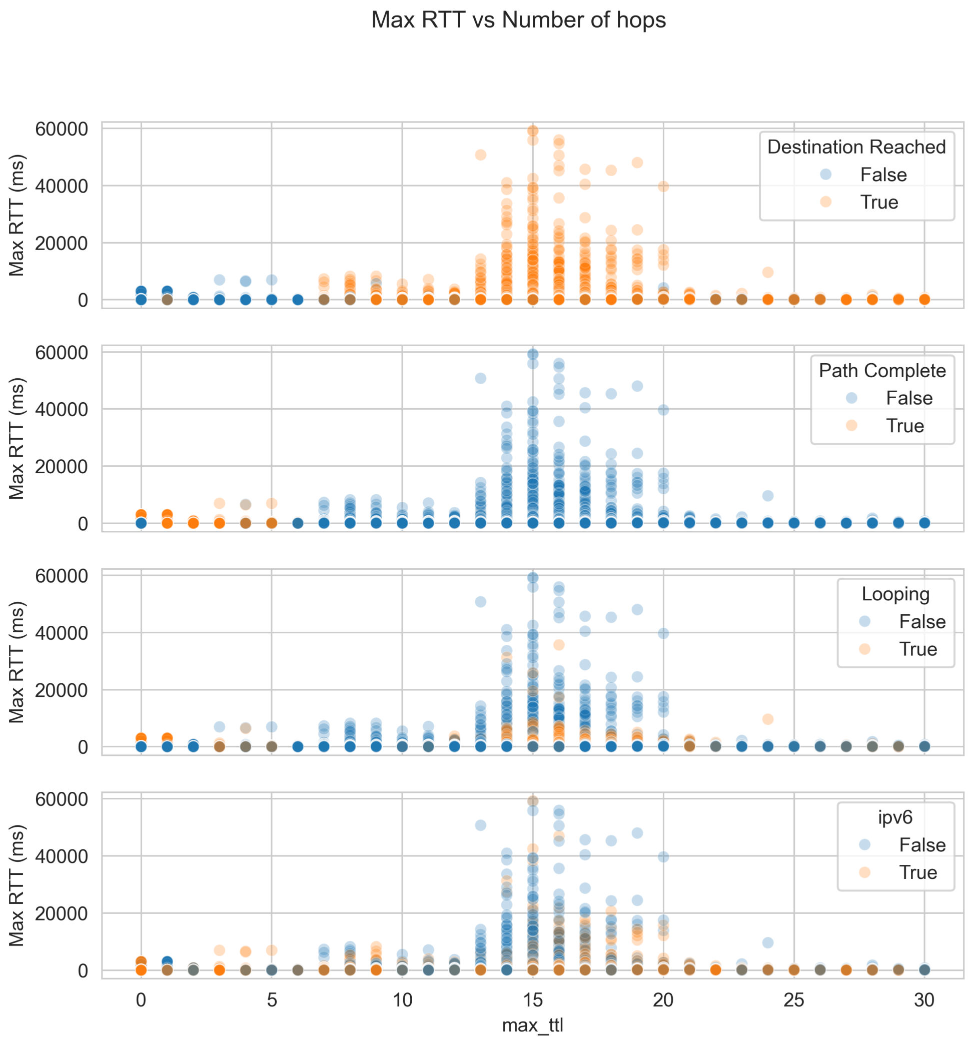


Figure 2.7 Comparison of RTT and TTL values coloured based on values from each Boolean field

In Figure 2.8, the records are grouped according to the combination of Boolean values and their percentage representation within the data set, which results in 16 bins. The first notable thing is that there are fewer than 5% complete paths. Then, for paths that are not complete, most of the records are traceroutes that reached the destination and are not looping. There are 45% of IPv4 and 37% of IPv6 records. There also seems to be a relatively high number of looping traceroutes, where most reach the destination (roughly 7%), but it seems that more of these are using IPv4. The last significant group would be traceroutes that did not reach the destination and are not looping, which is roughly 7.2%. However, it is interesting that the majority of these are using IPv6.

It might be interesting to analyse the traceroutes from each group separately to try to discover interesting statistics about them, which could then be used. But this would require substantial amounts of time and there is no guarantee that these statistics would be similarly distributed for traffic from all the sites, which would affect the overall performance of the model.

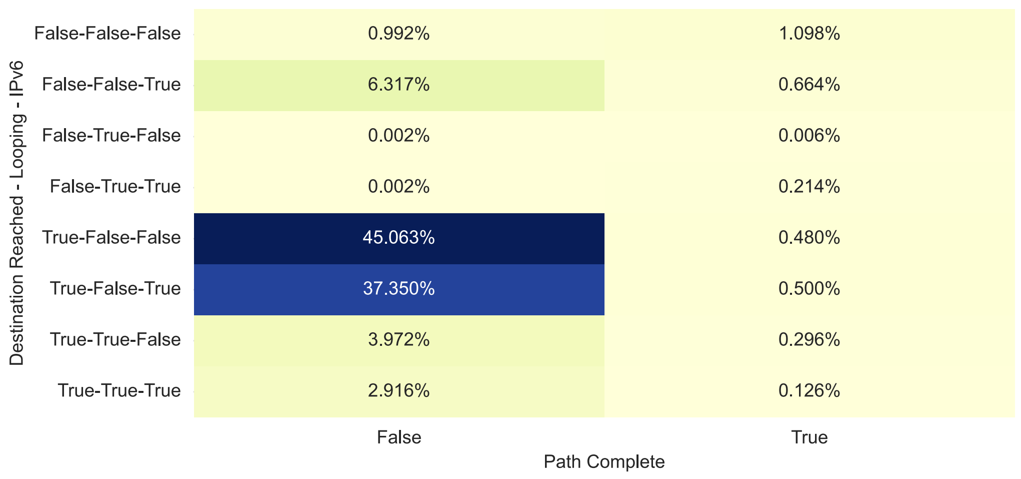


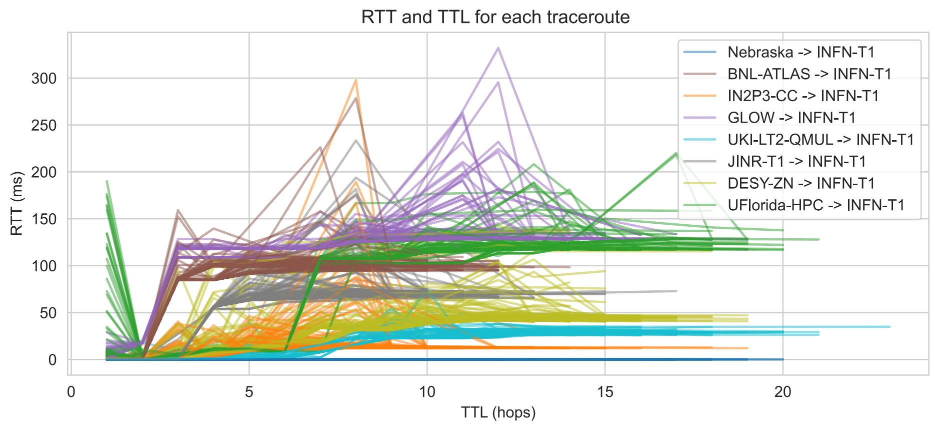
Figure 2.8 Number of traceroutes per combination of Boolean fields

2.2.2 Traceroute clusters

Another point of view for the analysis might be to take into account the common traffic for a given site or pair of sites. It might be valuable to determine if the RTT and TTL values are regular or irregular across different time periods. Due to the way the networks are configured, the routes between certain sources and destinations are also expected to follow a very similar path in the vast majority of cases, as the goal of the computer network is to deliver messages as soon as possible. There will definitely be some degree of variance, but for the same or very similar path, the RTT values measured for each segment of the path should also likely have similar values, unless an event with a negative impact on some part of the computer network occurs. This event can shift the RTT towards larger values and can even result in some devices being overwhelmed, rendering them unreachable for a certain period of time. Perhaps the most reasonable direction would be to take into account all sequences of the RTT and TTL values to gain insights, rather than just using their aggregations.

In Figure 2.9, six thousand traceroutes are represented with TTL values on the X-axis and their respective RTT values on the Y-axis for the ten most frequent sites. This figure shows that there is indeed a pattern for the routes between the pairs of source and destination sites. If a ML algorithm were able to learn the usual route, it could be used as a baseline for identifying irregular routes. For example, the usual RTT for the traceroute from GLOW to INFN-T1 (purple lines) takes approximately 130 ms for the hops from the ninth to the fifteenth hop. However, in some cases, there are routes for which the RTT is significantly higher, with some cases showing an increase of more than 50%.

Based on this graph, it is evident that the sequences of the RTT values for a given source and destination pair exhibit a similar pattern. This structural information can be leveraged to gain valuable insights about the data. Disciplines such as sequence-to-sequence learning [23], or sequence clustering methods described in [24] can be employed to achieve this; however, they might not be suitable for production use, as they are usually computationally very demanding.



■ **Figure 2.9** Traceroutes for 10 most common source sites communicating wiht INFN-T1

2.3 Alerts

The alerts generated by the algorithm described in Section 2.4 were analysed for the same period between January and March. In this case, there was no option to filter the alerts for a particular site, since each alert is basically a daily report. Within these alarms, there are 81 alert records in the database. The average number of daily sites for which traffic is not aligned with a baseline is roughly 46, the minimum over this period is 18 sites, and the maximum is 81 sites. Figure 2.10 contains a word cloud visualisation, where the size of the item is proportional to its frequency. Based on this visualisation, the top three sites for which most of the alarms were created are INFN-T1, FZK-LCG2, and PIC.

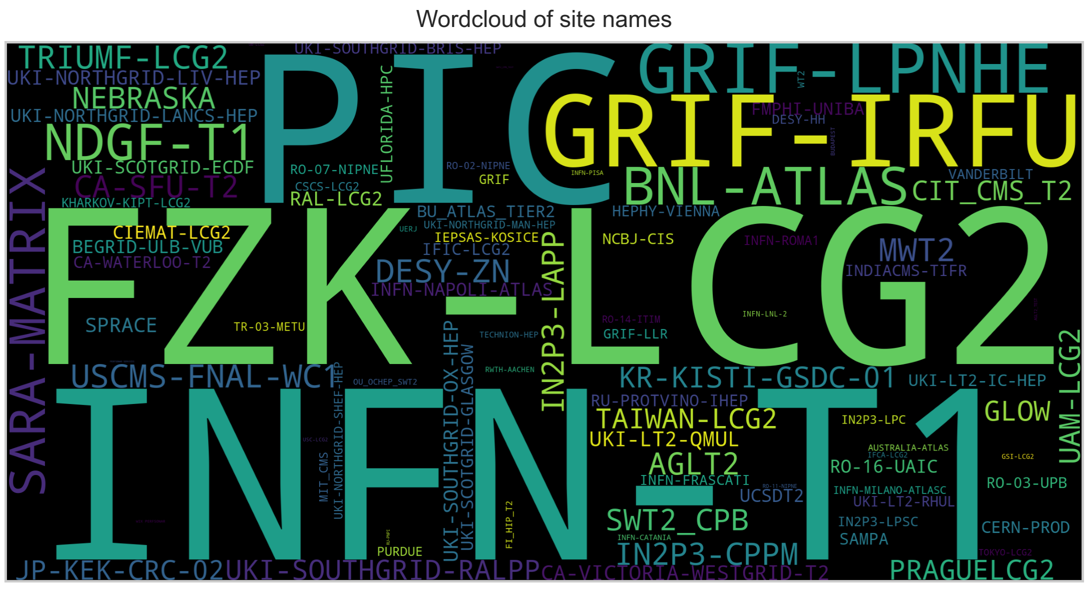


Figure 2.10 Wordlcloud of site names in alarms

When sites are aggregated according to their country of origin, the country with the most alerts is the United States with almost half of the alerts. Within the United States, there are several sites; the second most represented country is Spain with more than 10% of the alerts, followed by the Netherlands, Switzerland, Canada, Sweden, Korea, and Italy (see Figure 2.11).

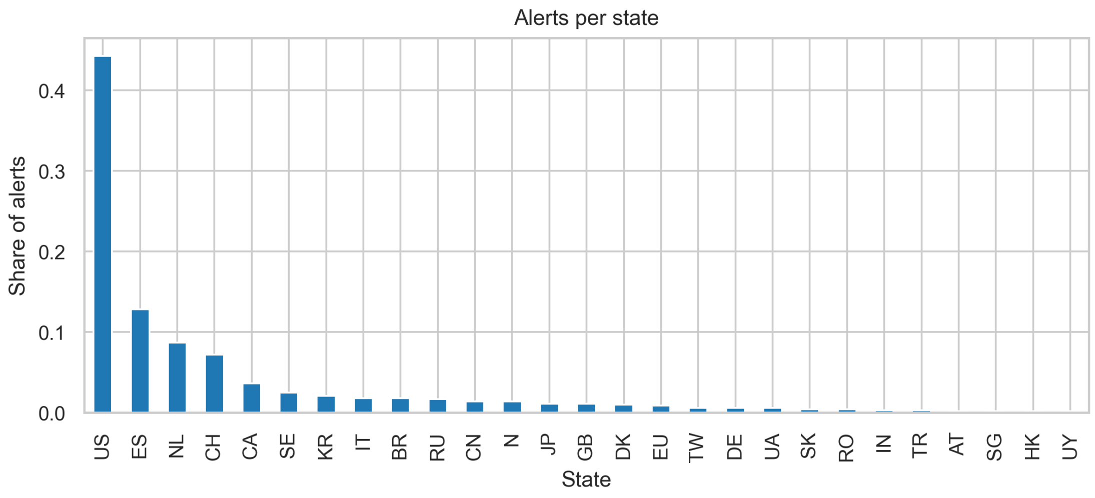


Figure 2.11 Anomalies per country (in percents)

2.4 Alert algorithm

Currently, there is already an algorithm whose objective is to uncover problematic traceroutes between all pairs of source and destination sites based on the sequence of the AS numbers. First, this algorithm collects all data from ElasticSearch for the last three days and then calculates, for each unique source and destination pair, how often each unique sequence of the AS numbers occurred within this time window. Then, if there is a sequence of AS numbers that occurred in more than 65% of cases, it is taken as the baseline path. Otherwise, the algorithm takes the path with the maximum number of unique AS numbers. If there is more than one path that satisfies this, they are sorted based on the frequency of occurrence of the AS.

After a baseline AS sequence is calculated for each pair of sites, each record is compared to its respective baseline by checking whether the set of AS numbers is the same for the baseline and the path. If they are not the same, the path is then marked as an anomaly, and an alert is created afterwards, which is uploaded to ElasticSearch.

In fact, this algorithm might capture errors in some scenarios. It is worth noting that, in the first place, the algorithm is deciding based on the majority of sequences, which might not be the best solution. Based on the observations in Section 2.2.1, most traceroutes contain unreachable devices, and the baseline is estimated from this data. In Table2.1, the maximum number of unreachable devices ($max(N_u)$ [1]) with their respective count for this dataset is displayed.

Also, for 63.3% of traceroutes, there are at least two consecutive devices missing from the data. Based on further analysis, the average ratio of the number of unreachable devices to the maximum TTL, which should reflect the real length of the traceroute, is about 22.8%.

This means that if the distributions were the same for each site pair subset, the respective baselines for each site pair would be heavily biased towards the incomplete traceroutes, which might lower the accuracy, since it could happen that traceroute with no issues will get reported.

max N_u	%
2	63.298
1	15.145
3	9.966
4	5.284
5	2.178
0	2.023
6	1.054

Table 2.1 Maximum TTL differences

The drawback of this alerting method lies in its inability to pinpoint affected records, exact time of occurrence, or severity of path change. The frequency of daily alerts makes it difficult to immediately investigate the issues by network administrators; thus, enhancing explainability, offering more insights and enabling more frequent monitoring could empower network administrators to address these issues more effectively and quickly. Nonetheless, it should be noted that this algorithm serves as a suitable baseline.

[1] $N_u = x_{i+1} - x_i - 1$

Chapter 3

Anomaly detection & traceroutes

Anomaly detection is a discipline in which algorithms try to detect irregular patterns in data.

Manual detection of anomalies by a network administrator is time-consuming. For a small network, manual detection is tedious and on a global scale, it becomes practically impossible due to the extensive volume of records generated by a large number of devices communicating within the network. That is why there is a focus on researching methods to automate anomaly detection. The goal of this research in computer network monitoring is to develop a universal and at least semi-automatic method to analyse network data and detect irregularities. These irregularities are expected to be caused by poor device configuration, hardware or software problems, or malicious attacks within the network. Each of these issues is expected to leave specific marks, which can then be analysed further to classify the root cause of the network's abnormal state.

Regarding anomaly detection using traceroute data, there is a limited amount of research and publications available. The range of possible methods is further constrained due to the absence of a labelled dataset, meaning that supervised models are unsuitable for this task, and the model must have the ability to learn in an unsupervised manner. The anomaly detection methods can be classified into three main categories, as follows:

- The most straightforward methods for detecting anomalies are heuristic algorithms, where, in most cases, observed variables are thresholds against a certain value, often determined empirically based on prior analysis of the problem or expert knowledge. Heuristic algorithms can range from simple to complex, affecting the interpretability of the results. An example of a heuristic approach would be the current alerting algorithm.

- The second group consists of statistical models and methods that are based on statistics, such as average, mode, median, or statistical distributions including Normal, Exponential, or Binomial distribution. These are utilized to model certain variables that are expected to be significantly influenced by malicious events. For example, [25] use changes in RTT together with time series methods, in order to detect anomalies.

- The last group consists of general ML algorithms for regression, clustering, or classification, such as decision trees, nearest neighbours, or neural networks, which can be applied to traceroute data. For example, [26] uses traceroute data complemented with throughput, packet loss, and one-way delay, which can be used to model a known set of anomalies using a decision tree and neural network.

As heuristic algorithms are usually based on certain assumptions, and if in this case they were created only based on a subset of the site-to-site communication, the final algorithm might not perform the same on each pair of sites, and could generate a large number of false positives and negatives due to poor generalisation. An alternative option would be to define a unique configuration of the heuristic algorithm for each pair of sites, which could be complicated if done manually and would require a significant amount of validation.

The problem with traditional ML methods is that simpler methods may not be able to learn complex patterns in massive datasets, while more complex methods could be computationally infeasible for training and inference. Furthermore, most ML methods do not consider the time factor, which means that the Machine Learning (ML) model is usually trained on historical data under the assumption that the distribution of live data will remain the same. This assumption may not always hold, although there is usually a period during which these models perform reasonably well. After this period, the models need to be retrained with a new batch of data to adapt to changes in the network. This does not necessarily guarantee that the new version of the model will, in general, perform better, and it requires additional validation of the new version of the model, which can be demanding in terms of human workloads.

Another very important aspect of the potential method is the explainability of the model, which can be a deciding factor in an interaction between the human and the ML algorithm.

For example, if banks were to decide on granting loans based on a predictive ML model, if police units used ML models for tracking and arresting suspects, or if hospitals determined patient survival chances in severe conditions using such models, the ability to explain the model's decisions is crucial. Errors in these domains could lead to the rejection of AI. Having an explainable AI algorithm also helps to detect biases that negatively influence the result [27]. In network monitoring, an explainable AI could help to understand the nature of detected problems, allowing network administrators to assess the severity of the problem and address them accordingly.

Making the results explainable can be challenging for some ML algorithms, but the gradual increase in the regulation of AI is shifting the attention to these methods quite rapidly [28].

After careful consideration of the possible limitations and needs, the choice of method was settled on Bayesian inference. Bayesian inference has been successfully applied to solve various problems in different domains, such as modelling ecosystem processes in biology [29], helping physics to find extrasolar planets [30], or object tracking [31], which is a sort of domain-independent problem and can have various applications.

Compared to perhaps more popular ML methods such as decision trees or neural networks, Bayesian inference is built directly on top of statistical distributions. These distributions inherently make Bayesian inference naturally explainable, as they can provide additional context for both observed and predicted variables. Furthermore, it also models the uncertainty of the model about the modeled variable, which provides additional and valuable information that is not provided by other methods.

Being an explainable method is one of the reasons why Bayesian inference may be suitable for network monitoring using traceroute data. Another important factor is the method's computational requirements and its ability to adapt to changes in distribution. These changes can be caused, for example, by replacing certain devices in the network. These properties are described in more detail in the following section.

3.1 Introduction to Bayesian inference

Bayesian inference is a method built on top of the Bayes' Theorem. It starts with prior knowledge about the modelled problem, which can for example represent expert knowledge or results of a survey, and uses the measured data to adapt the prior beliefs based on the data to get a better understanding of the problem. Given the fact that this is an iterative method, it is expected that with a small sample of data, the results will be biased towards the prior beliefs; however, after a certain amount of data samples, the learnt distribution should converge to the real distribution.

3.1.1 Bayes' theorem

As mentioned above, Bayesian inference is built on top of Bayes' theorem, which is written in Equation 3.1. The equation shows the relation between the conditional probability of the event A given that the event B occurred. On the right side of the equation, there is the conditional probability of event B given that event A occurred, which is then multiplied by the probability of event A. All this is then divided by the probability of the event B,

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \qquad P(B) > 0. \tag{3.1}$$

This formula can be interpreted as the process of learning, where the prior knowledge of the event A, which is represented by marginal probability $P(A)$, is updated after observation of the event B. Thus, after learning about the event B, the updated knowledge, also called posterior, about the event A is represented by the conditional probability $P(A|B)$.

Now, for a given dataset $Y \in \mathbb{R}^{m,n}$ the Bayes' Theorem can be rewritten in terms of probability density functions as follows:

$$\pi(\Theta|Y) = \frac{f(Y|\Theta) \cdot \pi(\Theta)}{f(Y)}, \qquad f(Y) > 0. \tag{3.2}$$

Here, $\pi(\Theta)$ is the posterior density function for the variable Θ, which is a set of parameters and/or latent variables $\Theta = (\theta_1, \theta_2, \ldots, \theta_n)$ used to represent prior knowledge, and $\pi(\Theta|Y)$ is the posterior conditional density function that represents updated beliefs. The $f(Y|\Theta)$ is the conditional density of the data Y given the parameters Θ, however, this is commonly referred to as the model of the data or likelihood function. The $f(Y)$ is the marginal probability density of the data Y, which is usually called evidence and can be denoted as

$$f(Y) = \int f(Y|\Theta) \cdot \pi(\Theta) \, d\Theta. \tag{3.3}$$

Equation 3.2 is usually simplified using the proportionality symbol ($\propto$), which means that the posterior density has the same shape as the right side of the equation, but is scaled by some factor, in this case the $f(Y)$. The simplified version of the equation is then

$$\pi(\Theta|Y) \propto f(Y|\Theta) \cdot \pi(\Theta). \tag{3.4}$$

This process of learning can then be repeated indefinitely, with respect to the limitations of hardware and software, with a new set of data using the same equation. Additionally, this method has native support for batched learning, which means that the Bayesian update does not have to be run for each measurement but rather on the whole batch, which also reduces the demand on computational power. These batches can have varying sizes everywhere from one sample to the limit of the computational system.

To demonstrate this mathematically, the respective measurements can be denoted as y_i, where i represents the index of the measurement with respect to the learning process. This index can also be interpreted as the measurement time. To describe a batch of data of size $l \in \mathbb{N}$, the batch of l samples can be written as $y_{i:i+l-1}$. Using this notation, the two-step Bayesian update for batches of size $j, k \in \mathbb{N} : j + k = t$ is

$$\begin{aligned}\pi(\Theta|y_{0:t}) &\propto f(y_{j+1:j+k}|\Theta) \cdot \pi(\Theta|y_{0:j}) \\ &\propto f(y_{j+1:j+k}|\Theta) \cdot f(y_{0:j}|\Theta) \cdot \pi(\Theta)\end{aligned}$$

and the posterior predictive density function has the following form

$$f(y|Y) = \int f(y|\Theta)\pi(\Theta|Y)\, d\Theta, \tag{3.5}$$

This can be used for example for prediction; however, getting a closed-form solution for complex integrals can be hard for computers, as they cannot solve them analytically. To solve this, sampling methods are used to create a large enough number of samples, which are then used to approximate the closed-form solution, which might slow down the computation if done for each new observation, which is why larger batch sizes are used. However, in some cases obtaining the posterior distribution and posterior predictive density function does not require sampling methods at all.

3.1.2 Point estimates

The Bayesian modelling assumes the existence of parameters Θ belonging to the model, which are, in fact, unknown and must be estimated in some way. The point estimates then represent an approximation of these parameters, such that they are the best approximation of the true value under a certain method.

There are several methods to obtain them from the data, and sometimes optimisation methods are required to do so. One of the methods is the Maximum A Posteriori (MAP). The optimisation task here is to find an estimate Θ that maximises the posterior distribution. Finding a maximum of the posterior function is the same as finding the maximum of product of likelihood function and the prior distribution.

$$\Theta_{MAP} = \underset{\Theta}{\operatorname{argmax}}\, \pi(\Theta|Y) \tag{3.6}$$

It can be shown that maximising the posterior probability density function is the same as maximising the following

$$\Theta_{MAP} = \underset{\Theta}{\operatorname{argmax}} \log f(Y|\Theta) + \log \pi(\Theta), \tag{3.7}$$

which is almost identical to finding the maximum likelihood of the data, but there is an additional term $\log \pi(\Theta)$, which means that unless the prior is uniform, the optimal set of parameters Θ must also maximise the prior beliefs [32].

3.1.3 Sufficient statistics

The sufficient statistic T is a function of the data Y that suffices for the estimation of Θ, which means that it is completely capable of representing the information carried by the data. Another useful property of sufficient statistics is that, with growing amounts of data, the dimensions of sufficient statistics do not grow, which means that they are highly memory-efficient, which also saves computation time.

In this way, the estimates of the parameters Θ refer to the set of statistics $T = T(Y)$ for which the posterior distribution $\pi(\Theta|Y)$ is the same, or at least very close to, as if it were based only on the sufficient statistic, in other words

$$\pi(\Theta|Y) = \pi(\Theta|T), \tag{3.8}$$

as described in [33].

This means that for the iterative run of the method, the parameters Θ do not have to be estimated from all observed values, but to obtain the posterior distribution, it is more feasible to only calculate and update sufficient statistics, which might make the whole computation easier. Sufficient statistics can also be used to obtain point estimates much more easily.

