It's Over 9000: Analyzing Early QUIC Deployments with the Standardization on the Horizon

Johannes Zirngibl
Technical University of Munich
Garching bei München, Germany
zirngibl@net.in.tum.de

Philippe Buschmann
Technical University of Munich
Garching bei München, Germany
buschman@net.in.tum.de

Patrick Sattler
Technical University of Munich
Garching bei München, Germany
sattler@net.in.tum.de

Benedikt Jaeger
Technical University of Munich
Garching bei München, Germany
jaeger@net.in.tum.de

Juliane Aulbach
Technical University of Munich
Garching bei München, Germany
aulbach@net.in.tum.de

Georg Carle
Technical University of Munich
Garching bei München, Germany
carle@net.in.tum.de

ABSTRACT

After nearly five years and 34 draft versions, standardization of the new connection oriented transport protocol QUIC was finalized in May 2021. Designed as a fundamental network protocol with increased complexity due to the combination of functionality from multiple network stack layers, it has the potential to drastically influence the Internet ecosystem. Nevertheless, even in its early stages, the protocol attracted a variety of parties including large providers. Our study shows, that more than 2.3 M IPv4 and 300 k IPv6 addresses support QUIC hosting more than 30 M domains.

Using our newly implemented stateful QUIC scanner (*QScanner*) we are able to successfully scan 26 M targets. We show that TLS as an integral part is similarly configured between QUIC and TLS over TCP stacks for the same target. In comparison, we identify 45 widely varying transport parameter configurations, *e.g.*, with differences in the order of magnitudes for performance relevant parameters. Combining these configurations with HTTP Server header values and associated domains reveals two large edge deployments from Facebook and Google. Thus, while found QUIC deployments are located in 4667 autonomous systems, numerous of these are again operated by large providers.

In our experience, IETF QUIC already sees an advanced deployment status mainly driven by large providers. We argue that the current deployment state and diversity of existing implementations and seen configurations solidifies the importance of QUIC as a future research topic. In this work, we provide and evaluate a versatile tool set, to identify QUIC capable hosts and their properties.

Besides the stateful *QScanner* we present and analyze a newly implemented IPv4 and IPv6 ZMap module. We compare it to additional detection methods based on HTTP Alternative Service Header values from HTTP handshakes and DNS scans of the newly drafted `HTTPS` DNS resource record. While each method reveals unique deployments the latter would allow lightweight scans to detect QUIC capable targets but is drastically biased towards Cloudflare.

IMC '21, November 2–4, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9129-0/21/11.
https://doi.org/10.1145/3487552.3487826

CCS CONCEPTS

• **Networks** → **Network protocols**; **Network measurement**.

KEYWORDS

IETF QUIC, TLS, Server Deployment, Internet Measurement

ACM Reference Format:
Johannes Zirngibl, Philippe Buschmann, Patrick Sattler, Benedikt Jaeger, Juliane Aulbach, and Georg Carle. 2021. It's Over 9000: Analyzing Early QUIC Deployments with the Standardization on the Horizon. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3487552.3487826

1 INTRODUCTION

QUIC, a new connection-oriented Internet protocol was finally standardized by the Internet Engineering Task Force (IETF) in May 2021 [21]. The protocol was initially developed and implemented by Google and made public in 2013 [23]. Afterwards, the official standardization process was transferred to the IETF within its own working group[1]. Over the years, the base draft passed through 34 revisions before entering its last calls.

The QUIC protocol combines functionalities from different layers of the network stack, including the transport layer, security in the form of Transport Layer Security (TLS) and stream control to optimize higher layer applications. The integration of QUIC into the protocol stack and the comparison to TLS over TCP can be seen in Figure 1. In addition to the QUIC base protocol, Hypertext Transfer Protocol (HTTP) Version 3 is drafted [4], specifically focusing on the deployment of HTTP on top of QUIC. This combination of functionality from multiple layers increases the overall complexity of a protocol and, therefore, increases the possibility of diverging implementations, potential errors or unintended behavior.

