# Analyzing a year of QUIC data

Paul-Vincent Kobow

HAW Hamburg, Hamburg Germany

**Abstract.** This work takes a closer look at QUIC traffic collected by Spoki over one year in a German IPv4 address space. We filtered out invalid packets, identified where connections came from, and tried to decode as much payload as possible. Most of the traffic came from just two sources—CENSYS and Alibaba—both running large-scale scanners. The decodable payloads were almost always DNS over QUIC requests. During this project, we added IPv6 support to Spoki. This required some changes to how packets are handled, especially because IPv6 has different rules for checksums. With these adjustments, Spoki can now process QUIC traffic over both IPv4 and IPv6. This sets the stage for future measurements as QUIC adoption grows and more devices support IPv6.

**Keywords:** QUIC · Reactive network telescopes · Internet background radiation · HTTP/3

## 1 Introduction

Based on the results of previous work [1] multiple questions could not be answered to the fullest or not at all. To address these questions, this paper will try to dive deeper into the usage of Quick UDP Internet Connections (QUIC) within the IPv4 address space, use more data for an improved view through the network telescope Spoki and adapt Spoki for the use in the IPv6 address space.

We will analyze the usage patterns and characteristics of QUIC traffic observed over a year-long period in a German IPv4 network. For this we will firstly filter the collected data to remove invalid packets, secondly we want to figure out the ownership of different IP's using the Registration Data Access Protocol (RDAP), in order to better identify traffic from scanners and research organizations, which can then be filtered out to allow for a focus on malicious or improper use of the QUIC protocol. We also want to decode traffic that we were not able to decode in my prior work [1]. This might allow for an in-depth view of the usage of QUIC and maybe even find some malicious intents.

Lastly, we want to look at the QUIC traffic over time to better analyze trends or temporal phenomena contained within the given dataset.

For a better understanding of this paper without context, we will first reintroduce some relevant keywords and principles.

## 2 A quick recap on QUIC and (reactive) Network Telescopes

Network telescopes are passive monitoring systems that observe unused IP address spaces to capture Internet Background Radiation (IBR), which includes unsolicited traffic such as scans, backscatter, or misconfigurations. Reactive network telescopes, like Spoki, extend this capability by responding to incoming traffic, enabling deeper analysis of protocols and interactions [2].

QUIC is a modern transport protocol designed to reduce latency by combining encryption and multiplexing over UDP. It serves as the foundation for HTTP/3 and DNS over QUIC (DoQ), among other applications. Unlike TCP, QUIC integrates security and reliability directly into the transport layer, making it faster and more resilient to network changes.

### 2.1 Spoki

Spoki, the reactive network telescope used in this study, was created in 2022 to investigate the use of irregular Transmission Control Protocol (TCP) handshakes used by various scanners to engage in two stage scanning of the whole IPv4 address space [2].

For this use case Spoki is required to scale to large networks and thus cannot use the default posix implementation of the TCP stack. Instead Spoki will attach directly to a network interface and process all incoming bytes itself. Incoming traffic is distributed among multiple Shards which receive data based on consistent hashing of IP-Packets. Each Shard will decide how to process packets and will forward information that will eventually allow Spoki to react to incoming packets using Scamper [2].

This Architecture allows for a relatively easy implementation of new protocols. Spoki is already able to process Internet Control Message Protocol (ICMP) and User Datagram Protocol (UDP) packets.

Due to the architecture of Spoki we cannot send out packets from the same process we received them on. Instead a dedicated process is used to send out requests by spoofing the source address.

Since QUIC is a rather complex protocol, compared to TCP and UDP we would like to leverage most of the implementation to existing libraries. The one we opted for is quiche, created by cloudflare and written in rust [3].

As we do not want to hand over every UDP package to quiche due to performance reasons, we need to determine wether a UDP packet contains QUIC or not. This is done using a heuristic, which is checking various aspects of a UDP packet to determine if it could be QUIC. The heuristic used to identify QUIC packets checks header type, version, and the lengths of source and destination connection IDs. This approach is prone to both false positives and negatives, but it is necessary to avoid passing every UDP packet to the quiche library. Given the volume of background traffic, this trade-off is acceptable for reducing server load and maintaining scalability .

