QUIC in reactive network telescopes

Paul-Vincent Kobow

HAW Hamburg, Hamburg Germany

Abstract. This paper investigates the use of QUIC traffic from the view of a reactive network telescope. Network telescopes have long been used to monitor Internet Background Radiation. As QUIC becomes increasingly prevalent due to its adoption in HTTP/3, and other applications, protocols, reactive network telescopes must adapt to effectively analyze QUIC traffic. This study focuses on explaining difficulties arising from implementing QUIC within Spoki, a reactive network telescope that provides low-resource, scalable responses, as well as analyzing collected data. By analyzing data collected over a month, we explore the characteristics and distribution of QUIC traffic on the Internet. The results demonstrate the feasibility and value of incorporating QUIC analysis in network monitoring for better visibility into emerging Internet traffic trends.

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

1 Introduction

Network telescopes are a way to monitor a portion of the Internet. With the use of such a telescope, researchers can monitor rogue packets, analyze bot traffic or the expansion of worms [1]. They have been in use for a long time [2]. In recent years Spoki was released, a reactive network telescope able to respond to incoming traffic in an efficient and sacalable manner. With the increasing deployment[3] of Quick UDP Internet Connections (QUIC)[4], a new network protocol designed to be fast and secure, spoki is required to be able to understand QUIC traffic to continue analysis of the Internet.

In this paper, we want to give an introduction to using QUIC in reactive network telescopes as well as a first analysis of collected data.

2 Network telescopes

The term network telescope is derived from an optical telescope. Similarly to an optical telescope, a network telescope is used to collect data to measure Internet traffic. This data can then be further processed to generate a detailed analysis of ominous traffic. Just like an optical telescope, a traditional network telescope is only receiving data. It will not react to incoming traffic.

The traffic received by a telescope is called Internet Background Radiation (IBR) because the telescope belongs to an unused IP address Space in which no legitimate traffic exists [5]. Thus, the received traffic belongs to different activities on the Internet.

Among these activities are Internet-Scanners, which often scan the whole IPv4 address space in a matter of minutes to identify hosts and applications. Another example of monitored IBR contains response packets of denial-of-service (DoS) attacks. These packets are called backscatter and are received, because the sender of a DoS packet does not want to receive a response and thus spoofs an IP packet with a random IP address.

Other packets received by a network telescope contain attempts of malware to spread to new systems [1] or are just received due to misconfigurations.

Some well known telescopes include the UCSD network telescope, maintained by CAIDA which concludes a whole /9 and /10 network[1], as well as the Orion network telescope, which contains 1856 /24 networks [6].

Network telescopes cannot monitor the whole Internet. Collected data will always be just a representation of the whole network and thus may not represent what is actually happening across the Internet. This is especially true, if the whole network is routed to a single location. Nationwide Firewalls and other interferences will influence what can be received [7].

Since most traffic on the Internet is connection oriented, meaning a connection has to be established and (in most cases) secured before any application data is sent, a passive network telescope will likely receive more initial packets than packets containing application data. There are however multiple exceptions, for example when dealing with backscatter, which are packets that already contain a response and thus valuable data. This is especially true for the Transmission Control Protocol (TCP). Most received TCP data consists of SYN packets, used to initialize a new connection [2][8].

Since a regular network telescope is designed to only receive, but never send traffic it is often not able to determine the reason a packet was sent, which is a rather interesting information for research. In order to solve this issue, a reactive network telescope was designed.

2.1 Reactive Network Telescopes

The idea of responding to traffic instead of only capturing it has been around for a long time. First implementations of so-called honeypots emerged in the late 90s[9] to early 2000s[2]. These implementations however are not scalable since they operate in a stateful manner on the application layer. To overcome this issue, one of the first scalable implementations has been created by Pang et al. in 2004 [2]. By applying rather strict filters to incoming traffic, Pang et al. were able to reduce traffic to a minimum. This traffic was then handled by a honeypot implementation

If we want to reply to all traffic (worth replying to), we need a method that is less resource intensive and yet still scalable to large networks. This can be accomplished by responding at the transport layer. This is known as a reactive network telescope.

In contrast to a honeypot, reactive network telescope works on the transport layer of the OSI-model. Honeypots not only reply with the correct structure to a request but also with the correct content [2]. A reactive network telescope will not try to imitate a certain application, but instead will reply with generic responses to traffic [8].