As new, fundamental network protocol, QUIC has the potential to drastically influence the Internet ecosystem. It attracted a variety of providers, developers and contributors even in its early stages. The QUIC working group already lists 22 different implementations [17]. Additionally, with the initial contribution from Google, and as shown by Rüth et al. [39] in 2018 significant, productive deployment of Google QUIC was already visible on the Internet in early stages of the specification. By now, QUIC carries over a third of the Google traffic [8] and Facebook reports that QUIC is

[1] https://tools.ietf.org/wg/quic/

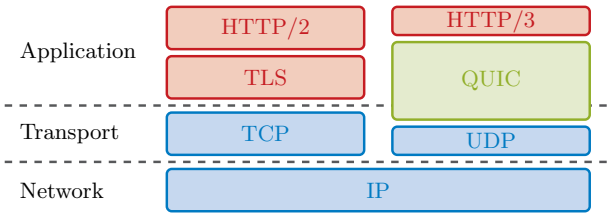

Figure 1: QUIC stack compared to TLS over TCP.

responsible for over 75 % of its traffic [24]. Furthermore, in April 2021, Firefox officially announced QUIC support in its Nightly and Beta release [7].

Due to the attraction of the protocol in the community and within large providers but also due to the increased complexity of the protocol based on the combinations of functionality from multiple layers, thorough research of the protocol, its deployment and its effects on the Internet ecosystem are necessary.

Therefore, research requires the means to identify QUIC capable targets. Furthermore, knowledge about the state of deployments, configurations but also involved parties provides a fundamental baseline for future research and developments regarding IETF QUIC.

In this paper, we compare different methods to identify QUIC deployments and analyze the availability of QUIC shortly before the standardization based on an Internet-wide measurement study. This allows to identify how many QUIC deployments can already be found and whether deployments are well-prepared for the final standardization already supporting latest versions. Besides the identification of deployments and supported versions, we examine characteristics in regard to successful handshakes, TLS behavior, transport parameters and HTTP/3 capabilities based on a newly implemented and shared QUIC scanner, namely *QScanner*.

Our contributions in this work are:

(i) We compare different methodologies to identify IETF QUIC deployments. On one hand, we execute large scale ZMap based IPv4 and IPv6 scans which detect QUIC servers and their supported versions. On the other hand, we compare our findings to alternative service discovery methods that can reveal QUIC deployments. Therefore, we analyze the HTTP Alternative Service (ALT-SVC) Header from TLS-over-TCP scans including HTTP requests and the newly drafted Domain Name System Resource Records (DNS RRs) for *Service Binding*, `SVCB` and `HTTPS` [41]. We scanned in regular intervals over a period of three months to analyze the development during the final steps of the standardization.

(ii) We deploy stateful scans that attempt complete QUIC handshakes with found deployments and analyze how many targets can be successfully connected to using QUIC. This allows us and future research to analyze the configuration of targets in regard to their transport parameters, TLS and HTTP.

(iii) To support the community to conduct substantial research on QUIC, we publish our versatile tool set including the ZMap modules to detect IETF QUIC deployments and our stateful QUIC scanner *QScanner*. Furthermore, we provide access to our analysis results and raw data:

https://quicimc.github.io/

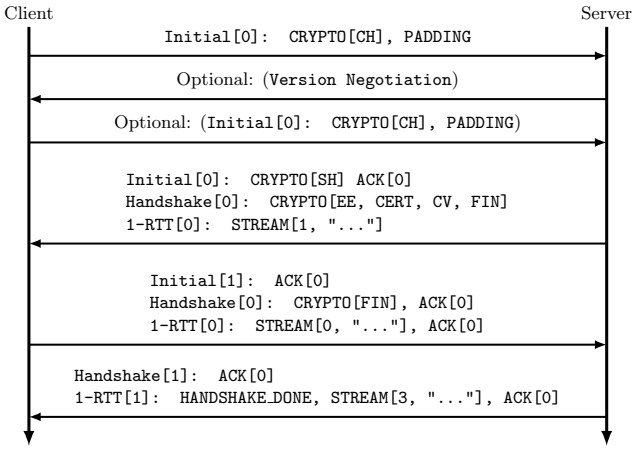

Figure 2: QUIC handshake including a version negotiation.