When a UDP packet is classified as QUIC, Spoki uses the quiche library to perform version negotiation if needed. If version negotiation is not required, Spoki completes the QUIC handshake and records the resulting payload. As Spoki is designed as a reactive telescope, it does not implement full application-layer protocol emulation for HTTP or DNS and focuses on efficient capture at the transport layer.

Received data is stored in multiple ways. All packets are stored in rolling csv files and pcap files. For the QUIC implementation we also store keylog and qlog files. Keylog files are a standardized way to store the secrets used for the TLS handshake in order to later decrypt data. Qlog files are a specialized for storing information on QUIC connections, such as timing information and packet sizes.

## 3 Analyzing the data

We have gathered IPv4 data of a 24 address space located in Germany. The first of all 3.877.034 packets was received at around 33 seconds after midnight on the 1. of January 2024, and the last around 33 seconds before midnight on the 31. of December 2024. It contained traffic from 13.488 different IP addresses.

Of all received packets that have been identified as QUIC by the heuristic used, Spoki was not able to handle 28%. Those are marked with "error", which simply indicates that either the used quiche library was not able to parse an incoming packet or the Packet did not match the protocol flow defined by the RFC 9000 [4].

Whilst analyzing packets that got marked with the "error" state, we noticed that the reason for the error of 37% of those packets was QUICHE_ERR_BUFFER_TOO_SHORT, which might indicate a suboptimal configuration of Spoki.

Nearly all of the observed exceptions were caused by invalid QUIC packets, which are likely not intended to carry valid QUIC data. Approximately 36% of all error packets result from the use of a short header packet to initiate a connection—an action that is not permitted under the QUIC specification [4]. These packets are likely back-scatter or IBR. This would be in line with [5].

Similar malformed packets attempting connection initiation have also been observed. Excluding these packets reduces the overall error rate to 6%, with the remaining errors primarily attributable to TLS or cryptographic failures.

Around 2.7% of all incoming packets contained a payload that can be further analyzed, which will happen in subsection 3.4. Other reactions sent out by Spoki can be observed in Figure 3.

Just like in my previous work [1], packets marked with "error" will be filtered out and are ignored from here on. Whenever we mention "all" packets, we mean all packets except the ones marked with "error".

Doing so reduces the total number of different IP addresses to 581.

Looking at the temporal distribution of incoming packets, we can observe a gradual increase from January until September, when the amount of traffic

suddenly increased drastically, as shown by Figure 2, due to more packets from CENSYS and Alibaba.

### 3.1 IP Origin

In contrast to our first work [1] we have switched to RDAP for IP ownership resolution. This is the successor of the WHOIS protocol and is better maintained. Thus, we were able to resolve the owners and origins of nearly all IP addresses.

In total, traffic originated from 98 different Autonomous Systems (ASs) located in 31 different countries all over the world, as depicted by Figure 1.

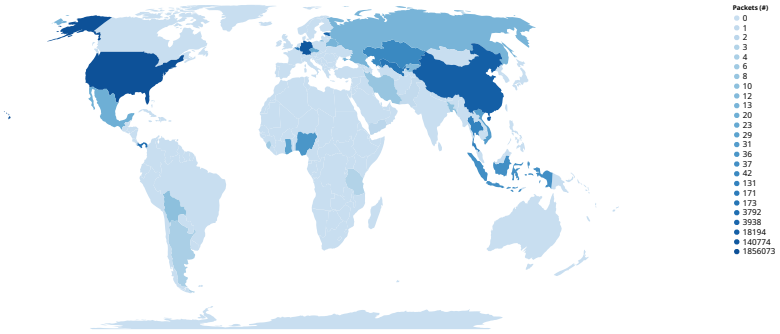


**Fig. 1.** Received Packets per AS origin

Looking more closely at this, we can observe that most traffic has its origin at one of two organizations. Most often, traffic originated from an AS belonging to CENSYS (47%). Following shortly after, Alibaba with 45% traffic share. These two created the most traffic, of which most was received after September 2024. The third-biggest contributor to the total number of packets was the TU München with just around 5% of total traffic share. This can be viewed in Figure 4.

Investigating the time each IP was active, depicted in Figure 8, we can observe that the average time was around 137 days, whilst the median was at 123 days, indicating single IPs being active for an extended period of time. Active time is defined as the time between the first received packet and the last received packet, no matter if data was sent in-between those timestamps. 18 IPs were active during the whole year of 2024 meaning the first packet was received on the first of January and the last on the 31. of December.