3.1.4 Conjugate priors

The problem with Bayesian inference is that obtaining the closed-form solution of the likelihood function might be easy to solve analytically by hand; however, solving it using a computer might in some cases be drastically harder. One solution to this can be to sample from the posterior density function in order to approximate the real posterior distribution; however, this might still be computationally demanding for a large enough model. An alternative to this is to use conjugate priors.

A prior distribution is called conjugate when for a given likelihood the posterior distribution is of the same type (also called a family) of distribution as the prior one, meaning that the only change is in the parameters of the distributions. When sufficient statistics exist for the conjugate prior distribution, the whole learning process becomes just updating the sufficient statistics, which requires minimal computational resources. This allows one to repeat the same operations in order to transition from a prior distribution to a posterior one and also track the changes between the distributions over time.

3.1.4.1 Examples of conjugate priors

Different kinds of conjugate prior distribution and model (likelihood) combinations used to model different types of variables exist, and it would be practically impossible to enumerate them all; however, there are some that are widely used, for example Beta-Bernoulli, Gamma-Poisson, Gamma-Exponential, Normal-InverseGamma, and Dirichlet-Multinomial [34], presented in Table 3.1 below.

Prior Distribution	Likelihood	Variable Type
Beta	Bernoulli	Binary data (success/failure)
Gamma	Poisson	Count data (event occurrence)
Gamma	Exponential	Continuous data
(Inverse-)Gamma	Normal	Continuous data (unknown σ^2)
Dirichlet	Multinomial	Categorical data

Table 3.1 Bayesian Conjugate Priors and their Corresponding Likelihoods

3.2 Bayesian model aggregation

So far the models discussed up until this point might be sufficient for some tasks; however, for more robust problems, they might not be able to fully adjust to them, resulting in under-fitting model. In order to solve more complicated problems, simple models are commonly aggregated in a certain way in order to leverage their respective strengths in order to get a better estimate.

3.2.1 Hierarchical models

One way to combine the models is to use them in a hierarchical manner, where one or more models are deployed to approximate certain variables, which are then used as parameters for the next layer of models. This hierarchical approach allows to utilise the models to learn simple relationships from data, and this knowledge is then used to model more complex relations. This approach also allows for mixing different families of models to compensate for the shortcomings of using only one type of model.

An example would be Bayesian networks, which are also a hierarchical model that incorporates the lesser models into a Directed Acyclic Graph (DAG), where the modelled variable represents the nodes of the DAG and the edges represent their dependencies, which means that for an orientated edge $u \to v$ the variable v directly depends on the variable u. Compared to feedforward neural networks, where usually only the output from the last layer is used, but in the case of Bayesian networks, not only can any layer be used, but if there were missing inputs for any underlying model, their values can be easily sampled.

The DAG representation can be used for every model. In Figure 3.1 an illustrative graphical representation with three nodes is shown. The first node denoted κ represents a constant variable, there is no circular border, and the arrow appears to connect the symbol directly. The circular grey node denoted by y then represents the observed variable, which is any value that can be measured or computed. The white circular node with symbol ρ represents a latent variable, which is modelled using this illustrative model.

$$\begin{tikzcd}\kappa \arrow[r] & \rho & y \arrow[l]\end{tikzcd}$$

Figure 3.1 Graphical representation of illustrative model showing constant, latent, and observable variables (from left to right)

3.2.2 Bayesian model averaging

The Bayesian model averaging [35] is an essential method to combine multiple models to improve the point estimates of a certain variable. This method takes the posterior predictive densities or point estimates (for simplicity, denoted as g_i) of each of m models, and calculates their weighted average to improve the final accuracy as

$$g(y) = \sum_{i}^{m} w_i \cdot g_i(y), \tag{3.9}$$

where w_i represents the weight assigned to the model. When no additional information about the models is known, for example, how accurate they are, they can be assigned uniform weights. This is a special case of weighted average, where the results of each of the models have equal influence on the result.

However, it can happen that a certain subset of the models that generally performs worse, for example, due to a higher level of noise. If equal weights were assigned, these models would have the same impact on the result as the models that perform well, which would skew the result. To mediate the influence of models that perform worse, weights that are proportional to the performance of each model are usually introduced to improve the overall performance.

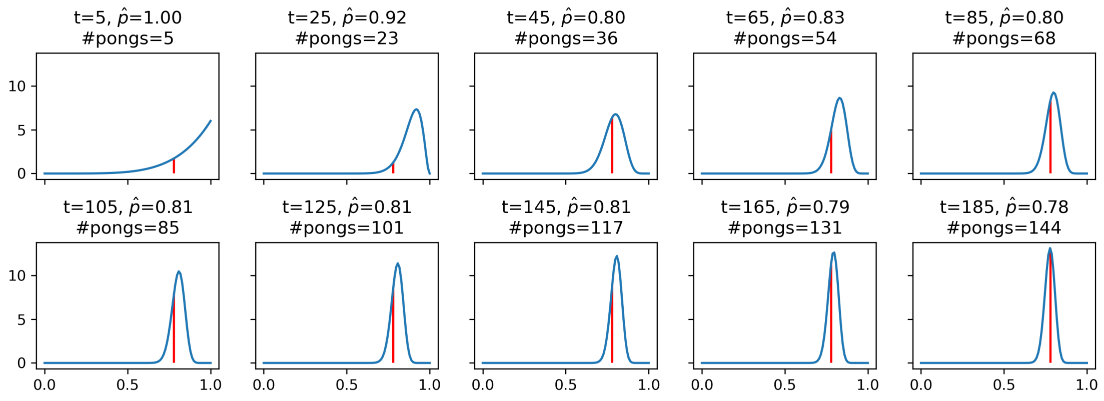
The weights can be assigned on the basis of expert knowledge, heuristic approach, or can be estimated using a meta-model. The meta-model learns the weights based on the performance of each model, for example based on the accuracy, or loss function.

3.2.3 Bayesian model switching

Model switching is another way to combine models with different properties in order to increase the accuracy. Imagine a task in which the model is supposed to predict the dosage of a certain drug. The usual parameters mentioned in the dosage instructions in the manuals are sex, weight, and age. Now, for a given patient, the method will take into account the known information about the patient and select the model with the most likely prediction. This illustrates the Bayesian switch of the model, based on the given context.s

3.2.4 Modelling a ping response probability

An example illustrating the iterative learning abilities of Bayesian inference is presented in Figure 3.2, where probability estimates over time are shown for the results of simulated ping requests, where the probability of answering with pong was 78%. Initially, after the first 5 updates, the estimated probability $\hat{p}$ was 100% since the model did not observe any failed pings, then over time, as the pings started to fail, the modelled probability started to converge to the real probability.



■ **Figure 3.2** Estimating probability of pong response to ping using Bayesian inference compared to real probability (red)

Chapter 4

Application of Bayesian Inference on traceroutes

Each record represents exactly one traceroute probe, and there is a relatively long delay between each new measurement. The decision was made to process them in a mini-batch manner. This way, each of the traceroutes is processed individually and instantly, in order to obtain as much useful information about the measurement as possible.

Traceroute measurements are expected to contain various pieces of information that would reflect problems in the network. It should be possible to model each of them using Bayesian inference and detect anomalies on them. Several experiments were conducted, each of which aims to model a different piece of information. These experiments are described in the following section. To develop these models, traceroute measurements between the sites ”CA-SFU-T2” and ”CSCS-LCG2” were used from January 1 to the last day of February.

4.1 Experiment: Boolean fields

The initial points for modelling were the Boolean fields mentioned in Section 2.2.1. Since the values of these fields can either be true or false, it might be useful to know how likely each event is to occur.

A Beta-Bernoulli model can be used to model the probability of each of the Boolean fields by utilising the Beta distribution as a prior distribution with Bernoulli likelihood to learn the probability of the field having the value true.

This is described using the following mathematical definition

$$
\begin{gathered}
y|\theta \sim Bernoulli(\theta) \\
\theta|\alpha, \beta \sim Beta(\alpha, \beta),
\end{gathered}
$$

where $y|\theta$ means y given the parameter θ, or by using the graphical representation below.

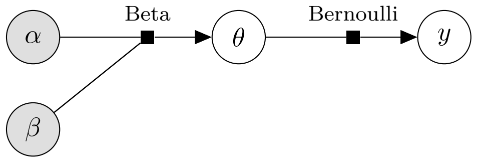


Figure 4.1 Graphical representation of the Beta-Bernoulli model

The Beta distribution is used to model the latent variable θ with the aim of getting as close to the real probability as possible, which is a parameter of the Bernoulli distribution. To do this, the Beta model uses two variables α and β, which serve as pseudo-counts of true and false values, respectively. The reason to use pseudo-counts instead of normal counts is that there can be a learning rate parameter $\gamma \in (0, 1]$ that is used to slow down the convergence.

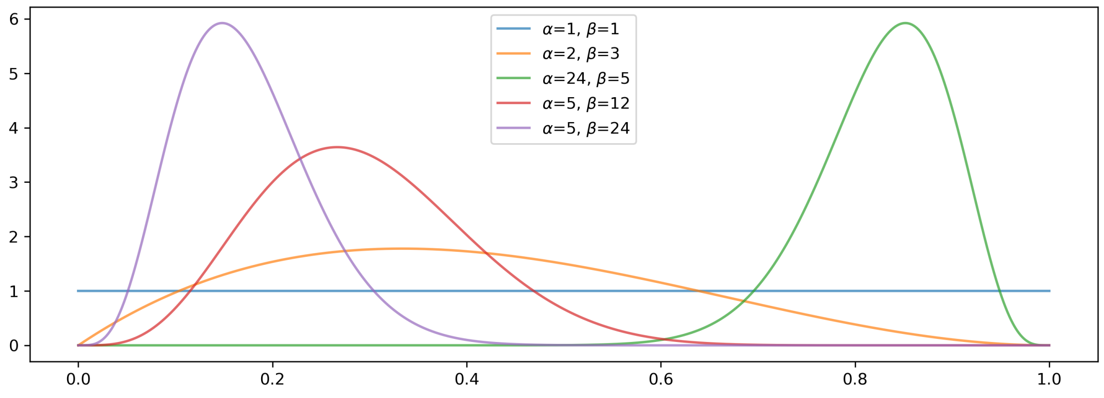


Figure 4.2 Beta distribution with different combinations o f parameters

Figure 4.2 displays multiple Beta distributions with various parameters. It is noticeable that the larger the values of α and β, the more concentrated the distribution around a certain value. This represents the model's belief that the real value is likely close to the MAP value; however, with less evidence, the model is less certain, resulting in a flatter distribution. The parameters α and β are also interchangeable in the sense that if their values were swapped, the resulting probability estimate would be complementary to the original, which can also be observed by inspecting the purple and green line distributions and their parameters. A special case is $\alpha = \beta = 1$, which gives the same weight to all values. This combination is commonly used as an uninformative prior, which is then gradually updated with new data using the following equations

$$\begin{aligned}\alpha_0 &= 1,\ \beta_0 = 1\\ \alpha_{t+n} &= \alpha_t + \gamma c_n\\ \beta_{t+n} &= \beta_t + \gamma(n - c_n)\\ \theta_{t+n} &= \frac{\alpha_{t+n}}{\alpha_{t+n} + \beta_{t+n}}\\ \sigma^2_{t+n} &= \frac{\alpha_{t+n} * \beta_{t+n}}{N^2 * (N+1)},\end{aligned}$$

as stated in [36], where c_n is the count of values *true* in the batch of size n and N is the total number of observations.

4.1.1 Results

In Figure 4.3, the blue line represents the value of the learnt parameter θ, surrounded by credible intervals, representing the uncertainty about the value of the parameter.

In Figure 4.3 the true values of the field are shown as green dots for the true values and red dots for the false values. The transparency of the dots was increased to display the density of these points to get a better idea of the real distribution. The estimated probability starts at 100%, then rapidly drops to 50% when false values start to appear, and gradually converges to approximately 60% while the credibility interval shrinks, giving more confidence in the estimate over time. This means that for the traceroutes between ”CA-SFU-T2” and ”CSCS-LCG2” the majority, however small, contains information about the whole path. There is also a significant decrease in records in the period of late January, which corresponds to the observation in Section 2.1.

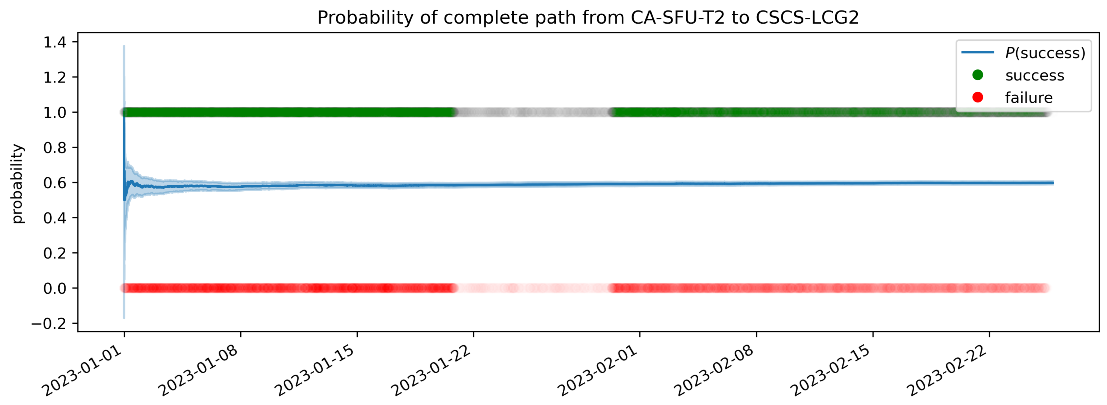
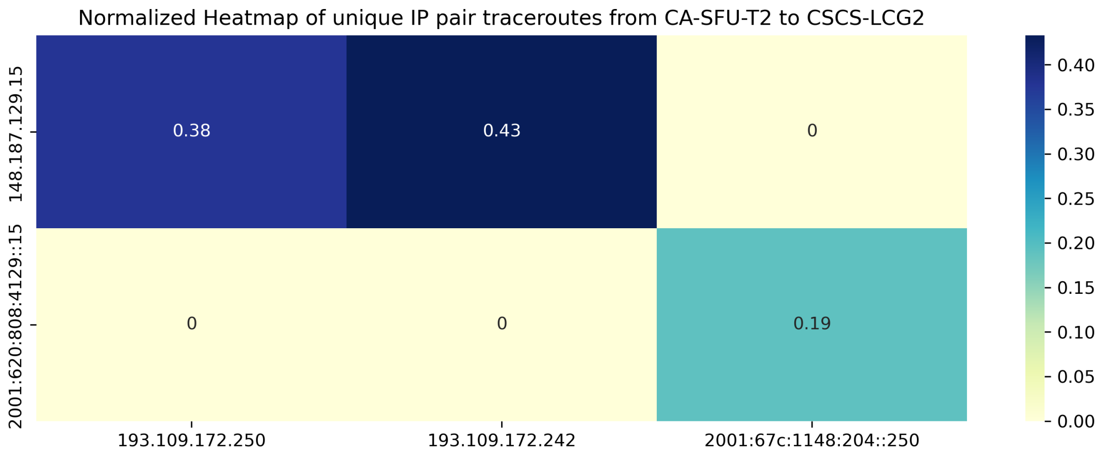


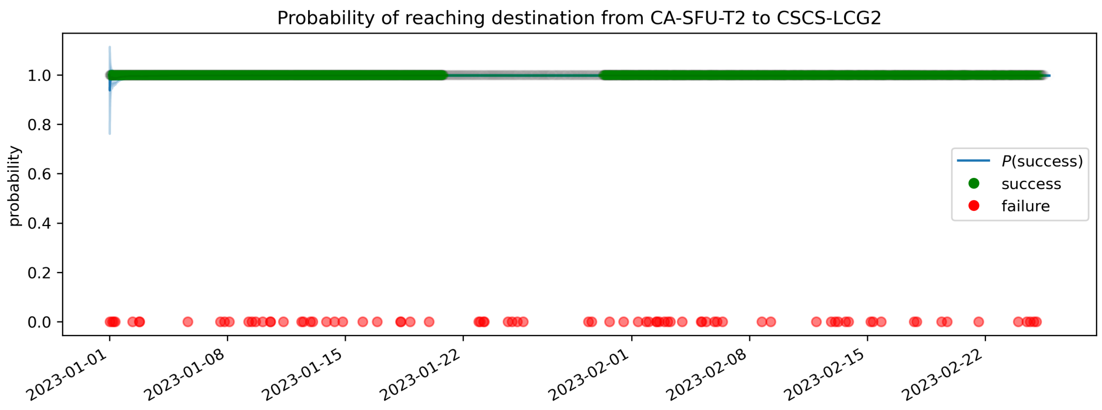
Figure 4.3 Estimated probability of path being complete

The probability of the traceroute reaching the destination in Figure 4.5 is very close to 100%, which means that the vast majority of traceroutes reach the destination; however, there were still traceroutes that did not reach it, but since this happens quite rarely, it might be sensible to consider them as anomalies. On the other hand, looping traceroutes never occurred within the observed period, resulting in the estimated probability of looping traceroutes being 0%. Figure 4.7 shows the results for IPv6 field. The estimated probability of IPv6 communication quickly converged to 20% and after the beginning of February the probability dropped to 15%, which shows that the dominant communication between these two sites was carried out using version 4 of the protocol.

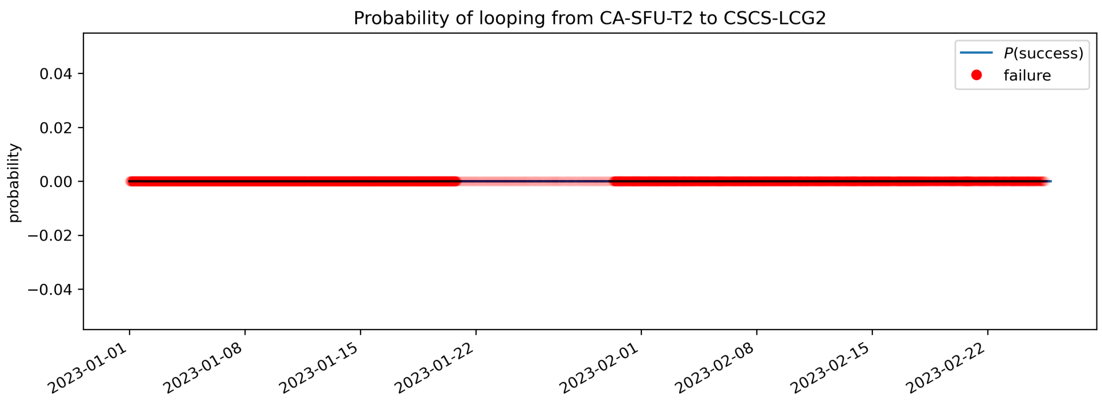
In Figure 4.4 the proportions of the number of traceroutes are shown on a heat map. From the heatmap, it is clear that there are two distinct IPv4 device pairs, where one is more common than the other with a roughly 5% difference. There is one pair of IPv6 devices doing traceroute probes, however, the number of records is significantly lower compared to the second protocol. After discussion with co-supervisors, it was made clear that sites may aggregate multiple data centres, which might or might not be physically close to each other, which means that there is no guarantee that they use same physical paths for communication; the decision was made to model the data on the device-to-device level and if needed, propagate the results into site-to-site level.



■ **Figure 4.4** Heatmap of IP pairs



■ **Figure 4.5** Estimated probability of traceroutes reaching the destination



■ **Figure 4.6** Estimated probability of encountering looping traceroute

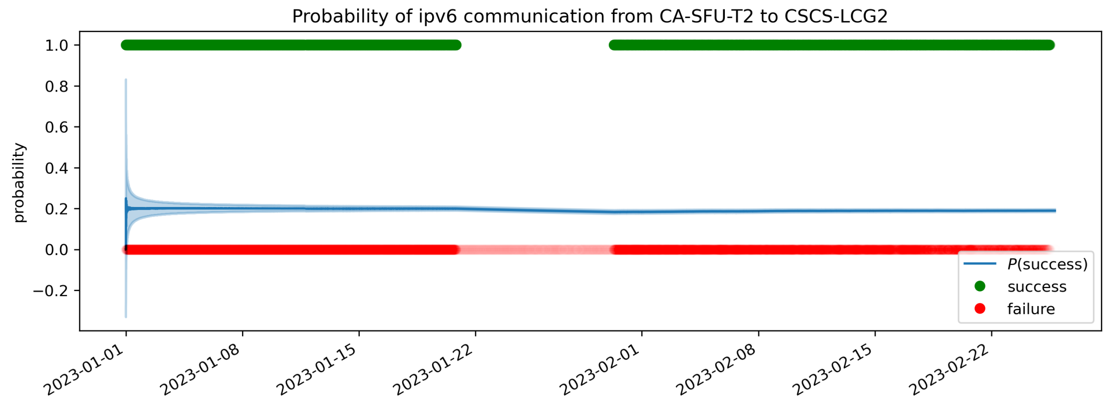


Figure 4.7 Probability of IPv6 communication

This was later shown to be a step in the right direction, since mixing the data with different sources and destinations IP addresses slightly skewed the probability of observing a complete path toward a lower probability, as shown in Figure 4.8, since the IPv6 communication was having fewer complete paths in general; however, since it also has fewer records, the change was not that significant, which might not be true in general for different pair of sites, and mixing could result in a loss of important piece of information if the distributions are significantly different. In this case, this is true only for the "path_complete" field.

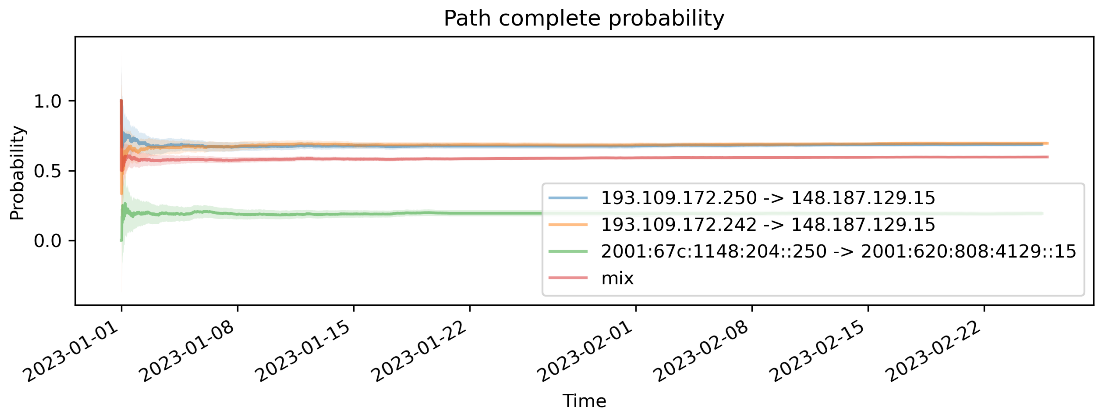


Figure 4.8 Comparison of device-to-device and site-to-site probability estimates showing that modelling marginalised data might improve the performance

4.1.2 Anomalies on Bernoulli distribution

Due to the nature of the Bernoulli distribution, to detect anomalies within the data, only observations and learnt probabilities can be used to create a simple and straightforward detection system. When each of the Bernoulli models is initialised, it can be given a scorer function s, which takes the observed value x and the prior probability of the event p, which is then used to categorise the measurement as anomaly or normal; otherwise, the model does not detect anomalies at all.

Few heuristic approaches were tested, from which one uses a simple probability threshold combined with the value of observation, but it seems that it performed better than the rest. This method can be setup for each model individually to better accomplish the goal. However, the looping model was an exception. For this model, every measurement with a true value is classified as an anomaly, since the looping event is considered problematic. This means that the scorer form was a simple identity of x.

$$s(x, p) = x \tag{4.1}$$

This was not obviously applicable to the rest of the fields, since they might, and probably will, differ for each unique pair of devices, and there does not exist a general rule of thumb which would solve this. This led to the creation of a piecewise scoring function that classifies both based on the observed value and the probability.

$$s(x, p) = \begin{cases} x, & \text{if } p \leq L \\ \neg x, & \text{if } p \geq U \\ false, & \text{otherwise} \end{cases} \tag{4.2}$$

where $U, L \in \mathbb{R} : 0 < L < U < 1$. The scoring function from Equation 4.2 is based on an idea of two non-overlapping regions divided by a margin space of uncertainty. This equation uses two additional parameters L and U, which represent the lower and upper probability bounds. If the probability is less than or equal to the lower bound, the measurement is classified as an anomaly if the value is true. Similarly, if the probability is greater than or equal to the upper bound, the false values are marked as anomalies. The fact that the lower and upper bounds cannot be equal to each other creates an uncertainty margin between them. This margin represents a probability range for which it is difficult to distinguish normal measurements from anomalies. The larger the uncertainty margin, the less likely the scoring function is to create a false positive. For example, if the uncertainty margin ranged from 5% to 95% and the probability was 97%, it would make sense to classify measurements with false values, which would be expected to be approximately 3%, as anomalies. However, if the uncertainty range was very small, for example, from 49% to 51% and the estimated probability of 52%, the scoring function could classify a false value as anomaly, even though both outcomes have similar probabilities. It is also worth mentioning that the uncertainty margin does not necessarily have to be centred around 50%.

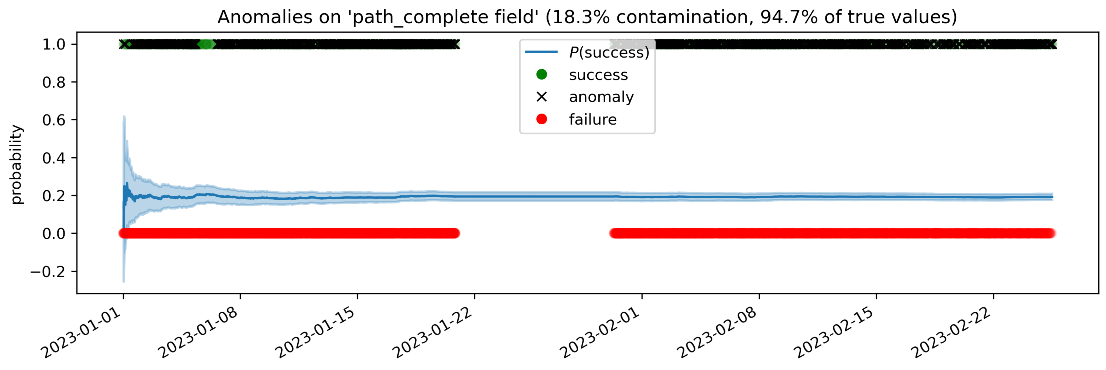


Figure 4.9 Anomalies on "path_complete" field of the IPv6 pair

The range for the "path complete" and "destination reached" were set to 20% and 80% and the result can be seen in Figure 4.9 which is the result of modelling the path completeness of the IPv6 addresses, where the only addition is that a black cross is rendered on top of the measurement that is classified as an anomaly. The anomalies in this case are roughly 18.3%, which is roughly 94% of the measurements with the path complete. There were two periods, January 1 and 7, where the probability of complete path was greater than 20%, therefore, the model did not classify them as anomalies.