In Section 2 important background is provided, followed by a description and motivation of newly implemented scan tools and conducted scans in Section 3. We analyze and compare different methodologies to discover QUIC capable hosts in Section 4 and investigate distribution and version support of deployments. We further analyze deployments regarding their TLS setup, QUIC specific transport parameters and HTTP capabilities in Section 5 followed by a comparison to related work in Section 6 and a discussion of results in Section 7.

2 BACKGROUND

We explain important background covering relevant parts of the QUIC handshake and version negotiation. Additionally, we cover the HTTP ALT-SVC Header and `HTTPS` DNS RR.

2.1 QUIC

QUIC is designed to be a general transport protocol combining features from different layers of the network stack. The protocol covers transport functionality, including reliable delivery and congestion control, allows network path independent connection migration, and integrates TLS 1.3 to provide confidentiality and integrity of data. To reduce latency, mainly during the handshake, it combines the exchange of transport and cryptographic parameters as shown in Figure 2.

As indicated in Figure 1, QUIC includes functionality, namely stream multiplexing and per stream flow control that is normally handled by higher layers, *e.g.*, HTTP/2. Therefore, HTTP/3 is standardized alongside QUIC [4] as HTTP derivative on top of QUIC. Furthermore, new Application-Layer Protocol Negotiation (ALPN) values are specified to indicate the support of HTTP/3. During the standardization process, it included the draft version, *e.g.*, h3-29 for "Version 29" but will be fixed to h3 [4].

Handshake. To begin a handshake, a client sends an *Initial* packet including a TLS 1.3 *Client Hello* and transport parameters. The latter are sent as TLS extension to offer integrity protection. Furthermore, the *Initial* packet needs to be padded to at least 1200 B. This ensures that a reasonable Path Maximum Transmission Unit (PMTU) is

supported and reduces the amplification factor to prevent potential misuse. If the server is able to pursue the handshake, it replies with an *Initial* packet containing the TLS 1.3 *Server Hello* in a *Crypto* frame, and further TLS messages in *Handshake* frames. The client needs to conclude the handshake with necessary *Acknowledgment* frames, a *Crypto* frame finishing the TLS handshake and can start data transmission using *Stream* frames.

Version Negotiation. Due to the possibility to implement QUIC in user space, different implementations are available [17] which can be quickly adapted to new versions [38]. Besides, Google advanced its own QUIC specification. This resulted in a multitude of existing versions [18] and their deployment in parallel. To allow successful handshakes without previous knowledge about supported versions by a server, a simple version negotiation mechanism was specified alongside QUIC [21]. If the server does not support the initially offered version, it can reply to the client's *Initial* frame with a version negotiation including its set of supported versions. A specific set of versions with the pattern `0x?a?a?a?a` is specified to enforce a version negotiation [21]. This mechanism might be updated in the future with more functionality as proposed in an additional draft [40]. Similar to Rüth et al. [39], we use this to discover QUIC capable hosts and their supported versions described in Section 3.1.

Transport Parameter. QUIC allows each client and server to specify an individual set of transport parameters [21]. To send these parameters early during the handshake and provide integrity protection, a new TLS extension was defined. Currently, 17 different parameters exist in the specification, which can be extended in the future. For example, peers can set the initial size of the flow control window for the connection or streams, the maximum number of allowed streams, and options regarding connection migration. Some options are server-only and cannot be used by the client. Also, options set during the handshake can later be updated with corresponding QUIC frames. We look into deployment specific parameters on the Internet in more detail in Section 5.2.

2.2 Alternative Service Discovery

With the increase of different services sharing the same domain or even the same port and the necessity to reduce latency, different mechanisms exist to discover alternative service endpoints and their parameters.

HTTP Alternative Services. HTTP provides a functionality called *Alternative Services* [36], to enable servers to redirect clients. Alternative service endpoints are defined as ALPN protocol name, a host and port. They can be served using the HTTP ALT-SVC Header or in a dedicated ALTSVC frame using HTTP/2. The new HTTP/3 ALPN value can be added accordingly to indicate support. Due to the strict relation between HTTP/3 and QUIC, receiving an HTTP ALT-SVC Header with an ALPN value indicating HTTP/3 implies QUIC support.