Since many protocols are stateful, replying to each request in a correct manner would require a lot of memory to store the state, so we need to reduce implementations of network protocols to a bare minimum and somehow make them stateless or at least reduce the required state to as little as possible.

The first implementation to fulfill this requirement is Spoki. Spoki is a reactive network telescope developed at HAW [8].

2.2 Spoki

Spoki, released in 2021, has proven to enable researchers to gather valuable insights into two- or multi-stage TCP scans. Originally designed to react to TCP SYN packets and force a stateless Scanner to further interact with Spoki, the codebase is flexible enough to also support different protocols such as User Datagram Protocol (UDP) or ICMP, which however do not require a response and are thus just monitored.

UDP is especially easy to implement since at the transport-layer there is no need for a response. This however changes if you implement a protocol that is mixing OSI-layers, such as QUIC.

3 QUIC

Most of the traffic on the Internet is based on HTTP [3]. HTTP (up until version 2) however, is based on TCP, which is a stateful protocol that requires a Handshake before sending any application data. This causes an additional latency for receiving data, but is required for an application protocol such as HTTP. To overcome this issue, a new transport protocol is required that is fast, secure and reliable.

As pointed out by Rosenberg in 2008 if you want reliable transport between any two endpoints on the Internet you need to use TCP, UDP or ICMP due to the way that Firewalls and NAT have evolved over time to improve performance of these protocols [10]. Thus, creating and implementing a new transport-layer protocol to quickly and reliable transport data over the Internet is a non-trivial problem.

QUIC is a UDP based protocol designed for low latency connection establishment as well as secure and reliable data transportation. By merging encryption with some TCP capabilities such as congestion control and retries, QUIC is able to reimplement certain aspects of communication with extremely high efficiency [4]. This is visualized by Figure 5. It is the transport-layer protocol for HTTP/3, DNS over QUIC and other protocols.

3.1 Amplification Problems in QUIC

Amplification in network traffic describes a behavior often used for malicious activities. If a server is receiving a rather small packet, but responding with a rather large response, an attacker is able to spoof the response address and thus amplify the traffic by using an arbitrary server to create more traffic to DOS some target.

This issue is even covered in the RFC for the QUIC protocol. As per the RFC, any QUIC implementation has to limit the amplification to a factor of three [4].

In reality, however, many implementations of large providers seem to not follow this requirement and allow for an amplification of up to 45x [11].

Whilst a network telescope might be able to observe the results of such an amplification attack, it is not able to observe how such an attack is initiated. A reactive network telescope might be able to inspect this behavior.

3.2 QUIC in Spoki

Since QUIC is a rather complex protocol, implementing it ourselves would take a great amount of resources. Thus, we would like to rely on existing implementations, if these are able to cope with the extreme performance requirements of Spoki. There are multiple implementations for the QUIC protocol, for example *Aioquic*, *quic-go* and *quiche*. Due to the way a network telescope work, it cannot send any data vie the listening interfaces. Spoki uses additional hosts, that spoof the sender IP and is then able to send data. Because of this, we need to be able to send UDP the packet ourselves. The only library capable of this at the time of implementing Spoki was quiche. Quiche is an QUIC implementation by Cloudflare written in Rust [12]. There are existing C APIs, so we can easily use quiche within Spoki, which is written in C++.

Since Spoki is a reactive network telescope, it works on the transport layer, but QUIC is UDP based, so we need a way to determine if a packet is QUIC or some other packet using the UDP protocol. To solve this issue, we need a heuristic that will most likely determine if a packet is QUIC or not. The implemented heuristic will apply a few simple filters to the raw byte data. First, we check that the packet has an adequate size (larger than 9 bytes). We then check the QUIC header flags to check whether the received packet is supposed to be a short header. If it is, a packet is required to have a size of more than 19 bytes, some fixed bits in the header and a valid Destination Connection Identifier (DCID). Otherwise, we check that the bytes that are supposed to specify the version have reasonable values. Afterward, we check the DCID and Source Connection Identifier (SCID) length. Both of them are supposed to be less than 20 bytes. If all of the above checks are successful, we consider the packet to belong to the QUIC Protocol.

This, however, is just a heuristic, so some of the gathered data might be labeled incorrectly or missing certain requests. To improve this heuristic, we also assume that every UDP packet received on port 443, the default port for TLS encrypted HTTP traffic, is also QUIC.

In the next chapter, we want to further process collected data to gain some insights into the current use of QUIC from the perspective of a reactive network telescope.

