JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Gerald Ortner

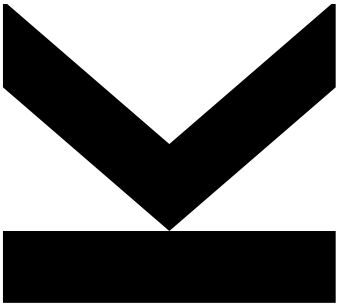
Submitted at
Institute of Networks and Security

Supervisor
Univ.-Prof. DI Dr. Renè Mayrhofer

February 2018

Mapping The Internet World

An Alternative To IP Geolocation



Diploma Thesis

to obtain the academic degree of

Diplom-Ingenieur

in the Diploma Program

Computer Science

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

Many IT security systems rely on IP geolocation to block Internet traffic from or to certain countries or geological locations. As more large companies run global networks with their public IP addresses roaming regularly between different geographical locations and with the increased use of Content Delivery Networks for serving web sites and other content, blocking traffic based on IP geolocation is not sufficient anymore in all cases. Therefore, a new approach to classify and summarize IP addresses as an alternative to IP geolocation was suggested. Furthermore, a visualization method for the classified data was evaluated. The focus of this work was on identification of Content Deliver Networks (CDN) and anonymization services like VPN and TOR. A new approach was designed and implemented to reliably identify web sites served by a CDN. The collected raw data and the final classification data set have been publicly available for future research. Additionally, an extensible prototype for collecting and classifying network traffic was implemented and release to public.

Zusammenfassung

Viele IT-Security Systeme nutzen IP-Gelocation um Internetverkehr aus bzw. in bestimmte Länder oder Regionen der Erde zu blockieren. Da immer mehr große Unternehmen weltumspannende IT Netzwerke betreiben und dabei öffentliche IP Adressen oftmals zwischen verschiedenen Standorten auf der Welt, je nach Bedarf, hin und her geschoben werden und da auch immer häufiger Content Delivery Networks (CDN) zum Bereitstellen von Webseiten und anderen Inhalten genutzt werden, ist das blockieren von Internetverkehr auf Basis von IP-Geolocation Daten oft nicht mehr ausreichend. Daher wird ein neuer Ansatz, als alternative zu IP-Geolocation, vorgestellt um IP Adressen zusammenzufassen und zu klassifizieren. Des Weiteren wird eine mögliche Visualisierung Variante für die klassifizierten Daten evaluiert. Das Hauptaugenmerk dieser Arbeit liegt auf der Erkennung von CDNs und Anonymisierungs-Services wie VPN und TOR. Es wurde eine neue Vorgehensweise entwickelt und implementiert um CDNs zuverlässig zu erkennen. Die dabei gesammelten Rohdaten und die klassifizierten Daten wurden für weitere Analysen öffentlich gemacht. Zusätzlich wurde ein erweiterbarer Prototyp zum Sammeln und klassifizieren von Netzwerkverkehr implementiert und veröffentlicht.

Contents

List of Figures

Listings

1. Introduction

Today, for IT departments it becomes increasingly important to have control over their company in- and outgoing Internet traffic and data. The reasons for that can be roughly separated into three groups, namely security, legal, and commercial. These groups usually overlap to some extent:

- Security concerns range from detecting, mitigating and stopping attacks on the company network over data leakage protection, e.g. intended or unintended disclosure of company trade secrets or customer data, to having an overview where traffic is flowing to.
- Legal concerns are typically related to privacy laws like the General Data Protection Regulation of the European Union. The GDPR harmonies the data privacy laws for the EU members and regulates how organizations within the union approach data privacy [16]. Legal concerns can also be certain provided content on a company web service which could be illegal in some countries, like copyright regulations or pornographic content.
- Commercial concerns are usually related to providing content and services only to a certain group of customers or audience. For example free vs premium content of online newspapers. Commercial reason could also be bound to legal contracts, for example as a contractor might have exclusive broadcast rights for certain media streaming content, like a TV show, within a region. This is typically enforced by blocking this specific content for own customers within the region in question.

Technical measures to tackle these issues exist in various flavors. Firewalls are employed to allow or block certain egress or ingress traffic. Where egress traffic defines traffic that starts within the company network and targets an outside destination Such traffic could be sending an e-mail to a recipient out the company or browsing a website on

the Internet. Ingress traffic on the other hand defines traffic that is originating from the internet and targets a destination inside the company network, like an incoming email. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are used to detect, alert, and block new or already ongoing attacks. Data leakage prevention (DLP) tools and measures are employed to detect and mitigate intentional or unintentional disclosure of business critical data. Websites and web services are protected from attacks by Web Application Firewalls (WAF). Remote connections of mobile workers outside the company network were not covered by the previous mentioned approaches. Such connections can be secured by employing Internet traffic policies and context aware security. Context aware security means to employ situational data like user, location, time of day for security decisions. For example if a company only has remote workers within the EU, a login from France at 4 p.m. CET is perfectly ok while a connection from Asia at 1 a.m. CET might not and therefore should trigger an alert.

All of these previously mentioned solutions use IP geolocation for some of their features. "IP geolocation" is the term used for assigning IP addresses to a certain geographical location, whereas this could mean anything from country over city to street level. The accuracy varies greatly between different geolocation approaches. Very basic approaches are just relying on preexisting information like location of IP address registration and DNS LOC entries which are not mandatory and usually not maintained regularly. More sophisticated techniques rely on latency or topology based information. Whereas latency describes the time passed by from sending a message to a remote location until an answer was received. Topology based describes approaches which enhance the latency based algorithms by using additional information gained from the internet infrastructure. IP geolocation is employed by web services and Web Application Firewalls (WAF) to restrict access to certain content according to the clients geographical locations. Management views of IDS/IPS systems show ongoing attacks based on the geographical location of the attacker on a map of the world like shown in Figure 1.1.

In firewall systems administrators and security teams often use Geo-IP Filter to block connections to or from a geographical location, mainly on country level. Decisions on which countries should be blocked are usually based on strategic considerations like no

business relations exist to these countries or on security reports like AKAMAIs "State of the Internet Security Report"[6]. According to the latest AKAMAI report the top 5 "Global Web Application Attack Source Countries" are U.S, Netherlands, Brazil, China, Germany [6]. While it seems to be suitable for most companies in Europe to block China or Brazil, only based on registration and LOC information, it will eventually cause problems to block all traffic to or from the U.S, as many global operating organizations like Facebook, Google, Amazon have registered most of their IP addresses in the U.S. For U.S. based companies blocking communication to and from U.S is out of question too. This means relying just on the location of registration is not enough. Therefore, firewall vendors usually use their own proprietary, non-disclosed algorithms for geolocation or external services like GeoIP, IP2Location, IPligence, who also rely on proprietary algorithms, with varying accuracy[30]. How often these databases are updated and if rules relying on these data are kept in sync depends on the vendor. This information are usually not made available in public documents. Albeit these databases are updated regularly using a sophisticated approach, there would still be false positives or negatives due to IP addresses moved across different geographical locations within a global BGP Autonomous Systems between the update intervals. This happens for example if a new service need to be hosted in a certain region, e.g. China, and the IP addresses used for this new service have been previously used in a different region, e.g. Europe.

With the increased use of Content Delivery Networks (CDN) for serving content an additional problem with blocking traffic based on geolocation arises. If an enterprise-critical website for company X is distributed by a global operating CDN. The content for company X is usually provided by a edge server at data center A as it is the most suitable. Due to an outage in A or due to changes in the global routing table the edge server at A isn't the most suitable anymore and a server in data center B located in a different geographical location, e.g. China, is taking over. If the resolved IP address of the edge server at B, even though registered anywhere else in the world, is geographically assigned to China and company X is blocking any traffic to and from China, the website would not be reachable anymore by any user at X.

In this thesis I propose an alternative to IP geolocation, allowing a more advanced and fine grained summarization and classification of IP addresses. The classification

Figure 1.1.: IDS/IPS Management View Ongoing Attacks World Map [22]

will allow a clear distinction between different service types like cloud computing and anonymization services and various providers of such services. The proposed approach should however not exceed the network footprint of IP geolocation. Where network footprint means the amount of traffic, in terms of packets and data transferred, produced while classifying traffic. As geolocation is performed passively there will be no active probing of remote sites at classification level for each connection. But like geolocation, active probing will be performed globally on previously collected or public available data and its results are input for the classifier. This work further proposes a graphical representation for classified connections, as counterpart to the IP geolocation world map typically used in IDS/IPS management views.

1.1. Goals Recap

The scope of this work consists of multiple distinct but linked goals.

- A novel approach for classifying IP connections as an alternative for IP geolocation should be proposed and implemented. The classification has to assign IP

addresses to Services or Service-Providers instead of geographical locations. The classification should be based on passively collected data only.

- For storage and processing of the classified packets a record (data structure) should be defined.
- For these records a graphical representation, "a new map of the Internet world", should be suggested and demonstrated. The graphical representation has to clearly outline the "borders" and edges between distinct Services or Service-Providers.

1.1.1. Constraints

Main focus of this work lies on identification and classification of Content Delivery Networks (CDN),ingress traffic from anonymization services like VPN and TOR, and the top 4 ranked Cloud Service Providers(according to Datamation [24]), namely AWS, Microsoft, Google and IBM.

In this work I will rely on different data sources and algorithms for classification:

- Already existing approaches for detecting and categorizing network/Internet traffic are evaluated, adopted and adapted.
- TCP/IP fingerprinting is revisited to find possible undetected patterns, beside the well-known mostly client specific ones for OS, Tethering and VPN detection.
- Preexisting data sources, like BGP dumps and public available IP lists from service providers, are employed for more detailed classification and/or fallback.

1.2. Structure of this Work

The remainder of this thesis is structured as follows. First I will give an overview on the current state of the art in chapter 2 and background work employed in the prototype. The approach and test setups are presented in chapter 3 and the implementation of the prototype in chapter 4. The results of the measurements and the prototype are shown at a glance in chapter 5 and evaluated in chapter 6. Finally, the work is concluded and further steps are suggested in chapter 7

2. Related Work

This chapter will discuss already existing approaches and techniques for classifying Internet and network traffic and how these are visualized. This includes existing approaches for IP geolocation with a focus on Anycast IP addresses as this is of special interest for CDN detection. Further typical setups and technologies for Content Delivery Networks are described. Additionally to the existing approaches for visualizing IT network data, a diagram type called "Sankey diagram" is introduced as it should serve as the tool for visualizing this new classification approach. In the end existing technologies to detect connections origin from VPN connections and TOR exit nodes are described.

2.1. IP Geo-Location

IP geolocation is the art of assigning IP addresses to geographical locations. In the most simple way information from various existing data-sources like WHOIS, DNS and local RIRs are put together to estimate the location of the IP. These data are usually not accurate or complete, not even in combination. Therefore, more sophisticated approaches utilize distance-range correlations by estimating the location using triangulation of a target IP according to its response time and the known geographical location of multiple vantage points[5]. Others combine the distance-range correlation approach with information gathered from BGP routing data to determine interfaces belonging to the same router, which lets infer a more accurate router location and finally using these locations as constraints for geolocating client IPs[28].

2.1.1. BGP Anycast Detection, Enumeration and Geo-Location

IP anycast allows to provide a service on multiple servers/locations, so called replicas, using the same IP address. An active network device, usually a router, is responsible for directing the IP packets sent by a client to at least one nearest, according to distance metrics and architectural decisions [9], neighbor (server), preferably only one. [41] Historically anycast was mostly used for DNS and DDoS protection but latest studies show an increasing number of anycast CDN services[11]. How DNS and BGP anycast are used in CDN services is discussed further in section 2.2.

Currently, only a few techniques for detection, enumeration and geolocation of anycast IP addresses exist as most research focuses on performance analysis, load balancing, and optimization. Existing techniques for detecting anycast IP addresses employ latency measurements from different vantage points and BGP routing information. Latency based anycaster detection is straight forward if the latency between two geographically distinct sources is greater than the sum of the latency from both sources to a single target IP, then the target must be present in more than one location[13][31]. BGP information are used, as a single IP prefix originating from different geographically distinct source routers is most likely anycast, or part of an BGP hijacking attack[13].

The detection method however does not enumerate, find all or nearly all, IP replicas. To tackle this problem in [19] a method to enumerate anycast DNS replicas using DNS CHAOS queries, which are DNS queries of a special class. But as it turned out not all DNS servers reply to these queries and if they do, the answer does not follow any standard. Another drawback of this proposal is that it is only targeting DNS services.

In [13] and [12] a method for IP geolocation based on latency measurements is suggested. Instead of just using two vantage points to detect anycast two research platforms, PlanetLabs and RIPE Atlas, with several hundreds of vantage points have been employed to enumerate and geolocate anycast IP addresses. According to their data geolocation is correct in 78% percent of the cases with and median error distribution of 384km for the other 22%, which is similar to unicast techniques.

2.2. Anycast in Content Deliver Networks

Anycast is used in CDN services in two ways, namely DNS server anycast and webserver/BGP anycast. The first approach assigns an anycast IP to any DNS server serving the CDN. These DNS servers are usually strategically placed all over the globe to have a minimal distance to all clients, this can of course vary depending on the size of the CDN and its target market. Each of the DNS servers resolves all served websites of the CDN to a webserver located in one of its, in best case the closest, data centers.

For webserver anycast the anycast IP is assigned to the webservers instead of to the DNS server. In this case a website is always resolved to the same IP but the actual targeted server depends on the shortest BGP route from the client to the next replica server.

Both approaches are used by the biggest players in the CDN market. AKAMAI[2], Amazons Cloudfront [4] utilize the first while Cloudflare[14] and Edgecast[45] implement the latter. Using one above the other is an strategical decision. BGP based anycast is ment to be more flexible as users can be sent for some webservers to location A and for others to location B. The drawbacks are that changes, e.g. to the above mentioned example, have to be announced to every peering partner. Further it is not possible to enforce a data center priority on BGP. For example it is possible in BGP to suggest preferences like go first to A and then to B but it is not guaranteed that this will be respected by others.

DNS server anycast does however allow faster switch of users in certain areas from location A to B, this is especially important in case of an outage or overload due to an DDoS attack. What is not possible for DNS server anycast CDNs is to redirect user A to location X and user B to location Y unless it has IPs in multiple prefixes with a name server in each [21][18].

2.3. Traffic Classification

Internet traffic classification is a research topic that has become more popular in the last 10 years as the Internet is becoming the most important and critical communication infrastructure in the world. The need for classification of the internet traffic has several

reasons. ISPs want to prioritize, especially for real time applications like VOIP or for paying customers, to block or protect certain traffic. But traffic classification research is very heterogeneous using inconsistent terminology, making it hard to compare different approaches. We can at least define the following classification variants:[17]

1. Services: Usually traffic originating from an IP/Port pair
2. Hosts: Classification of hosts according to their dominant traffic (both sides of the connection need to be observed)
3. Traffic profiles: (bulk, interactive)
4. Application categories: (e.g. chat, streaming, web, mail)
5. Applications: (e.g. HTTP, HTTPS, IMAP, POP, SMTP, Edonkey)
6. Content type: (e.g. text, binary, encrypted)

This classifications are usually done by the following techniques.

- By either the classical port to application mapping, which becomes more inaccurate lately as less new applications register new ports at IANA as they just use already registered ones[17].
- Employing deep packet inspection to analyze the payload, which is also becoming outdated due to more encrypted traffic[17].
- Machine learning using either supervised techniques with the drawback of having to keep the training set up to date with new applications or unsupervised techniques with the drawback of needing to provide sufficient features for categorization[17].
- Employing traffic flow pattern analysis to recognize hosts and applications[17] based on their communication behavior.

A drawback all the classification research approaches share is the missing ground truth due to lack of sufficient data for privacy and security reasons[17]. Another problem most of these approaches meet is the scalability and accuracy[17]. Therefore, most of the research is based either on a single sided view like from a university or home network or the working dataset is simplified for performance reason, e.g. excluding UDP traffic[17]. Therefore none of these approaches guarantees a 100% classification coverage nor classification accuracy.

2.4. TCP/IP Fingerprinting

TCP fingerprinting (often referred to as OS fingerprinting) is the technique to identify operating systems according to specifics in the TCP header. Usually it is distinguished between active and passive fingerprinting. While in active fingerprinting a remote host is actively probed and the response is analyzed, the passive approach packets originating from the target system are analyzed. Most of the techniques either rely only on the SYN or SYN-ACK packet, depending on if the remote system is server or client of the communication. Usual metrics utilized are congestion strategy, TCP window size, initial MSS, TTL, don't fragment bit, and/or order of TCP options[10]. Most of the approaches are also based on an expert system relying on previous knowledge but there are also multiple machine learning algorithms proposed yielding good performance for TCP traffic[3][8]. For [3] the accuracy ranges from 14% to 98.4% depending on the Layer 4 protocol used with TCP reaching the highest accuracy. TCP/IP fingerprinting is also used for NAT and tethering detection in the way that each IP address classified with more than one operating system is thought to hide a private network using NAT[10].

2.5. Graphical Representation of Internet/Network Connections

To represent Internet or network connections many plot types have been utilized. For auto generated/automatically collected data, typically world maps or network graphs/hypermaps are used. The first one typically shows the location of IP addresses on a map of the world, whereby the coordinates are determined by IP geolocation. How such a map could look like is illustrated in Figure 1.1. Such maps are used by IDS/IPS systems for showing the location of current attackers or by SIEM (Security Incident and Event Management) and Log-analysis tools to display the location where an event happened.