`SVCB` *and* `HTTPS` *DNS RR.* Besides HTTP ALT-SVC Header, IETF works on new DNS RRs, namely `SVCB` and `HTTPS` [41] to allow a client to learn about endpoints or additional information for a specific endpoint before an initial transport layer handshake using DNS.

The `SVCB` DNS RR stands for *Service Binding* and represents a more general record, while the `HTTPS` DNS RR is specifically designed to be used with HTTPS. They accomplish two major goals, directing a client *(i)* to another alias or *(ii)* to an alternative endpoint including service information. In the scope of this work, the latter goal is mainly of interest. A major use case of the `HTTPS` DNS RR is to direct clients to QUIC endpoints [41]. Besides ALPN values, these DNS RRs can directly contain *ipv4-* and *ipv6hints* including IP addresses for the service. This allows to identify, whether a domain can be accessed using QUIC and which IP address can be used, using a single, recursive DNS query. We use addresses from these hints in combination with resolved domains for further analysis (see Section 4) and as targets for stateful scans (see Section 5).

3 CONDUCTED SCANS

This section explains our measurements and describes all implemented or used tools to conduct scans including QUIC, DNS and TLS over TCP. We publish all implemented scanning tools to support the community with research regarding QUIC. For all scans, we apply ethical measures described in Appendix A.

3.1 ZMap Scans

ZMap allows detecting targets supporting a specific transport protocol on a scanned port without previous knowledge about targets (at least for IPv4) [10]. Therefore, we implemented a ZMap module to detect QUIC capable hosts on the Internet as first scan. This module is similar to the work from Rüth et al. [39]. We are not able to directly reuse their published module, since the available code was built to detect Google QUIC versions and is neither up to date with the current IETF drafts nor does it support IPv6.

The implemented ZMap module sends IETF draft conform QUIC packets enforcing a *Version Negotiation* packet as response. The remaining content is neither encrypted nor does it contain a *Client Hello* message. This is not necessary since the server must process the invalid version first and respond with a version negotiation [21]. This reduces the computational overhead at the scanner and theoretically allows higher scanning rates. Furthermore, padding is added to reach the required 1200 B. Using ZMap with the implemented module originates at least a magnitude more traffic as a simple TCP SYN scan with the same rate, *e.g.*, discovering HTTPS capable hosts on port 443. This research is limited to a single vantage point. However, we use a dedicated, unfiltered network research prefix part of our university infrastructure for scans. This network is capable to handle the scanning load, and we are in contact with the network administrators. With the increased usage and specification of newly means to discover alternative services including QUIC, future research might be able to reduce the number of required scans.

We tested whether a version negotiation can be triggered without padding and found a drastically lower response rate. Only 11.3 % of IPv4 addresses found with padding responded and 95.4 % of these are from a single Autonomous System (AS). With a subset of targets, we tested to complete a handshake with an *Initial* packet without padding, but were unsuccessful in all tests. Thus, all tested deployments follow specifications in draft 34 [22] at least for the *Initial* packet without a version negotiation.

For IPv4, we scan the complete address space filtering the local blocklist following the ethical considerations discussed in Appendix A. Before scanning the Internet, we tested the module against local QUIC setups and officially provided test servers [17]. We slowly increased the number of scanned targets and the packet rate per second. After we saw no issues in local tests and received no complaints, we started to scan Internet-wide with up to 15 k packets per second and are capable of scanning the reachable IPv4 address space in under 56 h.

For IPv6, we use AAAA records from domain resolutions explained in Section 3.2 in addition to the inputs provided by the IPv6 Hitlist service [14]. Combining both input sources we scan 24.5 M IPv6 addresses.