4 Analysis of Traffic collected by Spoki

As part of this work, we were able to gather data throughout May of a European /24 routed networks. This concluded to a total amount of 265.571 packets, identified by the QUIC Heuristic. Traffic was received mostly equally distributed over all 256 addresses, except for one address (...61/32) that received an unproportionate amount of traffic. This can be viewed in Figure 7. The data used to create the figure still contains packets that have been flagged as "error" by Spoki. Packets will get flagged with "error" if they are either invalid QUIC packets or could not be processed due to some other reason. We can use this flag set by Spoki to filter all received packets for those that do not contain the "error" flag.

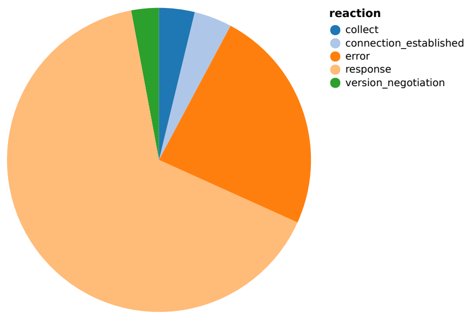

Fig. 1. Distribution of Spoki reactions in unfiltered data

As displayed in Figure 1 these "error" packets amount to around 24% of the total traffic.

Interestingly, most of the "error" packets were received at a single IP (...16/32) of the telescope. It is the same IP depicted in Figure 7 that received an unproportionate amount of traffic. Removing the "error" packets also causes the distribution of destination addresses to become equally distributed, as displayed in Figure 8.

"Error" packets originate from a variety of IPs, as visualized in Figure 9. One IP however is accountable for around 22% of all "error" packets. The IP belongs to China-Based Internet Measurement Company named *Ki3*.

From this point onward, we remove all packets that are flagged as "error" by Spoki. We will call this data filtered data.

As displayed in Figure 2 only four different ports received actual QUIC traffic.

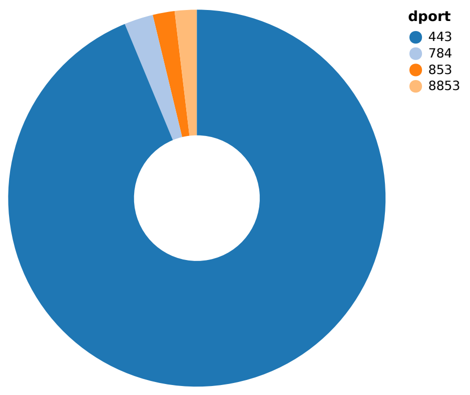

Fig. 2. Share of received packets in filtered data per destination port

- Port 443 is the default port for TLS encrypted HTTP traffic. It is also the default port for HTTP/3 (Http over QUIC).
- Port 784 was used in early drafts as the default port for DNS over QUIC. It was later switched to Port 853 [13]
- Port 853 is the default Port for DNS over QUIC
- Port 8853 is not officially assigned, but is often use as an alternative port to serve DNS over QUIC

Most of the traffic is received on port 443.

When looking at the origin of different traffic, we can observe that all traffic not destined for port 443 originates from the same IP. It is the same IP, that is accountable for a large share of invalid packets and belongs to already mentioned Internet Measurement Company Ki3.

If we aggregate source IPs into different prefixed networks, assuming most IPs are not given out solely but in a group of at least /24, we clearly see that most traffic originates from one of nine different prefixes.

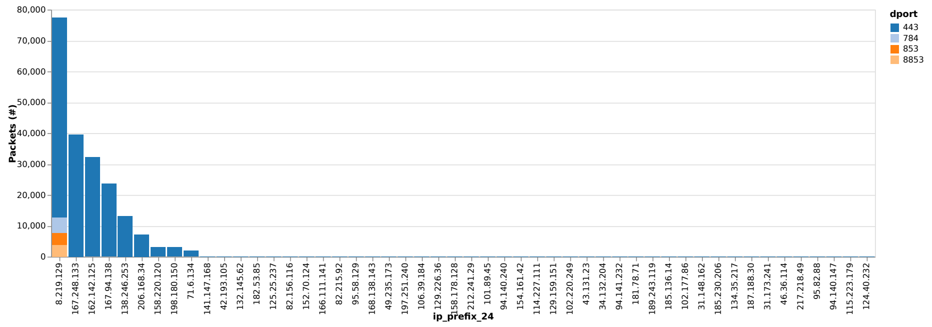

Fig. 3. Share of received packets in filtered data per source /24 prefix

When looking at the different QUIC versions used by senders, we can observe mostly three different versions, displayed in Figure 10. These versions are 0x00000001, 0x00000000 and 0x?a?a?a?a. As per the QUIC RFC, Version 0x00000000 is reserved for version negotiation. Version 0x00000001 is reserved for the RFC-9000 implementation. Version *0x?a?a?a?a* is defined as a pattern that is reserved for "ensuring that version negotiation remains viable" [4].