The latter one typically displays the connections between nodes in a network. This can be a map of all BGP routers on the Internet, the Spanning-Tree links of switches on a company network or the connection of hosts on a network. Such graphs are usually created by some sort of monitoring tool like Nagios [36] or Nedi[37] based on their

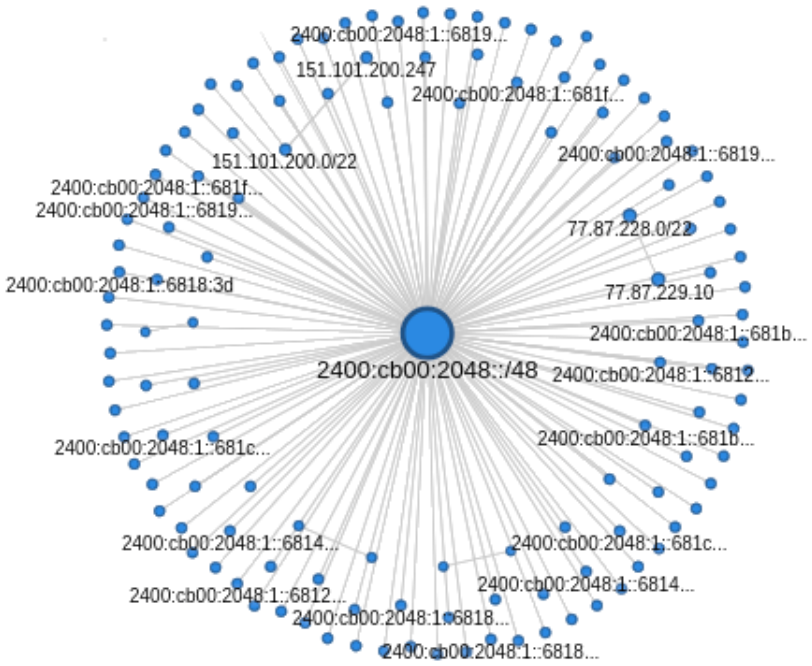


Figure 2.1.: Example Network Graph

collected data or view on the network. A sample network graph is shown in Figure 2.1. It is easy to imagine that such a graph does not scale very well, in terms of usability, with an increasing number of nodes.

2.6. Sankey Diagram

Sankey diagram is a chart type usually used for visualization of flows within a system. Such a diagram consists of nodes which represent resources or components of the system and the width of the edges indicates the proportional flow quantity. They are usually used to display government spendings (see Figure 2.2), energy production resource flows and voter flows [44].

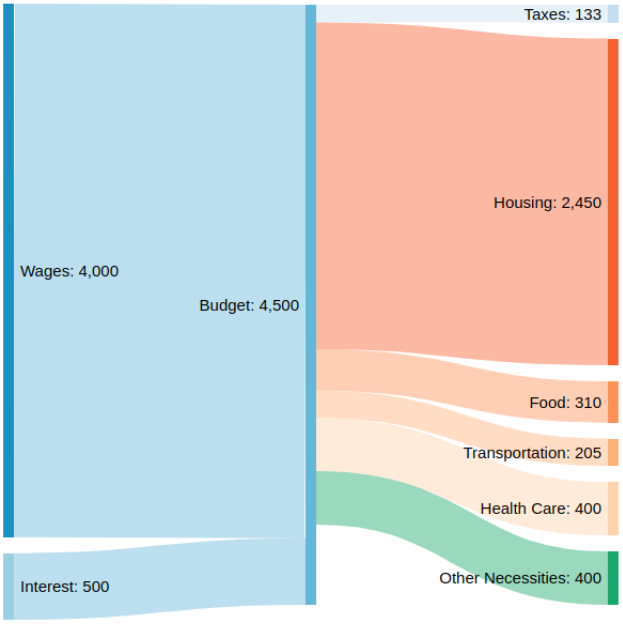


Figure 2.2.: Sankey Diagram example

2.7. P0f v3

P0f v3 is passive OS and application fingerprinting tool. It utilities the concepts described in section 2.3 to detect the targets OS (in most cases a simple SYN packet analysis) and to reason about the application by analyzing the packets payload[49]. Further some basic link type detection mechanisms are implemented to determine if packets of a host are originating from an Ethernet link or have been tunneled through a VPN or IPSec. tunnel[49]. This detection is usually just based on the default MSS sent in the SYN packet of the target[49]. P0f also performs NAT detection by detecting back and forth jumping TCP timestamps or changes in MTU and TTL[49]. Two of the main features making it interesting to utilize it in the prototype are that its detection database is easily extensible with custom metrics and that an API is provided to query P0f for its opinion about a host[49]. The drawbacks of P0f v3 are that the promised application detection currently only covers http and the development seems to have stalled as new minor releases are published every other year.

2.8. TORDNSeL

TOR exit node detection and enumeration seems not to be a primary target in researches on the TOR network as most of the research focuses on security of the TOR network. The reason for that there is no research on exit node enumeration is most probably the high number of publicly available lists of TOR exit nodes collected by crawlers and the TORDNSeL tool[47]. The publicly available lists are most of the time not complete, and maybe also not trustworthy, as the crawlers usually rely on that exit nodes declare and announce themselves as such to some central service[47]. TORDNSeL in its early version did also just rely on this information to detect exit nodes[47]. As this was not satisfying TORDNSeL is currently undergoing a major rework with more sophisticated approaches to detected exit nodes, even those who are only occasionally available[47]. TORDNSeL is based on DNS queries and therefore is easily queryable with already existing DNS tools. It defines 3 types of queries whereas only type 1 queries of format `{IP1}.{port}.{IP2}.ip-port.torhosts.example.com` are currently implemented[47]. A type 1 query returns `127.0.0.2` if a TOR server at {IP1} exists that permits connections to {port} on {IP2}. According to the design document the reason for implementing currently only type 1 queries is that they first want to see what the demand is like[47]. The TOR project also states that "the current public service is operating on an experimental basis and hasn't been well tested by real services"[47].

3. Approach

In the following approach for the three main parts of this work, namely classification of services and their provider, definition and implementation of a graphical representation for the classified services, and the implementation of a prototype, are described.

3.1. Classification

In chapter 2 already known approaches and techniques for classifying network traffic have been reviewed, but most of these are not applicable as they either focus on OS or on Layer 7 application detection or their field of application is to narrow, like anycast IP detection. Therefore new patterns and approaches for classification need to be examined and existing ones adopted.

For classification three categories, service, service-provider, and link type, are defined. For service classification the following labels have been derived from the problem description in chapter 1:

CDN Content Delivery Network
DNS Distributed/Managed Cloud DNS Service
CloudStorage Cloud Storage like Google Drive
CloudComputing Cloud Computing like Amazon EC2
VPN VPN
TOR TOR
OTHER No classification possible

Service labels are defined as mandatory, therefore a OTHER label was defined to tag services not hit by any classifier. These labels can be used to determine how much traffic was or was not covered by the classification.

For service providers no labels have been predefined. Service provider names will be defined manually when static data, e.g. service provider information, is used or derived from WHOIS information of the BGP prefix for the IP utilized by the service.

As it also could be interesting for companies to detect what link type the remote site is operating on, the following labels have been defined:

Fixed Cable/DSL
Mobile GSM/EDGE/HSDPA/LTE

3.1.1. CDN Detection

CDN providers usually use one of the following approaches to serve their content. Either, as already described in section 2.2, using Anycast IP addresses for their webservers or the DNS based approach by asking their customers to add a CNAME record pointing to a subdomain owned by CDN provider for their website. Some providers like Fastly support both techniques. Whilst in section 2.2 some reasons for using the one or the other approach are discussed from a performance and reliability point of view, there is also a reason for DNS based CDNs to provide an optional anycast IP web-server service. The DNS approach does not allow to use the root domain, e.g. `example.com`, to CNAME to the CDNs subdomain. The reason for that is section 3.6.2 of RFC1034 which points out that "if a CNAME RR is present at a node, no other data should be present; this ensures that the data for a canonical name and its aliases cannot be different"[34]. A root domain has other records, like NS or MX records, and therefor violates "no other data should be present"[34][20].

The CDN detection therefore has to pay respect to these different types of CDN networks.

The detection is based on measurements performed on a set of domain names based on Majestics million websites [32]. Majestic million websites is a list of the top one million ranked web-sites based on their backlink rate. The actual survey is done in two steps:

1. For each of the domain names in the dataset a DNS lookup for A, AAAA and CNAME records will be performed.

2. Subsequently, the RTT (round trip time) to the resolved IP addresses will be determined.

Each of the steps is performed from multiple vantage points (VP) positioned all over the globe. The results returned by each VP are represented by the 3-tuple (q, c, M) with $M = (I_1, I_2, \ldots, I_n)$, $I = (i, d)$ where:

q Queried domain name
c The resolved CNAME. (Can be empty)
I Resolved IP addresses and corresponding RTT for this domain
- i IP address
- d RTT to the IP

Anycast Webserver Based CDN Detection

For anycast CDN detection the latency based approach introduced in [13] and [31] was adopted. Instead of applying the latency based measurements to a known set of websites served by specific CDNs to geolocate their various points of presence, they will be applied to the previously described measurement results. The actual classification is summarized in the following. If the resolved IP addresses $IP(ip_1, ip_2, ..., ip_n)$ from multiple (at least two) VPs $VP(vp_1, vp_2, ..., vp_m)$ for a single domain are the same $ip_1 = ip_2 = ... = ip_n$ and the distance d from vp_x to vp_y denoted as $d_{vp_x vp_y}$ holds

$$d_{vp_x vp_y} > d_{vp_x ip} + d_{vp_y ip}$$

, with $d_{vp_x ip}$, $d_{vp_y ip}$ as distance from an target ip to vp_x, vp_y respectively, for at least 2 VP pairs then the IP can be considered anycast. This rule is illustrated in Figure 3.1.

DNS Based CDN Detection

For DNS based CDNs the same measurement results, excluding already classified ones, as for its anycast counterpart will be used. The detection consists of multiple steps:

1. Determination if it is a subdomain. If and only if it is a possible candidate.

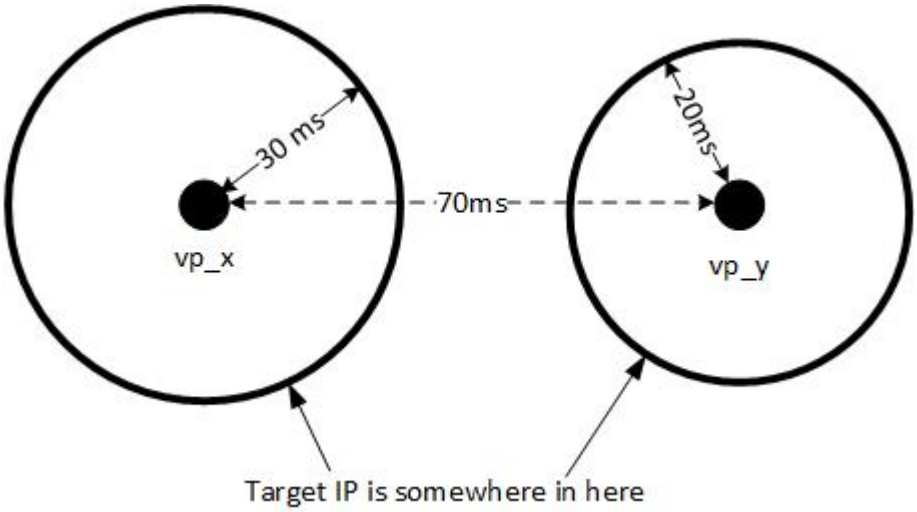


Figure 3.1.: Diverse target location detection

2. Calculate the minimum RTT for each VP per domain

3. Perform the same distance calculations as for Anycast Webserver CDNs.

3.1.2. Network Traffic Analysis

For detection of incoming connections from anonymization services but also for determination of additional information like link type (Ethernet, mobile,. . .), TCP metadata analysis for a range of devices, operating systems, services and link types are performed. Detection of outgoing connections to the TOR network or to a VPN server are not part of this work.

3.1.3. Categorization of Cloud Platforms

Detection of Cloud Platforms (CP) in a single approach seems almost impossible due their nature. Many cloud platform provider have their own custom infrastructure hardware from server to network equipment[29][33][43]. Their services include IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service), SECaaS (Security as a Service), MBaaS (Mobile back-end as a service), Serverless computing, not to mention hybrid clouds. The range of applications running on these services

range from simple websites over virtual networks to code execution. Therefore, it seems not possible to detect cloud platform providers or even services running on them by simple network traffic metadata analysis. It is of course possible to detect some servers belonging to a certain cloud provider by some metric specific to a customized OS they use or service they provide. For example Google Linux Server have a TCP Window Size of 5720[39]. But this information is just for a single OS for a single Provider and might even be used by other services of Google outside their CP. This means that for detection of CP services, first a deep insight into the infrastructure of each CP is needed to gain information on some of their services or server. This is however outside of the scope of this work and therefore I will rely only on open available IP address information provided by the biggest players in the Cloud market. The provided data range from assigning them to specific services, like Amazon does for EC2, Cloudfront, Route 53, . . . ,[15] or to just stating that certain ranges are used for their cloud services, e.g. Google [23].

3.2. Graphical Representation

For graphical representation a Sankey diagram on top of an existing charting framework is implemented and compared to a typical geolocation world map. Sankey charts are commonly used for static and not dynamic data as suggested for this work. Therefore, next to viability the scalability will be one of the main evaluation criteria.

In this work Sankey diagrams are used to visualize in- and outgoing network flows, where classification tags are displayed as nodes. The edge direction represents the communication direction (ingress and egress traffic), whereas the edge width displays the relative number bytes transferred. A possible representation is illustrated in Figure 3.2.

3.3. Classification Prototype

The prototype is targeted to analyze and classify live traffic of a networks Internet link and therefore has to be placed right in front or right after the firewall (see Figure 3.3

Figure 3.2.: Sankey Diagram usage concept

Deployment 1 and 2 respectively). For each detected TCP session a record, defined as

$$R = (id, ch, sh, srcip, srcp, dstip, dstp, proto, a, d, C, S, T) \tag{3.1}$$

$$C = S = (t, l, iphl, ipl, TO, mss, tos, ws, wsc, thl, ttl, p, dt) \tag{3.2}$$

$$TO = (to_1, to_2, \ldots, to_n) \tag{3.3}$$

$$T = (SP, NT, SC) \tag{3.4}$$

$$SP = (sp_1, sp_2, \ldots, sp_n) \tag{3.5}$$

$$NT = (nt_1, nt_2, \ldots, nt_n) \tag{3.6}$$

$$SC = (sc_1, sc_2, \ldots, sc_n) \tag{3.7}$$

Where:

id: is the session id

ch: session hash $h(srcip, srcp, dstip, destp, proto)$ from client point of view

sh: session hash $h(srcip, srcp, dstip, destp, proto)$ from server point of view

$srcip$: TCP source IP

$srcp$: TCP source port

$dstip$: TCP destination IP

$dstp$: TCP destination port

$proto$: IP protocol

a: indicates if the session is still active

d: indicates the session direction (in or outgoing)

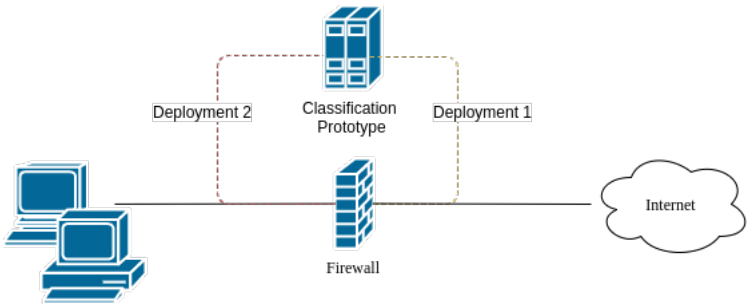


Figure 3.3.: Prototype placement

C/S: Client/Server packet information

- t: timestamp of the last seen packet
- l: SYN/SYNACK packet length
- $iphl$: initial IP header length
- ipl: initial IP packet length
- TO: initial TCP options
- mss: initial TCP Maximum segment size
- tos: initial TCP Type of Service
- ws: initial TCP Window Size
- wsc: initial TCP Window Scale
- thl: initial TCP Header Length
- ttl: Time to Live
- p: Packets transferred
- dt: Total Data transferred

T: Classification Tags

- SP: Service Provider List
- NT: Link Type List
- SC: Service List

These records are subsequently used as source for the graphical representation described before and for dynamic packet filter rule generation.

The classification module providing the T subset of the record is based on the findings for detecting CDNs, cloud platforms and anonymization services. The classification

module has to distinguish between ingress and egress TCP sessions as not all classifiers are applicable in both cases. For example does the TOR exit node classification only apply to ingress traffic. Which classifiers are used in which case is described in chapter 4.

4. Prototype Implementation

The prototype was designed to consist of three main components (see Figure 4.1), namely a data storage back-end, a worker for collecting and classifying IPv4 UDP/TCP packets, and a web front-end for the graphical representation.