We show in Section 4 and 5, that a major disadvantage of these scans is missing information about domains resolving to found IP addresses supporting QUIC. During stateful scans, missing knowledge about associated domains results in a low success rate due to the strict integration of TLS into QUIC and the usage of Server Name Indication (SNI). We argue that statistics regarding the deployment of QUIC only based on ZMap scans need to be analyzed carefully. Furthermore, we show in Section 4 that some deployments do not react to the forced version negotiation as specified in RFC9000 [21] and used by the ZMap module.

3.2 DNS Scans

To circumvent the disadvantages of QUIC ZMap scans, mainly missing domains and the high bandwidth requirement, we evaluate additional means to discover QUIC deployments based on domains that can be used as SNI. In theory, the newly drafted HTTPS DNS RR provides a mechanism to quickly identify whether a service can be accessed using QUIC based on its domain. This allows to set up lightweight scans based on DNS resolution. In contrast to ZMap scans, previous knowledge about potential targets is required in the form of domains.

To analyze the effectiveness of this approach and the quality of found QUIC capable targets for further research, we actively resolve domain lists searching for their SVCB and HTTPS DNS RRs as explained in Section 2.2.

We use MassDNS[2] with a local Unbound[3] resolver to resolve domains from different input sources. Resolved domain lists include the Alexa Top 1M [2], Majestic Top 1M [32] and Umbrella Top 1M [5]. Furthermore, we request and scan available zone files from the Centralized Zone Data Service (CZDS) [20], including com, net and org. CZDS offers us 1.1 k Top Level Domains (TLDs) accounting for around 211 M domains. 180 M of these are part of com, net and org. We scanned all sources once a week between March 1st, and May 9th, 2021. While we successfully resolved HTTPS DNS RR of domains, none successfully resolved to an SVCB DNS RR. Therefore, we focus on results from HTTPS DNS RR scans throughout the remaining paper. Nevertheless, we show in Section 4 that these scans still extend our view on the state of QUIC deployments. Analyzing the state of these new DNS RRs and their deployment in the future provides research a lightweight mean to detect service information of domains.

These DNS scans additionally resolve A and AAAA DNS RRs to be used as a basis for further scans including TLS and IPv6 ZMap scans. For the latter, we use AAAA records as input. Furthermore, DNS resolutions are combined with ZMap scans for stateful QUIC scans and TLS over TCP scans. The information from domain resolutions is used as SNI to increase the success rate of TLS handshakes.

3.3 TLS over TCP Scans

Due to the low success rate of HTTPS DNS RR scans (see Section 4), we additionally investigate the value of the HTTP ALT-SVC Header to detect QUIC capable targets. Compared to the DNS scans, collecting HTTP ALT-SVC Headers requires more costly scans but due to the advanced maturity of the methodology, we are able to detect more QUIC deployments as shown in Section 4. We rely on regular TLS over TCP scans including HTTP/1 or HTTP/2 requests, already conducted within our research group. This allows us to collect HTTP ALT-SVC Headers, and thus potential QUIC deployments as explained in Section 2.2.

To scan TLS over TCP we firstly conduct standard TCP SYN ZMap scans targeting port 443. We expect the highest response rate on this port for TLS over TCP but also for QUIC scans. We use the ZMap fork[4] from Gasser et al. [14] that supports IPv6. While we scan the complete IPv4 address space, we use domain resolutions from our DNS scans and data from the IPv6 Hitlist Service [14] as input for IPv6 scans.

The ZMap scans are followed by stateful TLS scans finishing a complete handshake. We use the Goscanner, introduced by Amann et al. [3] and used for example in related work by Holz et al. [19], for stateful TLS scans. The Goscanner is used instead of the more prominent tool ZGrab 2.0[5], because it supports TLS 1.3, the mandatory version for QUIC [43]. This allows us to extract TLS information from these scans to compare to our QUIC results besides the collection of HTTP ALT-SVC Headers (see Section 5). We scan the set of addresses from ZMap twice, once without and once with SNI. SNI information for each target is collected by joining the results from the DNS A and AAAA resolution with the results from the ZMap scans.