When grouping received packets by the received TTL of the underlying IPv4 packet, we can identify multiple frequently used values along with some rather rarely used values. This is depicted in Figure 4. Most packets have a TTL of either 47, 50, 51, 53 or 57. These can be used to identify distance and in some cases the sender of the packet or the running operating system. The default TTL for Unix and Mac OSX Systems is 64, explaining all received TTL below this value. Windows systems use a TTL of 128. Yet there are still a lot of packets received with a TTL above 128. The origin of these packets is unknown.

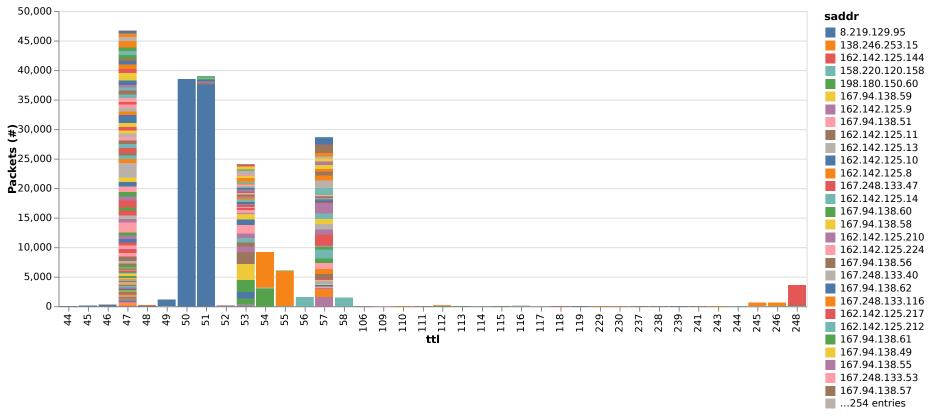

Fig. 4. Number of received Packets per TTL and IP

4.1 Interesting Findings

Around 4000 packets contain the following payload:

```
GET / HTTP/1.1
Host: x.x.x.x:443
```

When grouping by the /24 subnets we can identify four different networks (167.248.133.0/24, 162.142.125.0/24, 167.94.138.0/24, 206.168.34.0/24), from which the traffic originates. These correlate with IP ranges owned by Censys, a well known internet research company. It is still unclear why Censys would send HTTP/1.1 payloads via QUIC.

There are 6.408 other packets that also contain a payload. When grouping by the first few (8) bits, in order to identify the type of payload by common headers, we can identify three different major types of packets. This is displayed in Figure 11.

The most common Header seems to be x01, followed by x47 and x00. x47 correlates to the already described HTTP/1.1 packets received from Censys via UDP. Other data seems to be encoded and decoding this data is out of the scope of this work. Most of the received traffic originates from regular Internet scanners of well known companies. This is in accordance to the QUICSand paper, which states that QUIC IBR is *dominated by research scanners* [14].

5 Conclusion and Outlook

QUIC has become an important protocol for the Internet. This work examined the integration of QUIC into a reactive network telescope. As the use of QUIC becomes increasingly widespread (driven by its use in HTTP/3) adapting network telescopes to effectively monitor QUIC traffic is essential. Whilst regular network telescopes are able to monitor the amount and sources of such packets, they are not able to analyze payload. We have shown that reactive network telescopes are able to respond to incoming traffic and thus can extract more data. Through a month of data collection, we identified distinct characteristics of QUIC traffic and its use on the Internet. Overall, we have shown a feasible approach to collect and analyze QUIC traffic, but further research is required. We have seen a lot of traffic from well known research scanners. Existing data could be further analyzed by removing these scanners from the dataset. In order to better understand received data, we need to decode the payload. As displayed in Figure 1 around a quarter of the traffic determined to be QUIC by the heuristic cannot be handled by the quiche library. This is probably due to the fact, that the heuristic is not designed well enough. A follow-up work could improve this heuristic and thus reduce the server load and required resources.