For these components several constraints have been defined:

- Traffic capturing should be performed with as little overhead as possible.
- For capturing already existing libraries or tools should be used as the focus of this prototype is on classification
- The utilized libraries or tools for capturing should allow live analysis of single packets.
- Classification should start with the first/second, depending on if ingress or egress traffic, packet seen.
- The back-end for storing sessions and classification information should be optimized for a large amount of writes and fast read of aggregation data.
- The prototype should be runnable on common Linux distributions.
- For the graphical representation an already existing graphing library with support for Sankey Diagrams should be utilized as proper dynamic node placing on a plot is mathematical complex and very time-consuming in implementation and therefore outside of the scope of this work.

4.1. Backend

For the back-end MongoDB was selected as database system as it is designed for a large amount of write operations with a good query performance on large data. MongoDB is one of the leading NoSQL databases and is designed to work with so called documents.

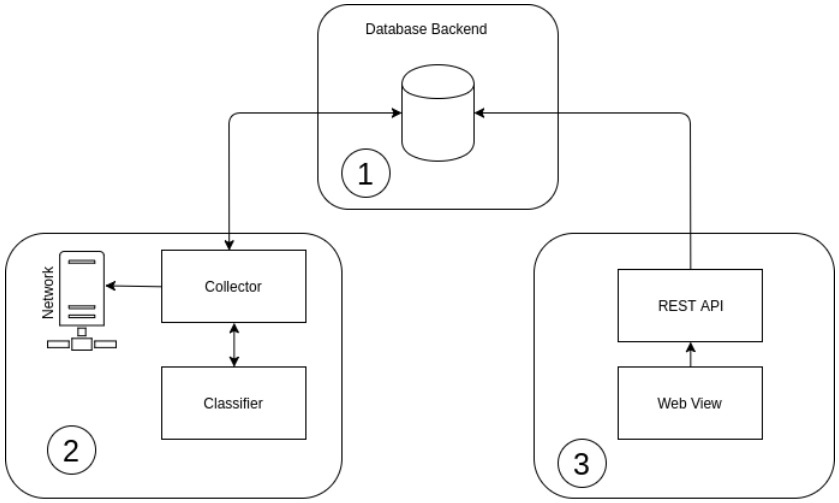


Figure 4.1.: Prototype Architecture

Such documents have a similar structure/syntax as the JSON format, and they do not need to be predefined before inserting them into a collection. This means there is no fixed table structure like in RDBMS and the client defines the structure of the document at insertion time. An example document for the data structure defined in chapter 3 is listed in Listing 4.1.

```
{
  "_id":ObjectId("598b7f3219985902437c63be"),
  "clienthash":"8483f093ffbe7add608a2a72cd74620d",
  "serverhash":"80062c53366aea07ef2317200636db1c",
  "srcip":"192.168.1.11",
  "srcport":59796,
  "dstip":"104.16.3.9",
  "dstport":443,
  "active":0,
  "proto":6,
  "client":{
    "timestamp":NumberLong(1502314290),
    "len":74,
    "iphl":20,
    "iplen":15360,
    "tcpoptions":[
      2,
      4,
      8,
      1,
      3
```

```
    ],
    "mss":1460,
    "tos":0,
    "winsize":29200,
    "winscale":7,
    "tcpheaderlen":10,
    "ttl":64,
    "pktcnt":8,
    "datatransfered":922
  },
  "direction":1,
  "server":{
    "timestamp":NumberLong(1502314290),
    "len":66,
    "iphl":20,
    "iplen":13312,
    "tcpoptions":[
      2,
      1,
      1,
      4,
      1,
      3
    ],
    "mss":1460,
    "tos":0,
    "winsize":29200,
    "winscale":10,
    "tcpheaderlen":8,
    "ttl":58,
    "pktcnt":11,
    "datatransfered":922
  },
  "classification":{
    "SERVICE" :[
      [ "CDN" ]
    ],
    "NETWORK" :[
      [ "OTHER" ]
    ],
    "SERVICEPROVIDER": [
      [ "CLOUDFLARE" ]
    ]
  }
}
"classification" : { "SERVICE" : [ [ "CDN" ] ], "NETWORKT" :
 ↪  [ [ "OTHER" ] ], "SERVICEPROVIDER" : [ [ "AMAZON", "AWS
```

```
↪ ", "CLOUDFRONT" ] ] }
```

Listing 4.1: Example Classification Record

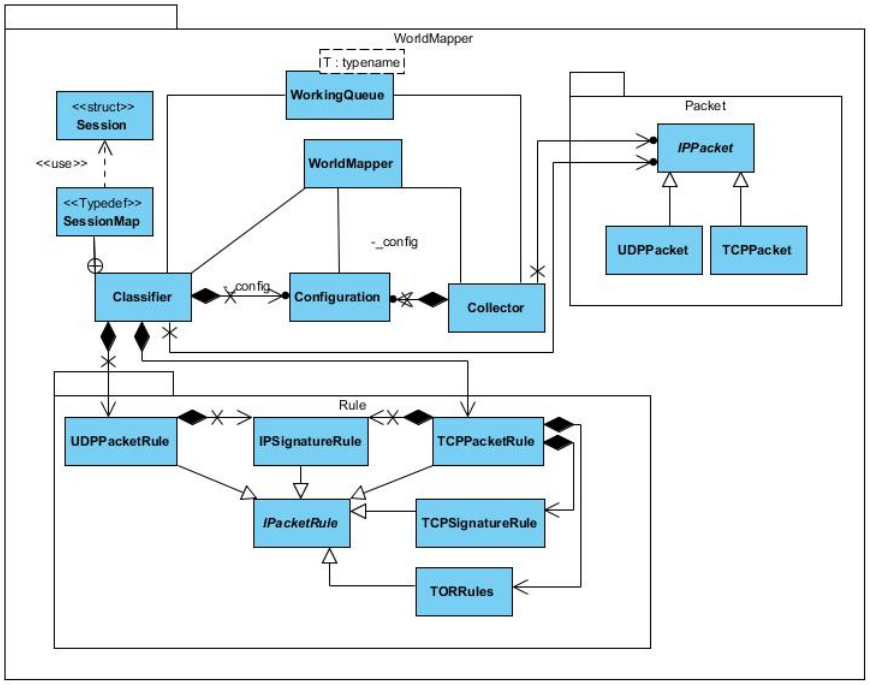


Figure 4.2.: Worker Class Diagram

4.2. Worker

The worker consists of two modules as shown in Figure 4.1 (2). The Collector is responsible for capturing packets from the network interface and enqueuing them in a working list for further processing in the classifier.

The Classifier is responsible for keeping track of TCP and UDP sessions, storing session information and statistics in the database. Further it is, as the name implies, responsible to classify packets in the working queue by applying a rule set (a chain of rules) to each packet. The worker was implement in C++ as it allows fast capturing and processing data. An overview class diagram is shown in Figure 4.2. Initially the use of some scripting language like Python was considered but the best library for packet capturing (Scapy) is too slow for live capturing and adds extra delay. It would also have

been possible to use RAW sockets in Python but this would have meant a lot of basic work on packet processing, which was not in the scope of this work. Another option would have been to utilize already existing packet capturing tools like Wireshark [48], but these are not designed to run continuously and forward single packets to other processes on the fly.

The initial class is `WorldMapper` which creates an instance of `Configuration` and `WorkingQueue` which are then passed to `Collector` and `Classifier`. `Configuration` reads and holds content from the configuration file. This configuration file contains information for database connections and tables, IP addresses to exclude from classifying and session timeouts for UDP and TCP. A sample configuration file is listed in Listing 4.2. The `WorkingQueue` is a thread save list which holds all collected packets which need to be processed by the classifier. It is defined as shared resource between `Collector` and `Classifier`.

```
{
"localIPRanges": [
"192.168.0.0/16",
"10.0.0.0/8",
"172.16.0.0/16"
],
"udpTimeout": 1800,
"tcpTimeout": 1800,
"mongoDBUri": "mongodb://localhost:27017/?minPoolSize=3&
 ↪ maxPoolSize=10",
"mongoDBName": "connclassifier",
"collectiontcpIPv4": "tcpsessionsv4",
"collectionudpIPv4": "udpsessionsv4",
"classifierThreads": 7
}
```

Listing 4.2: Example Configuration File

4.2.1. Collector

The Collector was implemented on top of libpcap [46], which is the base packet capturing component of tools like TCPDump [46] and Wireshark. Libpcap defines functionality to capture and process raw packets collected from a network interface. As most of the functions implemented in `Collector` are common for libpcap implementations only the

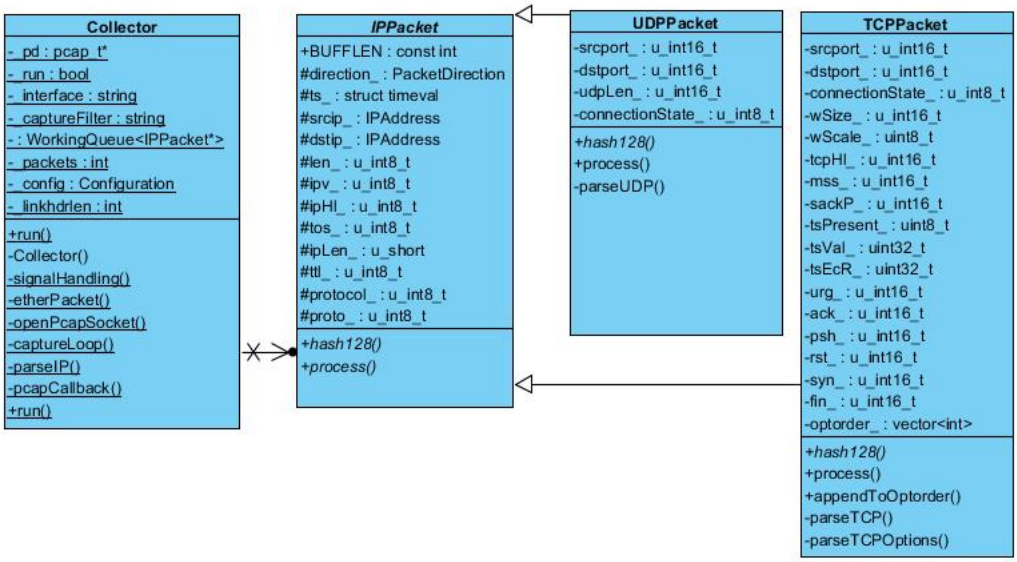


Figure 4.3.: Collector Class Diagram

function `parseIP` will be discussed as it is the core method for packet handling. It is called by `pcapCallback` as soon as a new packet is captured. This captured raw packet is then handed over to an instance of either `UDPPacket` or `TCPPacket` for parsing the IP-Header (defined by `IPPacket`) and the transport protocol specific information from it. Figure 4.2 shows the important fields and functions of these types. Especially worth to notice is the function `hash128` (see Listing 4.3) as it calculates the hashes for session identifiers. The hash is created from a string of destination-IP + destionation-Port + IP-Proto+source-IP + source-Port, or with source/destination swapped if reversed is set to false.

```
std::string WorldMapper::Packet::TCPPacket::hash128(bool
  ↪ reversed) {
uint64 hash1 = 1;
uint64 hash2 = 2;
char *inBuf = new char[BUFFLEN]();
if (reversed) {
snprintf(inBuf, BUFFLEN, "%s%u%u%s%u", this->dstip_.toString
  ↪ ().c_str(), this->dstport_
, this->proto_, this->srcip_.toString().c_str(), this->
  ↪ srcport_);
} else {
snprintf(inBuf, BUFFLEN, "%s%u%u%s%u", this->srcip_.toString
```

```
 ↪ ().c_str(), this->srcport_
, this->proto_, this->dstip_.toString().c_str(), this->
 ↪ dstport_);
}
SpookyHash::Hash128(inBuf, BUFFLEN, &hash1, &hash2);
std::stringstream ss;
ss << std::hex << hash1;
ss << std::hex << hash2;
return ss.str();
}
```

Listing 4.3: Session Hash Calculation

After parsing the packet it is checked if the conversation happens between two excluded IP addresses. If not, the packet is enqueued into the working list.

4.2.2. Classifier

As already stated the Classifier implements functionality for session management, classification, and persisting this information to the mongoDB back-end. The class diagram in Figure 4.4 shows the main fields and functions of the classifier. It is implemented to process the working queue multi threaded as the session management and classification are time-consuming and would led to a congestion in the queue. The process runner function is `process`, it contains the logic for taking elements from the queue and process them further. The number of threads is defined in the configuration file and therefore allows to be modified to yield the best performance on each hardware the prototype is running on. In the following the main processing steps for the packets, namely session management and classification are described.

Session Management

This is done by adding new sessions into the `SessionMap` sessions and removing closed sessions from it. New sessions are determined by either receiving a SYN packet or the first packet seen for TCP or UDP respectively. For each session two entries are stored in sessions map, one entry for each packet direction (in or outgoing). Where the first packet of a session is always thought to come from the "client" and any response from the "server". For each session two entries are stored in the `SessionMap` on for each calculated hash. This is done to allow fast lookup if the session already exists,

no matter which direction the packet flows, and to retrieve the actual session ID. The map key is the calculated hash (see Listing 4.3) for the packet, where both entries in the map are created with the first packet, hence the `reversed` parameter on the hash function. The actual session ID, named `identifier` corresponds to the `_id` field of the according document stored in the database collection. This document is created with the first packet seen and updated for each consecutive packet. These updates usually increase packet counter, total amount of data transferred, and the timestamp of the received packet for server and client respectively. Closed sessions are either determined if a packet with FIN or RST flag set was received or the last packet received was older than `udpTimeout` or `tcpTimeout` accordingly. Further if a TCP SYN packet was received when a session was thought to be already open a close will be forced before the new session entry is created. A session close always contains a remove from the `sessions` map and setting the `active` field of the document in the database to 0. On start of the application all documents with active $=$ 1 are update to yield active $=$ 0.

Classification

After updating the session information for the packet in the database and determining if the session still needs to be classified, the packet is handed over to the according Rule-chain. Where it is distinguished between a TCP and an UDP Rule-chain. The class diagram for the Rule-chain are shown in Figure 4.5. The rule parents are `TCPPacketRule` and `UDPPacketRule` for TCP and UDP respectively. These two classes define what and in which order rules are applied. The rule order for both is shown in Listing 4.4 and Listing 4.5. The TCP Rule-chain contains IPSignatureRule, TCPSignatureRule and TORRules whereas the UDP Rule-chain only contains the IPSignatureRule. This is because no UDP traffic was analyzed in this work and therefore classification of UDP traffic will only happen on an IP basis. The single rule types will be explained in the following.

IPSignatureRule IP signature rules determine the Service and Service-Providers only based on the assigned IPs in the rule sets. These are used for IP addresses retrieved from the provider directly as already mentioned or IPs/subnets determined by the CDN detection algorithm. These rules are stored in rules files in JSON

format. A sample rule file for Cloudflare based on provided information is shown in Listing 4.6.

TCPSignatureRule TCP signature rules determine Services according to their specific TCP flags and options signature. These rules are retrieved from the results of the TCP traffic analysis defined in chapter 3. These rules are stored in a similar JSON format as the IP signature rules. A sample rule file for TOR exit node detection is shown in Listing 4.7. The service provider is taken from the rule file or if not present from the IP signature classification or defaults to OTHER.

TORRules This rule was implemented as fallback if the TOR exit node detection on TCP base is not possible to have more data for the graphical representation. This rule uses TORDNSeL to determine if the remote site is a TOR exit node. As this service currently only provides type 1 queries as already mentioned in section 2.8, which only allow to query for specific ports provided by a specific exit node, this rule performs classification only on incoming SYN packets as outgoing packets would always return false.