3.4 *QScanner*: A Stateful QUIC Scanner

The previous scans only allow to detect QUIC deployments and their supported versions. Collection of further information about the deployment, *e.g.*, whether complete handshakes succeed or information regarding its QUIC specifics, including TLS and HTTP properties is not possible. Therefore, we implemented a stateful scanner similar to the used Goscanner for TLS over TCP. It is based on the QUIC implementation *quic-go*[6] and *qtls*[7]. Both libraries are under active development and implement new draft versions quickly. The version used for scans analyzed in Section 5 supported draft 29, 32 and 34. However, it was updated shortly after the release of RFC9000 to support IETF version 1, usable with the published *QScanner*. According to the *Interop Runner* from Seemann and Iyengar [42], it is compatible to most implementations. Therefore,

[2] https://github.com/blechschmidt/massdns
[3] https://www.nlnetlabs.nl/projects/unbound/about/
[4] https://github.com/tumi8/zmap
[5] https://github.com/zmap/zgrab2
[6] https://github.com/lucas-clemente/quic-go
[7] https://github.com/marten-seemann/qtls-go1-16/

Table 1: Found QUIC Targets (calendar week 18).

		Scanned Targets	Results Addresses	Results ASes	Results Domains
ZMap	IPv4	3 023 298 514	2 134 964	4736	30 970 316[1]
	IPv6	24 434 296	210 997	1704	17 972 799[2]
ALT-SVC[3]	IPv4	375 338 772	232 585	2174	36 907 770
	IPv6	69 458 318	283 169	292	16 979 759
HTTPS	IPv4	213 689 057[4]	85 092	1287	2 962 708
	IPv6		69 684	112	2 736 040

[1] Join with DNS scan, 10 % of IPv4 addresses map to a domain
[2] Join with DNS scan, 62 % of IPv6 addresses map to a domain
[3] Extracted from existing, regular scans (see Section 3.3)
[4] A and AAAA records are additionally resolved to join with ZMap scans (see Section 3.2)

we expected a high success rate with a scanner based on *quic-go* and are able to effectively parallelize our scan reducing the overall scan duration, while respecting the ethical considerations from Appendix A. We only altered the respective QUIC and TLS libraries to expose information about QUIC, TLS and HTTP. The *QScanner* allows to either scan IP addresses individually, IPv4 as well as IPv6, or combined with a domain used as SNI.

We used the *QScanner* for stateful scans covering all found targets, *(i)* from ZMap scans in combination with DNS A and AAAA resolutions, *(ii)* from HTTP ALT-SVC Header data and *(iii)* from HTTPS DNS RR scans.

4 QUIC DEPLOYMENTS ON THE INTERNET

The following section analyzes the results from scans in regard to their detection rate of QUIC deployments, seen versions and potential biases towards providers. Most of the following analyses focus on the scans from calendar week 18 (May 3, until May 9, 2021). Table 1 provides a general overview about found targets from each source.

ZMap results. With 2.1 M IPv4 addresses reacting with a *Version Negotiation* packet, our ZMap scan results in the most targets based on IP addresses. Furthermore, these addresses are located in over 4.7 k ASes. Compared to results from Rüth et al. [39], the number of addresses has tripled since 2018 and the involved ASes increased by 50 %. Joining the result with our DNS resolution reveals 30 M domains with potential QUIC support resolving to 10 % of found IPv4 addresses, while no domain resolves to the remaining addresses in our scans. Most IP addresses without an associated domain belong to large Content Delivery Networks (CDNs), mainly Cloudflare (AS13335) with 28 %, Google (AS15169) with 22 % but also Akamai (AS20940, 16.9 %) and Fastly (AS54113, 12.3 %). We argue that our resolved domain set can only be associated to a subset of their IP addresses due to load balancing mechanisms. Furthermore, our list of resolved domains is not exhaustive. Thus, for some IP addresses we might not be aware of associated domains.