Another way to improve results of this work would be to use more data by either deploying Spoki to a larger network or for a longer period of time.

References

[1] "The UCSD network telescope," CAIDA. Section: projects. (Jun. 12, 2012), [Online]. Available: https://www.caida.org/projects/network_telescope/ (visited on 08/28/2024).

[2] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, Taormina Sicily, Italy: ACM, Oct. 25, 2004, ISBN: 978-1-58113-821-4. DOI: 10.1145/1028788.1028794.

[3] D. Belson. "Cloudflare 2023 year in review," The Cloudflare Blog. (Dec. 12, 2023), [Online]. Available: https://blog.cloudflare.com/radar-2023-year-in-review (visited on 08/27/2024).

[4] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," Internet Engineering Task Force, Request for Comments RFC 9000, May 2021. DOI: 10.17487/RFC9000.

[5] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Network telescopes: Technical report," Cooperative Association for Internet Data Analysis (CAIDA), Jul. 2004. DOI: https://catalog.caida.org/paper/2004_tr_2004_04.

[6] "Orion network telescope – merit." (2024), [Online]. Available: https://www.merit.edu/initiatives/orion-network-telescope/ (visited on 08/19/2024).

[7] A. Dainotti, C. Squarcella, E. Aben, *et al.*, "Analysis of country-wide internet outages caused by censorship," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11, New York, NY, USA: Association for Computing Machinery, Nov. 2, 2011, pp. 1–18, ISBN: 978-1-4503-1013-0. DOI: 10.1145/2068816.2068818.

[8] 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," presented at the 31st USENIX Security Symposium (USENIX Security 22), 2022, ISBN: 978-1-939133-31-1.

[9] C. Stoll, *The cuckoo's egg: tracking a spy through the maze of computer espionage*. New York: Pocket Books, 2005, 399 pp., ISBN: 978-1-4165-0778-9.

[10] J. Rosenberg, "UDP and TCP as the new waist of the internet hourglass," Internet Engineering Task Force, Internet Draft draft-rosenberg-internet-waist-hourglass-00, Feb. 11, 2008.

[11] M. Nawrocki, P. F. Tehrani, R. Hiesgen, J. Mücke, T. C. Schmidt, and M. Wählisch, "On the interplay between TLS certificates and QUIC performance," in *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, ser. CoNEXT '22, New York, NY, USA: Association for Computing Machinery, Nov. 30, 2022, ISBN: 978-1-4503-9508-3. DOI: 10.1145/3555050.3569123.

[12] *Cloudflare/quiche*, Aug. 28, 2024. [Online]. Available: https://github.com/cloudflare/quiche (visited on 08/28/2024).

[13] C. Huitema, M. Shore, A. Mankin, S. Dickinson, and J. Iyengar, "Specification of DNS over dedicated QUIC connections," Internet Engineering Task Force, Internet Draft draft-huitema-quic-dnsoquic-07, Sep. 7, 2019.

[14] 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, ISBN: 978-1-4503-9129-0. DOI: `10.1145/3487552.3487840`.

[15] G. Woods. "HTTP/3 | software AG." (Mar. 2021), [Online]. Available: `https://techradar.softwareag.com/technology/http-3/` (visited on 08/28/2024).

Appendix

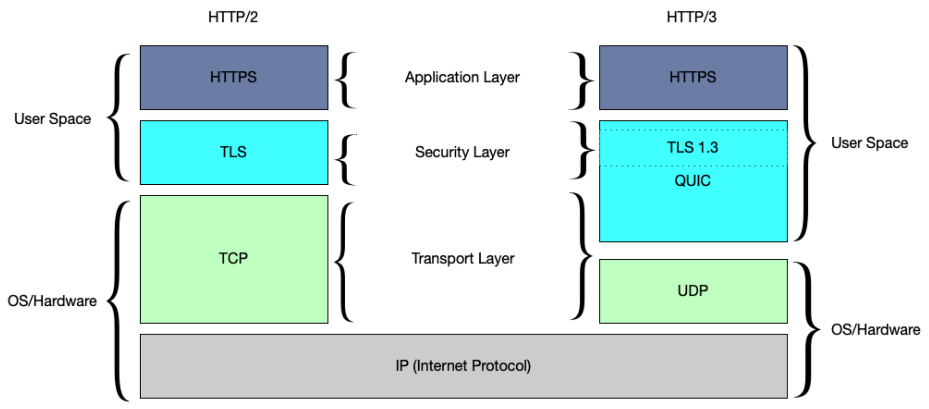

Fig. 5. QUIC in the OSI-Layer Model [15]