4.2 Experiment: Number of hops

The next information considered as a potential source of valuable information was the number of hops on the traceroutes, or, in other words, the number of devices that respond to the ICMP request, since it is expected that the number of hops between two devices over the Internet should be similar, at least in a reasonable period of time, and that each traceroute whose number of hops would be significantly different from the usual might point towards a potential problem in the network. For example, if the common number of hops would be twelve for a certain pair of devices, traceroutes with three and twenty hops might be understood as traceroutes that ended prematurely and traceroutes that had to undergo unexpectedly larger path, respectively.

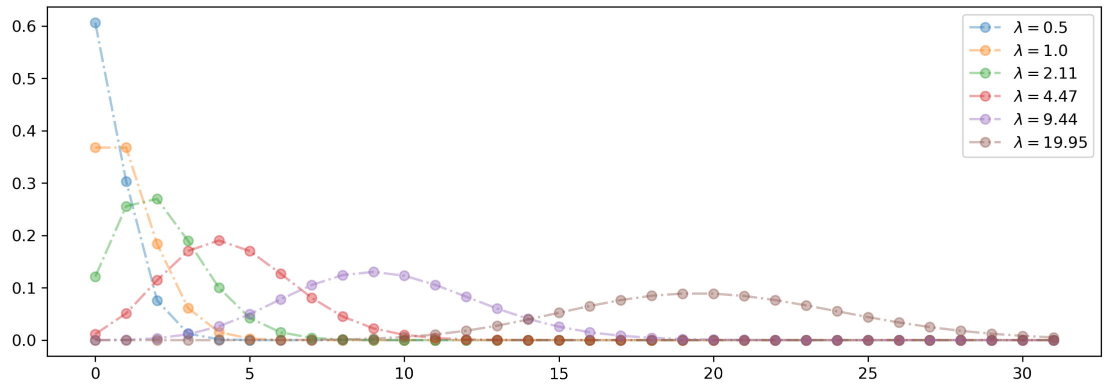
Insights gathered from this information might then be taken into the context of output from the models in the previous section, which can leverage the output from models from the previous section. For example, to see whether the twenty hop traceroute reached the destination, or whether or not it was looping.

The number of hops is a discrete random variable Y, which by the definition of the traceroute utility can take on any value in the range $0, 1, 2, \ldots, 32$. However, the value 0 is never present in the data set, since if traceroute fails while contacting the first router on the path, the list of TTL values is empty. Since the traceroute probes are expected to be independent of each other, the number of hops should be independent as well.

This means that the variable can be, in fact, modelled using the Poisson distribution, which is used to model the number of events occurring during the separate time periods. The Poisson distribution has only one parameter $\lambda \in (0, \infty)$, which represents the average rate of events that occur. It represents both the mean and the variance of the distribution. The probability mass function of the distribution is

$$f(y|\lambda) = P(k = y) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{4.3}$$

by definition, the distribution works with non-negative and discrete support. In Figure 4.10 are multiple Poisson distributions, each with a different parameter λ. The closer the value of λ is to 0, the steeper and more condensed the distributions, but as the value increases the distributions shift towards larger values, but at the same time they also become more flat, which is reasonable since λ is both the mean and variance of the distribution.

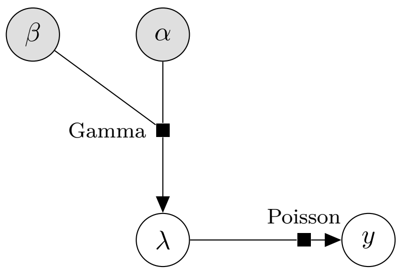


■ **Figure 4.10** Poisson distribution with different values for λ showing the change in shape of the distribution

The Poisson likelihood can be paired with the Gamma distribution with parameters α and β as a conjugate prior to modelling discrete values that represent the number of events over a period of time.

$$
\begin{aligned}
y|\lambda &\sim Poisson(\lambda) \\
\lambda|\alpha, \beta &\sim Gamma(\alpha, \beta),
\end{aligned}
$$

which can be described again, also using a graphical representation below.



■ **Figure 4.11** Graphical representation of the Gamma-Poisson model

In this case, the hyperparameters α and β are used as input to the Gamma distribution, which is used to estimate the value of the latent parameter λ. At each step, for a given batch of size n, the sum of the observer variables $\sum_i^n y_i$ is added to the parameter α, while β is used to store the total number of events that occur by adding n to it. The whole process of learning is shown in the following equations

$$\alpha_0 = 1,\ \beta_0 = 1$$
$$\alpha_{t+n} = \alpha_t + \sum_{i}^{n} y_i$$
$$\beta_{t+n} = \beta_t + n$$
$$\lambda_{t+n} = \frac{\alpha_{t+n}}{\beta_{t+n}}$$

, which means that the estimated value for λ is estimated as the average number of events per number of trials, as shown in [36].

4.2.1 Results

The number of hops on each trace route is shown in Figure 4.12, from which it can be deduced that most traceroutes have ten or eleven hops, but there are few measurements that contained only nine hops; however, there were also few measurements in late March, which had only a few hops, and there were also eight traceroutes that took more than twelve hops.

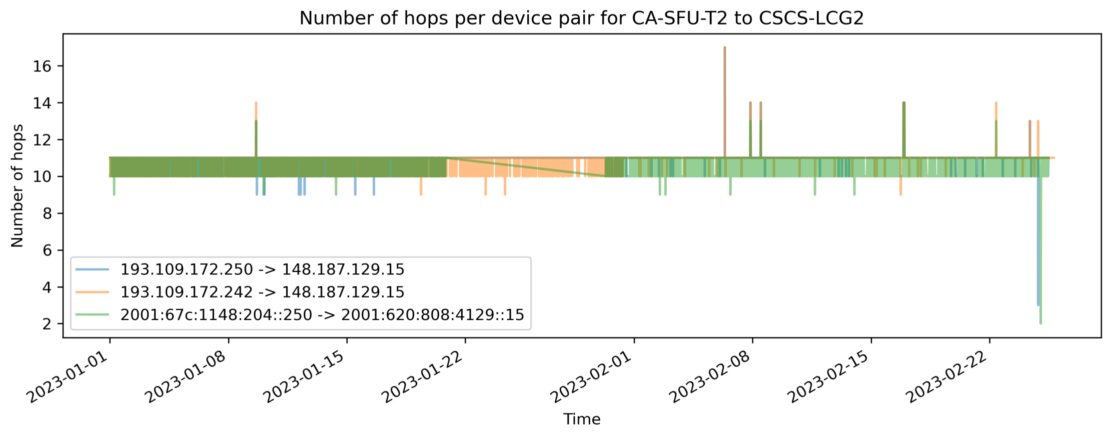
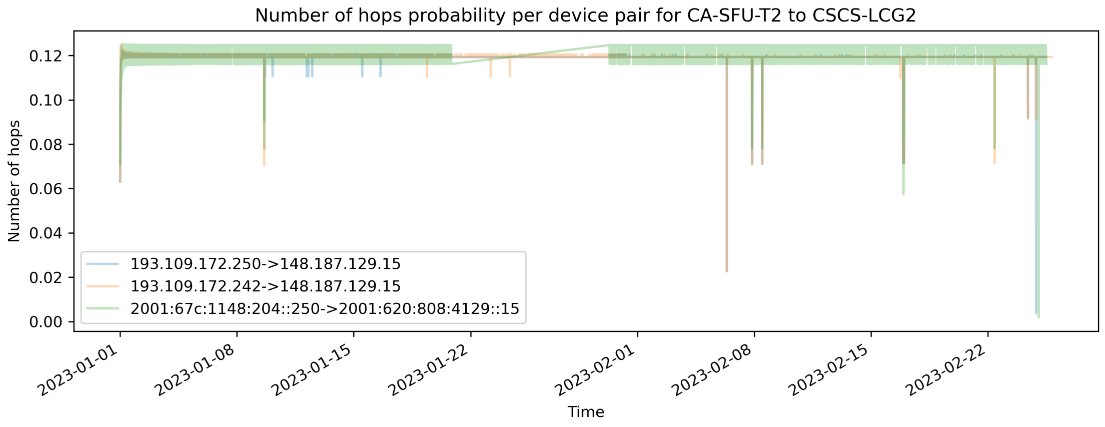


Figure 4.12 Number of hops from traceroutes between January and February for each device pair

When these values were converted to probabilities using a separate model for each unique pair of devices, some interesting things occurred. The probability of IPv4 quickly converged to 12% on average, which is true even for the IPv6, however in this case there was more variance in the probabilities.



■ **Figure 4.13** Estimated probabilities for number of hops of each traceroute from January to February

There are traceroutes with probability less than 10% and most of them were traceroutes with more than 13 hops, but two of them were traceroutes with fewer than ten hops. The first of them was in early January, where there was still a large degree of uncertainty in the model, and the second was in late March, where there were traceroutes with two or three hops. Some of the nine hop traceroutes are also distinguishable in the plot. This gives enough confidence that the Poisson distribution is a suitable way to model the number of hops.

4.2.2 Anomaly detection on number of hops

Similarly to the Bernoulli model, the only things that can be used for the classification of anomalies are the observed values and the model parameter. However, because of the properties of the Poisson distribution, it is possible to effortlessly calculate the probabilities for each of the values, which provide additional information about how the measurement aligns with the distribution.

The decision of the scoring function was based purely on the probability of the observed value given the Poisson distribution with parameter λ. If the probability of the observed value was less than a probability threshold, it was marked as an anomaly; however, the right question is what the threshold value should look like. Meaning that the scoring function has the following form:

$$s(y, \lambda) = P(y|\lambda) < T, \tag{4.4}$$

where T is the probability threshold.

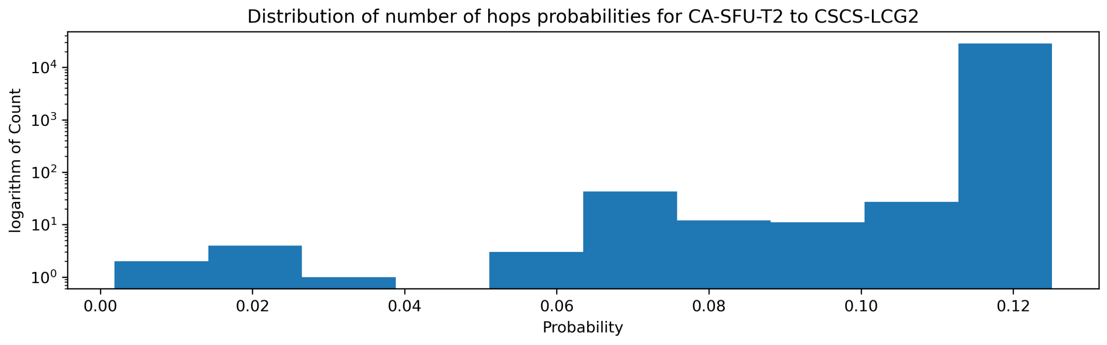


Figure 4.14 Histogram of estimated number of hops probabilities with outlying values with probability less than 5%

In Figure 4.14 the histogram of the probabilities of the observed values is shown. In the histogram, the probabilities of the vast majority of values are greater than 10%, which are values likely close to the MAP estimate. There are still a significant number of values with probabilities within the range of 6 to 10%. The last group consists of values with probability less than 5%, which contains a rather tiny number of samples.

On the basis of the histogram, the decision was made to heuristically set the probability threshold at 10%. However, this does not directly imply that 10% of the data will be marked as anomalies. Figure 4.15 contains the same data as Figure 4.13, but this time the observations are with a probability lower than the threshold. Using this approach, 76 values were marked as anomalies, which is roughly 0.26 percent of the data set. However, it is worth mentioning that few of them were from the early process of learning, while there was still a high uncertainty about the learnt distribution, which can be solved by starting the anomaly detection only after certain number of updates.

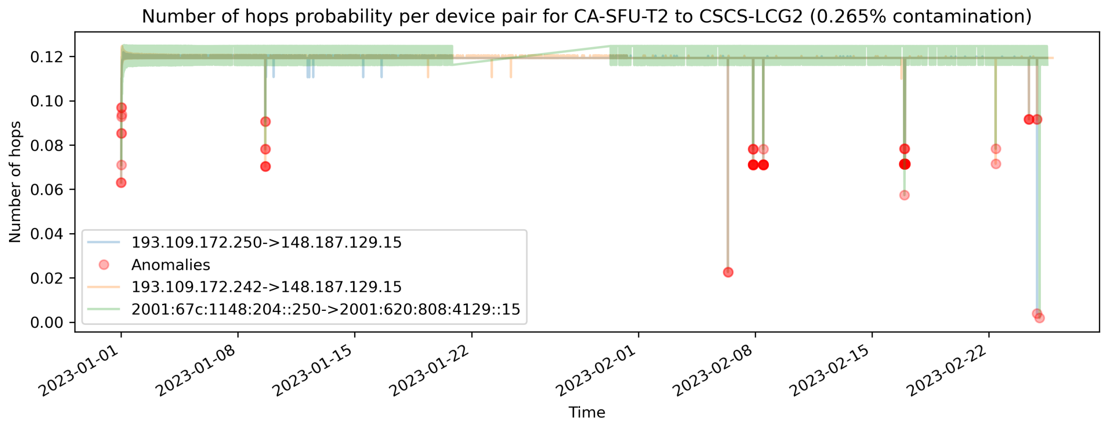


Figure 4.15 Anomalies detected on number of hops based on the probability threshold

To eliminate false positive anomalies at the beginning of learning, a decision was made to allow the Poisson model to inherit the prior estimate of λ from the value of the first observation. Since normal values are expected to be observed more frequently, the first observed value would likely be closer to the real value of λ than a constant, and if the initial value were significantly distant, the model would be able to correct the parameter with additional data. This also solves the problem of each pair of devices having a different number of hops.

4.3 Experiment: Duration of the traceroute

After modelling the lengths of the traceroutes, another interesting variable was the duration of the traceroutes. Modelling the duration enables detection of traceroutes that take significantly more or significantly less time. The only problem is that the traceroute duration field is not present in the data and would have to be added to model it. An alternative to this, which would not automatically discard historical records, since there is no way to obtain their duration, is to use the sum of RTT, which would act as an approximation of the duration of the traceroutes.

The reason why it would only be an approximation is due to the fact that most traceroutes have at least one unreachable device, as discussed in Section 2.4, which means that only for complete paths the sum of the RTT values would equal the real duration, and traceroutes with missing hops would represent only the lower bound of the duration. This also means that there will be a significant amount of noise in the data; however, if the lower bound of the duration shows signs of abnormality, the real duration of the traceroute would also show them. The only downside here is that the usefulness of the measurement decreases proportionally with the number of unreachable devices, which means that the lower the number of hops, the harder it would be for the model to detect the anomalies.

The visualisation in Figure 4.16 shows, that the sum of RTT was roughly between 170 and 300 milliseconds in most cases, however, for the IPv6 the range was between 210 and 300 milliseconds, meaning that the IPv6 traceroutes might be taking more time in general. There are seven periods with an unusually large sum of RTT. The first between 8th and 11th of January, the second between 15th and 18th of January, and 3 occurred between 5th and 10th of February, then there were two occurrences roughly on 17th and 22nd of February. There are few traceroutes with significantly lower total RTT between 8th and 30th January, but there is one cluster at the end of February, where the total RTT is less than 30 milliseconds.

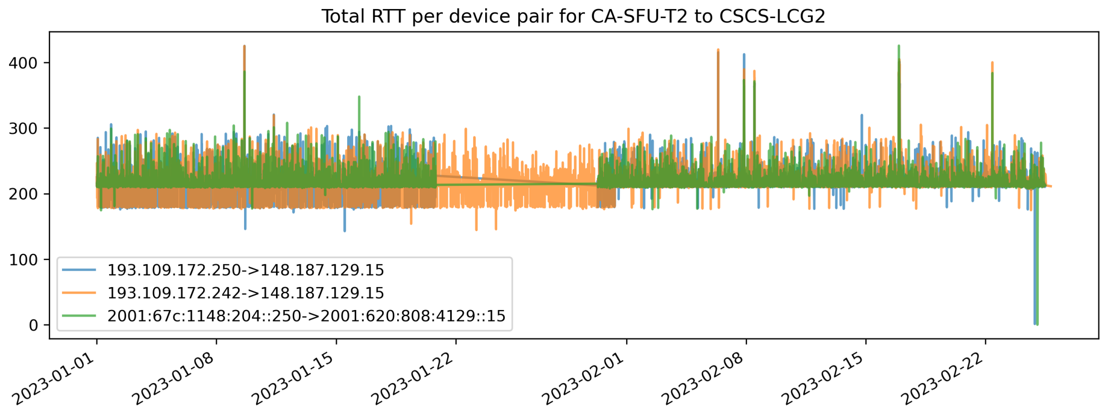


Figure 4.16 Total RTT over time for device-to-device communication showing

Since RTT is a continuous variable, none of the previously defined models can be used to reasonably model this variable. The real distribution of RTT is unknown, but it is assumed that the RTT is independent and identically distributed. Now, given the Central Limit Theorem (CLT), which states that the sum, or average, of samples converges toward a Normal distribution as the number of samples increases.

This means that it should be possible to use the Normal distribution as the likelihood of the data. There are two possible options for the conjugate prior, as shown in Table 3.1; however, since the variance of the distribution for the sum of TTL is unknown, the obvious choice is the Inverse-Gamma distribution.

$$y|\mu, \sigma^2 \sim Normal(\mu|\mu_0, \sigma^2)$$
$$\sigma^2|\alpha, \beta \sim InverseGamma(\alpha, \beta),$$

where μ and σ^2 are parameters of the Normal distribution. Alternatively, by using the Gamma distribution

$$\sigma^{-2}|\alpha, \beta \sim Gamma(\alpha, \beta), \tag{4.5}$$

since if it is true that $X \sim Gamma(\alpha, \beta)$, then also $X^{-1} \sim InverseGamma(\alpha, \beta)$.

This can be again represented using the directed graph where α and y denote the observed variables, and are used to model the parameters of normal distribution μ and σ^2.

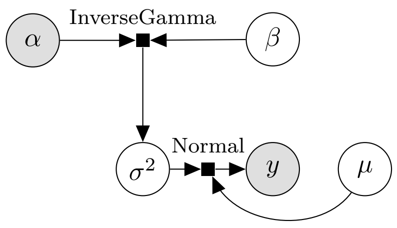


Figure 4.17 Graphical representation of Normal-InverseGamma model

This time, since there are two parameters, the learning process consists of two parts that both depend on the error between the observed and expected variables $(x - \mu)$. The first part uses the squared error to update the variance of the model, and the second uses the fraction of the error to update the parameter μ, which will be used in the next round to recalibrate the model once again. The update equation are defined as follows:

$$\alpha_0 = 1,\ \beta_0 = 1,\ \mu_0 = 0$$
$$\beta_{t+n} = \beta_t + \gamma\frac{t}{t+1}(y - \mu_t)^2$$
$$\alpha_{t+n} = \alpha_t + \gamma n$$
$$\sigma^2_{t+n} = \frac{\beta_{t+n}}{\alpha_{t+n}}$$
$$\mu_{t+n} = \mu_t + \frac{1}{t+1}(y - \mu_t),$$

as shown in [37].

The goal here is essentially to minimise the error by getting as close to the real value of the parameter μ as possible. Additionally, the factor $\frac{1}{t+1}$ allows one to make large corrections at the beginning of learning, which then becomes smaller as the model observes more data over time.

In Figure 4.18 the process of learning the parameters of the normal distribution is shown. From the line graph, it is clear that the model was able to quickly converge with the estimate for μ and σ for the three pairs of devices. The learnt value of μ was very similar for both devices using the IPv4 communication. The IPv6 communication had slightly higher value for the estimates of μ, however, over time the differences between them began to vanish. The process of learning the parameter σ was slightly slower to converge, and there are also few sudden increases in the estimated value, likely caused by the peaks described above. However, in the end, the estimates were also very close to each other.

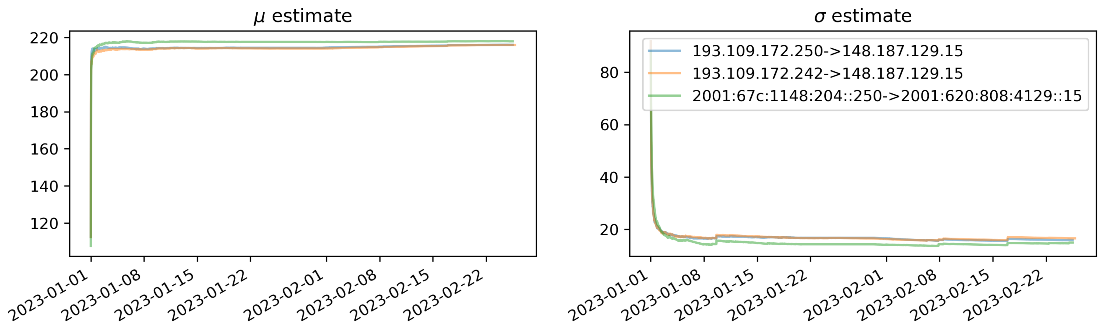


Figure 4.18 Learning process visualisation showing fast convergence

4.3.1 Anomaly detection on Normal distribution

For the Normal distribution, the most likely to occur is the expected value μ. Since some degree of variance is expected, values that are, in some sense, close to μ should also be considered normal, but if the values are far enough from μ, they might represent actual outliers.

The empirical rule of Normal distribution, sometimes called ”68–95–99.7” rule, states that for the standard Normal distribution with parameters μ and σ^2, approximately 68% of the observations are in the range of one standard deviation from the expected value; 95% of the observations are in the range of two standard deviations, and for three standard deviations, the percentage of observations should be approximately 99.7% [38].

This method can be used for the detection of anomalies for data from Normal distribution. If any observation is outside a predefined σ factor belt around the expected value, it might be considered as anomaly. The problematic part is finding the optimal factor, which is why the final scoring function allows us to define the σ factor parametrically. The scoring function has the following form:

$$s(y, \mu, \sigma, \alpha) = \begin{cases} true, & \text{if } y < \mu - \alpha\sigma \\ true, & \text{if } y > \mu + \alpha\sigma \\ false & \quad \text{otherwise,} \end{cases} \tag{4.6}$$

where $\alpha \in (0, \infty)$.

Using this approach, 204 traceroutes have been marked as anomalies, which is roughly 2% of the traceroutes between the devices with IP addresses ”192.109.172.250” and ”148.187.129.15”, as shown in Figure 4.19 (for results of the other two pairs of devices, see Figures A.1) and A.2. From the graph, it is clear that all the significant peaks mentioned previously were classified as anomalies; however, it seems that there are significantly more anomalies with a higher sum of RTT, which started the suspicion that these could be false positives; however, after further analysis it was made clear that in the vast majority of them, the device before the last one had a lot larger RTT than usually.

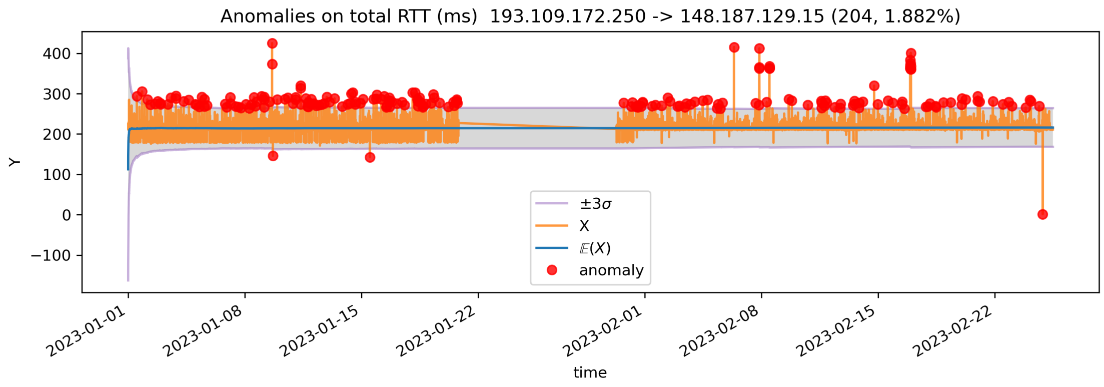


Figure 4.19 Anomalies on total RTT for an IPv4 device-to-device communication

4.4 Experiment: RTT and TTL delay

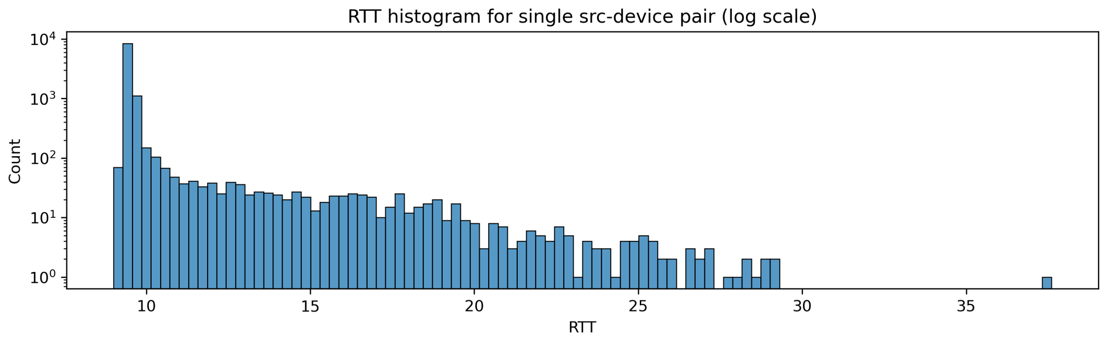
Modelling the length and duration of the traceroute definitely brings information about the usual state of the of the traceroute probes and should be indeed able to discover measurements which RTT, or TTL values are deviating from the range of expected values.

Since the values used for the two previous models are simple aggregates, they do not provide any information about the specific devices, which is not entirely helpful to the network administrators, since they would be aware that there might be an issue, but they would not be able to narrow down the exact location of the problem.