Looking at the distribution of Active Days, we can see that Alibaba seems to rotate IPs every $\approx 100$ days, as this is the time most IPs by Alibaba are active. Interestingly, some IPs of Alibaba are even active for 226 days. In Contrast, CENSYS does not seem to enforce a new IP lease after a given amount of time, as the active time is much more distributed, but most IPs of CENSYS are active for more than 144 days.

### 3.2 The data without scanners

In line with [5], the data collected by network telescopes is currently dominated by research scanners. Thus we want to further inspect our collected data without these overshadowing possible other insights.

Based on superficial research we did on the 98 different ASs, we will define all ASs mentioned in Table 1 as Scanners used for research or educational Purposes.

Removing all packets received with an origin at one of these ASs, leaves us with just 1.315.397 packets. Of those, 49.442 even contained a collected payload. All packets with a payload contained a DoQ request with either GET or POST and originated at an IP address belonging to Alibaba. In [1] we were able to find a WHOIS entry belonging to ki3, a china based internet measurement company. By switching to RDAP as our source of information, we are no longer able to find any IP address belonging to ki3 directly. They might be part of some AS belonging to Alibaba.

Observing the time and date these packets were received, as done in Figure 6, we notice that only CARINET has sent out packets all year long. CARINET is a US based cloud provider. The constant traffic received might indicate a running scanner. This is also proposed by [6]. Since CARINET is a hosting provider, we currently don't know if the scanner is malicious or for educational purposes, and thus have not added it to the list of scanners for research purposes.

On the 12th of May, another scanner became active, after an initial test phase during March. It has an IP belonging to the AS of Alibaba and has drastically increased traffic in September. The purpose of this traffic is currently unknown.

### 3.3 Outliers

**High TTL's** During the writing of [1] we noticed multiple packets with a Time-to-Live (TTL) of more than 128. These packets also appear in the larger dataset used this time. Most of these packets originate from the scanners defined in subsection 3.2. This is depicted in Table 3. In contrast to the overall QUIC traffic increasing over the year, the traffic with a TTL of more than 128 decreases, as visualized by Figure 7.

Filtering out the scanners identified in subsection 3.2, we can get a more detailed view on comparable small ASs. Most packets originate from an AS belonging to the "Private Layer INC" company. The company itself is a Swiss hosting provider, but the AS is registered in Panama. The second most packets are received from China. Other sources contain the Tencent Company, a German hosting provider (CONTABO) and the oracle cloud.

It is still unclear why the TTL of those packets is set to such a high value.

**IPs with Lots of packets** Traffic originated at lots of different IPs, but some IPs were the source of especially many packets. As depicted by Figure 5 a single IP address contributed nearly 14% of all packets. Four different IPS contributed around 5% to the total number of packets each, which is a lot, considering the average IP only contributed around 0.2% of packets.

The top IP address belongs to Alibaba, as do all other mentioned IPs except for the IP address responsible for the second most packets, which belongs to TU München. The TU München is actively scanning for QUIC since 2021 [7].

### 3.4 Analyzing Collected Payloads

Since the writing of our first paper on this topic, we were able to decrypt further payloads. In our first paper, we were only able to decrypt some misleading payloads that contained regular HTTP/1.1 requests sent over UDP. These payloads still make up around 45% of all received payloads.

By using stored "keylog" files to decrypt available payloads, another 19% of all collected payloads can now be analyzed, combining to a total of 64% of explainable payloads. A Keylog file is a file containing all secrets used during a single SSL session. These files can be combined to decode multiple Sessions.

As QUIC is using QPACK to efficiently compress headers for HTTP, we also had to decode the now decrypted payloads. QPACK is a version of HPACK implemented to support the out of order delivery of packets, that QUIC supports [8]. Most packets were easy to decode as QPACK uses a static header table to encode common headers. There are however more complex QPACK payloads making use of dynamic tables [8]. This hindered me on decoding all payloads, so only 84% of the decrypted payloads could be decoded.

For the decoding we used the python library p̈ylsqpack, which is a Python-wrapper for the C-library l̈s-qpack. The ls-qpack library is supposed to be able to decode even dynamic tables, but we were not able to get this to work. Further investigation is required.