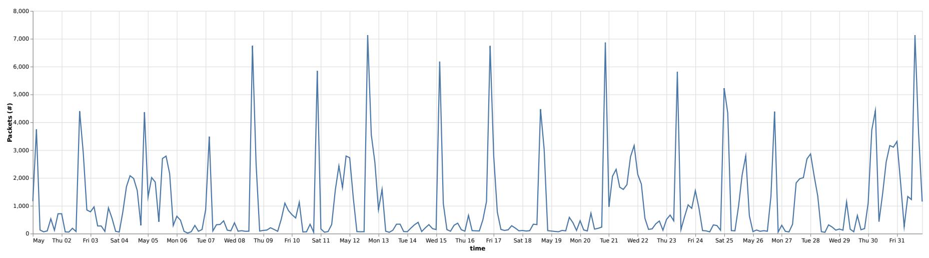

Fig. 6. Number of QUIC packets over time in a /24 network

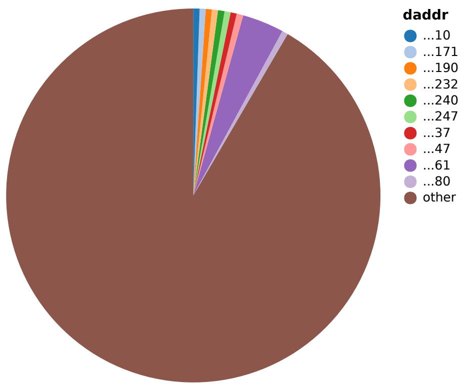

Fig. 7. Share of received traffic in the unfiltered data per destination IP

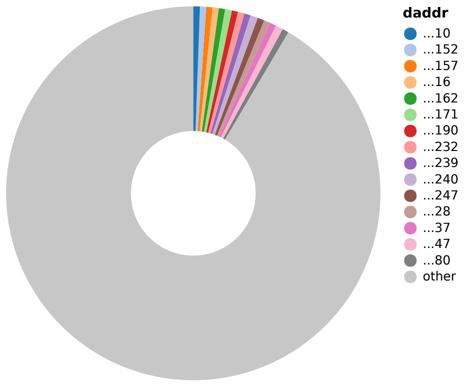

Fig. 8. Share of received traffic in the filtered data per destination IP

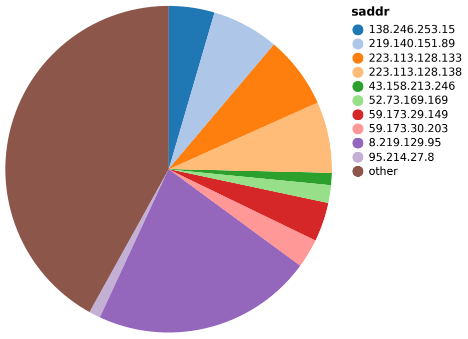

Fig. 9. Share of received traffic for "error" marked packets per source Addresses

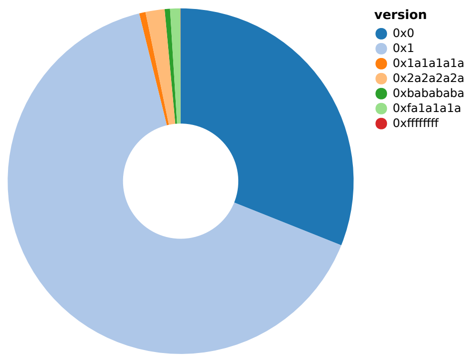

Fig. 10. Share of received packets in the filtered data per QUIC version

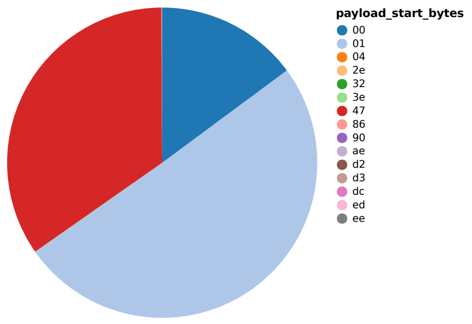

Fig. 11. Share of received packets in the filtered data per first byte of payload encoded as hex chars