In contrast to IP addresses without associated domains, we are aware that not all joined domains might be QUIC enabled but offer different functionality. Especially for large CDNs, a domain resolving to an IP address does not necessarily mean it is used for

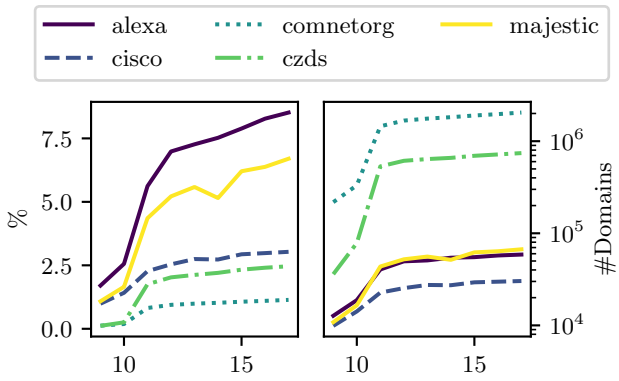

Figure 3: Success rate of HTTPS DNS RR scans over multiple calendar weeks (x-axis) in 2021. We resolve around 180 M domains from `com/net/org`. TLDs from CZDS (without `com/net/org`) yield in 31 M additional domains.

QUIC. We evaluate the success rate of QUIC handshakes with these targets in more detail in Section 5.

For IPv6, we find considerably fewer QUIC capable targets and ASes. The ZMap scan results in 210 k IPv6 addresses out of 24 M probed targets (see Section 3). Furthermore, IPv6 addresses are located in 1.7 k ASes compared to 4.7 k for IPv4. On one hand, this difference is based on the fact, that an IPv6 ZMap scan relies on an input and cannot scan the complete address space. On the other hand, IPv6 still lacks deployment reducing potentially found targets. Similar differences can be seen for our TLS over TCP scans with 53 M IPv4 but only 3 M IPv6 addresses with an open port 443.

A domain can be found for 62 % of IPv6 addresses. This is substantially higher than for IPv4 but one of the input sources for the IPv6 ZMap scans are the DNS scans. Similar to IPv4, most IPv6 addresses without domains are from CDNs (31.7 % Google (AS15169), 28.5 % Akamai (AS20940)).

Alternative service results. As shown in Figure 3, the overall success rate of HTTPS DNS RR per input is low with ~1 % for Com/Net/Org and up to 8 % for top lists but increases over time. HTTPS DNS RRs for 2.9 M domains contain information about an IPv4 QUIC deployment but hint to only 85 k distinct addresses located in 1.2 k ASes. Regarding IPv6, 2.7 M domains indicate QUIC support including 69.7 k addresses located in only 112 ASes.

Extracting HTTP ALT-SVC Headers from TLS over TCP scans results in 232 k QUIC capable IPv4 addresses located in 2 k ASes. While this methodology reveals more QUIC deployments, it hints to a magnitude less IPv4 addresses compared to the ZMap scan. Regarding IPv6, HTTP ALT-SVC Headers reveal a similar number of addresses, but they are located in considerably fewer ASes.

We do not additionally scan addresses from these targets with ZMap but only with the *QScanner* (see Section 5). The following results regarding the distribution across ASes and supported versions is directly extracted from HTTP ALT-SVC Headers or from resolved domains and their HTTPS DNS RR.

Overlap between sources. All three input sources provide unique targets. The overlap of IPv4 addresses is 69.5 k, while 146 k IPv4 addresses are unique based on HTTP ALT-SVC Headers and 2 M IPv4 addresses are only found by ZMap. Even though the success rate of HTTPS DNS RR scans is low (see Figure 3), it offers 12 k unique addresses.

While the overall number of seen IPv6 addresses is smaller, the overlap is similar with 68 k addresses. Only 855 addresses are uniquely seen in HTTPS DNS RR and 136 k addresses are from ZMap. The largest share of unique IPv6 addresses is from the HTTP ALT-SVC Header with 208 k IPv6 addresses.

While some addresses found by alternative service discovery mechanisms are missed by ZMap due to network events, most differences are due to deployments not implementing the version negotiation mechanism as used by our ZMap module. Stateful scans presented in Section 5 show, that even though found deployments do not react to the implemented ZMap module, the *QScanner* is able to communicate with multiple of these targets, either resulting in successful handshakes or QUIC specific alerts. In contrast, ZMap is able to detect further IP addresses supporting QUIC, missed by domain based scans from a single vantage point due to load balancing.