After the rule chain has returned the classification result is added to the document in the database and the `classifed` filed in according `Session` objects is set to true, to prevent further classification runs.

```
1  WorldMapper::Rule::TCPPacketRule::TCPPacketRule() {
   IPSignatureRule *iprule = new IPSignatureRule();
3  iprule->readRules();
   rules_.push_back(iprule);
5  TCPSignatureRule *tcprule = new TCPSignatureRule();
   tcprule->readRules();
7  rules_.push_back(tcprule);
   TORRules *torrule = new TORRules();
9  rules_.push_back(torrule);
}
```

Listing 4.4: TCP Rule Order Implementation

```
WorldMapper::Rule::UDPPacketRule::UDPPacketRule() {
2  IPSignatureRule *iprule = new IPSignatureRule();
   iprule->readRules();
4  rules_.push_back(iprule);
}
```

Listing 4.5: UDP Rule Order Implementation

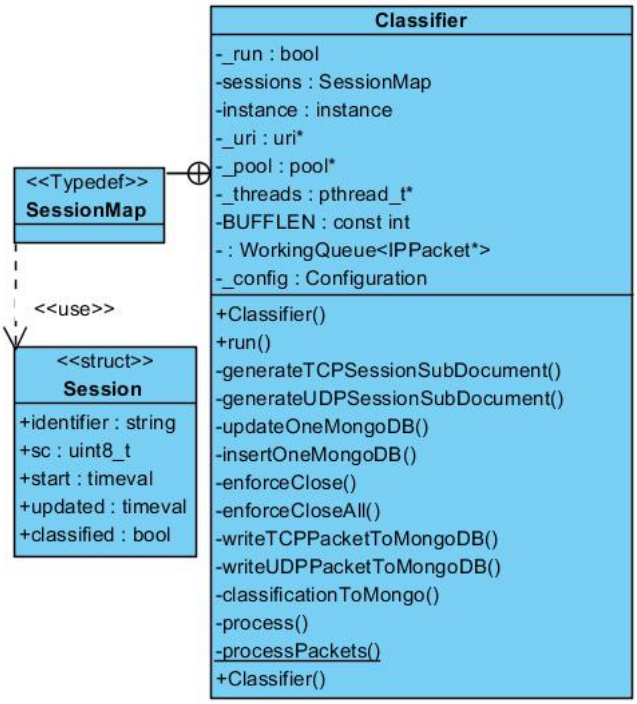


Figure 4.4.: Classifier Class Diagram

```
{
"IPRulesIPv4": [
    {"ip": "103.21.244.0/22","serviceprovider":["CLOUDFLARE"
↪ ], "service": ["CDN"]},
 {"ip": "103.22.200.0/22","serviceprovider":["CLOUDFLARE"],
↪  "service": ["CDN"]},
 {"ip": "103.31.4.0/22","serviceprovider":["CLOUDFLARE"], "
↪ service": ["CDN"]},
 {"ip": "104.16.0.0/12","serviceprovider":["CLOUDFLARE"], "
↪ service": ["CDN"]}
]
}
```

Listing 4.6: Example IP Rule

```
{
"TCPRulesIPv4":[
   {
  "name":"TOR TCP",
  "direction": 0,
  "options":{
```

```
    "MSS":[
      1310
    ],
    "WinSize":[
      13100
    ],
    "TCPHL":[
      40
    ],
    "LEN":[
      74
    ]
  },
  "service":[
    "TORTCP"
  ],
  "serviceprovider":[
    "TOR"
  ]
}
```

Listing 4.7: Example TCP Rule

name Name for the rule

direction Session direction to which the rule should be applied. 0 Ingress, 1 Egress, 2 Both

options The actual rule settings. All options allow multiple values.

SRCPRT Source Port

DSTPRT Destination Port

WSIZE Window Size

WSCALE Window Scale Factor

TCPHL TCP Header Length

MSS Maximum Segment Size

SACKP SACK Permitted

TSPRESENT Timestamp Present

LEN Packet Length

IPHL IP Header Length

IPLEN IP Frame Length

TOS Type of Service

service The Service tag that should by applied to a found session. Multiple values are allowed.

service The Service Provider tag that should by applied to a found session. Multiple values are allowed.

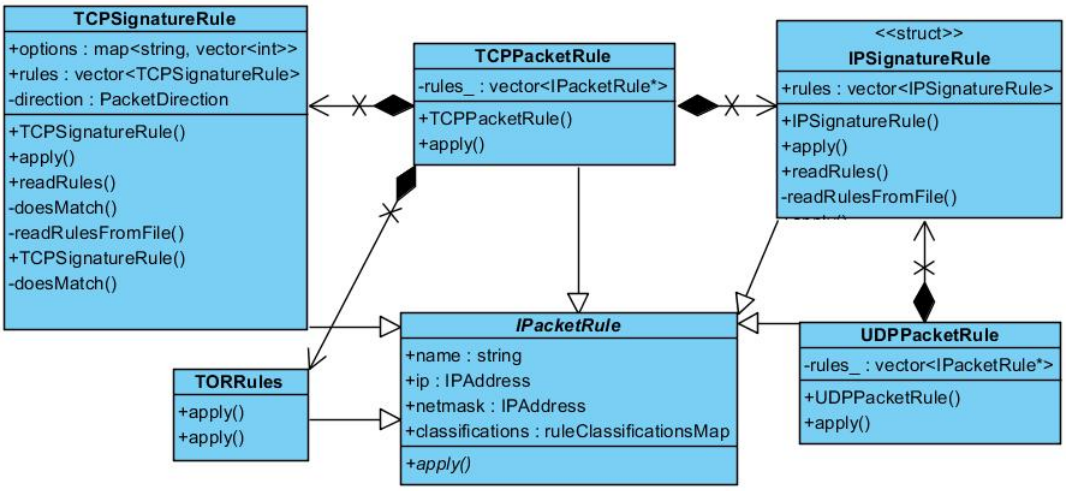


Figure 4.5.: Rule Class Diagram

4.3. Web Frontend

The graphical representation for the data is implemented as web interface which consists of two components. A REST API for providing correlated data from the database as web service and a web view for rendering a Sankey diagram with the provided data.

4.3.1. REST API

This REST service was implemented with Eve [26] a Python REST API. It allows fast and easy setup of RESTful Web Services and comes with native support for MongoDB [35]. A sample configuration which returns aggregated data grouped by classification is shown in Listing 4.8.

```
X_DOMAINS = '*'
```

```
PAGINATION = False

DOMAIN = {
    'tcpout':{
     'datasource':{
     'source': 'tcpsessionsv4',
     'aggregation' : {
     'pipeline': [
       {"$match" : {
         "$and" : [
           { "direction" : 1 },
           { "active" : 1 }
         ]
          }
       },
       { "$unwind" : "$classification.SERVICE" },
       {"$group" : {
         "_id": {
         "service" :  "$classification.SERVICE",
         "sp" : "$classification.SERVICEPROVIDER",
         },
       "ccount" : {"$sum" : "$client.datatransfered" },
       "scount" : {"$sum" : "$server.datatransfered" }
       }}
       ]
       }
     }
   }
}
MONGO_HOST = 'localhost'
MONGO_PORT = 27017
MONGO_DBNAME = 'connclassifier'
```

Listing 4.8: Example REST Configuration

This configuration list the service for retrieving aggregated data for egress TCP sessions. The data for all active sessions is grouped by Service and the hierarchical Service-Provider list (e.g. Amazon, AWS, Cloudfront). The fields `ccount` and `scount` are representing the summed transfered data for client and server respectively in the group. Next to this service the API provides three more web services with the same data structure for egress UDP and ingress TCP/UDP sessions.

4.3.2. Web Interface

The web interface is implemented in JavaScript with graphs based on bihisanky [7] a library for drawing bidirectional hierarchical Sankey Diagrams. It queries the REST API with AJAX every 10 seconds to retrieve in and -egress traffic for TCP and UDP. These retrieved data is then merged and transformed to fit the data structure requirements of bihisanky. Which means Services and Service-Providers are declared as nodes. Further bidirectional links between Service nodes, Service-Provider nodes and within the Service-Provider node hierarchy are created. In contrast to the initial idea of having in- and egress traffic displayed in one diagram the prototype was implemented to use two separated graphs, as shown in Figure 4.6. The main reason for that is, that it would not have been possible to distinguish between in and egress sessions but only between ingress and egress packets, which is not desired as lot of information would get lost. Each chart consists of three different types of nodes. The root node which represents the local site- This node is directly linked to a service node representing the type of service. These service nodes are then linked to the hierarchical structure of service-provider nodes. For better visibility of the hierarchy each layer was assigned a specific color.

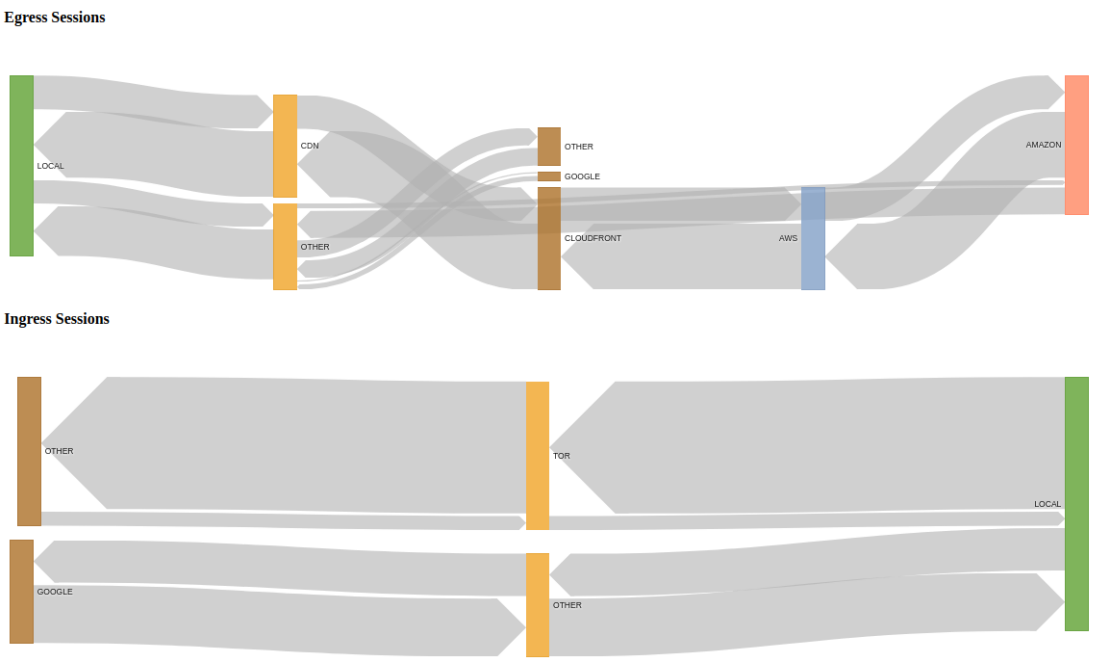


Figure 4.6.: Prototype Web Interface

5. Results

In the following the test setups and results for CDN detection and the network traffic analysis are described. Further the prototype which is partly based on the results of the measurements and the performed test runs on it are presented.

5.1. CDN Detection

The measurements have been performed on a set of domain names based on Majestic million websites [32] as already described in subsection 3.1.1. Most of the domains are only listed with their root/apex domain name, therefore for each domain an entry with common website subdomains "www", "www2", and "www3" was added to adhere to the constraints of anycast DNS based CDNs. After adding the additional domains the dataset contained 3,994,354 entries.
The actual survey was done in two steps:

1. For each of the domain names in the dataset a DNS lookup using the `dig` command from multiple vantage points (VP) over the globe was performed.

2. Subsequently, the RTT (round trip time) from each VP to the resolved IP addresses was determined by performing a TCP Ping with NPing [38] on Port 443. Whereas a TCP Ping is just sending a packet to the target IP with the SYN flag set and waiting for the corresponding SYN-ACK packet. The time between sending the SYN and receiving the SYN-ACK is taken as RTT. This approach was chosen as first test runs with an ICMP ping showed that most of the ICMP echo requests are blocked by a firewall at the remote site.

Each of the steps was performed from seven vantage points all over the globe. These VPs have been placed in different regions of the AWS EC2 cloud as shown in Figure 5.1.

	Oregon	Ohio	London	Frankfurt	Mumbai	Sydney	Sao Paulo
Oregon	X	68	156	160	217	161	182
Ohio	68	X	86	98	190	194	130
London	156	86	X	17	111	280	193
Frankfurt	160	98	17	X	109	290	207
Mumbai	222	190	111	109	X	220	301
Sydney	161	194	280	290	220	X	330
Sao Paulo	182	130	193	207	301	330	X

Table 5.1.: RTT between the vantage points

1. US West 1 (Oregon)
2. US East 2 (Ohio)
3. EU West 2 (London)
4. EU Central 1 (Frankfurt)
5. Asia Pacific Souths 1 (Mumbai)
6. Asia Pacific South East 2 (Sydney)
7. South America East 1 (Sao Paulo)

The RTT between the different vantage points is listed in Table 5.1. The round trip time was calculated as the average of thirty ICMP pings. The pings have been performed in both direction as Internet routes are not always symmetrical. The table shows that the RTT is constant in both direction. There was only a slight difference for Oregon $\longleftrightarrow$ Mumbai. For this pair another measurement was performed on a different day, which yielded the same result. As the difference is only 5 ms which is about 2% of the RTT this is negligible. For any calculation or comparison the smaller value has been used.

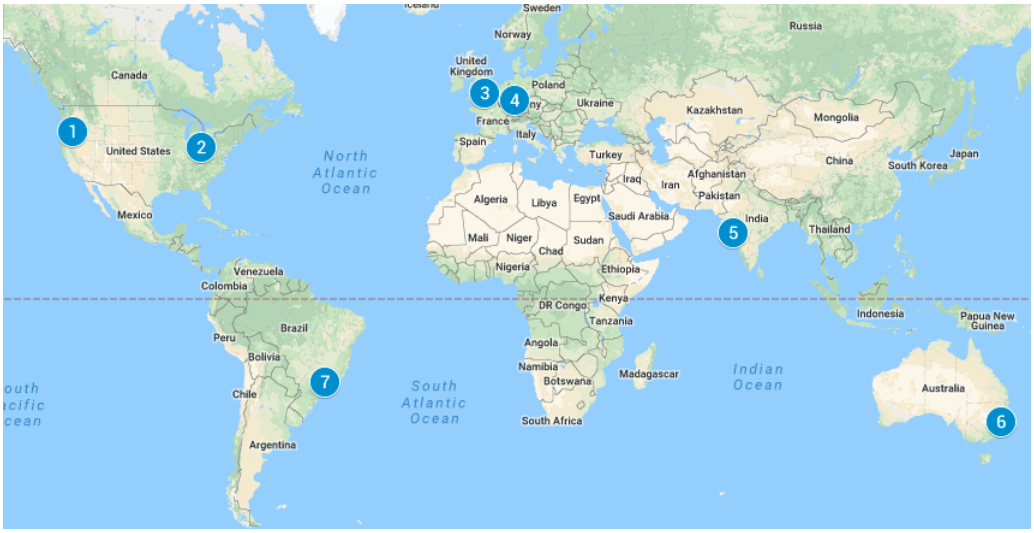


Figure 5.1.: Measurement vantage points map

5.1.1. Survey

In step one of the survey, the DNS resolution, 2,297,819 of the total 3,994,354 domains have been resolvable. Some queried domains employ a wildcard record, which resolves all not explicitly declared or non-existent subdomains to either their apex domains A record or to some dummy address. Further, as not all additionally added (www, www2, www3) subdomains do really exist this number seems plausible. These domains resolved to an actual number of 887,077 unique IP addresses. Therefore, we can assume many domains share IP addresses as they are most likely hosted on the same web server, or they are provided either by the same web server instance/load-balancer of a CDN. Further there have been 380,427 CNAMES resolved, which lets infer that there can be no more than 380427 domains that belong to an anycast DNS CDN within the result dataset.

In the second step of the domain survey the response time of the resolved IP addresses has been determined utilizing a TCP ping on port 443. Of the 887,077 resolved IP addresses only 706,523 have been responding. The difference of about 180,000 IPs is explained due to the fact that not all the queried websites listen on port 443. Further some domains have been resolved to private IP addresses. Even though DNS records on public DNS servers should not contain private IP addresses for security reasons like

Total Domains	3,994,354
Resolvable domains	2,297,819
Unique IPs	887,077
Responsive IPs	706,523

Table 5.2.: CDN survey figures

preventing information disclosure [42] it is not forbidden. Sometimes it seems to be done on purpose as security/sanitation measures to resolve wildcard entries to invalid/private and therefore not public routed IP addresses. A. Kalafut et al. have done a survey on the use of wildcard records in [27]. A summary of all the figures is listed in Table 5.2.

5.1.2. Anycast IP Web-server based Content Delivery Networks

For recognition of anycast web-server CDNs' a subset (a_dataset) of the dataset retrieved in the domain survey was used. All IP addresses yielding RTT results from only a single Vantage Point have been removed as these do not fulfill the requirements for the CDN type in question. This pre-filtering removed 131,476 IP addresses from the total 706,523 leaving 575,047 IPs for analysis. In order to apply the algorithm defined in section 3.1.1 the a_dataset had to be transformed from a domain to an IP centric representation. For each of the resolved IP addresses a record was created containing the IP as ID and its distance (the RTT) to each of the Vantage Points. Missing values have been set to a RTT of 9999 ms, so they won't ever create a candidate record.

Ground Truth As there does not exist any ground truth for CDN detection to measure the effectiveness of the applied algorithm a sample dataset has been created. This dataset consists of n samples (where $n = 1000$) which have been randomly drawn from the a_dataset. These 1000 samples have been manually classified to yield either 0 or 1 where $0 =$ no CDN and $1 =$ CDN. Each of 1000 samples has then been manually reviewed and classified. Of this drawn sample dataset 254 samples have been classified TRUE and 746 samples FALSE. Even though this dataset could be to unbalanced for

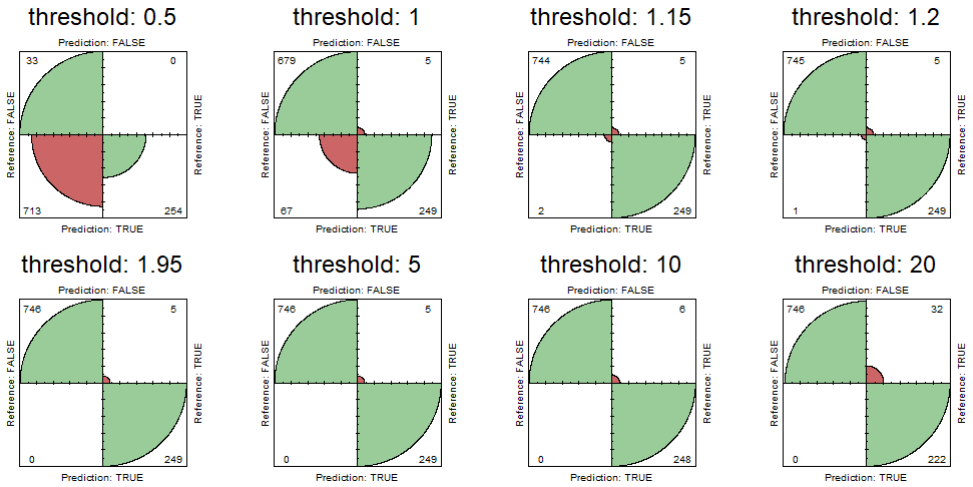


Figure 5.2.: Anycast web-server CDN threshold selection

training machine learning classifiers, as the employed algorithms could make false assumptions according to the balance in test and new data, it should not have a negative impact on manual classification algorithm used in this paper as it does not consider the number of elements in each class. Further for analysis of the results, only metrics and tactics from machine learning to combat imbalanced dataset are employed. This includes using Confusion Matrix, Recall (True Positive Rate, Sensitivity), and Cohen's kappa. After the manual classification the algorithm described in chapter 3 was applied to the sampled dataset. The result yields a Cohen's kappa of 0.8242 and a Sensitivity of 0.9803, the confusion Matrix of the absolute values is shown in Figure 5.2 threshold 1. While this seems to be a relatively good results there is still space for improvement.