This was an important reason for researching the possibilities of modelling the RTT and TTL values in order to detect anomalies in the network even for intermediate devices. On the basis of which decision was made to model the traffic between the source device and every intermediate device on the path separately, in order to obtain information about each respective segment of the traceroute probe.

As mentioned, traceroute measurements contain sequences of RTT and TTL values for each device that responded along the way. This presents the first obstacle; since the device may not respond every time, the model must be able to work with incomplete information. Furthermore, it is assumed that when the source device communicates with another device in the network, the distribution of TTL and RTT values would be different from that of another device with the same distance (in terms of hops) from the source. There can be many reasons for this, for example different type of cables, networking and routing policies, etc.

This means that for each observed device, there should be one model for learning the RTT distribution, and one for learning the TTL distribution, which also allows one to work with sequences of different lengths, since evaluation of the sequence segments is independent from each other and for each unique sequence, a unique set of models would be used; however, the models themselves can be used for evaluation of multiple paths, which can be understood as a model switching. Another advantage of this approach is that, if a new device occurs, new models are created and used immediately for the evaluation based on the prior setup.



■ **Figure 4.20** Histogram of RTT values on source-device communication with one outlier bin with values around 37 ms

As mentioned above, TTL represents the number of hops it took to reach a certain device, which can be modelled using the Poisson distribution. This was a straightforward application of a statistical distribution and worked very reasonably for the TTL sequences; however, the Poisson distribution cannot be used for RTT, since it is not a discrete variable. Based on the histogram of the RTT values in Figure 4.20, it can be seen that the most common values were around nine milliseconds, but due to the nature of the variable, there are also values larger than that, but with increasing values, their frequency decreases rapidly. There is also a bin of outliers with a value of more than 37 ms.

Given the empirical distribution and the similarity to TTL, it seems that an Exponential distribution might be a suitable choice. The Exponential distribution has a single parameter λ, which represents the average rate of traceroutes per millisecond. Since λ is the rate, it is possible to model it using the Gamma distribution using the duration of the traceroute and a counter. In other words, it should be possible to infer the statistics about RTT using the following model:

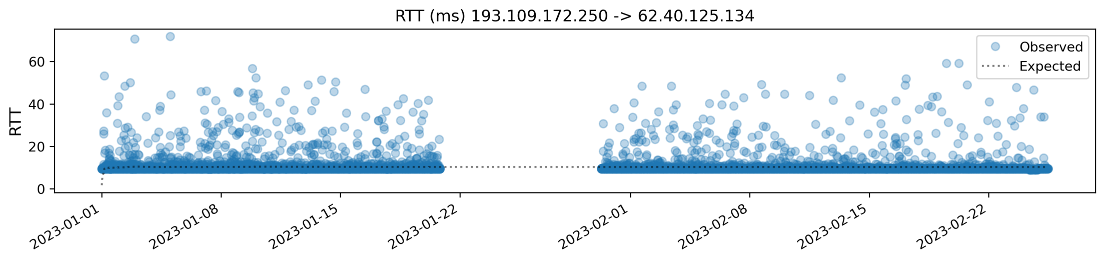
$$\begin{aligned} y|\lambda &\sim \mathit{Exponential}(\lambda) \\ \lambda|\alpha,\beta &\sim \mathit{Gamma}(\alpha,\beta), \end{aligned}$$

such that the update equations for the implementation look like follows

$$\begin{aligned} \alpha_0 &= 1,\ \beta_0 = 1 \\ \beta_{t+n} &= \beta_t + \sum_i^n y_i \\ \alpha_{t+n} &= \alpha_t + n \\ \lambda_{t+n} &= \frac{\alpha_{t+n}}{\beta_{t+n}}, \end{aligned}$$

according to [36], where α follows the number of observations and β is the cumulative sum of RTT values.

Based on the results shown in Figure 4.21, the expected RTT value $\mathbb{E}[y|\lambda] = \frac{1}{\lambda}$ is in this case approximately 10 milliseconds and is highlighted using the dotted line; however, it is noticeable that for several traceroutes, the TTL value was up to 7 times higher than expected.



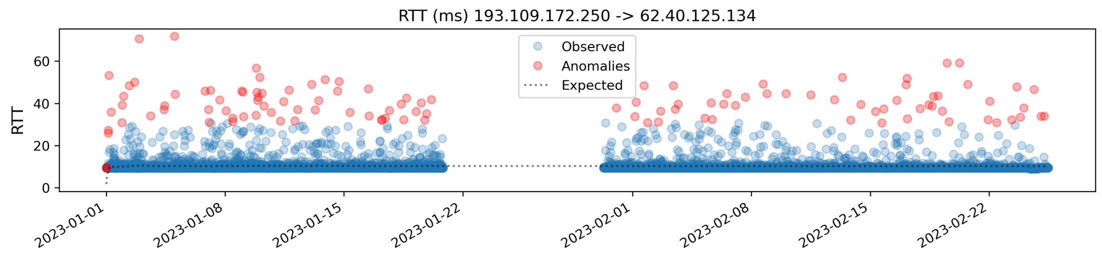
■ **Figure 4.21** Result of modelling RTT showing the expected value within the most dense cluster of measurements

The last thing for the RTT model is to determine the way to detect the anomalies. Given the nature of the distribution, it seems logical to perceive values close to the expected one as normal, and the more the value increases, the more likely they should be marked as anomalies; however, this again creates the need to explicitly set a threshold, which is impossible to set as a fixed value for all models, but thanks to the properties of the Exponential model, it is possible to once again deploy a scoring function which works with the distribution. After quite a few experiments, the final scoring function was chosen to use the survival function of the Exponential distribution, which represents the probability $P(Y > y)$. The final scoring function

$$s(y, \lambda) = \exp(-\lambda \cdot y) < T \tag{4.7}$$

was introduced with a probability threshold parameter T. This allows us to parametrically control the threshold for classifying the anomalies. The threshold value could also be learnt using another model, but for the needs of this experiment, the percentage threshold was heuristically set to 5%.

In this case, the scoring function marked 116 measurements as anomalies, which is approximately 1% of the RTT values observed for this intermediate device. After a short learning period, the value threshold converged to approximately 35 milliseconds.



■ **Figure 4.22** Anomalies on RTT model based on survival function threshold

For the TTL values, the results could be of two types. The first is the case where the TTL values are having constantly the same value, which is correct but useless for anomaly detection, and the second type, which has some variance in the measurements. Figure 4.23 with the results for the TTL values show that the model was able to learn the expected value to be 12, and also marked any measurements with the TTL value 16 and more as an anomaly.

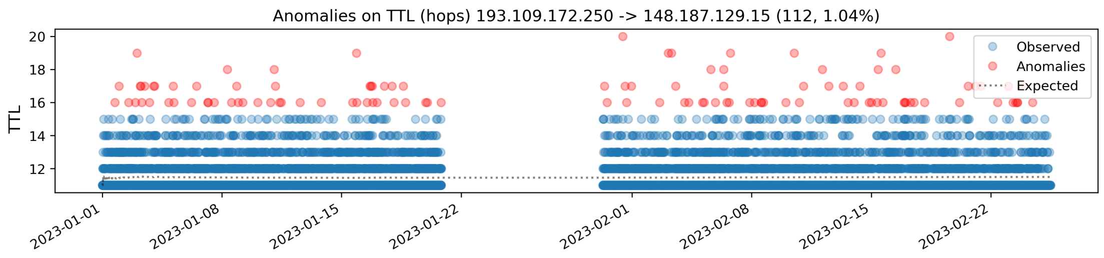


Figure 4.23 Anomalies on TTL model based on probability thresholding approach

Having a model for each device is definitely worth it for thorough network problem analysis, but it might be hard to interpret the results over time even for one source-destination communication, which becomes nearly impossible with numerous sites communicating with each other in real time. For this reason, aggregation of the output of the device-level model is necessary. The most useful information is probably the expected RTT and TTL values, which are expected to not change over time, unless there is a significant change in topology. After a few experiments, the final model was designed to calculate the difference between the expected and observed values ($y - \mathbb{E}[y]$) using devices-related models, which are also known as error or residuals.

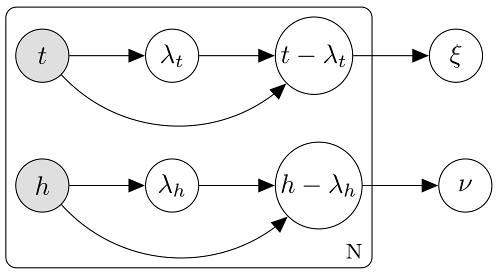
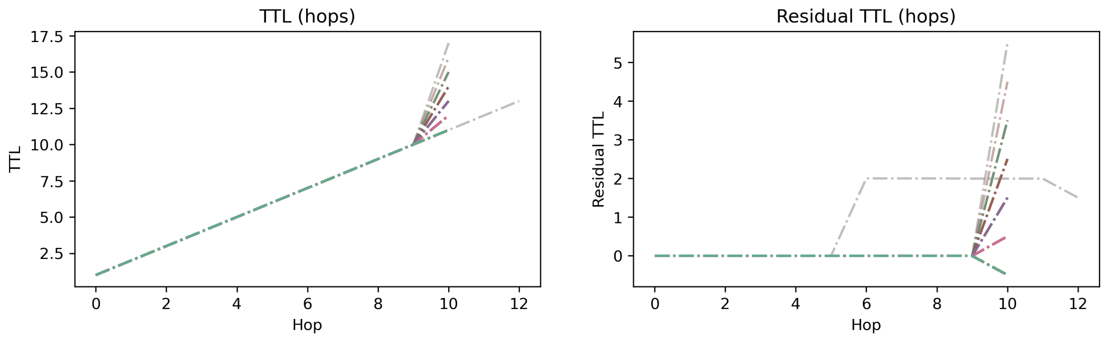


Figure 4.24 Proposed model for RTT and TTL sequences extracted from traceroute measurements

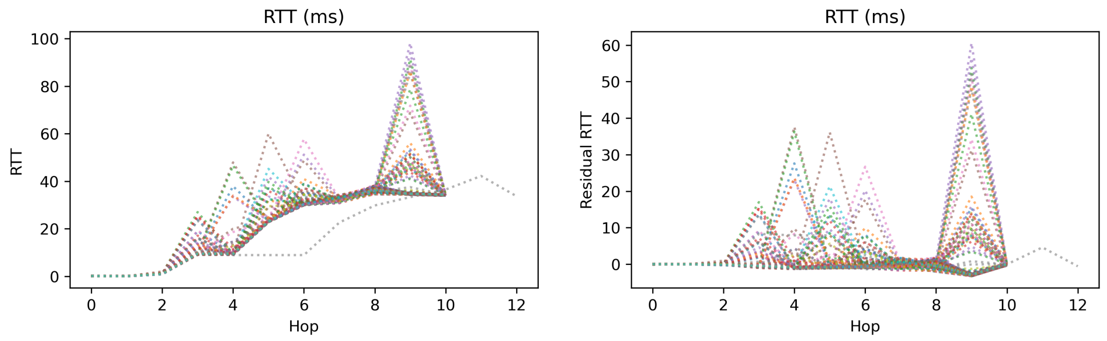
This model can be represented using the graphical model in Figure 4.24, where t represents the RTT observed for a given device and h is the value of the number of hops observed for that device. These values are used to calculate the expected values and, in addition, to calculate the residuals. This happens N times for each device in the traceroute IP address sequence, which is the reason why this process is enveloped in a "plate" that represents the repetition of this process. In the end, the residual sequences are used as input to ξ and ν, which represent the final aggregation of both sequences.

The residual transformation of the TTL sequences, which is shown in Figure 4.25, is capable of removing the positional bias introduced by the position of the device in the path, which means that the residual values are zero for most traceroutes; however, there is a significant fan-shaped deviation for the last hop, which in most cases is probably due to the final device not responding, which could be caused by it being overwhelmed, or it could be subject to some problems. There is also one traceroute displayed, which deviates from the others at the fifth hop, where it's residual has gone to 2, which can mean that the traceroute was delayed.



■ **Figure 4.25** Comparison of TTL and residual TTL showing the influence of removing learned positional biases

For the RTT sequences (see Figure 4.26), the transformation works similarly; however, there are some traceroute measurements with abnormal RTT values at some hops, which are not much affected by the transformation, as they are usually multiple times higher than the expected values. Due to this, the expected value becomes larger in value, which then causes the residual to be negative for some paths (hops 8-10). However, note that in some cases the less significant peaks might be due to a low number of observations for that specific device, which means that the expected value might not yet have converged.



■ **Figure 4.26** Comparison of RTT and residual RTT showing the influence of removing learned biases

Residual plots for a single traceroute could be useful for detecting anomalous behaviour of devices; however, due to the large volume of data, it could be difficult for the system administrator to manually check each of the traceroutes, so there needs to be some statistic that can describe the state of the traceroute, in terms of TTL and RTT values, which can then be monitored instead. The statistic might also help to remove some degree of noise from the data, making it more clear.

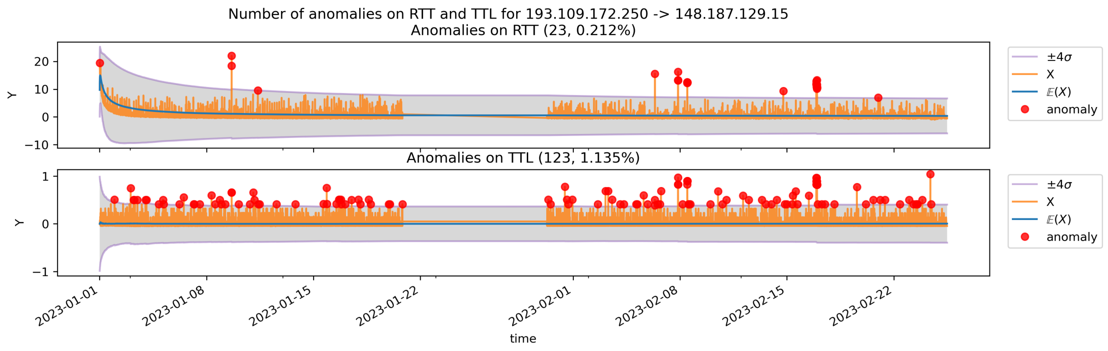
4.4.1 Aggregation and anomaly detection

For both, the TTL and RTT, the sequences of residual were used to detect anomalies. As was previously discussed, sequences can be of various lengths; therefore, a model that would be able to process sequential data, such as RNN might be suitable; however, since the goal is to create a computationally lightweight model, since it would be used on a large number of data samples, the decision was made to use simple aggregation of the sequence.

The selected aggregation was a simple average of the residuals

$$\frac{1}{N}\sum_{i}^{N}(y_{i,t} - \lambda_{i,t}), \tag{4.8}$$

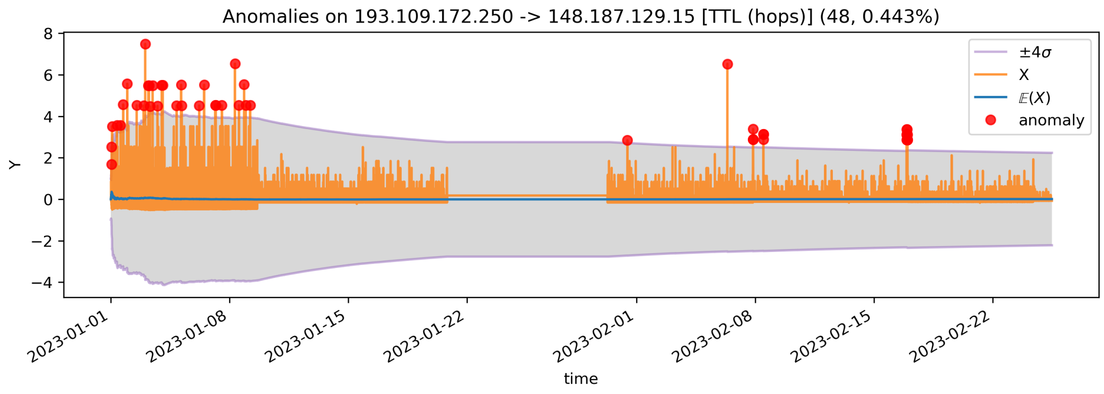
where i corresponds to the i-th hop of the traceroute and index i, t corresponds to the variable for i-th hop in time t. The aggregations were then modelled using the Normal distribution with the same assumptions based on the CLT, the same way as for the total RTT. The aggregations were then marked as anomalies if they were outside the $\mu \pm 4\sigma$ interval, which can be reconfigured again based on the knowledge of the network specialist. This aggregation can be understood as the average delay and the average number of devices that did not respond.



■ **Figure 4.27** Anomalies detected on average residual RTT (top) and TTL (bottom)

Based on the results for RTT aggregated sequences, 23 traceroutes were classified as anomalies between devices "193.109.172.250" and "148.187.129.15", which is approximately 0.2% of the total records for that period (see Figure 4.27). Traceroutes marked as anomalies occur in the second week of January, at the turn of the first week of February, and in the third week of February. After comparing the result with the results from the previous section, the large peaks are also captured there, but this result is much less noisy.

For the TTL sequence, the model classified significantly more traceroutes as anomalies; however, their TTL did not indicate anything alarming. After further investigation, it was clear that in most of these cases there are many devices without residuals that outweighed the devices with nonzero errors. Since the devices without any residual in TTL values do not provide any information that can help identify anomalous traceroutes, they were removed from the final aggregation. This at least partially solved the issue; however, this approach could still be imperfect. The model resulted in the classification of 48 traceroutes as anomalies (see Figure 4.28), which is significantly less than previously. The anomalies in the first 10 days of January are usually due to a single device having a nonzero residual at each time the source device had to send at least four additional ICMP requests. After the first ten days, the traceroute probes started to produce TTL sequences with fever non-zero residuals, as there was probably more traffic after the start of the new year over the Internet in general. After a further learning period, the model classified several measurements as anomalies in late January and three clusters in February.



■ **Figure 4.28** Average RTT residual anomalies with filtered zero residuals

After further inspection of the sets of anomalies produces for the RTT and TTL sequences it was found that most (77%) of the traceroutes that were marked as anomalies by the RTT model were also marked as anomalies by the TTL model . This is not true for the opposite direction, probably mainly due to the large number of anomalies at the beginning of January for the TTL model. For a more detailed comparison, see Tables B.1,B.2 and B.3, which contain results for traceroutes, which have both anomalies, only RTT, or only TTL anomalies respectively.

Alternative aggregations, such as sum and weighted average with weight vector based on anomaly sequences and total number of anomalies were tried out; however, they did not seem to be useful in this case. Additional experiments with calculating weights vector based on the anomalies detected from each node, in order to limit the influence of constantly alerting models, were tested. Unfortunately, none of the tested ways of calculating the weights did not improve the result, and in addition the final interpretability was not as clear as the original one. Therefore, further research in this direction was abandoned.

4.5 Experiment: Network paths

As mentioned in Section 2.1, the traceroute paths are defined by a sequence of IP addresses, and after the post-processing step also by a sequence of AS numbers, which is used by the original alert algorithm to create a baseline path, which was explained in Section 2.4.

Modelling the probabilities of the paths is probably the closest thing to what the original alert algorithm was trying to achieve, that is, being able to determine whether a given path is problematic or not. The main weakness of the alert algorithm is that it considers only one path to be correct, while others are deemed problematic, which might be the correct assumption, since there can be multiple unique paths for communication between two fixed devices, and it would be beneficial to track all of them to get more information. For example, if a certain traceroute is indeed problematic, knowing whether or not it is common could change the reasoning about it. For a very unusual path, the reason might lie in the nature of the path itself and the real problem might be the reason why the traceroute did not travel in the usual way; on the other hand, if the path is common, it might indicate that the problem is indeed with some of the devices within it.

The task of modelling how each of the paths is likely to occur can be translated into modelling the probabilities of classes in categorical distribution, where categories are represented by the paths themselves. The paths themselves are disjoint and the only way the path A can only be part of the class B if it is its prefix, in which case the path A likely has failed and is incomplete;

however, they are still counted as two distinct paths.

To model the probabilities of the paths, a Dirichlet-Categorical model was implemented. The model uses Dirichlet's distribution as a prior distribution, which is used to count the occurrences α_i of each class i, where classes represent individual paths. These counts are then used to calculate the parameters of the Categorical distribution. In other words,

$$y|\theta \sim Categorical(\theta) \tag{4.9}$$

$$\theta|\alpha \sim Dirichlet(\alpha), \tag{4.10}$$

where p is the probability of observing the category i, given the set of parameters $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$, which was obtained using the number of occurrences of each class $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$, which can be again represented using the graph representation (see Figure 4.29), where the only variable observed is the occurrence of each class as shown in [39].

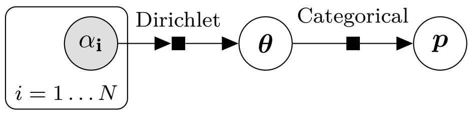


Figure 4.29 Graphical representation of the Dirichlet-Categorical model

This means that the probability estimates $p_{i,t}$, where the index i, t refers to the specific class i in time t, are obtained as

$$p_{i,t} = \frac{\alpha_{i,t}}{\sum_{j=1}^{N} \alpha_{j,t}},$$

which are then used for for estimating the final probability from the Categorical distribution using its probability mass function is obtained as

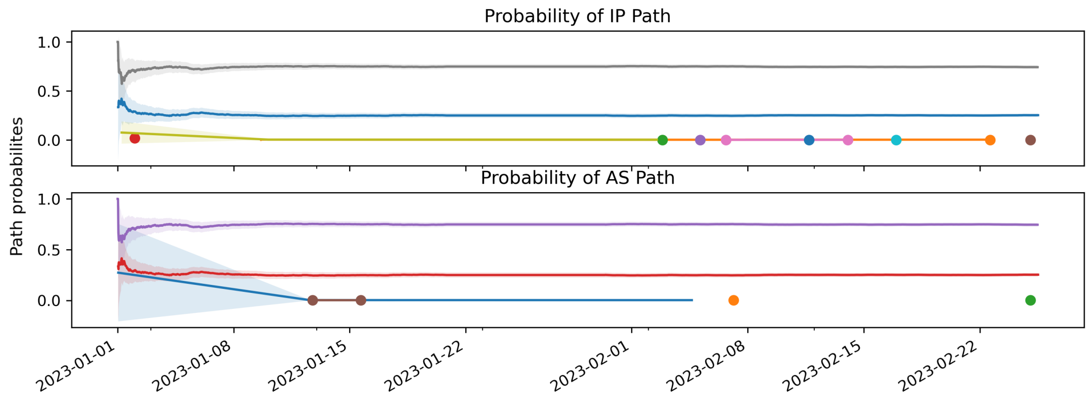
$$f(y_{i,t}|p) = \prod_{j=1}^{N} p_{j,t}^{\delta_{ij}}, \tag{4.11}$$

where δ_{ij} represents the Kronecker delta, which is defined by the following piecewise function

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases} \tag{4.12}$$

Since the sequences of IP addresses can be rather long, a decision was made to create hashes from the concatenated IP addresses using the SHA-1 hashing function. This was inspired by the path identification in the data set, which unfortunately cannot be used as it is computed from the AS numbers and some pre-processing is also performed. The first k characters of the hash are then used to represent each path. Another thing that was considered prior to running the experiment and that also affected the actual form of the paths was the need to avoid modelling the looping paths as unique classes, as there could potentially be many of them. This led to the decision to initially remove duplicates from each path before modelling them.

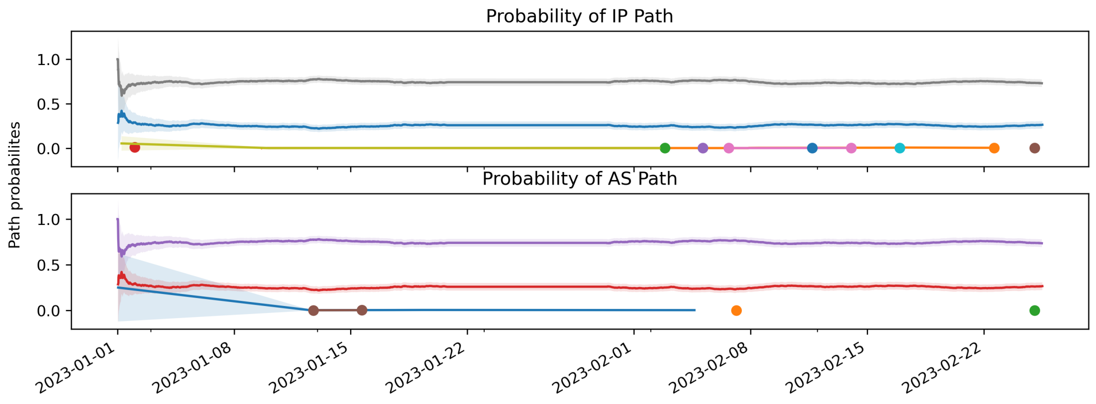
The initial results for IP and AS paths for communication between "2001:67c:1148:204::250" and "2001:620:808:4129::15" present in Figure 4.30 show that in both cases there were initially two paths with their probabilities close to 50%; however, after some time their probabilities started diverging from each other as one of them was more frequently occurring. In the graphs, there are also some paths with probability less than 5% for both types of paths, where some paths were observed less than three times, and since they would not be visible in the graph, each time they occurred is represented with a marker of the same colour. Since multiple devices can share the same AS number, some of these paths are not present in the second plot, as they merged with other paths.



■ **Figure 4.30** Estimated probabilities of observing specific IP and AS paths with two paths occurring in majority of traceroutes (coloured per unique path)

One thing that was noticed after inspecting the observed paths and their probabilities over time was that even after a longer period in which a certain path did not occur, the probabilities of the paths did not change much because the whole history was taken into account. This could be potentially an issue, for example, in cases when the new cable infrastructure is created which affects the shortest path from the source to destination, or, for example, switching to a different bandwidth, which can sometimes mean that the traffic has to go through a different set of routers. This could create a scenario where the model considers both the old and the new paths, where the old path has some decently large probability and the new one starts at zero and gradually increases, which itself is not the issue, but rather the fact that the old path will still maintain large probability value, even though it will probably never occur again.

In other words, if some of the events change the path itself, which might be considered as an anomaly, the model should not be giving that much importance to the whole history of the communication, but rather should be reflecting some smaller time frame to provide insights about the current state of the network. To solve this, the model was changed so that it considers only the k most recent paths. The value was then set to hold up to the last 1000 measurements, which should be roughly equal to 10 days of traffic, since the average number of records per day is roughly 100; however, the value is fully configurable for the needs of each site.