**HTTP over UDP** Over the whole year of 2024 we have received 38.193 packets that contained a HTTP payload delivered via UDP. All of those requests were GET requests to the "/" URI. Since this work is on QUIC we will not go into further detail.

**Decoded QUIC traffic** Of all decrypted and decoded payloads (including HTTP over UDP), around 50% were actual DoQ requests. Most of those ($\approx 87\%$) tried to do a GET request on the path "/dns-query". Another 12% tried a POST request on the same URI. All of these DoQ Requests originated at Alibaba.

Other QUIC Connections established an HTTP/3 Connection, of which all were HEAD requests to the "/" URI. This can be viewed in Table 2. All HEAD requests originated at the TU München, which only sent HEAD requests. As HEAD requests are often used as a preflight check to avoid large, unnecessary downloads, the given response created by Spoki might not trick the scanner into completing another request and thus no other request methods have been recorded.

Both GET and POST requests originated at IPs belonging to Alibaba, which, as already mentioned, might belong to the internet research company ki3.

Interestingly, even though the traffic originated from different sources, all requests used a user agent that was either "quic-go HTTP/3" or "quic-go-HTTP/3", as seen in Table 2.

"quic-go HTTP/3" matches the default user agent of the well known QUIC implementation in go "quic-go" [9]. We were not able to figure out the software accountable for other user agent "quic-go-HTTP/3", but are assuming an older version of the same library.

The TU München only used the "quic-go-HTTP/3" user agent. Requests from Alibaba used "quic-go HTTP/3".

All received queries tried to fetch the "A" Record of either "www.example.com" or "example.com", as observable in Table 4.

## 4 Conclusion, Outlook and future work

Currently only two organizations seem to have some kind of always-active QUIC scanner in place. Both seem to have either an educational or a monitoring purpose, as they constantly send out the same requests, suggesting a standardized testing rather than any malicious intent.

We were not able to find any other use of the QUIC protocol in our data. This however might not be true as we still have more than 30% of unexplained payload packets, which might hide malicious activity. The absence of other activity might correlate with the built-in security standards by QUIC such as encryption and amplification limits.

Monitored QUIC traffic mostly originated in the US, China, and Germany reflecting the distribution of QUIC-enabled infrastructure, which however is just based off data from Germany. A worldwide deployment of Spoki might lead to different results.

As there are still many payloads that we were not able to decode, future work might focus on improving the decoder script as well as the collection of keys used to decrypt payloads.

### 4.1 IPv6 Integration

In a recent paper by TUM, the QUIC deployment of various libraries was analyzed using a scanner in both the IPv4 and IPv6 address space [10]. By using a IPv6-hitlist around 7.9 million hosts supporting QUIC where found, which indicates that the deployment of Spoki to the IPv6 address space might gain valuable insights. Most of these addresses belong to large Content Delivery Networks (CDNs) and thus configuration issues that would cause traffic to be redirected to a Spoki probe are rather unlikely.

To confirm this hypothesis we are working on an integration of QUIC for IPv6 to Spoki. With some slight modifications to the used C++ struct, Spoki was able to process IPv6 packets. Currently the posix socket API has slight variations between IPv4 and IPv6. These differences force us to handcraft the IPv6 ethernet headers ourself. The information for the next router is determined by the *nexthop* table. This however causes problems on certain systems as the table does not always contain all known addresses. For this deployment this doesn't cause any issues, but later deployments will need to check this behavior.

The quiche library is already able to handle IPv6 Packets and thus only slight modifications are required. One major difference is the use of the UDP checksum. The IPv4 IP-Header already contains a checksum for each IP-Packet. IPv6 in contrast does not. Due to ever increasing link-speeds the validation of checksum is becoming a major overhead if checked with software. Thus most Ethernet cards implement the validation in hardware. This obfuscated the problem. Multiple Packets were received by the connected client but simply ignored. Others got accepted. But after introducing some tests for this functionality the issue is now fixed and Spoki is now able to respond to incoming IPv6 connections with valid UDP packets.

At the time of writing this paper Spoki with IPv6 support is deployed to the same host that was used to gather the data analyzed in this paper, but newly gathered data has not yet been reviewed.

### 4.2 Outlook

Most packets Spoki has received are not from Germany, where the server is currently located. Thus, it might be interesting to deploy Spoki on different servers around the world.

Due to the lack of interesting traffic, it might be interesting to repeat this experiment in a couple of years, when QUIC has become more wildly adapted.