Therefore, the False Positives have been manually reviewed. It turned out that most of them resulted from a minimal (in relation to the actual value) faster, accumulated, response time, of about 5—30 milliseconds, from the Vantage Points to the target IP than between the two VPs. For these False Positives the measurements of the survey phase have been redone to validate the results. These retests yielded similar values. The reason for this behavior is easily explained with the Internet architecture as the packets used for determining the distance between VP_1 and VP_2 have most probably taken a different BGP-Path than $VP_1 \leftrightarrow Target$ or $VP_2 \leftrightarrow Target$. Hence it is possible that sum of distances of $VP_1 \leftrightarrow Target$ and $VP_2 \leftrightarrow Target$ is smaller

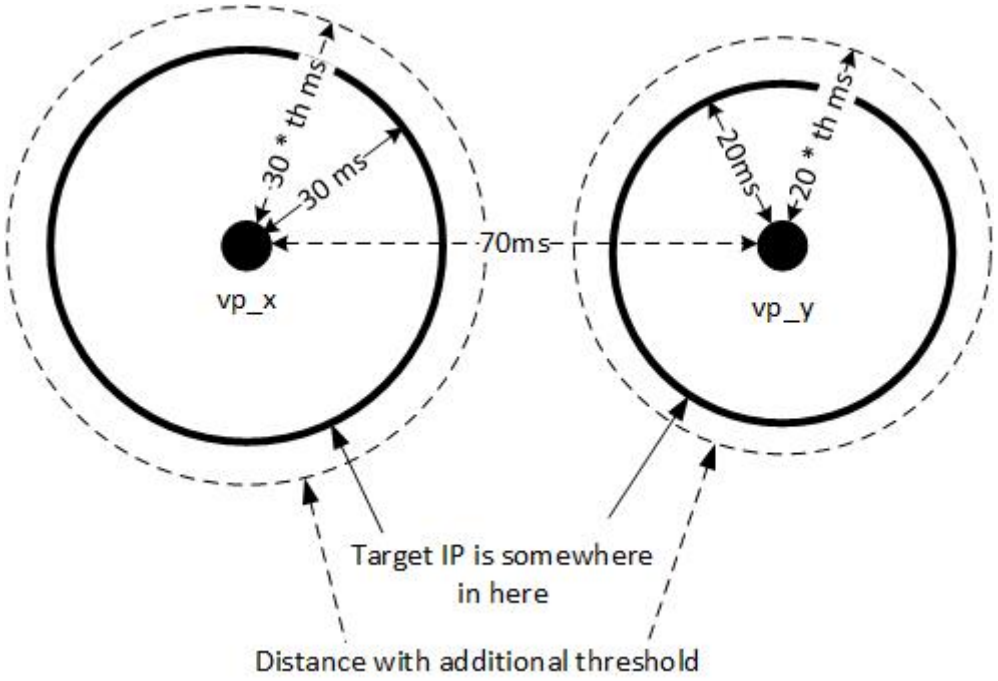


Figure 5.3.: Diverse target location detection with threshold

than the distance $VP_1 \leftrightarrow VP_2$. To tackle this problem a dynamic, percentage based, threshold was introduced. A static threshold wouldn't be sufficient in this case as the variance increases with the distance and a threshold of for example 30 ms is larger than the distance between the Vantage Points in Frankfurt and London. The threshold was applied by multiplying it to the sum of distances of $VP_1 \leftrightarrow Target$ and $VP_2 \leftrightarrow Target$. The updated scheme is visualized in Figure 5.3.

To select the appropriate threshold that returns less FALSE-POSITIVES but does not increase the TRUE-NEGATIVES a range of thresholds was tested. Therefore, the algorithm was again applied to the sampled dataset with a varying percentage threshold multiplied with the VP $\longleftrightarrow$ Target distance. The thresholds test range was defined between 0.5-20.0 in 0.05 steps. The outcome of these runs is visualized in Figure 5.4 and some selected Confusion Matrices, to show the influence of several thresholds, in Figure 5.2. According to these results the best threshold would be 1.95 at a True-Positive-Rate (TPR) of 0.9803 and a False-Positive-Rate (FPR) of 0.0 with a Kappa of 0.9867. As this would mean almost a doubling of the calculated distance values and as

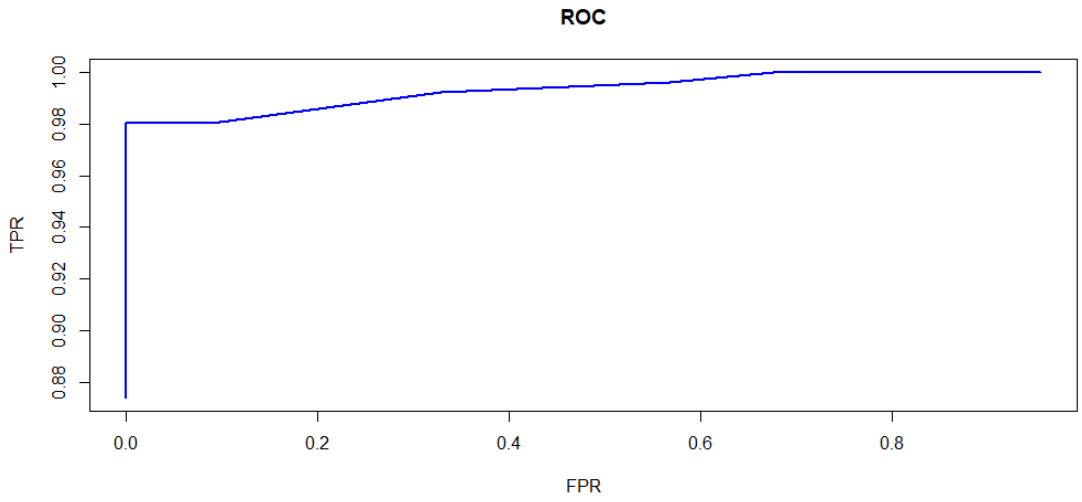


Figure 5.4.: Anycast web-server CDN ROC

there could also happen an over-fitting to the test-dataset a threshold yielding a slightly worse FPR should be selected. Therefore, a threshold of 1.2 has been selected as it is the smallest threshold having the second least FPR of 0.0013 with the same TPR of 0.9803 and a Kappa of 0.9841.

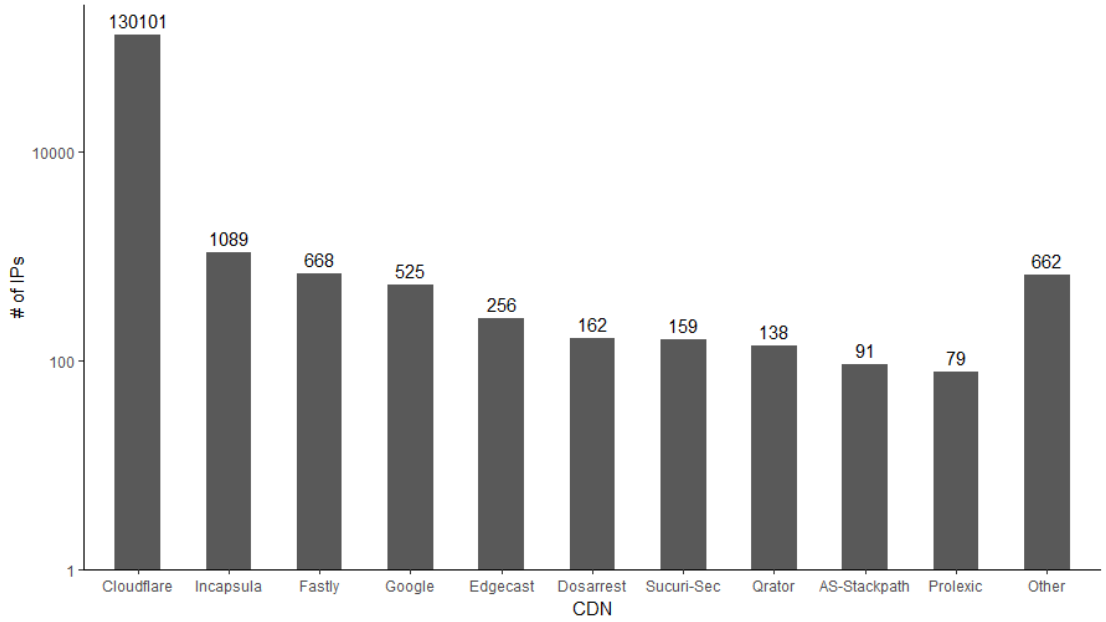
Application After having selected the target threshold the algorithm defined in section 3.1.1 was applied to the a_dataset. From the 575,047 IP addresses in the a_dataset 133,930 have been identified as part of a CDN that is about 23.3%. These IP addresses belong to 71 different Content Delivery Networks. The top 10 CDNs by number of IP addresses in the a_dataset plus a summed value for all other CDNs are listed in Figure 5.5a. The largest CDN, namely Cloudflare, represents about 98% of all IP addresses classified as part of a CDN, therefore the figure has a logarithmic scaled y axis. All of these top 10 do either provide a CDN and/or a DDoS mitigation service. For completeness the algorithm was also applied with a threshold of 1.95 (see Figure 5.5b). In comparison to the threshold of 1.2, 1519 fewer IPs have been classified as part of a CDN, where 1058 of them belong to Incapsula [25], a DDoS protection provider. A manual review of the RTTs of these IPs and research on Incapsula showed that the algorithm does not fit the setup of Incapsula. Incapsula is using anycast IP addresses for their web and DNS servers but only regional [25]. This means that the infrastructure is split up in several

regions and only within a single region the same IP address is used at different points of presence. As the measurements have been only performed from one or two Vantage Points the results are not fine-grained enough to meet the criteria for reliably detecting such CDNs, or at least not with a certain threshold applied.

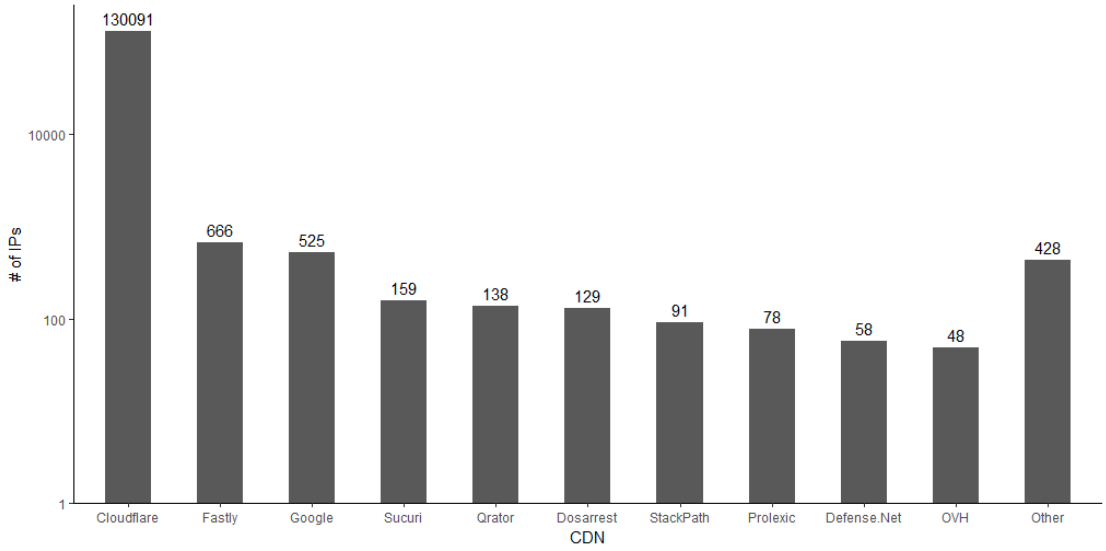
5.1.3. Anycast IP DNS server based Content Delivery Networks

For recognition of anycast DNS server based CDNs a subset (b_dataset) of the dataset retrieved in the domain survey was used. In contrast to the web-server based CDN recognition in this b_dataset the records had to be domain centric as typically for anycast DNS server based CDNs the IP of the web-server is different for each region of the CDN. As these type of CDN does rely on CNAME DNS records, only domains showing such a record have been selected for the dataset. After filtering non responsive and domains with only one result 363685 candidate entries have been left. For each of these domains a record was created containing the domain name as ID and the minimum measured distance, of all resolved IPs at each Vantage Point for the specific domain, to each Vantage Point. For this correlation of RTT results some assumptions have been made. First it is expected that if a CDN is used by a domain, all resolved IPs belong to the same CDN. This means that there are no additional IPs pointing to the actual web-server of the site as this would render some CDN features like DoS protection useless. Second if the domain is not served by a CDN the IP closest (in terms of RTT) to each VP is used. As before, missing values have been set to a RTT of 9999 ms, so they won't ever create a candidate record.

Ground Truth As with anycast web-server based CDNs, there also does not exist any ground truth for anycast DNS-server CDNs detection to measure the effectiveness of the applied algorithm. Therefore a sample dataset containing n samples (where $n = 1000$) has been randomly drawn from the b_dataset. These 1000 samples have been manually classified to yield either 0 or 1 where $0 =$ no CDN and $1 =$ CDN. All 1000 samples have then been manually reviewed and classified. This drawn and manually classified sample dataset consisted of 130 TRUE and 870 FALSE samples. As before this sample dataset is unbalanced but should not yield any problem in terms of reliability.



(a) Threshold: 1.20



(b) Threshold: 1.95

Figure 5.5.: Top 10 anycast web-server based CDNs by number of IPs

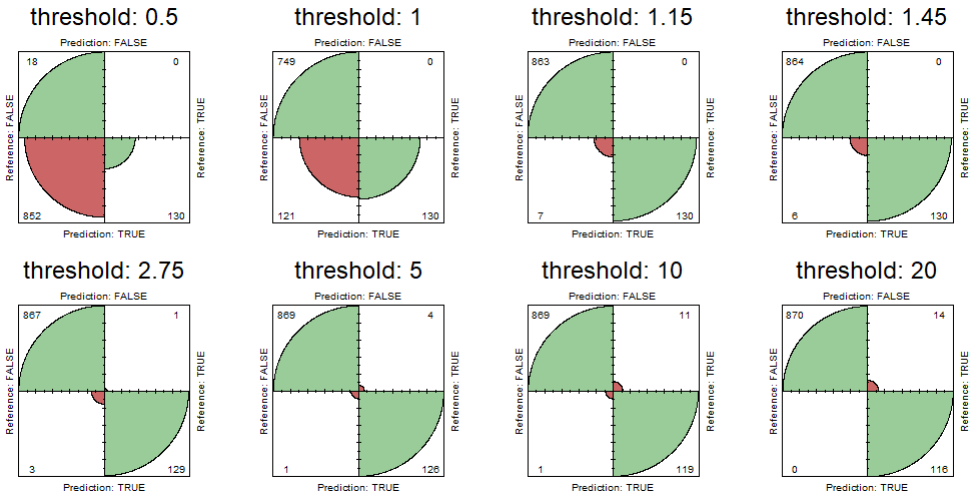


Figure 5.6.: Anycast DNS CDN threshold selection

Like for the anycast web server based CDNs for analysis of the results, only metrics and tactics from machine learning to combat imbalanced dataset are employed. This includes using Confusion Matrix, Recall (True Positive Rate, Sensitivity), and Kappa. After the manual classification the algorithm described in section 3.1.1 was applied to the sampled dataset. The result yields a Kappa of 0.6168 and a Sensitivity of 1.0000, the confusion Matrix of the absolute values is shown in Figure 5.6 threshold 1. While the Sensitivity yields a perfect, possibly too perfect (risk of overfitting), result the Kappa shows a different result which seems more reliable after checking the confusion matrix. As there is again a high False-Positive rate a similar threshold as in subsection 5.1.2 was introduced. The thresholds test range was again defined between 0.5-20.0 in 0.05 steps. The outcome of these runs is visualized in Figure 5.7 and some selected Confusion Matrices, to show the influence of several thresholds, in Figure 5.6. According to these results the best threshold would be 2.75 at a True-Positive-Rate (TPR) of 0.9923 and a False-Positive-Rate (FPR) of 0.003 with a Kappa of 0.9824. As this would mean to multiply of the calculated distance values with almost three and as there could also happen an over-fitting to the test-dataset a threshold yielding a slightly worse FPR should be selected. Therefore, a threshold of 1.15 has been selected as it is the smallest threshold having the second least FPR of 0.008 with the same TPR of 1.0 and a Kappa of 0.9697.