■ **Figure 4.31** Results showing influence of forgetting on the probability estimates of network paths (coloured per unique path)

The effect of forgetting is noticeable on the two paths with highest probabilities, as sometimes one is starting to occur more its probability is increasing, but the probability of the other one starts to decrease, which creates sort of wave pattern. This kind of equilibrium can give additional information about the paths in terms of the local trend.

4.5.1 Anomaly detection on network paths

To detect anomalies in the network paths, the objective was to base the approach on the distribution of the network paths at every point in time. This information would then be used to determine whether the path is an anomaly or not. After several experiments such as comparing the average probability with the probability of the observed path or comparing the rate of the probability of the observed variable with the maximum and minimum probabilities of the classes, a possible solution inspired by binary cross-entropy. Binary cross entropy defined as

$$H(p,q) = -\sum_{y} [p(y)\log(q(y)) + p(1-y)\log(q(1-y))] \tag{4.13}$$

is a measure of mutual information between two Bernoulli distributions p and q. On the basis of this equation, several more experiments were conducted in order to incorporate the logic of the cross entropy, one of which seemed to provide reasonable results.

The main idea behind this approach was to work with marginalised distributions for each of the paths, which, in fact, are Bernoulli distributions, since for the probability of an observed path p, the probabilities of the rest of the paths collapse to the complementary class, which represents the probability of not observing the path. For each marginalised distribution q, only the first term $(-q \cdot \log p)$ is calculated to act as a penalty that represents the degree of alignment of the marginal distributions. Based on the visualisation in Figure 4.32 it can be observed that when the probability of the observed path is low, the penalty has low values only if the probabilities of the other paths also have small probabilities, but as the probability of the rest of the paths increases, the penalty grows larger.

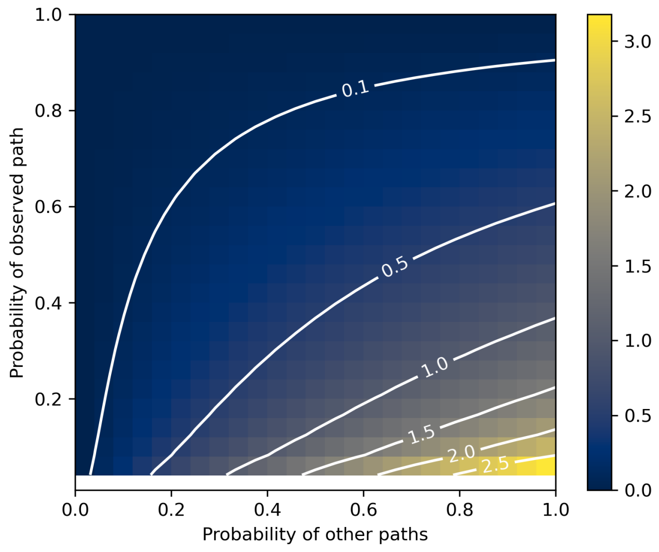


Figure 4.32 Visualisation of values of $-q \cdot \log p$ equation showing the effect on observed paths with low probability

The penalty scores between the probability of the observed path and the rest of the probabilities are then averaged, to obtain a single number that reflects the average relation between the probability of the observed path within the categorical model. This means that the final score depends on the entire categorical distribution. This means that the score of a path with probability 10% would be significantly based on the distribution. In Figure 4.33, three scenarios are shown. In the first one, the paths are uniformly distributed, and the penalty is low. In the second case, paths with various probabilities are present, but since there are two paths with probabilities greater than 30%, the final score is higher. In the third scenario, there are two paths which are dominant, complemented with several paths with lower occurrence. In this case, the score is even higher, which is caused by the two superstar paths in the distribution.

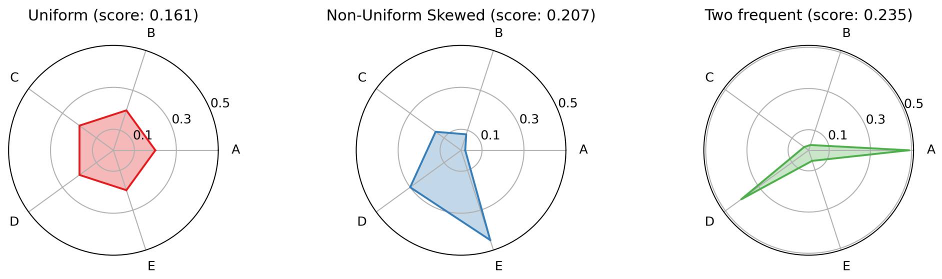
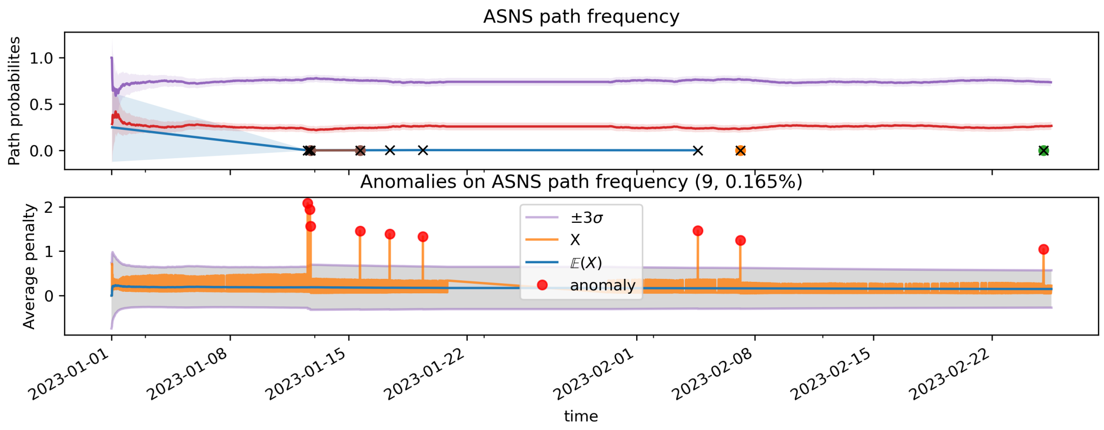


Figure 4.33 Average penalty for value 0.1 with differently distributed paths

Since the score itself can, and will differ for different sites, the score itself was then used as input to a Normal model, in order to detect the anomalies. The Normal model was set so that the prior hyper-parameter μ was set to 0, since in an ideal case there would be only one path, which means that the penalty would be 0; the hyper-parameter σ^2 was set to 1, which means that the standard Normal distribution was expected a priori. The goal of deploying a Normal model is that if the average penalty starts to diverge from 0, the model would be able to quickly adapt to the changes. The threshold interval for anomaly classification was set to $\mu \pm 3\sigma$.



■ **Figure 4.34** Anomalies detected using Normal model (bottom) and estimated path probabilities (top, coloured per unique path) with projected anomalies (cross marker)

The results of the previously discussed model that models the AS paths are shown in Figure 4.34. Using this method, the AS paths from nine traceroutes were marked as anomalies, which makes a lot of sense, as their probabilities were rather small. The expected score is approximately 0.15 with a high amount of variance, which is due to the frequent alternating between the two most common paths.

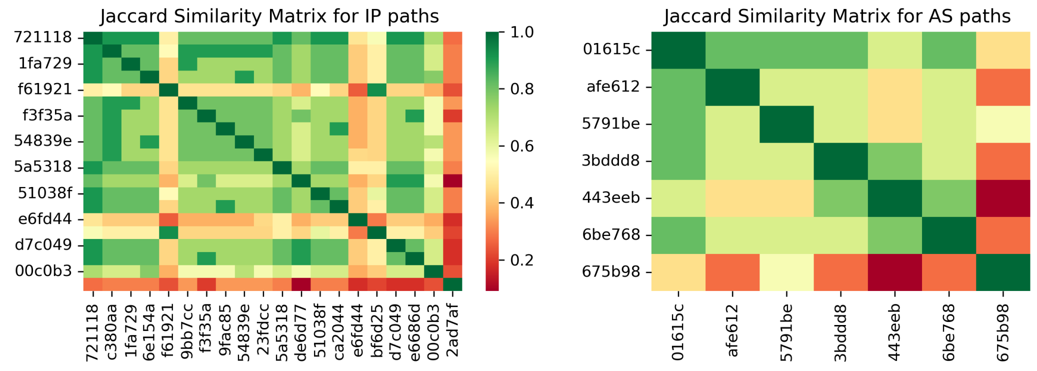
4.5.2 Problem of missing hops

After further examination of the paths modelled by the model, it was discovered that the only difference between the two paths is a device that is present as the device before the last on the path with the second highest probability and is missing in the path with the highest probability; also, some of the other paths marked as anomalies were also significant parts of the path similar. It was already discussed that paths are more common to be incomplete because some of the devices do not respond for whatever reason. For this communication there are only a few distinct paths, so there is not that much noise present, however, in a theoretical scenario, where each of the devices has a 50% probability of responding, for a complete path of length n there can be up to 2^n different paths observed. The number of these paths is expected to be significantly lower in practice, but there can still be scenarios where only a few of the devices do not respond, which the model could classify as anomalies.

In order to solve this, one had to solve how to determine whether or not there are two paths, actually one. After trying several metrics, a Jaccard similarity seemed to be the best choice. The Jaccard similarity is a metric, or similarity score, defined on two sets. The range of values ranges from 0 to 1 and is defined as follows:

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4.14}$$

where the A and B in this case would represents the sets of IP addresses, or equivalently AS numbers, of the two compared network paths. In this context, when the overlap between two sets is substantial compared to their combined size, the Jaccard similarity approaches 1, indicating a strong similarity between the sets. This concept can be extended to network paths. In situations where the source and destination are fixed, the positions of the intermediate devices are determined by their distance from the source. As a result, it is highly unlikely that a device x is observed both before and after another device y on the same path.



■ **Figure 4.35** Jaccard Similarity matrices for IP and AS paths showing similarities between different paths

Using precomputed similarities between all IP and AS paths observed by the previous model over time, based on the similarity matrices, which show how similar each pair of the paths is (see Figure 4.35), it was clear that most of the paths in both cases have quite high similarity score. There are four IP paths and two AS paths that deviate from this pattern.

Due to this, another experiment was done to see whether it is possible to utilise the Jaccard similarity score to find similar paths and model them as one in order to limit the amount of noise and simplify the task for the model. The first version of the experiment used a naive strategy, in which a global dictionary of known paths was created and with each new observation it was first checked if the path is already in the dictionary or not. If it was, it would be modelled as itself, and when it was not present, it was compared to the rest of known paths, and if any of them would have a similarity score greater than 0.7, it would be used instead of the original path. This approach was able to merge some of the similar paths; however, the result did not seem correct.

```
from typing import Dict

def get_closest(
        hash: str, # SHA-1 hash of path
        path: list[str | int], # path for reference
        threshold: float = 0.7,
        known_paths: dict[str, list[str|int]] # known complete paths
    ) -> str:
        if hash in known_pahts:
            return known_paths[hash]

        for other_hash, other_path in known_paths.items():
            if jaccard_similarity(path, other_path) > 0.7:
                return other_hash

        known_paths[hash] = path
        return hash
```

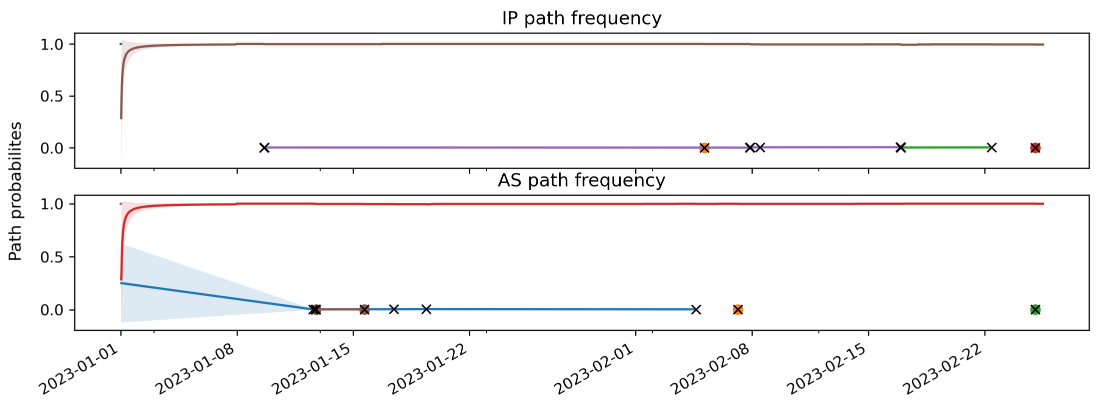
■ **Code listing 4.1** Finding closest path

After further investigation, an issue with this approach was found. This issue was caused by the fact that in the online modelling, there is no guarantee that while observing an incomplete path, a version of this path with fewer missing devices would be present. As it turned out, it could even happen that path with more nodes would be modelled only because the other one was seen first. The second version of this algorithm was created. The main difference is that in this version a path can be added to known paths only if the path itself is complete. In this way, incomplete paths are compared only to complete paths, and the whole self-destructive effect is eliminated. Additionally, the similarity threshold was made parametric for convenience.

```python
from typing import Dict

def get_closest(
        hash: str, # SHA-1 hash of path
        path: list[str | int], # path for reference
        path_complete: bool,
        threshold: float = 0.7,
        known_paths: dict[str, list[str|int]] # known paths
    ) -> str:
        if path_complete:
            if hash not in known_paths:
                known_paths[hash] = set(path)
        else:
            if known_paths:
                scores = [jaccard_similarity(path, known_paths[x])
                    for x in known_paths]
                max_idx = np.argmax(scores)
                if scores[max_idx] > threshold:
                    hash = list(known_paths.keys())[max_idx]

        return hash
```

■ **Code listing 4.2** Finding closest complete path

Using the Jaccard similarity threshold helped reduce the number of paths in the dataset for both IP and AS, which resulted in a single path being present most of the time for both types of paths. For the IP paths, the model marked 13 paths as anomalies, and for the AS paths it was only 9. After analysing the anomalous paths, in both cases, there were two paths which initially followed the most common path, but after a certain number of hops, the path diverged by communicating with previously unseen devices, and then the path returned to the expected path at some point. Since this behaviour obviously shows a weird and unusual pattern in terms of traversed nodes, the model result is probably correct here. Then another path marked as anomaly was a single traceroute, which had only two hops. This path is probably also an anomaly, but as it followed the expected path, it might also be possible to ignore these kind of paths and detect them for example using only based on the number of hops.



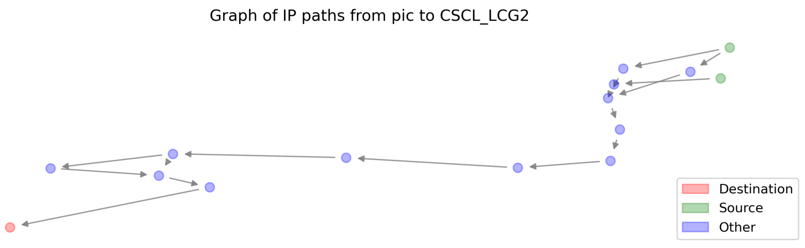
■ **Figure 4.36** Results for IP and AS paths with forgetting model and with similar paths merged (coloured per unique path)

Based on these findings, it seems that this approach might indeed be helpful in detecting anomalous paths from the traceroute data. However, an additional thing that should be mentioned is that the algorithm for finding the closest complete path is highly dependent on the fact that the complete path has already been observed, which might not always be true; thus improving this method by reducing false positives by checking whether a newly observed complete path is similar to any incomplete path might be step in a right direction.

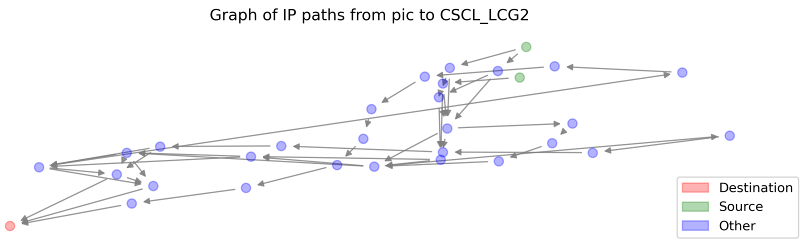
4.6 Experiment: Network path transition probability

The occurrence of network paths might be sufficient to track changes in for the fixed source and destination in the network; however, if a previously unseen path is observed, it might not directly mean that the path itself is wrong. For example, if the expected router was busy due to unusual traffic, the traffic may be routed to a second router, which can be exactly next to it, and then would get back regular path in order to keep the traffic flow stable. To model this, the experiment starts with an empty directed graph G, in which new vertices represent the routers present in the traceroutes, and the edges that connect them are added to the graph in an iterative way.

In Figure 4.37 are shown the initial and final states of the graph for communication for both IPv4 device pairs are shown. The first subplot shows the directed graph after 50 measurements. The graph contains two source nodes and one destination node, which is expected, and several intermediate devices. It seems that both source devices use a very similar path, but there is some variance in the first few hops. There are also a few segments in the graph where for vertices a, b, and c both paths $a \to c$ and also $a \to b \to c$ exist, which is most likely caused by missing hops in the traceroutes. The second subplot shows the final state of the graph after observing all of the traceroutes. There are the same number of source and destination devices, but significantly more intermediate devices that are densely connected, which makes the graph harder to read. This is probably due to the traceroutes frequently missing some hops. It also seems that there are multiple new paths from source to destination.



(a) After 50 traceroutes



(b) Whole dataset

Figure 4.37 Early and late form of transition graph inherited from data

The graph building process was then extended so that instead of learning only the representation of the network from traceroutes, it also accumulates the number of occurrences of each of the edges it observes. These occurrences were then converted to transition probabilities so that for each node u, the target nodes v_i connected to the directed edge $u \rightarrow v_i$ are considered. Given the similarities, the graph can be understood, in fact, as a large Markov chain built in an iterative way.

A Markov chain can be understood as a graph model in which each of the nodes represents an observable state and each directed edge $i \rightarrow j$ represents the probability of the observed variable to transition from state i to state j.

Based on this, the properties of the Markov chain are used with the learnt representation to use the properties of the Markov chain. The most important property is that the Markov chain is memoryless, which is also referred to as the Markovian property. This means that the probability of any future transition does not depend on the previous transitions. In other words, instead of computing the conditional probability of each previous node, the transition probability

$$P(Y_n = y_n | Y_{n-1} = y_{n-1}, Y_{n-2} = y_{n-2} \cdots Y_1 = y_1) \tag{4.15}$$

can be simplified as

$$P(Y_n = y_n | Y_{n-1} = y_{n-1}). \tag{4.16}$$

Thanks to treating the graph as a Markov chain, it is also possible to obtain the probability of transitioning through the whole sequence of routers Y, which is calculated as the product of the transition probabilities of each oriented edge $u \rightarrow v$ from the path as

$$P(Y) = \prod_{u,v \in Y} P(V = v | U = u). \tag{4.17}$$

Based on [40], the Dirichlet distribution can also be used as a prior distribution to model the probabilities based on the counts for each node in the Markov chain, meaning that

$$Y|P \sim MarkovChain(P)$$
$$P|\alpha \sim Dirichlet(\alpha).$$

Using this approach, the probabilities obtained for each pair of devices are shown in Figure 4.38, where each pair of devices is represented only with the source device. For each pair of devices, there are multiple "common" values that tend to repeat over time. For the traceroutes from device with address "193.109.172.242", the probability of the paths is mostly around 0.4, with some paths where the probability is very low, but they stop occurring at the end of January. For traceroutes from "193.109.172.250", most of the measurements have a probability of more than 0.8, which seems to be slowly converging to 1. For IPv6 traffic, the probabilities there seem to be some sort of symmetry with centre at 0.5, but for most of the paths, the probability converges to around 0.7 after a learning period. In all cases, the paths with lower probability were occurring with a significantly lower rate, which can be seen thanks to the transparency level of the markers in the plot.

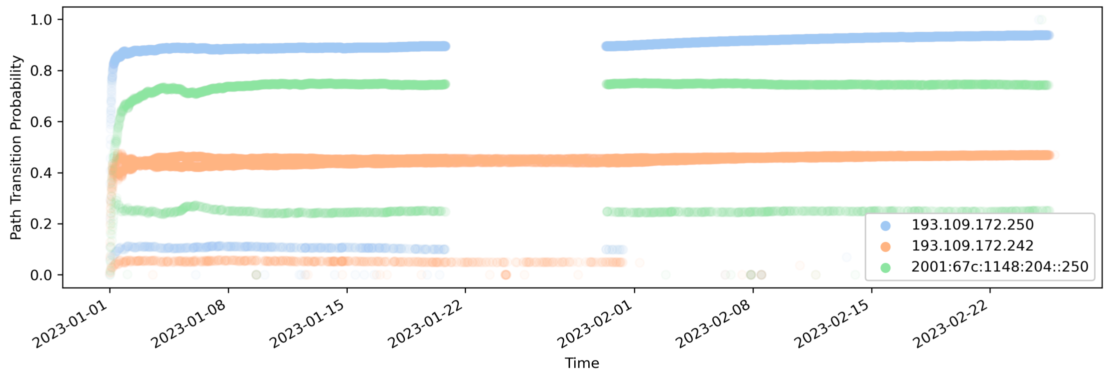


Figure 4.38 Markovian path probability using the estimated transition probabilities

It was suspected that this might be caused by the fact that the model reflects the whole history, which can contain transitions that occurred only few times, but the model still considers them as relevant. Due to this, the underlying Dirichlet model was altered to consider only the n most recent hops from the node for which it models the probabilities. This was also done to allow the model to be re-fitted more easily if more significant changes to routing are done, similarly as in the previous experiment. In this case, the memory size of each model was set to 1000 observations.

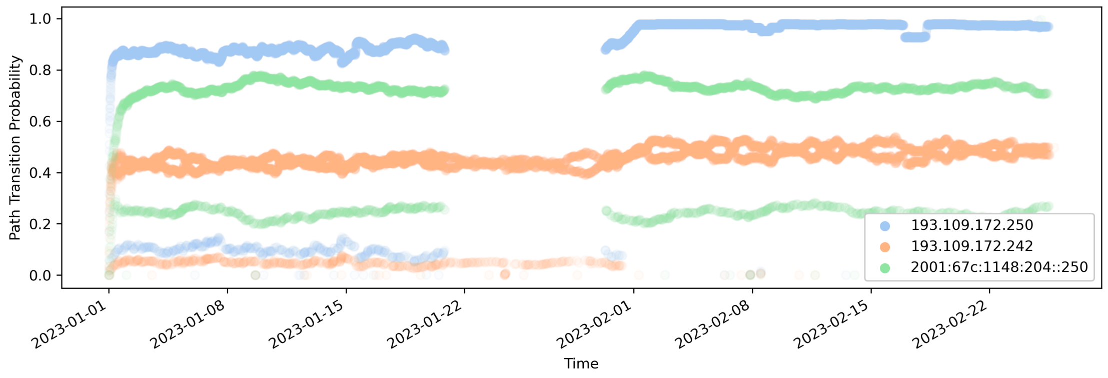


Figure 4.39 Markovian path probability using the estimated transition probabilities with forgetting

The result of this (see Figure 4.39) was that the probabilities of the paths over time were less stable because the probabilities were changing frequently. After further investigation, it was clear that the variance in probabilities between the same source and destination device was also caused by missing hops, which in this context means that instead of observing a subpath $a \to b \to c$, the model sometimes encounters a subpath $a \to c$. This causes the model to assign a low probability value, which after multiplying with the rest of the probabilities can have a destructive influence on the final value, which may or may not be an issue.

4.6.1 Anomaly detection

The detection of anomalies in this case is slightly complicated, since the output variable of the model represents the probability of observing the sequence, which ranges between 0 and 1. When the path has a high probability, it means that it is frequent and is likely good, but when the probability is low, it might indicate that there is an issue. To separate the paths with missing hops from paths that are indeed problematic, few changes were made.

After an investigation, it was determined that the main cause of the effect shown in Figure 4.39 was caused by traceroutes with segments with relatively small but still significant transition probability. For a large enough number of these transitions, the final probability score can quite fast reach the value of 0.

To solve the problem caused by multiplying relatively small probabilities, they were instead scaled using a logarithmic function, which transforms high probabilities into small negative values and low probabilities into large negative numbers. This was done primarily to help with numerical stability. For convenience, the absolute value of scaled transition probabilities was used in this way.

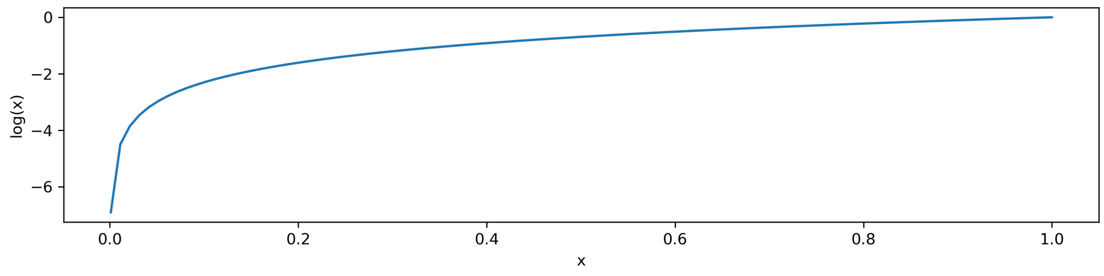


Figure 4.40 Influence of logarithmic transformation on values between 0 and 1

The transformed transition probabilities for each path were then averaged. The result of this manipulation is shown in Figure 4.41, from which it is noticeable that in most cases the value of the path is less than four, but there are multiple cases when the value is around 7.5 or more, which means that the probabilities of some transitions on these paths were very small and were able to outweigh the transitions with normal probability values.