While QUIC's current use appears benign and scanner-dominated, its growing adoption (especially with HTTP/3) necessitates continued monitoring. This study provides a baseline for longitudinal comparisons, particularly as QUIC becomes a larger attack surface. Reactive telescopes like Spoki, coupled with refined methodologies, will remain vital for tracking the protocol's evolution and security implications.

## References

[1] P.-V. Kobow, *QUIC in reactive network telescopes*, Aug. 30, 2024.

[2] R. Hiesgen, M. Nawrocki, A. King, A. Dainotti, T. C. Schmidt, and M. Wählisch, "Spoki: Unveiling a new wave of scanners through a reactive network telescope," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022, pp. 431–448, ISBN: 978-1-939133-31-1. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/hiesgen.

[3] *Cloudflare/quiche*, version v0.24.0. [Online]. Available: https://github.com/cloudflare/quiche (visited on 08/28/2024).

[4] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," Internet Engineering Task Force, Request for Comments RFC 9000. DOI: 10.17487/RFC9000.

[5] M. Nawrocki, R. Hiesgen, T. C. Schmidt, and M. Wählisch, "QUICsand: Quantifying QUIC reconnaissance scans and DoS flooding events," in *Proceedings of the 21st ACM Internet Measurement Conference*, Nov. 2, 2021. DOI: 10.1145/3487552.3487840.

[6] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," DOI: 10.5555/2671225.2671230.

[7] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle, "It's over 9000: Analyzing early QUIC deployments with the standardization on the horizon," in *Proceedings of the 21st ACM Internet Measurement Conference*, Virtual Event: ACM, Nov. 2, 2021, pp. 261–275. DOI: 10.1145/3487552.3487826.

[8] C. '. Krasic, M. Bishop, and A. Frindell, "QPACK: Field compression for HTTP/3," Internet Engineering Task Force, Request for Comments RFC 9204, Jun. 2022. DOI: 10.17487/RFC9204.

[9] M. Seemann, *A QUIC implementation in pure go*, version v0.52.0. [Online]. Available: https://github.com/quic-go/quic-go (visited on 03/30/2025).

[10] J. Zirngibl, F. Gebauer, P. Sattler, M. Sosnowski, and G. Carle, "QUIC hunter: Finding QUIC deployments and identifying server libraries across the internet." DOI: 10.1007/978-3-031-56252-5_13.