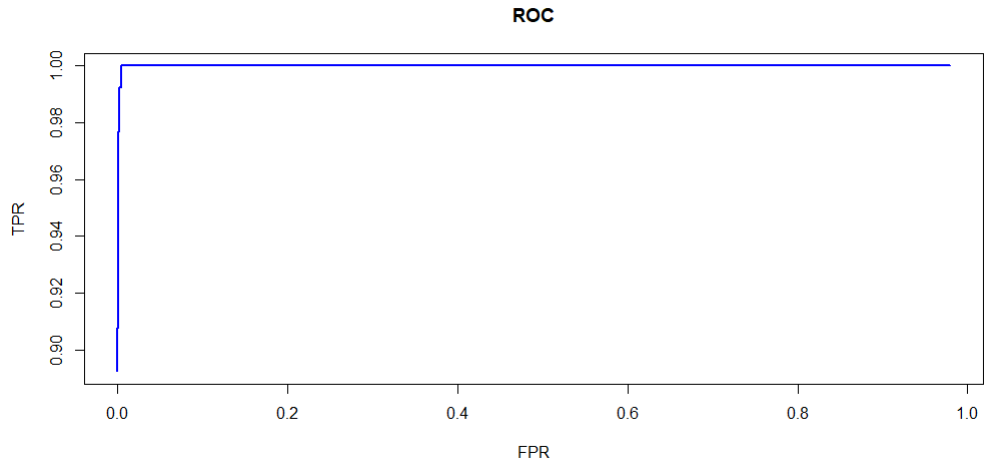


Figure 5.7.: Anycast DNS CDN ROC

Application After having selected the target threshold the algorithm defined in section 3.1.1 was applied to the b_dataset. From the 363,685 domains in the b_dataset 48,807 have been identified as part of a CDN, this is only about 13%. To assign these domains to certain CDNs each resolved IP is matched to its most specific BGP Prefix utilizing libbgpinfo [40]. This returned a total number of 140,396 IPs of 3770 BGP prefixes in 619 Autonomous Systems. After manually verifying the top 50 AS, in terms of IP count, it was identified that these belong to 47 different providers. The IPs for Autonomous Systems of the same provider have then been merged so that their count reflects the actual number of IPs for this provider. The remaining 569 AS only represent about 1.6% of the IPs and therefore have not been manually validated. Figure 5.8a shows the top 10 CDN providers, according to the number of assigned IP addresses. The manual review also showed that some providers without points of presence in every region do rely on third party companies. For example is Akamai using IP addresses of Bharti Airtel to provide services in India. Research on Bharti Airtel revealed that they are also partnering with Limelight another CDN provider [1]. All the top 50 companies are either CDN or DDoS providers, have a global infrastructure like Microsoft, Facebook, LinkedIn or provide network services, like Level 3 Networks or Bharti Airtel, which are used by CDNs. As IPs of the latter ones cannot be assigned to a single CDN they have been classified as a separate CDN on their own. Again, for completeness the algorithm was

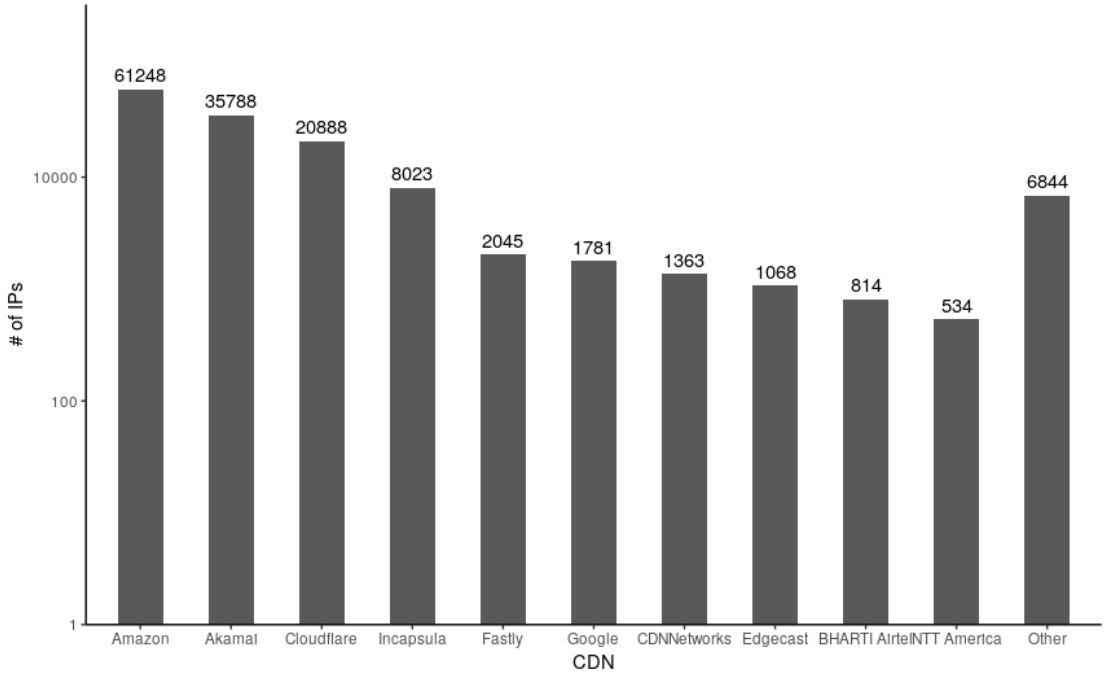
applied to dataset using a threshold of 2.75 which was yielding the best performance in the sample set. The top then for this run are listed in Figure 5.8b. This run returned 136,723 IPs of 2966 BGP prefixes in 392 Autonomous Systems which yields a difference 3673 IPs whereby these belong to 1233 prefixes in 360 Autonomous Systems. A manual review of the top 10 Autonomous Systems, which yield about 55%, for these IPs showed that all of them belong either to a CDN provider or to companies with global networks who run their own "private" CDN. Therefore, the selection of the conservative threshold was a good choice.

5.2. Network Traffic Analysis

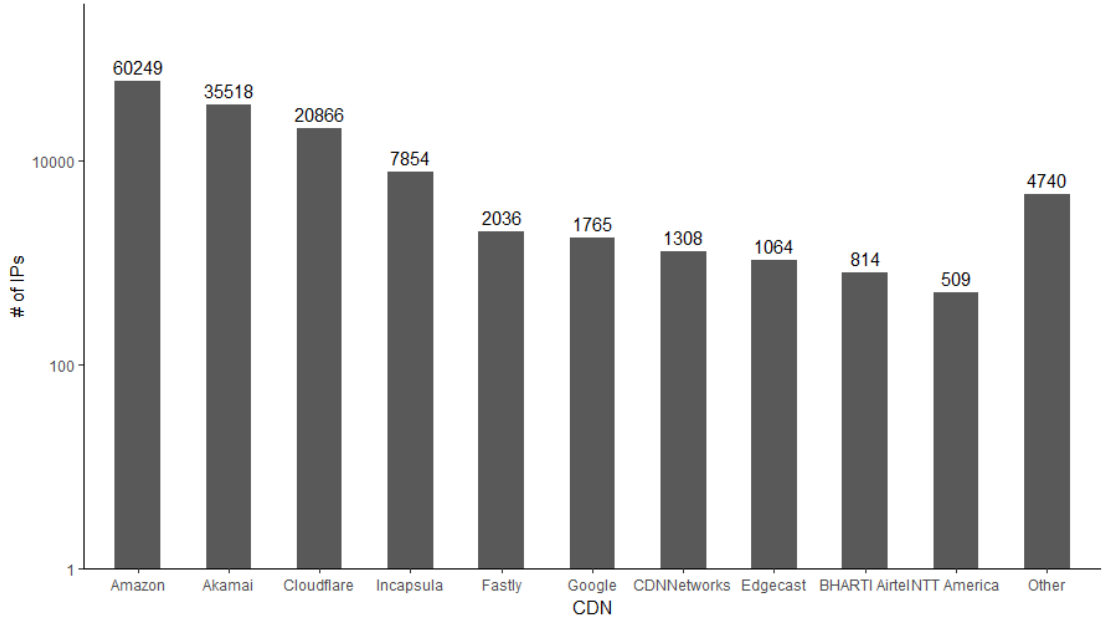
For detection of ingress connections from anonymization services and determination of remote site link type (Ethernet, mobile,...), TCP metadata analysis for a range of devices, operating systems, services, and link types was performed. The test setup included the operating systems, network carrier, and services is listed in Table 5.3.

The test was done in two steps, first data for the combinations listed in Table 5.4 Step 1 has been performed to get a better understanding which factors have an influence on the metadata and which not. In the second step additional combinations, listed in Table 5.4 Step 2, for the factors identified in step one have been performed.

For all combinations in Step 1 and 2, HTTP/HTTPS queries to the dedicated target server have been performed. TOR measurements for the iPad2 could not be performed as the device was not under full control by the author and had a mobile device management software installed which automatically removed the TOR-Browser as soon as it was installed. On the target server all traffic on port 80/443, while performing the queries was captured and assigned to the specific OS, carrier, service and connection type. The measurements for entries with service NONE are performed without any tunneling or anonymization as they should serve as a baseline. This captured data has been subsequently analyzed for patterns to identify the service and connection types used.



(a) Threshold: 1.15



(b) Threshold: 2.75

Figure 5.8.: Top 10 anycast DNS server based CDNs by number of IPs

Client devices and Operating Systems		Internet Carrier	
Device Type	Operating System	Carrier	Connection Type
PC	Windows 10	Liwest	Cable
Lenovo X1 Carbon	Ubuntu 17.04	3	GSM/HSDPA
Galaxy S3	Android 4.4.4	Yesss	GSM/HSDPA/LTE
Moto X Play	Android 6.1		
iPad2	iOS 10		

GSM Modems		Services
GSM Modem Type	Connection Type	Anonymization services
Internal	if present	TOR
Huawei E3372	USB	Random Open VPN provider
Huawei E3372 in TL-MR3420	WLAN	

Target/Capturing Device	
Operating System	Service
Ubuntu 16.04	Website on Port 443(HTTPS)

Table 5.3.: Network Traffic Analysis Test Environment

Step 1

OS	Connection Type	Service
Windows 10	Cable	TOR
Windows 10	Cable	VPN
Windows 10	Cable	NONE
Ubuntu 17.04	Cable	TOR
Ubuntu 17.04	Cable	VPN
Ubuntu 17.04	Cable	NONE
Android 6.1	Cable	TOR
Android 6.1	Cable	NONE
Android 6.1	Internal GSM	TOR
Android 6.1	Internal GSM	VPN
Android 6.1	Internal GSM	NONE
iOS 10	WLAN/Cable	VPN
iOS 10	WLAN/Cable	NONE
iOS 10	Internal GSM	NONE
iOS 10	Internal GSM	VPN

Step 2

OS	Connection Type	Service
Windows 10	USB $\rightarrow$ E3372	NONE
Ubuntu 17.04	USB $\rightarrow$ E3372	NONE
Ubuntu 17.04	WLAN $\rightarrow$ E3372	NONE
Android 4.4.4	WLAN $\rightarrow$ E3372	TOR
Android 4.4.4	Internal GSM	TOR

Table 5.4.: Network Traffic Analysis Test Phases

Step 1

The figures 5.9a and 5.9b show the results using MSS, RTT, TCP Window Size and as x,y,z axis respectively. Although the RTT does not add much value to classification as we can only conclude that it is increasing when VPN gateways are used. But RTT might also vary when the client is on a mobile network or when one of the communication partners has a bad connection therefore this seems not to be a good metric. Using the frame length instead of RTT as in Figure 5.9c and Figure 5.9d seems to aid the separation of some classes.

The tables 5.5 are listing all identified and correlated data from the collected baseline dataset. The correlated data for OpenVPN and TOR measurements for both steps are listed in 5.6.

Mobile The only difference between land line and mobile connection when no service is used is evident in the Maximum-Segment-Size. All other values like Frame-Length, Window-Size, and Header-Length are equal for all collected base data. The difference seems to originate either from the mobile carrier, as for Android and iOS different providers (3 and Yesss respectively) have been used, or the GSM/HSDPA modem of the devices. To prove that, further data using different OS types/versions, GSM modems and Mobile-Data-Providers has to be collected in step 2.

TOR For TOR exit nodes no client side or connection related influence on the collected data could be identified. This was however expected as the exit nodes setup a new TCP connection to the target host. Therefore, they often have the same fingerprint as other common Linux systems outside the TOR network. But there also exist exceptions as some exit nodes show a pattern in MSS and Window-Size, like `1310/13100`, `1460/14600`. If this is caused by configuration changes by the TOR exit node hoster or a special Linux distribution could not be identified. As there is no link to client, connection type, carrier, OS no further data will be collected in step 2.

OpenVPN As already widely known and used by tools like P0f, the collected data show that it is possible to identify connections which have been tunneled through a

Landline/Fixed

Step	Type	OS	MSS	WinSize	TCPHL	FRAMELEN
1	fixed	android	1460	65535	40	74
1	fixed	ios	1460	65535	44	78
1	fixed	linux	1460	29200	40	74
1	fixed	windows	1460	8192	28	62

Mobile connections

Step	Type	OS	MSS	WinSize	TCPHL	FRAME-LEN	Modem
1	mob	android	1370	65535	40	74	internal Moto X
1	mob	ios	1410	65535	44	78	internal iPad2
2	mob	android	1424	65535	40	74	Huawei E3372
2	mob	android	1318	11862	40	74	internal S3
2	mob	android	1318	13180	40	74	internal S3
2	mob	linux	1424	29200	40	74	Huawei E3372
2	mob	linux	1370	29200	40	74	internal Moto X
2	mob	windows	1370	8192	32	66	internal Moto X
2	mob	windows	1424	8192	32	66	Huawei E3372

Table 5.5.: Network Traffic Analysis Base Line Data

TOR connections

Step	Type	OS	MSS	WinSize	TCPHL	FRAMELEN
1	fixedtor	android	1310	13100	40	74
1	fixedtor	android	1460	29200	40	74
1	fixedtor	android	1460	14600	40	74
1	fixedtor	linux	1460	29200	40	74
1	fixedtor	windows	1460	29200	40	74
1	fixedtor	windows	1460	14600	40	74
1	fixedtor	windows	1310	13100	40	74
1	mobtor	android	1460	29200	40	74

VPN connections

Step	Type	OS	MSS	WinSize	TCPHL	FRAMELEN	Modem
1	fixedvpn	linux	1460	8192	32	66	
1	fixedvpn	windows	1351	8192	28	62	
1	fixedvpn	android	1365	65535	40	74	
1	fixedvpn	ios	1360	65535	44	78	
1	mobvpn	android	1368	65535	40	74	internal MotoX
1	mobvpn	android	1365	65535	40	74	internal MotoX
1	mobvpn	ios	1360	65535	44	78	internal iPad2

Table 5.6.: Network Traffic Analysis Data

VPN connection. There however could not be any relation to connection type, land line or mobile, identified. It is noticeable that most VPN servers enforce a MSS around 1360 as this yields the default settings in OpenVPN to mitigate packet fragmentation. Also the MSS seems to be untouched by the OpenVPN configuration. The only outliers in the dataset are from the Linux machine as there the MSS was increased and the Window-Size decreased and the TCP-Header-Length and Frame-Length differ from the base line results. As this seems not to be right the measurement was redone using a different server, but the result kept the same. Therefore, this is either coincidence that the tested servers use the same settings or it is OpenVPN client related. As there is no link to client, connection type, carrier, OS no further data will be collected in step 2.

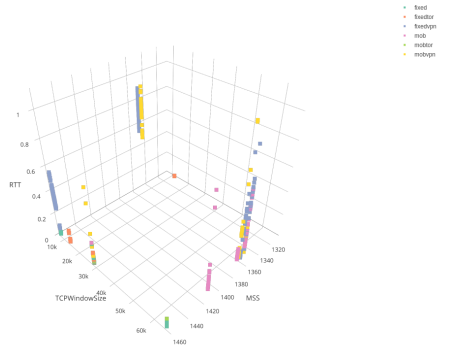
Step 2

The results from the collected data for step two show a clear picture on what was the cause for the differences in the data for mobile connections. The MSS seems only to be dependent on the GSM/HSDPA modem used. There is no difference if the Modem was connected directly via USB or indirectly, plugged int a WLAN-Router, via WLAN. TCP Window-Size, TCP-Header-Length and Frame-Length are similar to the base line, except from Frame-Len for Windows which is slightly increased. Therefore, it is possible to identify clients on mobile connections and maybe even the Operating System they are running.

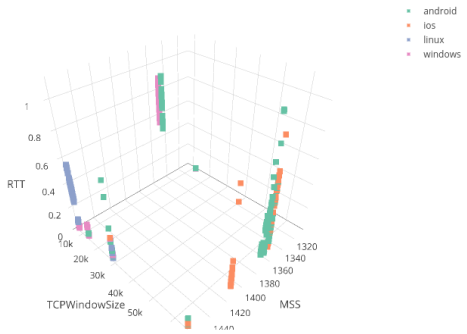
5.3. Prototype

The prototype was configured with the results of the previous two surveys, of CDN detection and network metadata analysis, installed on a virtual machine and placed within a home network, behind a NAT router. All traffic originating from the internal network interface of the router was mirrored to this machine as shown in Figure 3.3 "Deployment 2".