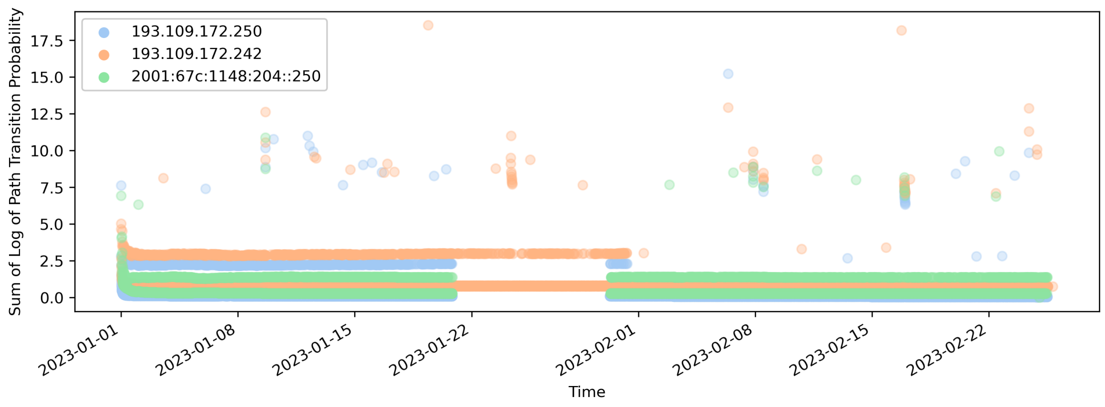
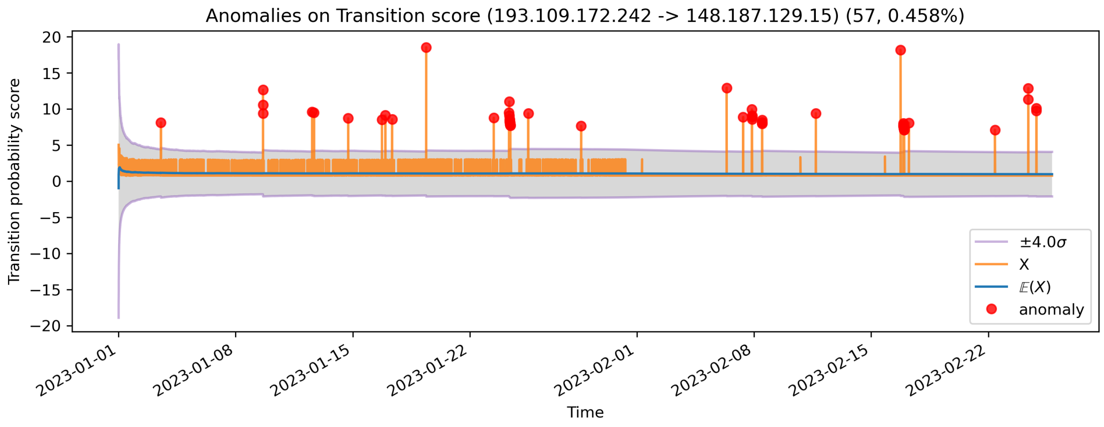


Figure 4.41 Average of log-transformed transition probabilities

Although the transition probabilities would be collected for all observed devices, the anomalies should probably still be classified on the device-to-device level. Thus, a Normal model was used to model this way transformed transition probabilities for each pair of devices. The main reason for this was that the Normal distribution is commonly used to approximate data from unknown distributions while still performing relatively well.



■ **Figure 4.42** Anomalies on transition probabilities

Given the result of the Normal model shown in Figure 4.42, it seems that in combination with the transformation the model classified 57 paths as anomalies, which was the highest between the three unique device pairs. The plots for the other two pairs of devices are shown in Figures B.1, and B.2. After further investigation of the traces classified as anomalies, in roughly half of the cases, the paths marked as anomalies contained devices common for the communication between source and destination, but it was infrequent to observe them in a sequence. The rest of the anomalies, on the other hand, contained 4 unexpected devices in 26 cases, 3 devices in 4 cases, and 8 devices in a single case.

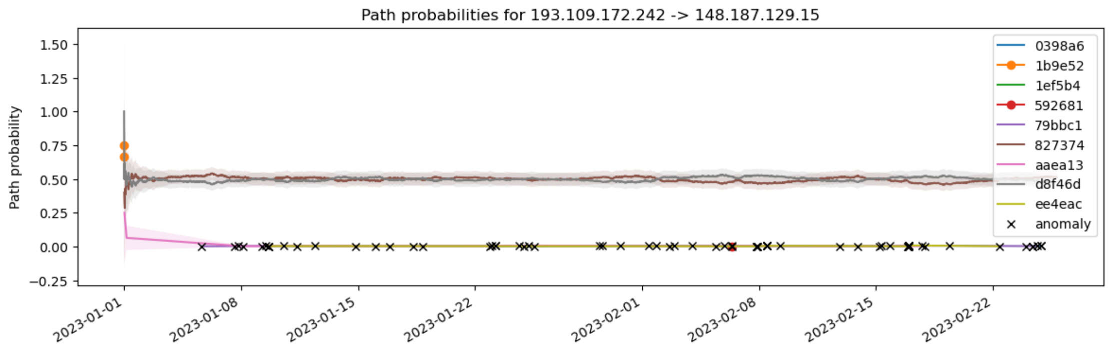
4.6.2 Path and transition model comparison

In order to compare the results of the network path model and the network transition model, traffic between the communication of the three pairs of devices was evaluated using both models on both the IP and AS paths. Based on the results shown in Table 4.1, the anomalies of the transition model seem to be consistent for both types of network paths, with slightly higher count for AS paths. For the model from the previous experiment, the number of anomalies on the AS paths was almost halved, but at the same time for the IP paths the number was more than three times higher compared to the transition model. It may seem that the transition model might perform better with the IP paths, but when the outputs of the models were compared to each other, it was clear that the anomalies of both models were vastly different and only 9 of them for IP paths, and 15 for the AS paths were reported by both models.

	IP paths	AS paths
Transition model	42	48
Path model	136	24
Shared	9	15

■ **Table 4.1** Comparison of IP and AS models

The higher number of anomalies reported by the model from the previous experiment were due to one device pair having two paths with probability close to 50% and several paths with low probability, which have been marked as anomalies (see Figure 4.43). For the other two pairs of devices, there was one path with high probability and few paths that were classified as anomalies.



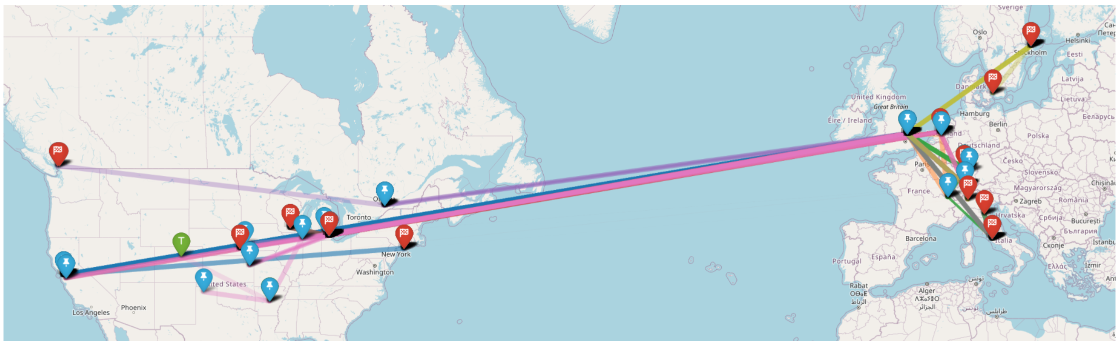
■ **Figure 4.43** Anomalies detected on device pair using IPv4 communication

Additionally, the learnt graph of transition probabilities could be a subject of further analysis, as it yields valuable information. It can be used to analyse the relationships of the devices, their similarity, or importance in the network.

4.6.3 GPS-Based Location Tracking

To further extend the capabilities of a learnt network topology used for data transfer, an idea was to determine the longitude and latitude of each device using a third-party service. The aim was to calculate the distance between each hop. Knowing the distance between each pair of consecutive devices, for example, could enable an analysis of how latency correlates with the physical length of the path.

To obtain the physical location of each router, free public services that translate the IP address to GPS coordinates were evaluated. To find the one that performed the best, the services were compared based on the precision of the location assigned to a subset of the source sites for which an approximate location was found using the name of the institutions. Based on these findings, the Ipstack [41] service was chosen, as it appeared to have the most accurate results based on the accuracy of the source and destination devices, for which the real location could be approximated thanks to the domain names of scientific organisations present in the data set.



■ **Figure 4.44** Screenshot of an interactive map displaying devices from different site paths

This allowed for the creation of an interactive visualisation (see Figure 4.44) using Leaflet [42], which is a free open map visualisation tool. However, after further analysis of the position of the nodes, it was clear that even this geolocation service was not correct every time, as some of the routes stretched as some devices were placed in seemingly random locations, for example, a path from CERN to New York through London, which might be possible, but then through Berkeley, which is on the other coast of the United States.

After further investigation, it was found that this is due to the devices having assigned the location of its AS provider headquarters or the headquarters of the scientific institute, which unfortunately would distort any further analysis of this kind of data, but if there were a way to more accurately pinpoint the location of these devices, there is definitely potential for research in this direction.

Although this particular experiment did not yield any important results, pairing the devices with their geo-location definitely has the potential to uncover interesting information about the network, so if reliable service that would be able to provide the correct location of the machine would be a valuable source of information. It could, for example, help answer questions about latency between countries, help calculate the distance of the routers, thus the minimal physical distance the data have to travel, and many more.

4.6.4 Mathis equation

Another experiment was done using Mathis equation [43] which is defined as the following inequality

$$\text{Throughput} \leq \frac{\text{MSS}}{\text{RTT} \cdot \sqrt{p}}, \tag{4.18}$$

which suggests that it's possible to approximate the maximum throughput based on RTT, maximum segment size, and packet loss p. This was done to verify whether it is possible using the equation to estimate the throughput between devices.

Since packet loss is not present in the dataset, it was substituted with probability of observing a device in communication between the source and destination.

However, this approach did not seem to provide reasonable output, probably because of this substitution for packet loss.

4.7 Local and global aggregation

Given the large number of sites where each of them communicates through possibly multiple devices, modelling the device-to-device communication might be helpful, but it would be impractical for a network administrator to manually check each of them regularly. Due to this, there is a need to somehow aggregate the findings from the device level to both the site and global levels.

Initially, the process of aggregating the information on the anomalies detected in device-to-device communication was based on the Poisson model. For each traceroute from the site-to-site level, the corresponding parts of the traceroute were evaluated by the models described in previous sections, and the total number of anomalies from each of the models was used as input to the Poisson model.

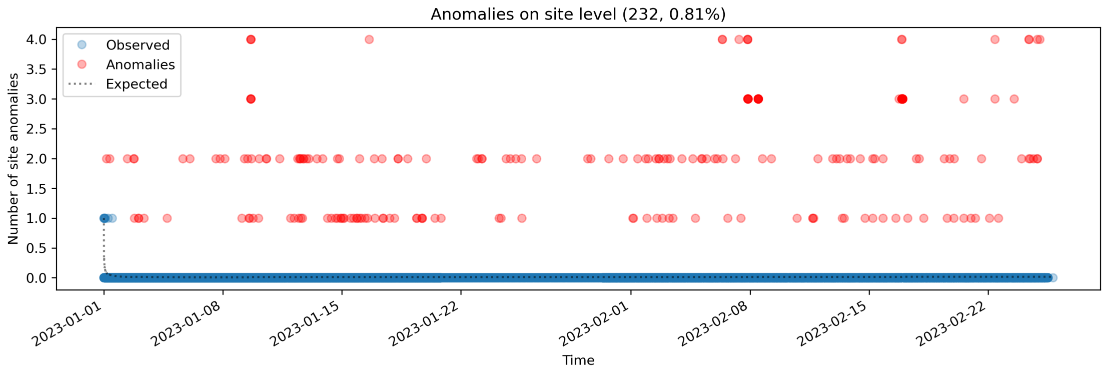


Figure 4.45 Results of Poisson model for site anomalies

The goal of this approach was to mitigate the potential over-reporting of anomalies, which could have been caused, for example, by an imperfect scoring function of some model. After evaluating the entire data set, based on the result of the Poisson model shown in Figure 4.45 it is clear that since most traceroutes were not classified as anomalous by any of the underlying models, as the Poisson model quickly learnt that any traceroute with even a single anomaly is essentially wrong. Using this approach would essentially mean that the number of underlying models classifying the traceroute as anomaly is irrelevant because any number of anomalies on this level would be promoted as uncommon and in addition each such combination would have the same weight, which probably does not make sense.

After trying several different approaches, where none of them provided reasonable results, a decision was made to simply aggregate the number of anomalies in a specific time interval. This allows one to calculate an aggregate value for each time period, which can be the subject of further analysis. The total number of anomalies in each model was calculated in 8-hour intervals, which were then plotted on a stacked bar graph. This visualisation allows us to show not only the total number of anomalies in a given time period but also the proportionality of anomalies from each model in that period. The bar plot in Figure 4.46 contains several periods with relatively

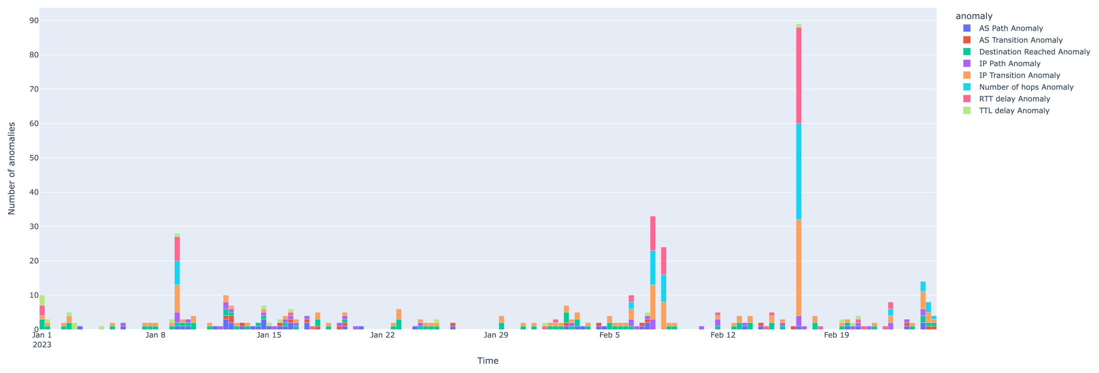


Figure 4.46 Results of Poisson model for site anomalies

small number of anomalies. There are a larger number of anomalies in the beginning, which is caused by the underlying models not being sufficiently trained yet. There is a higher peak of anomalies that occurred on 9 January between 8 AM and 16 PM, where several models classified the traceroutes as anomalies. The period contains anomalies for the AS and IP routes, delay in both TTL and RTT, complemented by the number of hops and destination reached anomalies, which could mean that for some reason the traffic was routed differently than usual, which could have made the route take longer and perhaps also affect the reach of the destination. There is also a cluster of anomalies for the AS path from January 13 to January 17, which could indicate a sudden change in the AS paths. The number of anomalies is then usually less than 10, until February 6, 7, and 8, where the number of anomalies has suddenly risen to 10 anomalies and more. The composition in all three periods is the same, IP transition probability anomalies paired with the number of hop anomalies and abnormal RTT delays, which could mean that the traceroutes in these periods transitioned through uncommon devices, which affected the duration and length of the traceroute. This is then repeated on February 16, but the period contains a significantly larger number of anomalies.

Using this approach, the user can identify problematic periods in the data, which can then be further investigated on the level of models, or the raw data itself. It can also be compared with results observed from different kinds of data in order to validate the results. This approach may not be revolutionary, but creating the visualisation does not require an additional model, which removes the need of interoperability of such a model, which might be complicated. However, the final aggregation may be a subject of further research.

4.8 Evaluation on Multi-Site Communication

After the prototyping period, another experiment was carried out, whose purpose was to observe how the models perform in multi-site communication, since, up to this point, only one pair of sites had been used to develop the models and validate the results of the whole pipeline. This dataset contains communication from three sites, each acting as a source and destination site in communication with the other two, meaning there are nine unique site-to-site communications. The chosen sites were ”FZK-LCG2” in Germany, ”INFN-T1” in Italy, and ”SARA-MATRIX” in t he Netherlands. For easier analysis of the results, an interactive dashboard application was built using Streamlit [44].

The application shows the global state of anomalies using the previously described bar graph, complemented with the geovisualisation of the devices occurring in the traceroutes at the global, site, and device levels to locate the exact point of the problems in the network. Screenshots of the dashboard are shown in Figures C.1 to C.11.

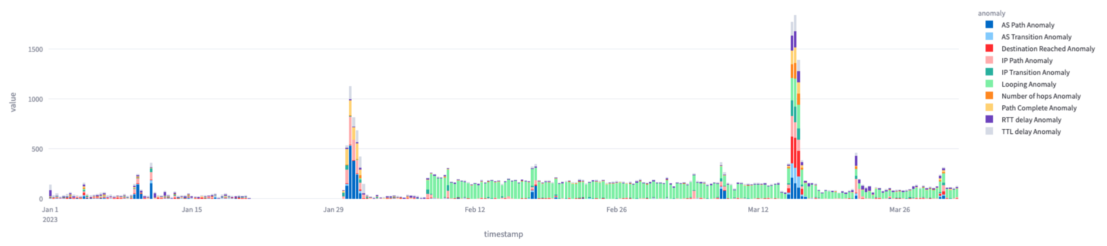


Figure 4.47 8-hour aggregation of all anomalies with two extreme peaks of anomalies showing 2 major peaks in number of anomalies detected

Figure 4.47 shows the aggregated anomalies at the global level. At the beginning of the year, the number of anomalies was approximately 50, but there were peaks on January 1, 4, 9, and 11, with the number of anomalies reaching up to 350 in a single period. In the second half of January, there were no anomalies, which is probably due to the decreased number of measurements. At the beginning of February, there was a large peak of anomalies with a significant number of IP and AS path anomalies, destination-attended anomalies, and TTL delays, which could mean that the traces observed during this period were uncommon and resulted in a change in reaching the destination. This may suggest that the traceroutes that previously did not reach their destination suddenly started to do so, but only for this brief period. This could also be caused by maintenance on the network or datacenter, which would explain the gap in the data, and after the maintenance period the paths would change, which would be interpreted by the model as an anomaly. After a small period with a relatively small number of anomalies, there was a huge increase in looping traceroutes, which, after further investigation, was found to be present only in traceroutes from ”INFN-T1” to ”SARA-MATRIX”. There were also two smaller peaks on February 17 and March 8, where in both cases anomalies were detected with respect to the usual path. After that, from March 15 to March 17, there was a huge peak in the number of anomalies. It practically contained all types of anomalies in large numbers, and it definitely represents some significant issue in the network. Then two more peaks occurred, one on March 21, where multiple RTT delays and IP path anomalies were detected, and on March 30, where the number of anomalies was lower, but with the same type of anomalies. This way, found anomalous events can then be compared with results from other methods or further investigated, for example, by using a smaller aggregation period, which would allow one to pinpoint a more precise period with detected issues, to filter out potentially bad traceroutes, and analyse them separately. However, as some types of anomalies might be less significant compared to others, it would make sense to enable weighting the models, and perhaps also enable different aggregations, such as average over the period.

An alternative approach might be to use the results of these models as input into another classifier, which would then classify the severity of the network problems in the given time period. However, this would require additional research, the creation of a labelled data set based on historical events, or a network expert to first analyse the results of these models.

4.9 Remarks on Implementation

All experiments were written using the Python[45] programming language. Python is widely used in the world of ML algorithms due to its ease of adoption and prototyping capabilities. More importantly, it is also used in the codebase of the alerting algorithm. The models, utilities, and the rest of the processing pipeline were converted into a Python module, which should allow for easy incorporation of the processing pipeline into the entire architecture. All mathematical operations, excluding basic arithmetic, were carried out using Numpy[46], an optimised mathematical module for Python.

All models support the export of data, meaning the inputs, outputs, and statistics learnt over time from the data, by creating Pandas[47] DataFrame. This can be used for further analysis or serialisation of the data in various formats. Each of the models also supports the creation of visualisations of the learning process using the Matplotlib[48] plotting module for Python and for interactive graphs Plotly[49], to easily visualise the results. To simplify working with graph structures, Networkx[50] functions and data structures were used, complemented by Netgraph[51] for better layout and visualisation of graphs.

Initially, the processing time for the data used in the multi-site experiment was almost 30 min. This was mainly due to the storage of too many values for each new traceroute and the fact that all models had their own copy of the time index. After pruning the values and making each group of models in the hierarchy use a shared index, the computation time was reduced to only 8 minutes, achieving a processing rate of almost 1250 traceroutes per second, which could be optimised to be more efficient.

It is fair to mention that, in its current state, the processing pipeline in the module can only process data on the local machine. To enable a full streaming mode, direct communication with the database, which would allow for direct reading and writing of data, is necessary and would need to be implemented.

Chapter 5
Conclusion

The goal of this work was to explore potential methods for detecting anomalies in the CERN traceroute dataset, with the goal of modifying or replacing the current heuristic algorithm.

Based on the prior analysis of the data, its volume, the existing heuristic algorithm, and the alerts generated by it, Bayesian inference was selected as an alternative approach. This method is expected to model complex relationships at the device level, even for network traffic on a global scale. Bayesian inference also requires minimal computational resources and, in the setup demonstrated in this work, proves to be time-efficient. Thanks to the statistical distributions Bayesian inference could be considered a naturally explainable AI algorithm, which is particularly useful in identifying the sources of network problems.

Several experiments, modelling aspects such as delay in RTT or TTL, path length, probability of observing a path, or probability of observing a sequence of device transitions, were conducted.

The findings suggest that models based on Bayesian inference can effectively model the traceroute data and identify anomalies in a more detailed manner compared to the heuristic algorithm. While the heuristic algorithm operates on a daily basis, the models can identify anomalies almost in real time (i.e., in a streaming/online approach), a significant strength of the Bayesian inference approach that could enable network administrators to respond almost immediately to detected anomalies.

Another advantage of these models is that Bayesian inference allows for the modelling of each device's communication separately, enabling network administrators to inspect results at the device level without information loss. Additionally, the results of these experiments could be used as input for training another model, to filter non-problematic traceroute measurements, or as labels for training a different model.

However, a potential limitation is that all models are currently assigned equal importance, which may not accurately reflect the real-world setting. Allowing network administrators to assign varying levels of importance to different models could be beneficial, reflecting the severity of the problems more accurately.

To fully validate the results of these experiments, creating a dataset with all known occurrences of network-related problems in the network and compare it with the experimental results.

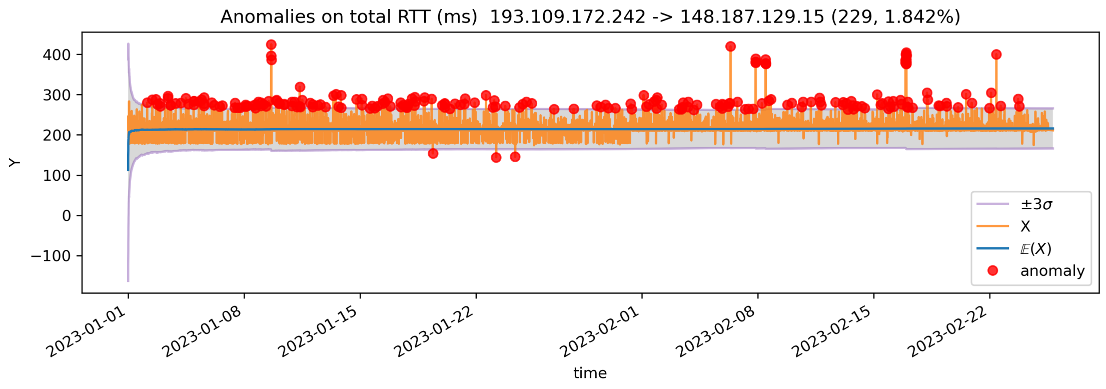
5.1 Future Works

The current implementation of the models presented in this work does not support a full streaming approach, which would include storing metadata and intermediate results in ElasticSearch. This aspect will be the focus of future work along with the productionalisation of these models. Additionally, further experiments are planned to automatically alert major events on the network using the final aggregations.

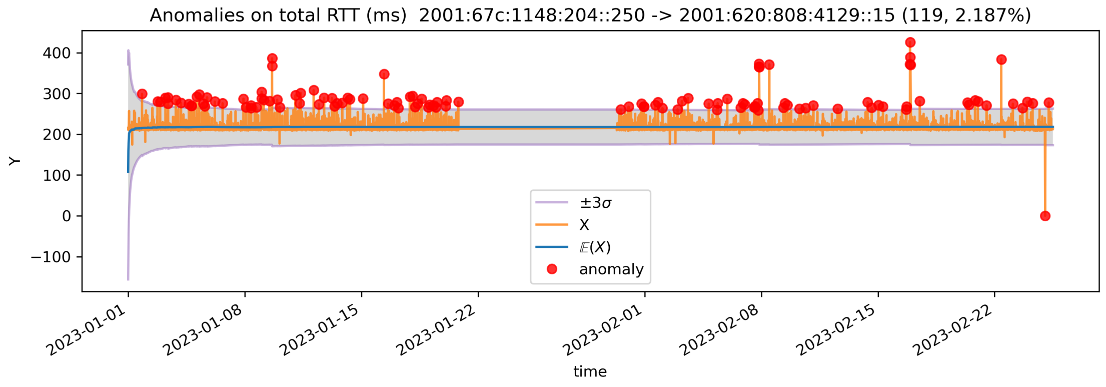
Another planned improvement is to add functionality that allows network administrators to preview the original traceroute measurement marked as an anomaly, to validate the model outputs.

As this approach generates additional data that are descriptive of the traffic between two nodes, another experiment could analyse the possibility of classifying or clustering nodes (similar to the method mentioned in Section 2.2.2) based on these data, which could help detect problematic devices.