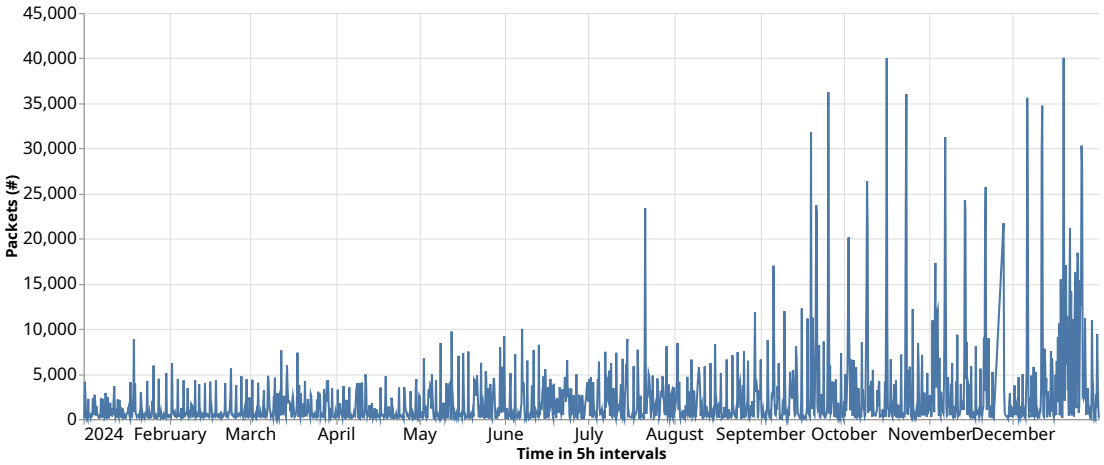


**Fig. 2.** Received Packets without Packets marked as "error" by spoki over the year 2024

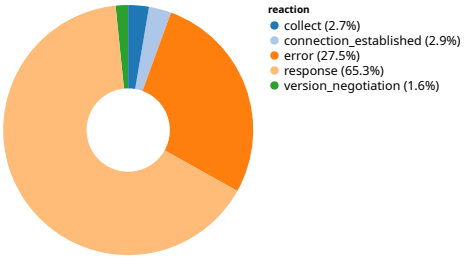


**Fig. 3.** Reactions by Spoki to incoming Packets

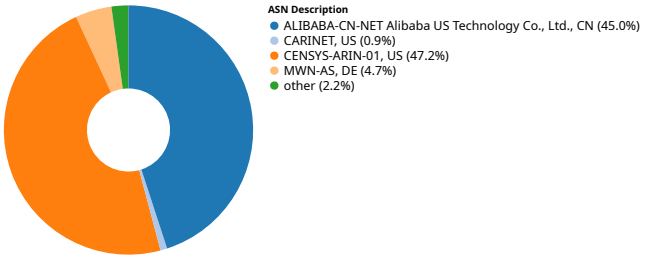


**Fig. 4.** Received Packets per AS Description

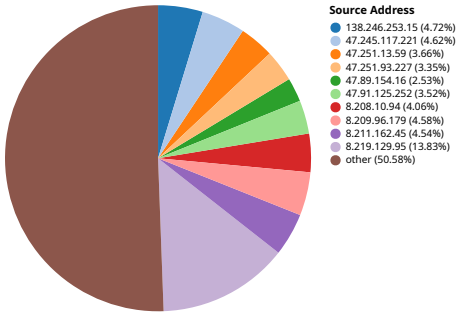


**Fig. 5.** Received Packets per Source Address

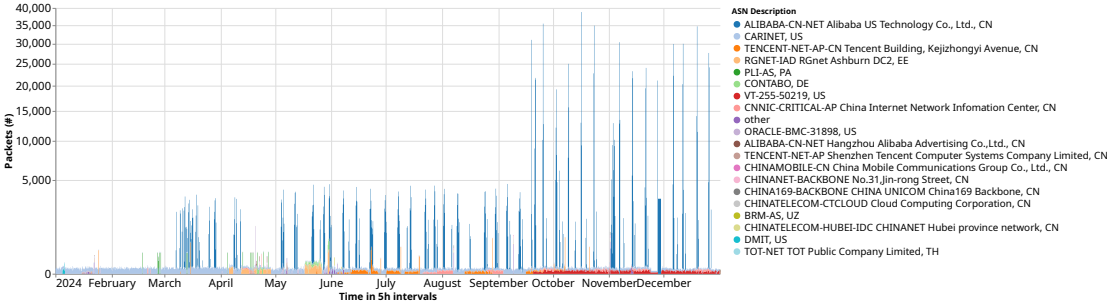


**Fig. 6.** Received Packets per AS over time, without identified scanners

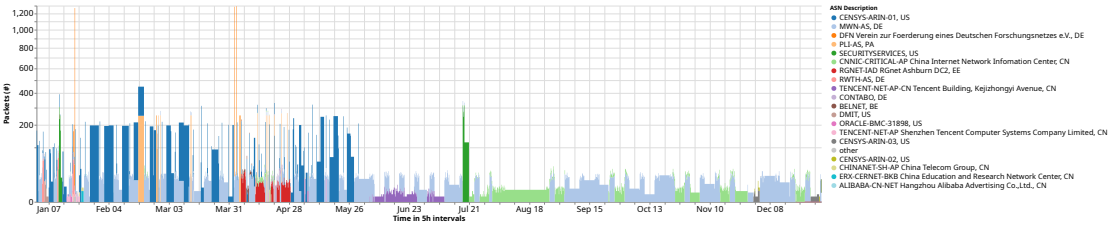


**Fig. 7.** Received Packets with a TTL $>$ 128 per AS over time

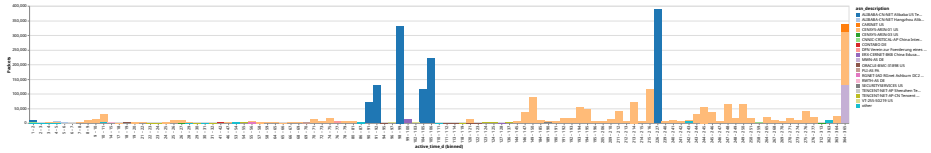

**Fig. 8.** Active Time of all Source Addresses, colored by their according AS

| ASN(s) | ASN Description | Country | Comment |
|---|---|---|---|
| 12816 | MWN-AS | DE | TU München |
| 680 | DFN Verein zur Foerderung eines Deutschen Forschungsnetzes e.V. | DE | |
| 398722, 398705, 398324 | CENSYS-ARIN | US | Censys Inc Internet Scanner for Thread Prevention |
| 4538 | ERX-CERNET-BKB China Education and Research Network Center | CN | |
| 19905 | SECURITY SERVICES | US | Arbor Internet Scanner |
| 2611 | BELNET | BE | Belgian Research Network similar to DFN |
| 47610 | RWTH-AS | DE | Aachen University |

**Table 1.** Scanners used for research or educational purposes

**Table 2.** DoQ requests per AS

| AS Description | method | path | content-type | user-agent | Protocol | Packets (#) | Share (%) |
|---|---|---|---|---|---|---|---|
| CENSYS-ARIN-01 US | GET | / | | | HTTP over UDP | 348631 | 41.06 |
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | GET | /dns-query | application/dns-json | quic-go HTTP/3 | QUIC | 190420 | 22.43 |
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | GET | /dns-query | application/dns-message | quic-go HTTP/3 | QUIC | 180896 | 21.31 |
| MWN-AS DE | HEAD | / | | quic-go-HTTP/3 | QUIC | 69979 | 8.24 |