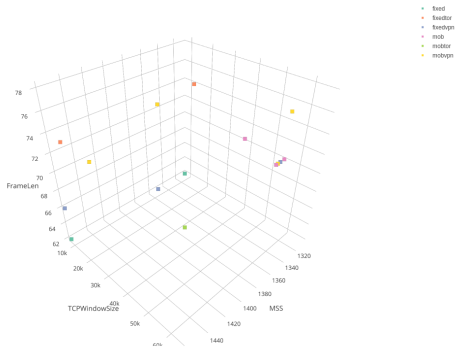
To test the prototype several outgoing queries to websites from the Majestic Million sites which are known to be served by a CDN have been performed. Further an Internet



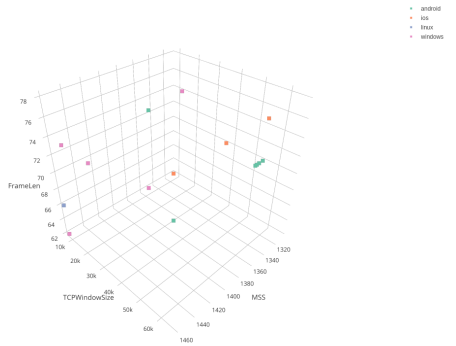
(a) Classification by type using MSS; TCP Window Size and RTT



(b) Classification by OS using MSS; TCP Window Size and RTT



(c) Classification by type using MSS; TCP Window Size and Frame Length



(d) Classification by OS using MSS; TCP Window Size and Frame Length

Figure 5.9.: Measurment results of first round

reachable, via NAT, web-server was hosted within the test-network. This web-server was queried from clients connected via land-line, VPN and TOR. For outgoing sessions all traffic was classified according to the configuration provided. For ingress traffic the additional NAT layer caused misclassification as the TCP-Header-Length for VPN connections differed from the measurements taken previously without a NAT router in between. The graphical representation at different states of the test is documented in Figure 5.10 and Figure 5.11.

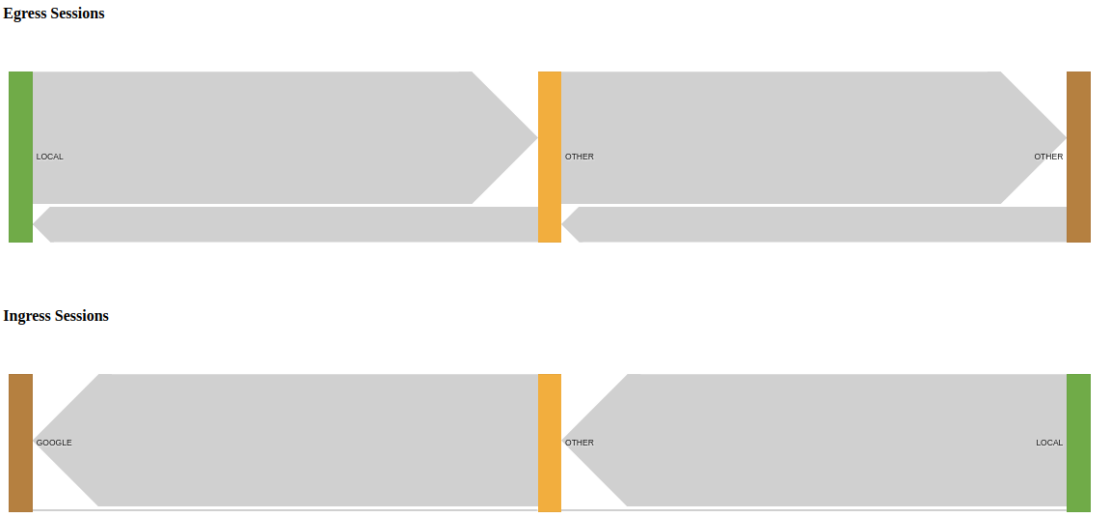
a Shows the state with only background noise as no active external connection was started. At this state there should not have been any ingress traffic classified. A review to the classification showed that there was some UDP traffic going on between a host (port 7909) on the network and an IP of Google (port 47873). What purpose this traffic had was not able to identify as it did not reoccurred during the test and was therefore not capturable. It could however be related to Googles QUIC protocol as a ongoing conversation between the host and IP in question was captured.

b Shows traffic after `aws.amazon.com` was queried in a browser.

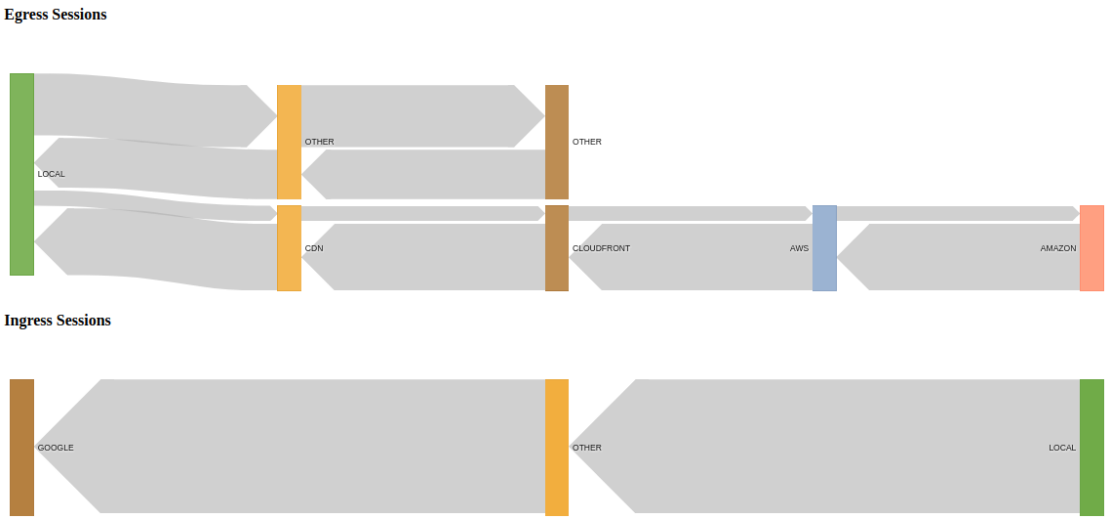
c Shows traffic after `fastly.com` was queried in browser. At the same time a page hosted on the local web-server was queried by a device using a VPN gateway.

d In addition to (c) several websites hosted by Fastly, Cloudflare, Cloudfron, Akamai were queried. Further the internal web-server was queried by several clients using either plain land-line, VPN or TOR connections.

With only a few Service and Service-Provider nodes as in Figure 5.10a to Figure 5.11a it is easy to perceive details of the visualization and get good information on how much traffic, in relation to total traffic, is flowing to a specific Service-Provider. Already with a slightly increased number of nodes as in Figure 5.11b, it is noticeable that the diagram could get quickly overloaded.

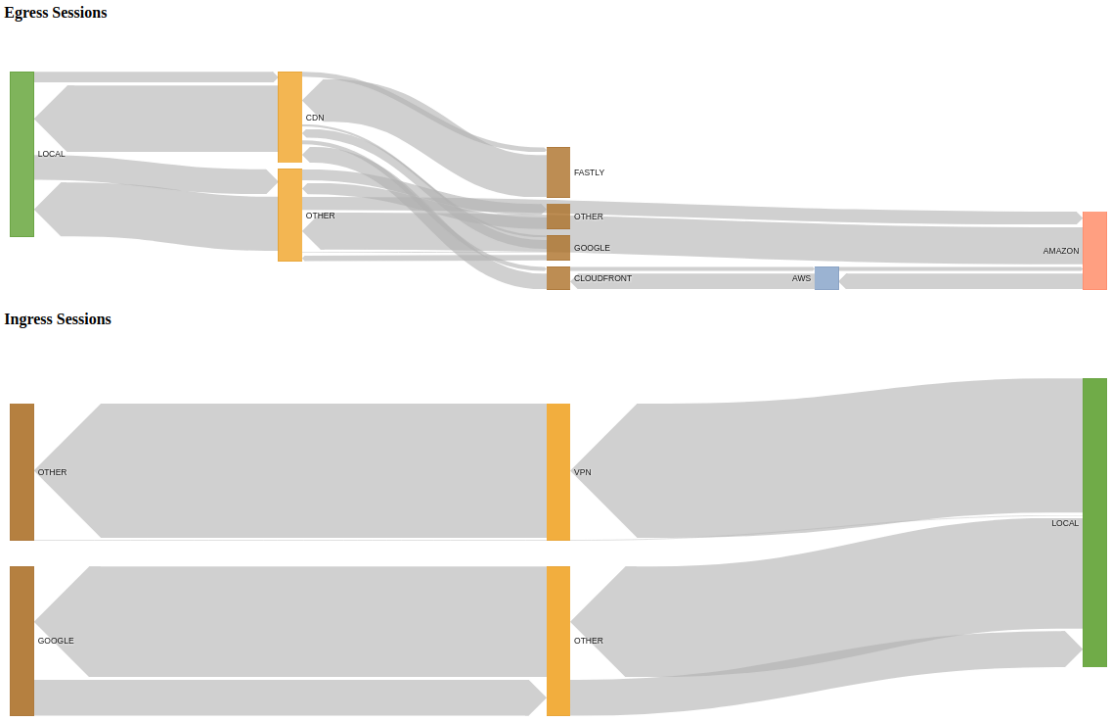


(a) Background noise

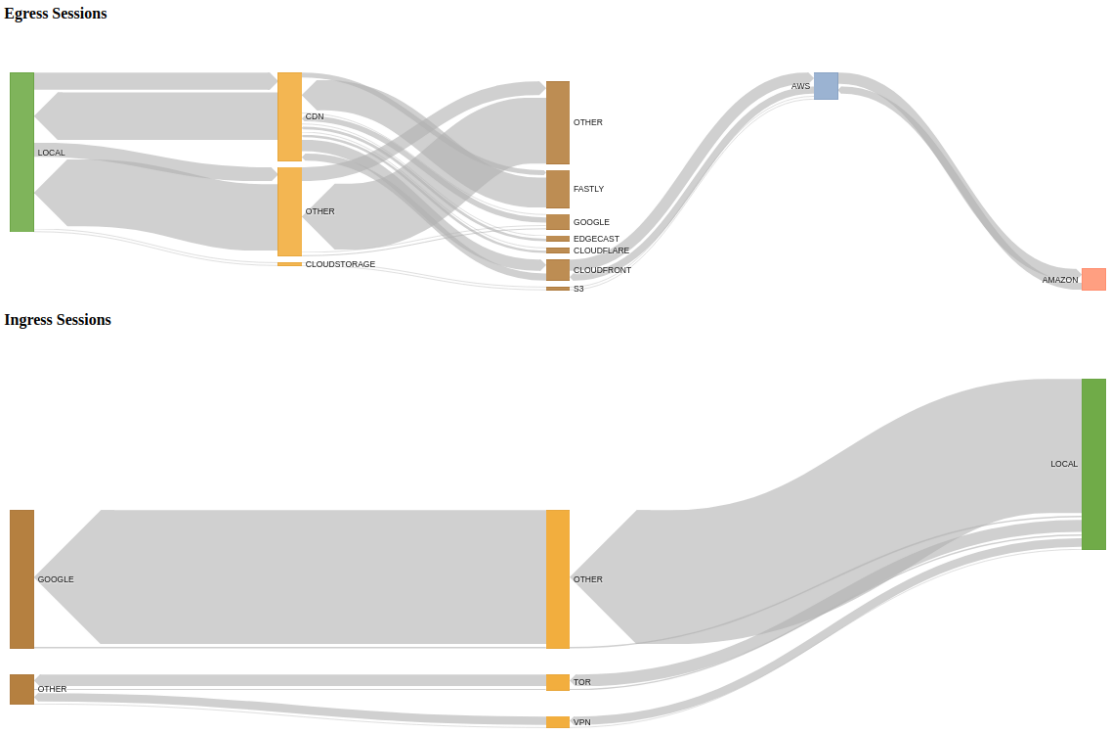


(b) Connection to aws.amazon.com

Figure 5.10.: Visualization prototype examples 1/2



(a) Connection to fastly.com



(b) Multiple ingress and egress sessions

Figure 5.11.: Visualization prototype examples 2/2

6. Evaluation

The introduced data record for holding packet/session and classification information was sufficient for this work. However additional fields should be added if information like time behavior is needed for classification. The classification part of the record is capable of holding several tags for Service and Service-Provider, which could lead to some confusion as it is not clear which one is the best fitting. Therefore, it would have been better to extend the record with a weight tag to allow modules to force their result. Also a reliability indicator could be useful if only a part of a rule, for example in TCP signature classification, was fitting. Currently it is all or nothing, which leads to instant false negatives if only one part is mismatching, as with VPN connection detection in combination with a NAT router. Compared with IP geolocation this approach does require much more work until all communication partners can be categorized. For IP geolocation only the coordinates have to be assigned according to some algorithm that can be applied globally. For the introduced classification approach for each Service, which can be infinite many, separate techniques for detection need to be found and tested. In the following the results for the distinct parts of this work will be recapitulated and evaluated.

6.1. CDN detection

The CDN detection approach does perform exceptionally well on the test dataset, with a sensitivity rate of about 98%, when a dynamic threshold is taken into account. Without this threshold the sensitivity is only about 80% which means 20% false negatives what is not acceptable. The comparison of the results of classifications for anycast IP web-servers and anycast DNS-server based CDNs for both the threshold that was the best according to the ROC and the one manually selected, showed that the algorithm does not work for all CDN architectures equally well. For both CDN types when the best

threshold according to the ROC was used, the false negatives slightly increased. Most of these false negatives for anycast IP web-server based CDNs belong to one company, Incapsula, which is using a regional anycast. With an IP only present in a single region and only one to two Vantage Points for measurement within this region, even a conservative threshold could easily lead to false negatives. For example is the distance Oregon-Ohio measured with 68ms. If the distance from these two Vantage Points to a target IP, only present in West-US region, in total is 58ms, a dynamic threshold multiplier of 1.2 would lead to an assumed distance of about 70ms which yields a false negative. In such a case the placement of Vantage Points was too coarse to get sufficient data to reliable detect CDNs with such an architecture.

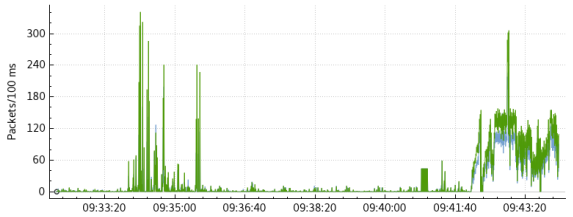
6.2. Network traffic analysis

The network traffic analysis did show that it might be possible to identify IPs/BGP Prefixes that belong to a mobile network provider by the fingerprint of GSM modems. But to prove that, real traffic metrics from a mobile provider are needed to identify and validate more GSM modems patterns. For service detection the traffic analysis did not add much value. Already known VPN patterns have been identified and complemented with some new metrics to possibly infer the operating system. During the test run of the prototype it turned out that these additional patterns lead to false negatives when analysis is performed behind a NAT router. For TOR exit node detection patterns have been found for indicating a relationship between MSS and initial TCP Window Size but this is only the case in about 50% of the recorded connections. It is also not clear if this pattern was caused by the used operating system of the exit node or has been set by the operator running it.

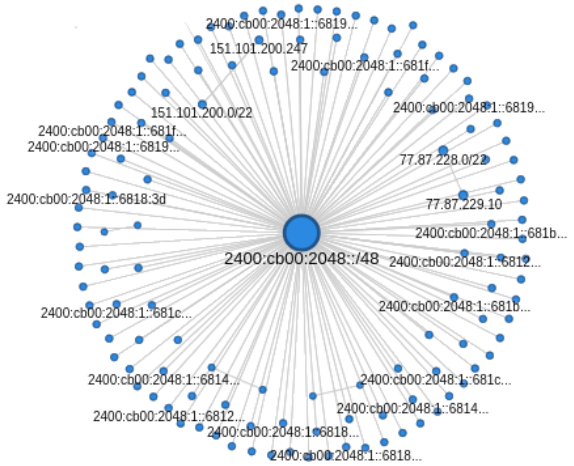
6.3. Graphical Representation

The chosen visualization based on Sankey Diagrams does combine many useful features of widely use visualization types for network and Internet traffic and connections, namely world maps, network graphs and bandwidth charts. But, it also has some of their

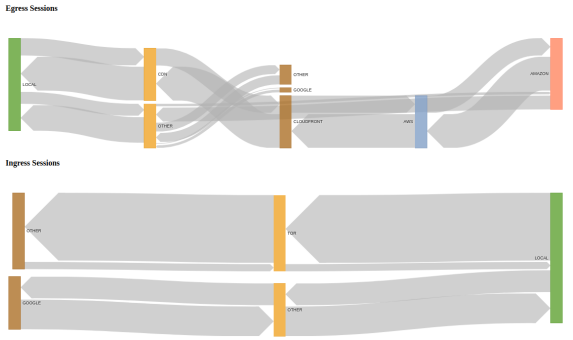
(a) Geo Location Map [22]



(b) Bandwidth Graph



(c) Network Graph



(d) Sankey Diagram

Figure 6.1.: Comparison of different visualization types

drawbacks. In Figure 6.1 examples for these three are shown next to a Sankey diagram from the prototype. When Sankey diagrams are used for connection visualization a user is able to see at a glance where traffic is coming from or going to, in terms of Service and Service-Provider. Further it can be determined what services are the busy talkers and how much traffic they generate in relation to others. It also allows to easily identify unusual ongoing traffic. For example if egress sessions show a large amount of outgoing data to a Cloud Storage Platform like Amazons S3 or even an unknown (OTHER) service which could indicate data leakage.