Appendix A
Appendix



■ **Figure A.1** Anomalies on total RTT for an IPv6 device-to-device communication



■ **Figure A.2** Anomalies on total RTT for an IPv4 device-to-device communication

Appendix B

Results for RTT and TTL delays

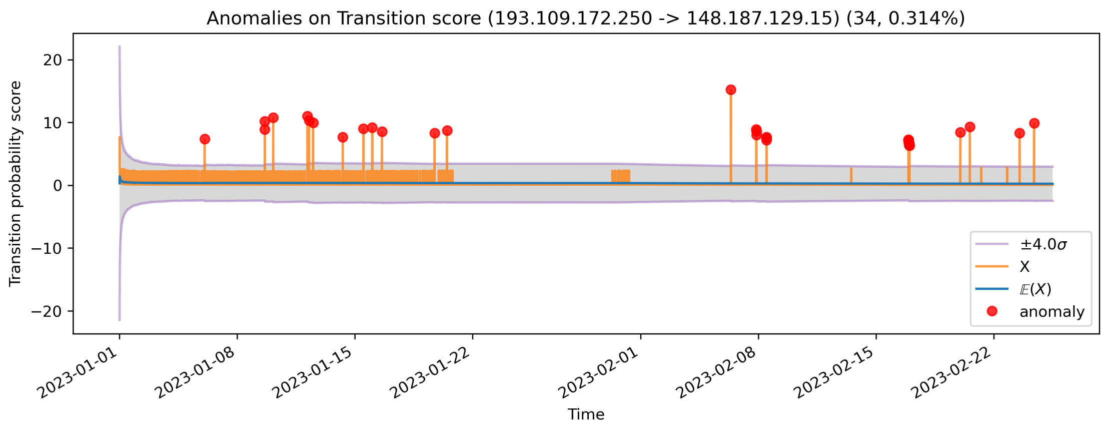


Figure B.1 Anomalies on transition score device, device pair 2

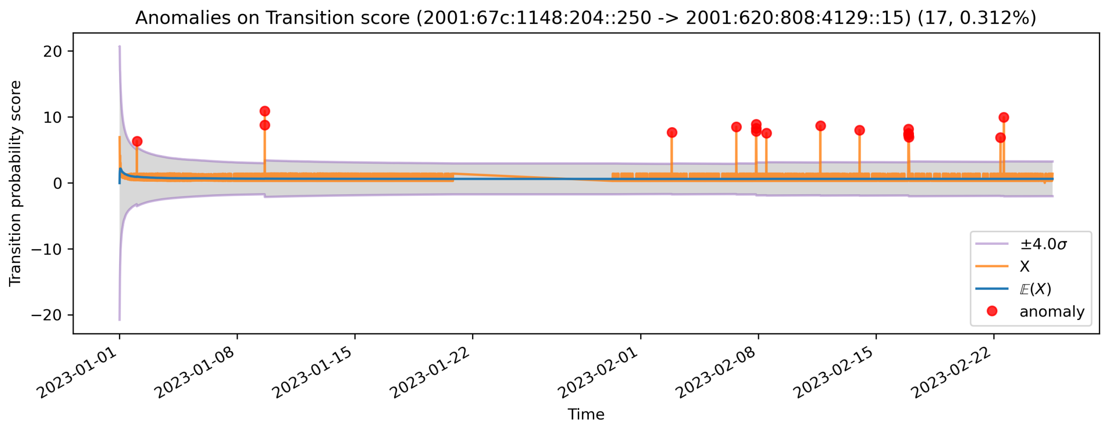


Figure B.2 Anomalies on transition score, device pair 3

Timestamp	RTT			TTL		
	Agg.	Expected	Total Resid.	Agg.	Expected	Total Resid.
02-06 08:37:37	15.56	0.46	264.57	6.53	0.00	6.53
02-07 20:45:14	13.28	0.44	185.85	2.88	0.00	11.53
02-07 20:56:54	13.21	0.44	184.96	2.88	0.00	11.53
02-07 20:58:58	16.31	0.44	228.29	3.38	0.00	13.52
02-08 11:25:54	12.58	0.44	176.11	3.13	0.00	12.52
02-08 11:26:22	12.24	0.44	171.35	2.88	0.00	11.52
02-08 11:36:59	12.45	0.44	174.32	3.13	0.00	12.52
02-16 22:06:38	11.87	0.39	166.22	3.13	0.01	12.50
02-16 22:06:44	11.67	0.39	163.43	3.38	0.01	13.50
02-16 22:15:19	12.87	0.39	180.16	2.88	0.01	11.50
02-16 22:28:46	11.78	0.39	164.87	3.13	0.01	12.50
02-16 22:37:06	11.08	0.39	155.18	2.87	0.01	11.50
02-16 22:46:15	10.77	0.40	150.72	3.37	0.01	13.50
02-16 22:55:10	11.63	0.40	162.81	2.87	0.01	11.50
02-16 23:09:53	13.25	0.40	185.45	2.87	0.01	11.50
02-16 23:19:22	10.21	0.40	142.89	3.12	0.01	12.49
02-16 23:29:35	10.59	0.40	148.31	2.87	0.01	11.49

Table B.1 Results for traceroutes with both types of anomalies from Trace Model

Timestamp	RTT			TTL		
	Agg.	Expected	Total Resid.	Agg.	Expected	Total Resid.
01-01 00:05:28	21.39	10.70	213.90	0.00	0.00	0.00
01-09 15:18:49	18.48	1.12	240.23	2.85	-0.01	8.54
01-09 15:25:05	22.03	1.13	286.44	2.85	-0.01	8.54
02-14 18:19:46	9.37	0.40	103.03	-0.12	0.01	-0.49
02-20 23:36:14	6.96	0.38	76.51	0.37	0.01	1.49

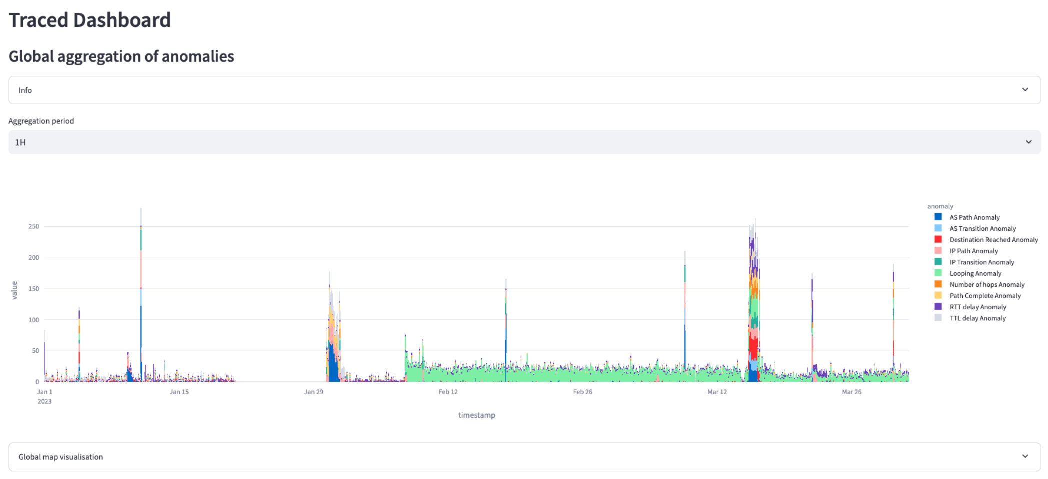
Table B.2 Results for traceroutes with only RTT anomalies from Trace Model

Timestamp	RTT			TTL		
	Agg.	**Expected**	**Total Resid.**	**Agg.**	**Expected**	**Total Resid.**
01-01 01:15:03	12.96	14.96	142.56	2.53	0.36	2.53
01-01 01:55:21	10.73	14.25	118.05	3.52	0.27	3.52
01-01 08:47:20	7.55	8.46	83.00	3.57	0.06	3.57
01-01 13:56:26	2.44	6.73	26.82	3.57	0.04	3.57
01-01 17:38:55	2.18	5.89	23.96	4.57	0.03	4.57
01-01 23:19:10	2.32	5.05	25.55	5.57	0.03	5.57
01-02 12:45:30	1.22	3.84	13.42	4.53	0.05	4.53
01-02 22:36:07	6.13	3.32	67.43	4.51	0.06	4.51
01-03 00:48:20	0.40	3.22	4.00	7.49	0.08	7.49
01-03 05:16:40	0.52	3.01	5.75	5.49	0.07	5.49
01-03 06:07:26	0.41	2.98	4.56	5.49	0.07	5.49
01-03 07:35:05	0.56	2.92	6.14	4.48	0.08	4.48
01-03 11:16:38	0.32	2.78	3.50	5.48	0.07	5.48
01-03 18:08:41	0.38	2.58	4.19	4.50	0.04	4.50
01-04 00:05:18	1.06	2.42	11.68	5.51	0.04	5.51
01-04 01:35:23	-0.03	2.39	-0.31	5.50	0.04	5.50
01-04 21:09:40	-0.18	2.03	-1.98	4.52	0.02	4.52
01-05 03:37:25	0.51	1.94	5.63	5.52	0.02	5.52
01-05 04:27:17	1.09	1.93	12.01	4.52	0.02	4.52
01-06 05:08:18	0.51	1.62	5.62	4.52	0.01	4.52
01-06 12:09:39	0.83	1.54	8.31	5.52	0.01	5.52
01-07 03:56:11	0.05	1.42	0.59	4.53	0.00	4.53
01-07 04:38:10	0.51	1.41	5.57	4.53	0.01	4.53
01-07 13:15:28	-0.04	1.35	-0.48	4.53	0.00	4.53
01-08 07:48:02	-0.06	1.24	-0.66	6.53	-0.00	6.53
01-08 13:25:47	-0.02	1.21	-0.19	4.53	-0.00	4.53
01-08 20:39:39	1.37	1.18	15.02	5.53	-0.00	5.53
01-08 22:59:05	-0.17	1.17	-1.92	4.53	-0.00	4.53
01-09 05:18:11	1.08	1.16	11.84	4.53	-0.00	4.53
01-31 11:05:27	-0.44	0.55	-4.87	2.85	-0.00	8.54

■ **Table B.3** Results for traceroute with only TTL anomalies from Trace Model

Appendix C

Streamlit dashboard application



Traced Dashboard

Global aggregation of anomalies

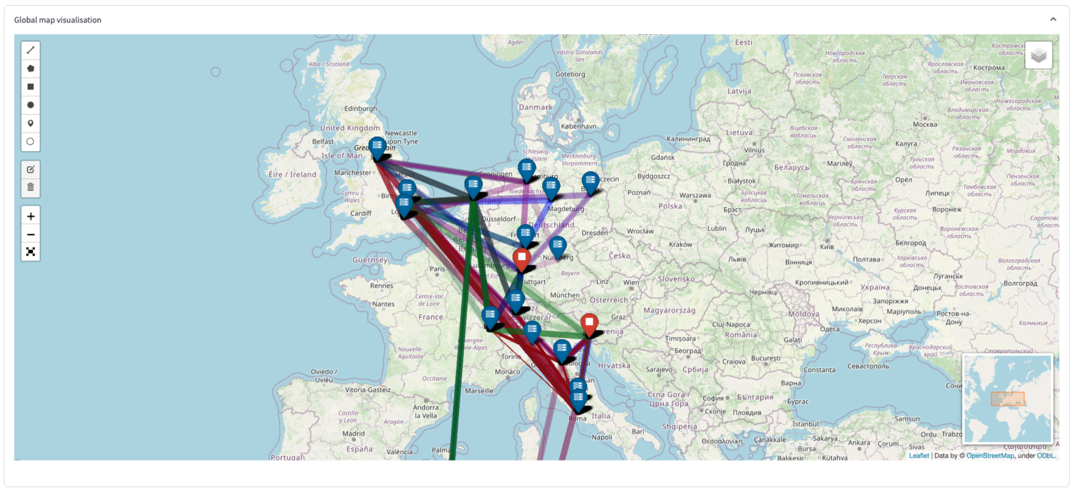
Info

Aggregation period

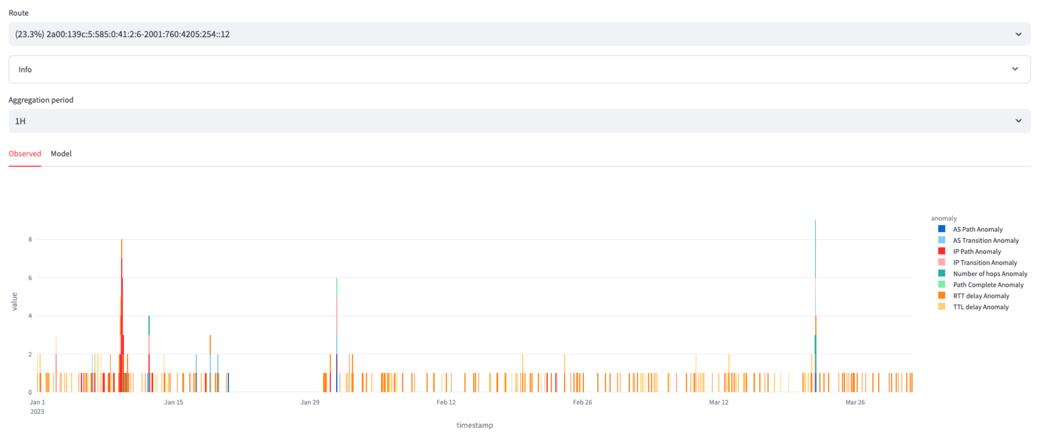
1H

Global map visualisation

Figure C.1 Screenshot of global aggregation in dashboard application



■ **Figure C.2** Screenshot of geo visualisation in dashboard application


Route

(23.3%) 2a00:139c:5:585:0:41:2:6-2001:760:4205:254::12

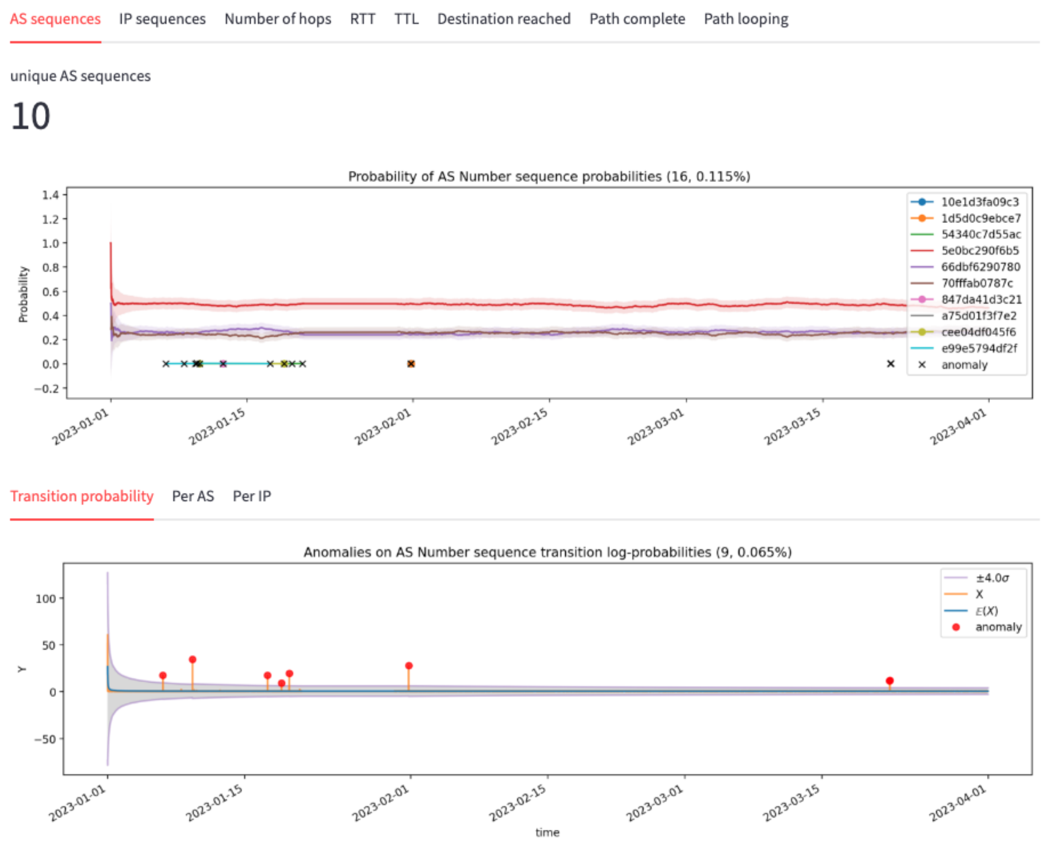
Info

Aggregation period

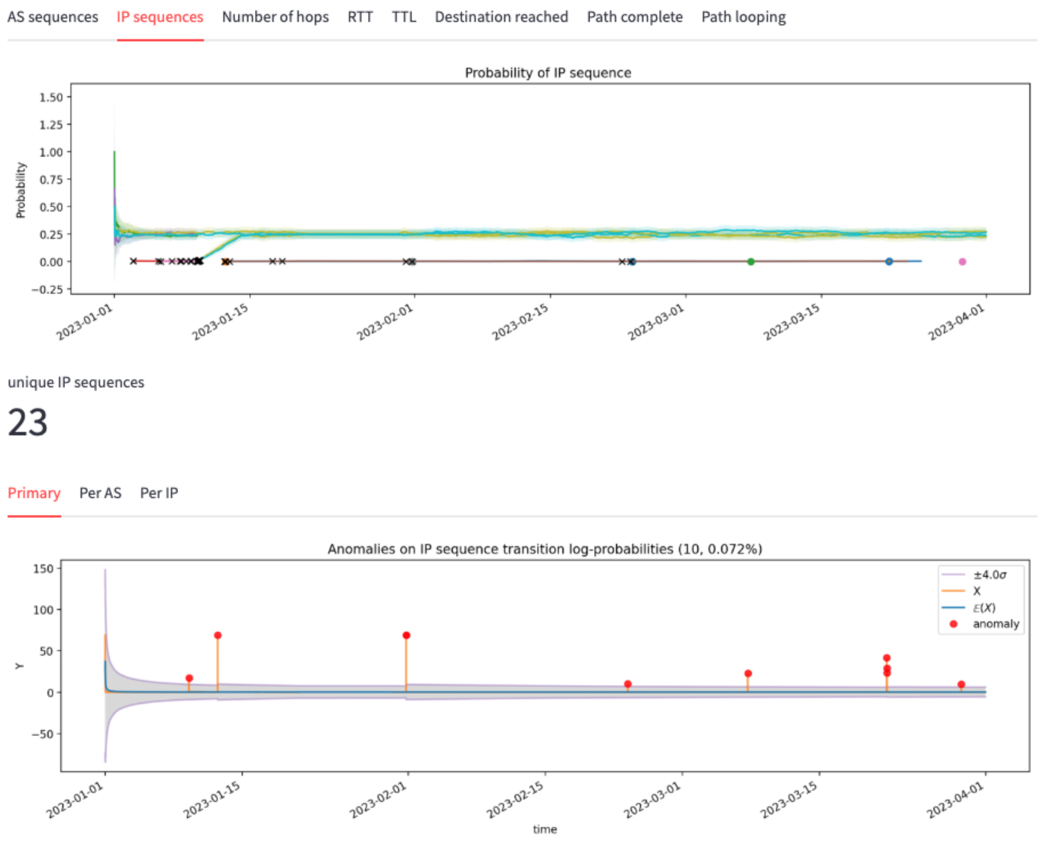
1H

Observed Model

■ **Figure C.3** Screenshot of device level aggregation



■ **Figure C.4** Screenshot of dashboard section related to AS network paths



■ **Figure C.5** Screenshot of dashboard section related to IP network paths

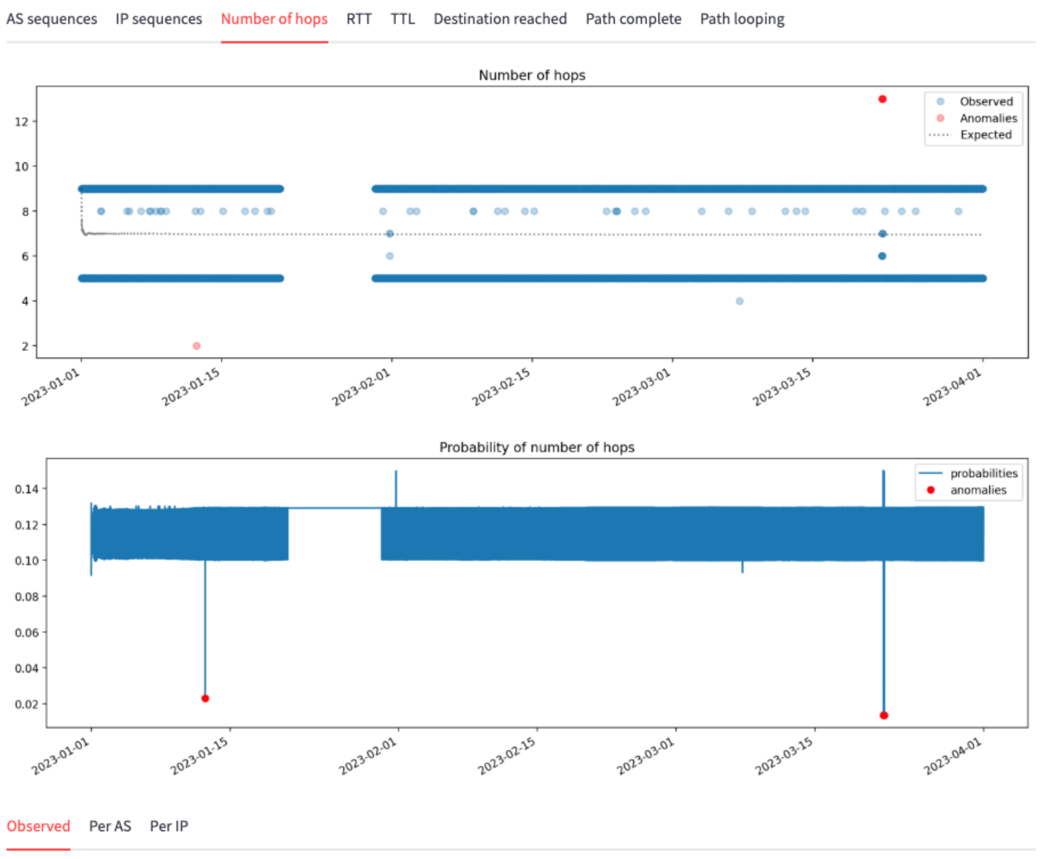
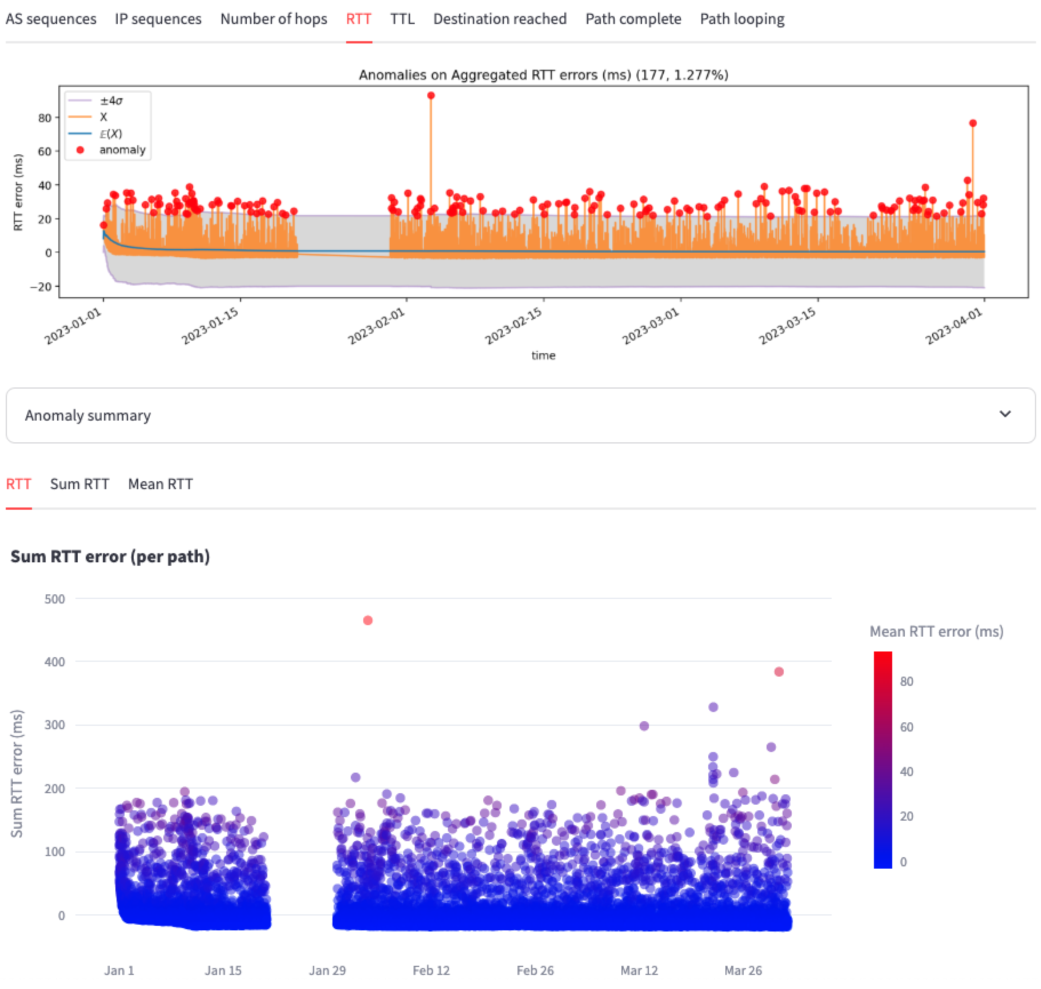
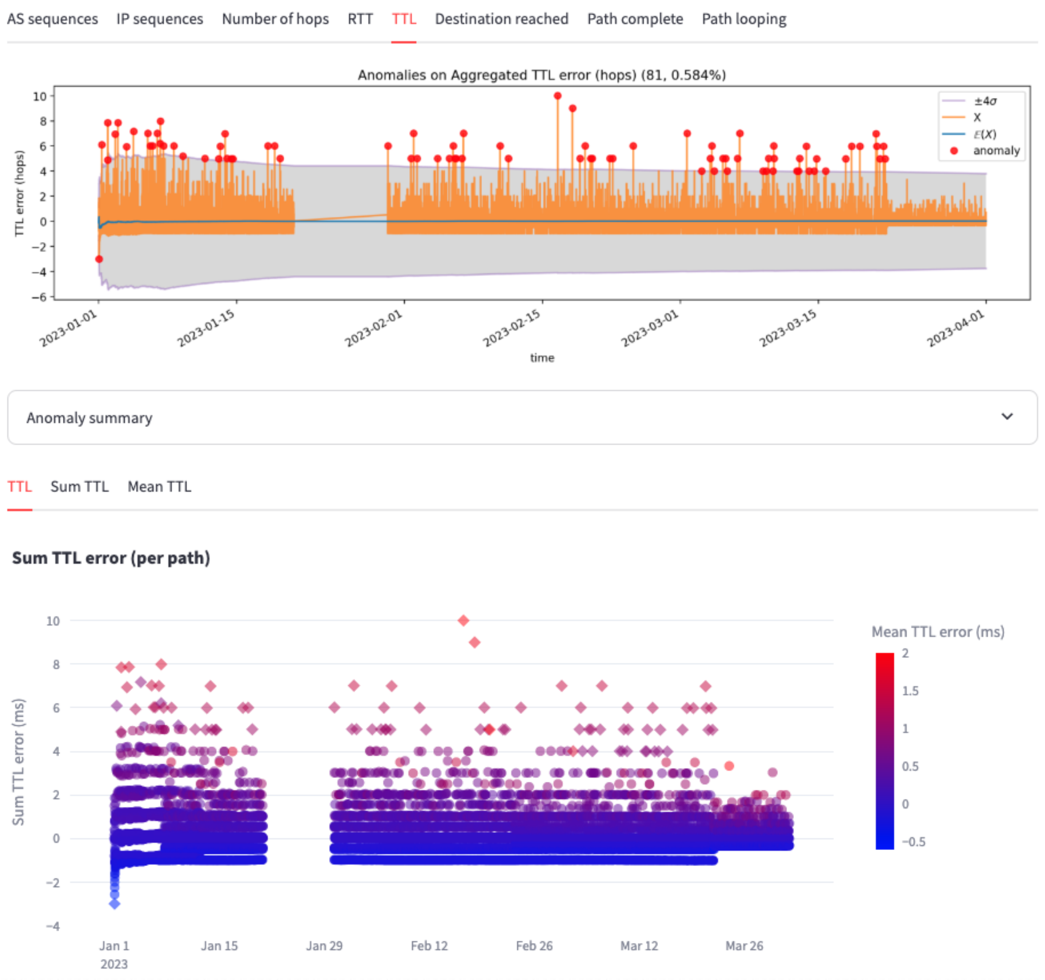