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | POST | /dns-query | application/dns-message | quic-go HTTP/3 | QUIC | 55238 | 6.51 |
| Unknown | GET | / | | | HTTP over UDP | 3495 | 0.41 |
| CENSYS-ARIN-03 US | GET | / | | | HTTP over UDP | 242 | 0.03 |
| CENSYS-ARIN-02 US | GET | / | | | HTTP over UDP | 80 | 0.01 |

**Table 3.** Packets with a TTL > 128 per AS

| AS Description | Packets (#) | Share (%) |
|---|---|---|
| CENSYS-ARIN-01 US | 14843 | 35.66 |
| MWN-AS DE | 12148 | 29.18 |
| DFN Verein zur Foerderung eines Deutschen Forschungsnetzes e.V. DE | 4054 | 9.74 |
| PLI-AS PA | 3792 | 9.11 |
| SECURITYSERVICES US | 2286 | 5.49 |
| CNNIC-CRITICAL-AP China Internet Network Infomation Center CN | 1859 | 4.47 |
| RGNET-IAD RGnet Ashburn DC2 EE | 902 | 2.17 |
| RWTH-AS DE | 761 | 1.83 |
| TENCENT-NET-AP-CN Tencent Building Kejizhongyi Avenue CN | 278 | 0.67 |
| CONTABO DE | 254 | 0.61 |
| BELNET BE | 171 | 0.41 |
| ORACLE-BMC-31898 US | 70 | 0.17 |
| DMIT US | 70 | 0.17 |
| TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited CN | 37 | 0.09 |
| CENSYS-ARIN-03 US | 28 | 0.07 |
| CENSYS-ARIN-02 US | 17 | 0.04 |
| CHINANET-SH-AP China Telecom Group CN | 15 | 0.04 |
| ERX-CERNET-BKB China Education and Research Network Center CN | 10 | 0.02 |
| ALIBABA-CN-NET Hangzhou Alibaba Advertising Co. Ltd. CN | 6 | 0.01 |
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | 5 | 0.01 |
| UNICOM-GUANGZHOU-IDC China Unicom Guangdong IP network CN | 5 | 0.01 |
| CHINANET-IDC-BJ-AP IDC China Telecommunications Corporation CN | 4 | 0.01 |
| CMNET-JIANGSU-AP China Mobile communications corporation CN | 3 | 0.01 |
| CHINANET-SCIDC-AS-AP CHINANET SiChuan Telecom Internet Data Center CN | 3 | 0.01 |
| HETZNER-AS DE | 2 | 0.0 |
| AMAZON-02 US | 2 | 0.0 |

**Table 4.** Queried DNS Entries

| AS Description | Queried DNS Entry | Packets (#) | Share (%) |
|---|---|---|---|
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | example.com | 334156 | 92.01 |
| ALIBABA-CN-NET Alibaba US Technology Co. Ltd. CN | www.example.com | 29010 | 7.99 |