What is not possible though, is to identify the actual conversation partners and the connection details like ports and protocols involved. Furthermore, does this kind of diagram not show the bandwidth over time, but only a total for all ongoing sessions to or from a certain service. This also means there is no information about past conversations.

The prototype implementation and tests let infer, that it might not scale very well when communications happen to lots of different Services, especially when lots of Service-Providers are involved at the same time. In the worst case nodes could look as overloaded as in Figure 6.1c. Considering that there are bidirectional, weighted edges between them, the user experience could get be even worse. Splitting up egress and ingress sessions into two charts was a good decision as it becomes less overloaded and therefore more usable.

7. Conclusion

Many IT systems and components rely on IP geolocation to block unwanted traffic from certain geographical areas or to visualize ongoing connections or attacks on a map of the world. There also exist many approaches to make IP geolocation more accurate like [5] or [28]. However, there is an increasing risk of false blocks as more large companies run global networks with public IP addresses wandering between different location according to the companies need. If the geolocation database is not updated regularly (for example once a day) IPs could still be assigned to a blocked country even though already been moved away. With more websites using global Content Delivery Networks for DDoS protection and content delivery acceleration this problem becomes even more evident.

As geolocation do not seem to be sufficient in such scenarios, a new approach was suggested to summarize and classify IP addresses. This classification allows distinction of service types and providers of such services. As services on the Internet vary largely in infrastructure, communication protocols, traffic flows, the focus in this work was on identifying CDNs, and ingress traffic from VPN servers and TOR exit nodes. For detection of CDNs an algorithm based on [13] and [31] was described. Furthermore, a data structure was suggest for processing and storing collected and classified data and a possible visualization approach was evaluated.

For testing the suggested algorithm for detection of Content Delivery Networks, a large set of websites has been queried from seven different locations around the globe. Initial analysis of the colleceted data showed a sensitivity of 80% which was not acceptable.Therefore, an additional dynamic threshold was introduced which lead to a sensitivity of about 98%. An analysis of the misclassified two percent showed that it was mostly caused by one service provider which run a CDN based on regional anycast IPs.

For VPN and TOR exit node detection traffic of a test setup was captured and the metadata analyzed. This analysis however showed no significant new data. Already

known VPN patterns have been confirmed and complemented with some additional metrics which might let infer the operating system. For TOR exit nodes a pattern has been found which was only true for about 50% of the tested cases. This pattern showed a direct relation between MSS and initial TCP Window Size. However, it was not possible to identify if it was caused by the operating system the TOR exit node is running on or by some configuration setting.

Furthermore, an extensible prototype was implemented to collect and classify ongoing Internet traffic. Classification done by the prototype was based on the findings of the CDN and metadata analysis and for TOR exit nodes on TORDNSeL. Tests of the prototype showed some problem with classification of traffic originating from VPN servers. It turned out that it was caused by the NAT router as it changed the TCP-Header-Length. All other tested traffic classifications did not show any errors.

Finally, a visualization prototype was implemented which reads live data created by the classification prototype. The chosen visualization type gives well insight on ongoing traffic and allows an easy identification of unusual patterns. But with an increasing concurrent use of many services it gets too overloaded. But in a scenario where only ongoing attacks are shown (similar to IDS/IPS world map) this would possibly not be the case.

Even though the approach for CDN detection works exceptionally well and the suggested visualization does provide a good insight, at least in some scenarios, on the classified connection, there is still a lot of work to do. More service types need to be defined and algorithms to detect them specified. This is also the main drawback of the suggested classification strategy, many of the algorithms might include active probing like for CDN detection, which cannot be performed in real time. Therefore, a database needs to be maintained and regularly updated which could lead to similar problems like with IP geolocation. However, the suggested classifications, evaluated approaches and visualization provide a good starting point for future work.

7.1. Future work

For future steps, approaches need to be found to identify already defined but not yet classified (apart from public available data from AWS and Google) services types like cloud storage and computing. Further, Internet traffic needs to be analyzed and unclassified traffic identified. Based on this identified traffic, new service types could be identified or patterns for existing ones found.

The classified data could also be used to suggest rules for firewalls or IDS/IPS system. For example could egress traffic to an IP address which was previously identified as TOR exit node trigger an alert as it could be considered as unusual behavior. Further could historical data of the classification be used to detect anomalies, like IP addresses which are at one point in time identified as TOR exit node and the next time as something different or even unclassified.

To improve the usability of the visualization prototype, especially with lots of open connections to various services, nodes should be made collapsible and navigable (drill-down).

Independent from the suggested work, could the CDN detection approach be used for easier maintenance of IP white lists in firewall. For example to exclude, certain, CDNs from IP geolocation blocks.

Bibliography

[1] Bahati Airtel. *Bahati Airtelt*. URL: http://www.airtel.in/about-bharti/media-centre/bharti-airtel-news/enterprise/bharti-airtel-and-limelight-networks-announce-strategic-partnership-for-global-content-delivery-network-(cdn)-services (visited on 11/26/2017).

[2] AKAMAI. *Designing DNS for Availability and Resilience against DDoS Attacks*. URL: https://www.akamai.com/us/en/multimedia/documents/white-paper/akamai-designing-dns-for-availability-and-resilience-against-ddos-attacks.pdf (visited on 08/12/2017).

[3] Ahmet Aksoy and Mehmet Hadi Gunes. "Operating System Classification Performance of TCP/IP Protocol Headers." In: *Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on*. IEEE. 2016, pp. 112–120.

[4] AMAZON. *Amazon Route 53 FAQs*. URL: https://aws.amazon.com/route53/faqs/?nc1=h_ls (visited on 08/12/2017).

[5] Mohammed Jubaer Arif, Shanika Karunasekera, and Santosh Kulkarni. "GeoWeight: Internet host geolocation based on a probability model for latency measurements." In: *Proceedings of the Thirty-Third Australasian Conferenc on Computer Science-Volume 102*. Australian Computer Society, Inc. 2010, pp. 89–98.

[6] Jose et. al. Arteaga. *State of the Internet/Security 2017 Q2*. URL: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q2-2017-state-of-the-internet-security-report.pdf (visited on 10/01/2017).

[7] Neil Atkinson. *bihisankey*. URL: https://github.com/Neilos/bihisankey (visited on 11/26/2017).

[8] Robert Beverly. “A robust classifier for passive TCP/IP fingerprinting.” In: *Passive and Active Network Measurement* (2004), pp. 158–167.

[9] Soraya Ait Chellouche, Daniel Négru, Eugen Borcoci, and Eric LeBARS. “Anycast-based context-aware server selection strategy for VoD services.” In: *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*. IEEE. 2010, pp. 1513–1517.

[10] Yi-Chao Chen, Yong Liao, Mario Baldi, Sung-Ju Lee, and Lili Qiu. “OS fingerprinting and tethering detection in mobile networks.” In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM. 2014, pp. 173–180.

[11] Danilo Cicalese, Jordan Augé, Diana Joumblatt, Timur Friedman, and Dario Rossi. “Characterizing IPv4 anycast adoption and deployment.” In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM. 2015, p. 16.

[12] Danilo Cicalese et al. “A fistful of pings: Accurate and lightweight anycast enumeration and geolocation.” In: *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE. 2015, pp. 2776–2784.

[13] Danilo Cicalese et al. “Latency-based anycast geolocation: Algorithms, software, and data sets.” In: *IEEE Journal on Selected Areas in Communications* 34.6 (2016), pp. 1889–1903.

[14] Cloudflare. *The Cloudflare Global Anycast Network*. URL: `https://www.cloudflare.com/network/` (visited on 08/12/2017).

[15] Cloudflare. *The Cloudflare IP Ranges*. URL: `https://www.cloudflare.com/ips/` (visited on 08/12/2017).

[16] EU Council. *General Data Protection Regulation*. URL: `https://www.datenschutz-grundverordnung.eu/wp-content/uploads/2016/04/CONSIL_ST_5419_2016_INIT_EN_TXT.pdf` (visited on 11/26/2017).

[17] Alberto Dainotti, Antonio Pescape, and Kimberly C Claffy. “Issues and future directions in traffic classification.” In: *IEEE network* 26.1 (2012).

[18] Oracle Dyn. *Steering Website Traffic with Managed DNS vs. IP Anycast*. URL: https://dyn.com/blog/steering-website-traffic-with-managed-dns-vs-ip-anycast (visited on 08/12/2017).

[19] Xun Fan, John Heidemann, and Ramesh Govindan. "Evaluating anycast in the domain name system." In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 1681–1689.

[20] Fastly. *Using Fastly with apex domains*. URL: https://docs.fastly.com/guides/basic-configuration/using-fastly-with-apex-domains (visited on 08/17/2017).

[21] Patrick Gilmore. *Serving at the edge: Good for performance, good for mitigating DDoS*. URL: https://blogs.akamai.com/2013/04/serving-at-the-edge-good-for-performance-good-for-mitigating-ddos-part-ii.html (visited on 08/12/2017).

[22] Google. *Google*. URL: https://www.google.com/maps/d/edit?mid=1Q6aK2lmOabuk3Eb638Ex6K4SS64&ll=8.666016771853986%2C-53.76708985000005&z=3 (visited on 11/26/2017).

[23] Google. *Static IP Addresses and App Engine apps*. URL: https://cloud.google.com/appengine/kb/#static-ip (visited on 08/12/2017).

[24] Cynthia Harvey. *Public Cloud Computing Providers*. URL: http://www.datamation.com/cloud-computing/public-cloud-providers.html (visited on 09/08/2017).

[25] Incapsula. *Incapsula Anycast*. URL: https://www.incapsula.com/cdn-guide/route-optimization-anycast.html (visited on 11/26/2017).

[26] Nicola Jarocci. *Eve. The Simple Way to REST*. URL: http://python-eve.org (visited on 11/26/2017).

[27] Andrew Kalafut, Minaxi Gupta, Pairoj Rattadilok, and Pragneshkumar Patel. "Surveying dns wildcard usage among the good, the bad, and the ugly." In: *Security and Privacy in Communication Networks* (2010), pp. 448–465.

[28] Ethan Katz-Bassett et al. “Towards IP geolocation using delay and topology measurements.” In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM. 2006, pp. 71–84.

[29] Yousef Khalidi. *Networking innovations that drive the cloud disruption*. URL: https://azure.microsoft.com/en-us/blog/networking-innovations-that-drive-the-cloud-disruption/ (visited on 09/08/2017).

[30] Robert Koch, Mario Golling, and Gabi Dreo Rodosek. “Advanced geolocation of IP addresses.” In: *International Conference on Communication and Network Security (ICCNS)*. 2013, pp. 1–10.

[31] Doug Madory, Chris Cook, and Kevin Miao. “Who are the anycasters.” In: *Proceedings of NANOG59* 10 (2013).

[32] Majestic. *Majestic Million CSV now free for all, daily.* URL: https://blog.majestic.com/development/majestic-million-csv-daily/ (visited on 08/17/2017).

[33] Rich Miller. *Amazon Building Custom Chips to Accelerate Cloud Networking*. URL: http://datacenterfrontier.com/amazon-building-custom-asic-chips-to-accelerate-cloud-networking/ (visited on 09/08/2017).

[34] P. Mockapetris. *Domain names - concepts and facilities*. STD 13. http://www.rfc-editor.org/rfc/rfc1034.txt. RFC Editor, Nov. 1987.

[35] MongoDB. *MongoDB*. URL: https://www.mongodb.com/ (visited on 11/26/2017).

[36] nagios. *Nagios*. URL: https://exchange.nagios.org/directory/image/678 (visited on 11/26/2017).

[37] Nedi. *Nedi*. URL: https://www.nedi.ch/ (visited on 11/26/2017).

[38] NMAP.ORG. *NPing*. URL: https://nmap.org/nping/ (visited on 10/01/2017).

[39] Sean-Philip Oriyano. “CEH v9: Certified Ethical Hacker Version 9 Study Guide.” In: Sybex, 2016. Chap. Scanning, p. 229. ISBN: 978-1-119-25224-5.

[40] Gerald Ortner. *LIBBGPInfo*. URL: hhttps://bitbucket.org/bgputils/libbgpinfo/wiki/Home (visited on 11/27/2017).

[41] Craig Partridge, Trevor Mendez, and Walter Milliken. *Host Anycasting Service*. RFC 1546. http://www.rfc-editor.org/rfc/rfc1546.txt. RFC Editor, Nov. 1993.

[42] R. Chandramouli, S. Rose. "NIST Special Publication 800-81-2: Secure Domain Name System (DNS) Deployment Guide." In: 2013, pp. 7–6.

[43] Uday Savagaonkar. *Titan in depth: Security in plaintext*. URL: https://cloudplatform.googleblog.com/2017/08/Titan-in-depth-security-in-plaintext.html (visited on 09/08/2017).

[44] Mario Schmidt. "The Sankey diagram in energy and material flow management." In: *Journal of industrial ecology* 12.1 (2008), pp. 82–94.

[45] Verizion digital media services. *CDN performance Faster means better*. URL: https://www.verizondigitalmedia.com/platform/edgecast-cdn/cdn-performance/ (visited on 08/12/2017).

[46] TCPDump. *libpcap*. URL: http://www.tcpdump.org/ (visited on 11/26/2017).

[47] TOR. *Design For A Tor DNS-based Exit List*. URL: https://gitweb.torproject.org/tordnsel.git/tree/doc/torel-design.txt (visited on 08/13/2017).

[48] Wireshark. *Wireshark, Packet capture library (libpcap)*. URL: https://wiki.wireshark.org/libpcap (visited on 11/26/2017).

[49] Michal Zalewski. *p0f v3: passive fingerprinter*. URL: http://lcamtuf.coredump.cx/p0f3/README (visited on 08/13/2017).

Appendices

A. Appendix

- The source code for the classification prototype can be found at `https://bitbucket.org/preciseitsol/connectionclassifier` alongside with a brief description and some hints for installation.
- The source code for the REST service and the web view can be found at `https://bitbucket.org/preciseitsol/connectionclassifierrest`
- Scripts created for processing the collected data are published at `https://bitbucket.org/preciseitsol/script-collection`
- Collected data including pcap files from the network traffic analysis and records from the CDN classification process are available at `https://bitbucket.org/preciseitsol/script-collection/downloads/`. This download also contains data from the several analysis steps.

Europass Curriculum Vitae

Personal information

Surname(s) / First name(s)	**Ortner, Gerald**
Address(es)	Raiffeisenstraße 30/3/7, 4300 St. Valentin, AT
Telephone(s)	0664/4246715
Email(s)	ortnge@gmail.com
Nationality(-ies)	Austria
Date of birth	13.07.1984

Experience

08.2017 - today	Security Consultant, VACE Systemtechnik Gmbh
08.2011 - 08.2017	Customizing Engineer, Meierhofer AG
01.2008 - 07.2011	System Engineer, Netzwerk & Sicherheit, GESPAG
05.2006 - 01.2008	Operator Network & Security, GESPAG
04.2004 - 04.2006	Electrician, Elektrotechnik Pfaffeneder
02.2003 - 04.2004	Electrician, Fa. Truhlar

Education

04.2016 - Today	**Master of Science (MSc.)** Johannes Kepler University Linz Computer Science (Focus: Networks and Security)
10.2012 - 04.2016	**Bachelor of Science (BSc.)** Johannes Kepler University Linz
09.2006 - 07.2010	**Reife- und Diplomprüfung** HTL Leonding HTL für Berufstätige für Informatik (Focus: Software Engineering)
09.1999 - 02.2003	Electrician Apprenticeship Fa. Truhlar

Languages

Mother tongue(s): **German**

Self-assessment *European level*(*)	**Understanding**		**Speaking**		**Writing**
	Listening	Reading	Spoken interaction	Spoken production	
Englisch	C1 Proficient user	C1 Proficient user	B2 Independent user	B2 Independent user	B2 Independent user

Interests

[*] *Common European Framework of Reference (CEF) level*

- IoT
- Elektronic
- Cooking

Publications

- Markus Weninger, Gerald Ortner, Tobias Hahn, Olaf Drümmer und Klaus Miesenberger. "ASVG - Accessible Scalable Vector Graphics: intention trees to make charts more accessible and usable". In: *Journal of Assistive Technologies* 9.4 (2015), S. 239–246. DOI: 10.1108/JAT-10-2015-0027. eprint: http://dx.doi.org/10.1108/JAT-10-2015-0027. URL: http://dx.doi.org/10.1108/JAT-10-2015-0027

Statutory Declaration I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references. This printed thesis is identical with the electronic version submitted.

Eidesstattliche Erklärung Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Gerald Ortner, BSc	Date