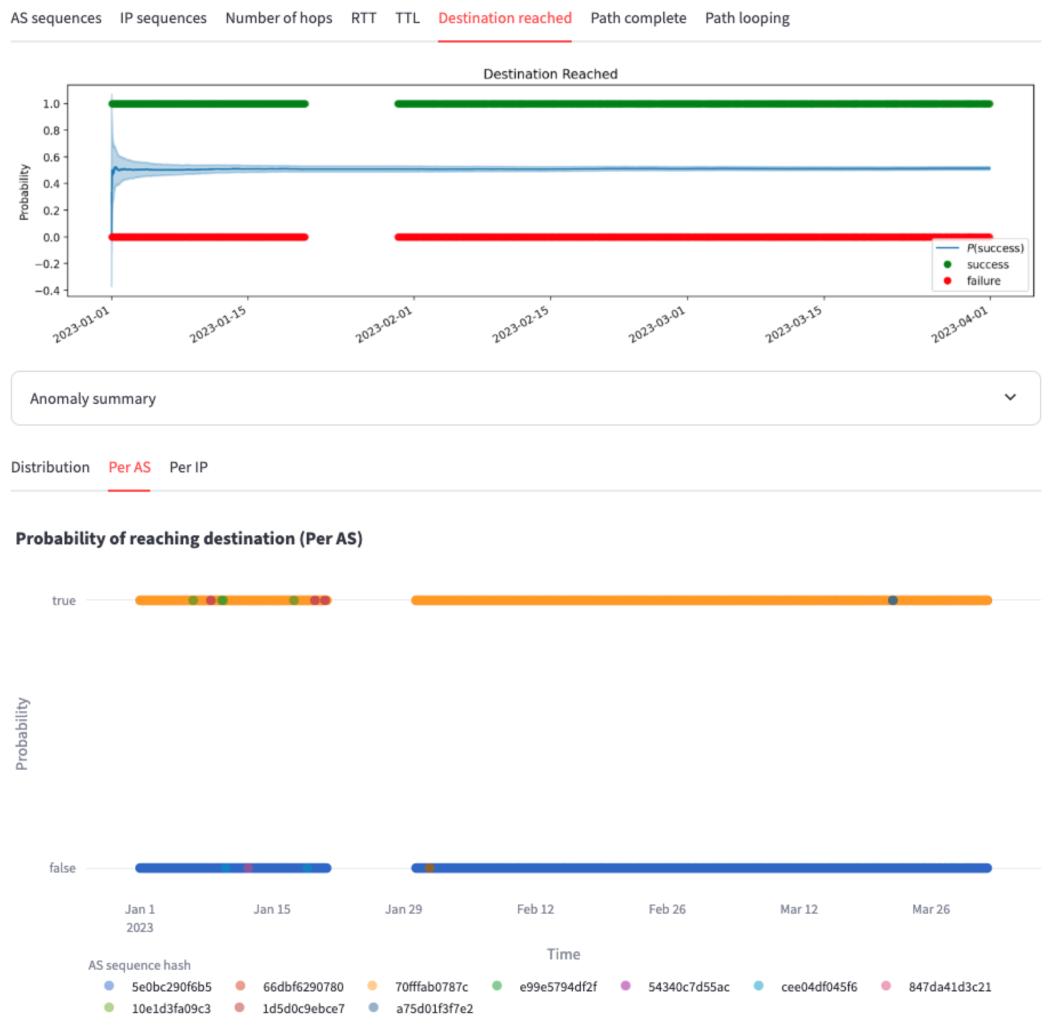
Figure C.6 Screenshot of dashboard section related to number of hops



■ **Figure C.7** Screenshot of dashboard section related to RTT



■ **Figure C.8** Screenshot of dashboard section related to TTL



■ **Figure C.9** Screenshot of dashboard section related to reaching destination

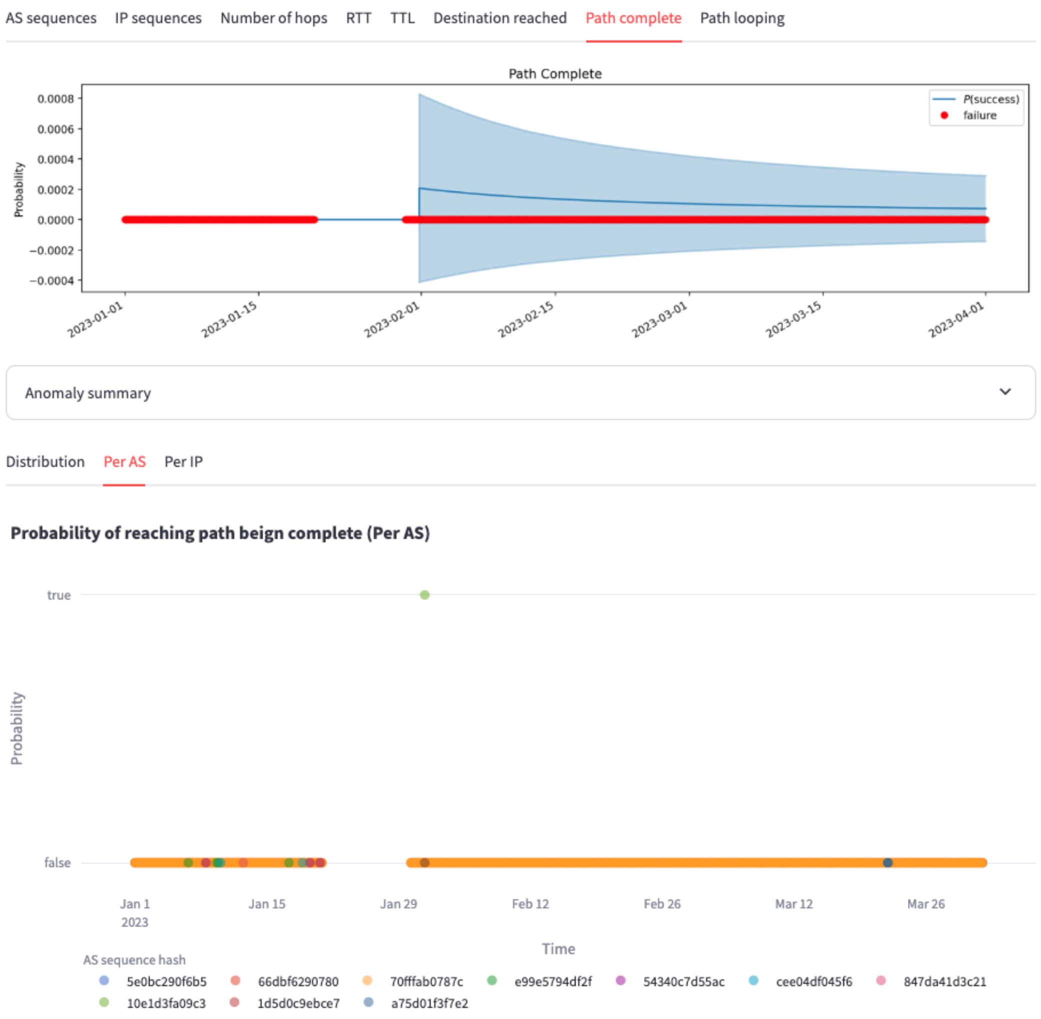
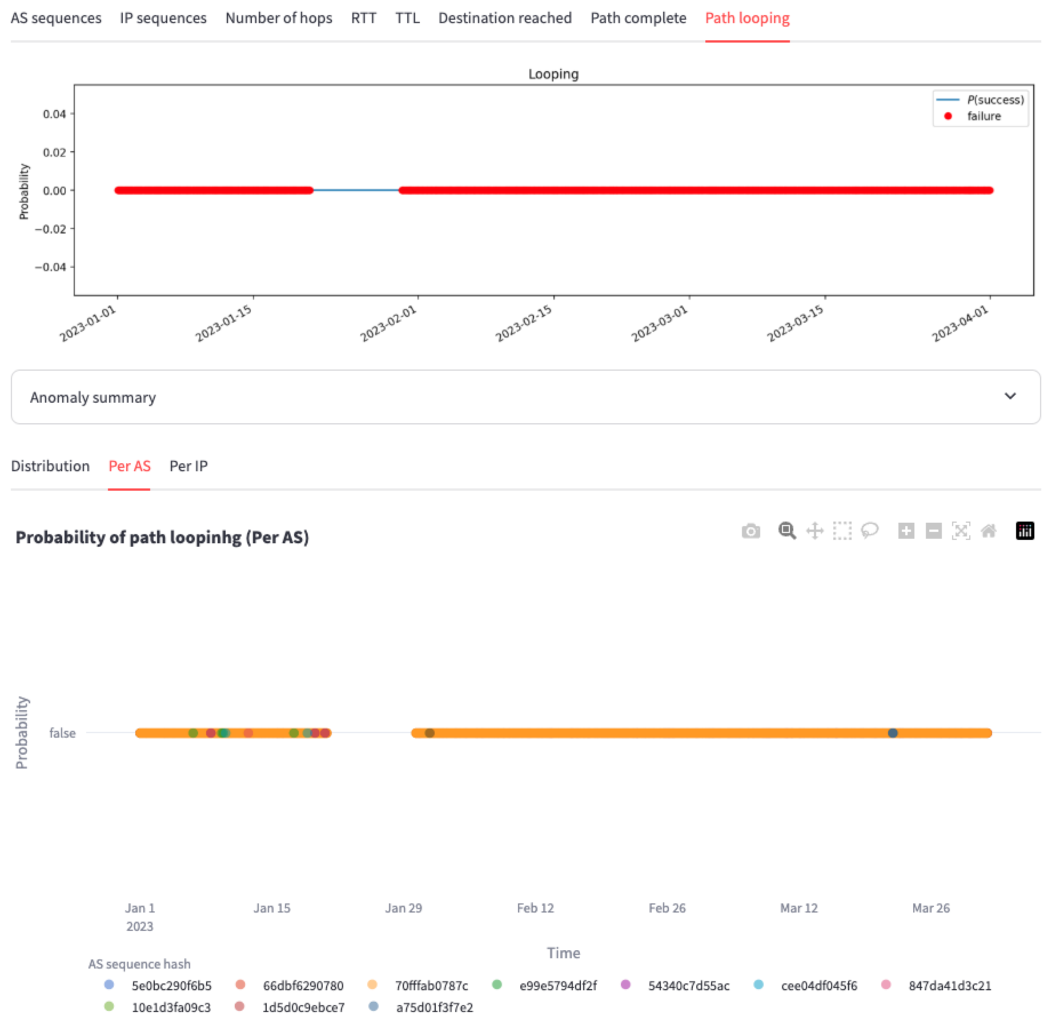


Figure C.10 Screenshot of dashboard section related to path completeness



■ **Figure C.11** Screenshot of dashboard section related to looping of traceroutes

Bibliography

1. *The Large Hadron Collider* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.web.cern.ch/science/accelerators/large-hadron-collider`.
2. *A Large Ion Collider Experiment (ALICE)* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.web.cern.ch/science/experiments/alice`.
3. *ATLAS* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.web.cern.ch/science/experiments/atlas`.
4. *Compact Muon Solenoid (CMS)* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.web.cern.ch/science/experiments/cms`.
5. *Large Hadron Collider beauty (LHCb)* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.web.cern.ch/science/experiments/lhcb`.
6. *CERN Tape Archive: High Performance Archival Storage System Developed at CERN for LHC Physics Data* [online]. CERN [visited on 2024-01-07]. Available from: `https://cta.web.cern.ch/cta/`.
7. *Storage: What data to record?* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.cern/science/computing/storage`.
8. *How a detector works* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://home.cern/science/experiments/how-detector-works`.
9. *WORLDWIDE LHC COMPUTING GRID* [online]. CERN, 2023 [visited on 2024-01-07]. Available from: `https://wlcg-public.web.cern.ch`.
10. *The Worldwide LHC Computing Grid (WLCG): Dealing with the LHC data deluge* [online]. CERN, 2024 [visited on 2024-01-07]. Available from: `https://www.home.cern/science/computing/grid`.
11. *History of WLCG* [online]. CERN, 2023 [visited on 2024-01-07]. Available from: `https://wlcg-public.web.cern.ch/about/history-wlcg`.
12. FEDOR, Mark; SCHOFFSTALL, Martin Lee; DAVIN, James R.; CASE, Dr. Jeff D. *Simple Network Management Protocol (SNMP)* [online]. RFC Editor, 1990-05 [visited on 2024-01-08]. Tech. rep., 1157. Available from DOI: `10.17487/RFC1157`.

13. ALHAIDARI, Sulaiman; ALHARBI, Ali; ALSHAIKHSALEH, Mansour; ZOHDY, Mohamed; DEBNATH, Debatosh. Network Traffic Anomaly Detection Based on Viterbi Algorithm Using SNMP MIB Data. In: *Proceedings of the 2019 3rd International Conference on Information System and Data Mining* [online]. Houston, TX, USA: Association for Computing Machinery, 2019, pp. 92–97 [visited on 2024-01-07]. ICISDM '19. ISBN 9781450366359. Available from DOI: `10.1145/3325917.3325928`.

14. LEE, Dong Cheul; PARK, Byungjoo; KIM, Ki Eung; LEE, Jae Jin. Fast Traffic Anomalies Detection Using SNMP MIB Correlation Analysis. In: *Proceedings of the 11th International Conference on Advanced Communication Technology - Volume 1* [online]. Gangwon-Do, South Korea: IEEE Press, 2009, pp. 166–170 [visited on 2024-01-07]. ICACT'09. ISBN 9788955191387. Available from: `https://dl.acm.org/doi/10.5555/1701955.1701986`.

15. GERHARDS, Rainer. *The Syslog Protocol* [online]. RFC Editor, 2009-03 [visited on 2024-01-08]. RFC, 5424. Adiscon GmbH. Available from DOI: `10.17487/RFC5424`.

16. CHEN, Zhuangbin; LIU, Jinyang; GU, Wenwei; SU, Yuxin; LYU, Michael R. Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection. *Computing Research Repository* [online]. 2021, vol. abs/2107.05908 [visited on 2024-01-07]. Available from DOI: `https://doi.org/10.48550/arXiv.2107.05908`.

17. POSTEL, Jon. *Internet Control Message Protocol: Protocol specification* [online]. RFC Editor, 1981-09 [visited on 2024-01-08]. RFC, 792. DARPA. Available from DOI: `10.17487/RFC0792`.

18. THE PERFSONAR PROJECT AND CONTRIBUTORS. *perfSONAR (performance Service Oriented Network monitoring ARchitecture)* [online]. The perfSONAR Project and contributors., 2005 [visited on 2024-01-07]. Available from: `https://www.perfsonar.net/`.

19. ELASTIC. *Elasticsearch Guide [8.11]* [online]. Elastic, 2023. Version 8.11 [visited on 2023-12-10]. Available from: `https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html`.

20. LOUGHEED, K.; REKHTER, Y. *A Border Gateway Protocol (BGP): Protocol specification* [online]. RFC Editor, 1989-06 [visited on 2024-01-08]. RFC, 1105. Available from DOI: `10.17487/RFC1105`.

21. HUSTON, Geoff. *Exploring Autonomous System Numbers* [online]. The ISP Column, 2005-08 [visited on 2024-01-03]. Available from: `https://www.dotnxdomain.net/ispcol/2005-08/`.

22. ALI, Amer Nizar Abu. Comparison study between IPV4 & IPV6. *IJCSI International Journal of Computer Science Issues* [online]. 2012, no. 1 [visited on 2024-01-07]. ISSN 1694-0814. Available from: `https://www.ijcsi.org/papers/IJCSI-9-3-1-314-317.pdf`.

23. SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to Sequence Learning with Neural Networks. In: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N.; WEINBERGER, K.Q. (eds.). *Advances in Neural Information Processing Systems* [online]. Curran Associates, Inc., 2014, vol. 27, pp. 3104–3112 [visited on 2024-01-08]. ISBN 9781510800410. Available from: `https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html`.

24. XU, Rui; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*. 2005, vol. 16, no. 3, pp. 645–678. ISSN 1941-0093. Available from DOI: `10.1109/TNN.2005.845141`.

25. HOU, Bingnan; HOU, Changsheng; ZHOU, Tongqing; CAI, Zhiping; LIU, Fang. Detection and Characterization of Network Anomalies in Large-Scale RTT Time Series. *IEEE Transactions on Network and Service Management*. 2021, vol. 18, no. 1, pp. 793–806. Available from DOI: 10.1109/TNSM.2021.3050495.

26. ZHANG, James; GARDNER, Robert; VUKOTIC, Ilija. Anomaly detection in wide area network meshes using two machine learning algorithms. *Future Generation Computer Systems*. 2019, vol. 93, pp. 418–426. ISSN 0167-739X. Available from DOI: https://doi.org/10.1016/j.future.2018.07.023.

27. ROYAL SOCIETY. *Explainable AI: The Basics: Policy Briefing* [online]. London: The Royal Society, 2019 [visited on 2023-10-26]. ISBN 978-1-78252-433-5. Available from: https://royalsociety.org/-/media/policy/projects/explainable-ai/AI-and-interpretability-policy-briefing.pdf.

28. *Explainable AI (XAI) market by offering (Solutions and Services), by deployment mode (cloud-based and on-premise), by application (medical, industrial, cyber security, financial services, customer service, and other applications), and by End-User Industry (BFSI, Healthcare & Biotechnology, retail & e-commerce, manufacturing, telecommunications, public sector, Military & Defence, and other industries) – global opportunity analysis and industry forecast, 2023–2030* [online]. Next Move Strategy Consulting, 2023 [visited on 2023-01-13]. Available from: https://www.nextmsc.com/report/explainable-ai-market.

29. ELLISON, Aaron M. Bayesian inference in ecology. *Ecology Letters* [online]. 2004, vol. 7, no. 6, pp. 509–520 [visited on 2023-12-03]. Available from DOI: 10.1111/j.1461-0248.2004.00603.x.

30. TOUSSAINT, Udo von. Bayesian inference in physics. *Reviews of Modern Physics*. 2011, vol. 83, no. 3, pp. 943–999. Available from DOI: 10.1103/revmodphys.83.943.

31. KERSTEN, Daniel; MAMASSIAN, Pascal; YUILLE, Alan. Object Perception as Bayesian Inference. *Annual Review of Psychology* [online]. 2004, vol. 55, no. 1, pp. 271–304 [visited on 2023-12-10]. Available from DOI: 10.1146/annurev.psych.55.090902.142005. PMID: 14744217.

32. KAK, Avinash. ML, MAP, and Bayesian — The Holy: Trinity of Parameter Estimation and Data Prediction. *An RVL Tutorial Presentation* [online]. 2008, pp. 1–60 [visited on 2024-01-07]. Available from: https://engineering.purdue.edu/kak/Trinity.pdf.

33. VIDAKOVIC, Brani. *Ingredients of Bayesian Inference* [online]. 2005. [visited on 2024-01-08]. Available from: https://www2.isye.gatech.edu/isyebayes/bank/handout3.pdf. ISyE8843A, Handout 3.

34. LIU, Han; WASSERMAN, Larry. *Statistical Machine Learning: Bayesian Inference* [online]. 2014. [visited on 2024-01-07]. Available from: https://www.stat.cmu.edu/~larry/=sml/Bayes.pdf.

35. HOETING, Jennifer A.; MADIGAN, David; RAFTERY, Adrian E.; VOLINSKY, Chris T. Bayesian Model Averaging: A Tutorial. *Statistical Science* [online]. 1999, vol. 14, no. 4, pp. 382–401 [visited on 2024-01-08]. ISSN 08834237. Available from: http://www.jstor.org/stable/2676803.

36. HITCHCOCK, David. *Introduction to Bayesian Data Analysis: The Gamma-Poisson Bayesian Model* [online]. University of South Carolina, 2014 [visited on 2024-01-07]. Available from: https://people.stat.sc.edu/Hitchcock/slides535day5spr2014.pdf.

37. STEORTS, Rebecca C.; QIAN, Lei. *Introduction to the Normal Gamma Model* [online]. Duke University, 2017 [visited on 2024-01-07]. Available from: `https://www2.stat.duke.edu/~rcs46/modern_bayes17/lecturesModernBayes17/lecture-4/04-normal-gamma.pdf`.

38. DUNN, Peter K. Identifying outliers. In: *Scientific Research and Methodology* [online]. Bookdown, 2023, chap. 13.5 [visited on 2024-01-07]. Available from: `https://bookdown.org/pkaldunn/SRM-Textbook`.

39. TU, Stephen. The Dirichlet-Multinomial and Dirichlet-Categorical models for Bayesian inference. *Computer Science Division, UC Berkeley* [online]. 2014, vol. 2 [visited on 2024-01-08]. Available from: `https://people.csail.mit.edu/stephentu/writeups/dirichlet-conjugate-prior.pdf`.

40. MINKA, Thomas P. Bayesian Inference, Entropy, and the Multinomial Distribution. In: [online]. 2003 [visited on 2023-12-03]. Available from: `https://tminka.github.io/papers/minka-multinomial.pdf`.

41. IPSTACK. *Locate and Identify Website Visitors by IP Address* [online]. ipstack, 2009 [visited on 2024-01-07]. Available from: `https://ipstack.com/`.

42. AGAFONKIN, Volodymyr. *Leaflet.js* [online]. 2012. [visited on 2024-01-07]. Available from: `https://leafletjs.com/`. An open-source JavaScript library for mobile-friendly interactive maps.

43. MATHIS, Matthew; SEMKE, Jeffrey; MAHDAVI, Jamshid; OTT, Teunis. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Comput. Commun. Rev.* [online]. 1997, vol. 27, no. 3, pp. 67–82 [visited on 2024-01-07]. ISSN 0146-4833. Available from DOI: `10.1145/263932.264023`.

44. TREUILLE, Adrien; TEIXEIRA, Thiago; KELLY, Amanda. *Streamlit* [online]. Snowflake Inc., 2019 [visited on 2023-12-10]. Available from: `https://streamlit.io/`. Streamlit - A faster way to build and share data apps.

45. VAN ROSSUM, Guido; DRAKE, Fred L. *Python 3.10 documentation* [online]. Ed. by SALGADO, Pablo G.Editor. Python software foundation, 2021 [visited on 2024-01-07]. Available from: `https://docs.python.org/3/whatsnew/3.10.html`.

46. WALT, Stefan van der; COLBERT, S. Chris; VAROQUAUX, Gael. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering* [online]. 2011, vol. 13, no. 2, pp. 22–30 [visited on 2024-01-07]. Available from DOI: `10.1109/MCSE.2011.37`.

47. MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In: WALT, Stéfan van der; MILLMAN, Jarrod (eds.). *Proceedings of the 9th Python in Science Conference* [online]. 2010, pp. 56–61 [visited on 2024-01-07]. Available from DOI: `10.25080/Majora-92bf1922-00a`.

48. HUNTER, John D. Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering* [online]. 2007, vol. 9, no. 3, pp. 90–95 [visited on 2024-01-08]. Available from DOI: `10.1109/MCSE.2007.55`.

49. PLOTLY TECHNOLOGIES INC. *Low-Code Python Data Apps* [online]. Montreal, QC: Plotly Technologies Inc., 2015 [visited on 2024-01-07]. Available from: `https://plot.ly`.

50. HAGBERG, Aric A.; SCHULT, Daniel A.; SWART, Pieter J. Exploring Network Structure, Dynamics, and Function using NetworkX. In: VAROQUAUX, Gaël; VAUGHT, Travis; MILLMAN, Jarrod (eds.). *Proceedings of the 7th Python in Science Conference* [online]. Pasadena, CA USA, 2008, pp. 11–15 [visited on 2024-01-08]. Available from: `https://conference.scipy.org/proceedings/SciPy2008/paper_2/`.

51. BRODERSEN, Paul J. N. *Netgraph: Publication-quality Network Visualisations in Python* [online]. Zenodo, 2023. Version 4.13.1 [visited on 2024-01-08]. Available from DOI: `10.5281/zenodo.8138403`.

Enclosed Media Contents

- `implementation.zip` .... implementation folder
  - `readme.md` .... brief description of the project, with set-up guide
  - `environment.yaml` .... Anaconda environment definition
  - `requirements.txt` .... pip requirements equivalent
  - `Makefile` .... automation script using make
  - `notebooks` .... Jupyter notebooks for proof of concept
  - `traced` .... old version of models for backward compatibility
  - `traced_v2` .... final and optimised and extended version of module
    - `models` .... sub-module containing model implementation
    - `ip.py` .... device to GPS location module
    - `site_analyzer.py` .... site aggregation logic
    - `trace_analyzer.py` .... device-to-device (trace) level logic
    - `utils.py` .... utility functions
  - `app` .... Demo dashboard application
    - `app.py` .... startup script
    - `utils.py` .... utility functions
    - `templates.py` .... templates definition
  - `tests` .... test folder
  - `results` .... plots and helper files
- `data.zip` .... archive containing serialised models for dashboard app (see readme)
- `thesis.zip`
  - `source` .... fodler containing LaTeX files
  - `thesis.pdf` .... thesis in pdf

The entire codebase, including experiment notebooks, is also available at `https://github.com/Eldeeqq/traced`.