Key take-away. Analyzing the found number of QUIC deployments based on the three described discovery methods illustrates their differences. Each methodology reveals unique QUIC deployments due to differences in QUIC implementations but also configurations and thus their behavior in respect to our scans. At the moment, research needs to rely on different sources of QUIC capable targets to allow a holistic analysis of QUIC and its deployment on the Internet.

ZMap indicates QUIC support for most IPv4 addresses but based on our DNS scans, domains resolve to only 10 % of found addresses. We analyze whether successful handshakes are possible without a domain in Section 5. In comparison, alternative service discovery approaches reveal fewer addresses but HTTP ALT-SVC Headers reveal similar amounts of domains reachable using QUIC. The low success rate of HTTPS DNS RR scans drastically limits the utility of the scan at the moment. We argue that the low success rate is mainly due to the early stage of the draft [41].

We suggest re-evaluating this finding in future work, to analyze whether the final standardization of QUIC results in the convergence of deployments towards a similar behavior and thus, whether a single detection approach suffices.

4.1 Who deploys and uses QUIC?

To evaluate the distribution of deployments in regard to providers, we analyze which ASes announce ranges containing QUIC capable IP addresses. Figure 4 shows the distribution of addresses, which indicate QUIC support either based on a successful version negotiation with ZMap, with an HTTP ALT-SVC Header during TLS over TCP scans or an HTTPS DNS RR, across ASes ranked by the number of addresses per AS. For IPv6 addresses, the overall number of hits and corresponding ASes is not only smaller, but the top AS already covers between 60 % for ZMap and 99 % for HTTPS DNS RR. Considering ZMap for IPv4, the top AS covers only 35 %, but the top 4 already covers 80 %. The most even distribution can be

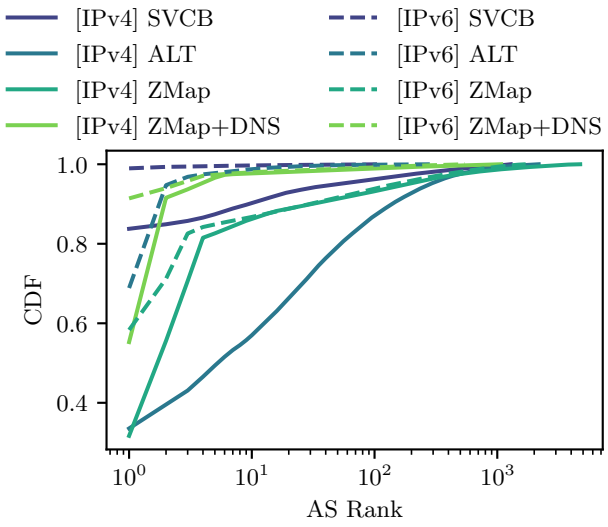

Figure 4: AS distribution of addresses indicating QUIC support during version negotiation with ZMap or ALPN values (Note the y-axis does not start with 0).

seen for HTTP ALT-SVC Headers where the top AS only covers 35 %, and only after 100 ASes, an 80 % coverage is reached.

Table 2 shows the top 5 ASes for IPv4 addresses (a mapping between AS numbers and their names can be found in Appendix B Table 7). Cloudflare (AS13335) originates most addresses for all sources but IPv6 addresses in combination with their HTTP ALT-SVC Headers. The latter source is dominated by Hostinger (AS47583). Interestingly, Google (AS15169) as initial force behind QUIC is Rank 2 based on ZMap scans, Rank 3 based on HTTPS DNS RRs and only Rank 6 based on HTTP ALT-SVC Headers. Most of the remaining top ASes are cloud providers like DigitalOcean (AS14061), Akamai (AS2094), or OVH (AS16276).

Key take-away. The analysis shows that the early deployments can mainly be found in ASes operated by large providers. Nevertheless, deployments can be found in more than 4700 different ASes. Using HTTPS DNS RRs to discover QUIC capable hosts is mainly limited to Cloudflare at the moment. We investigate whether deployments distributed accros ASes are set up by individuals or operated by large providers in Section 5 based on stateful scans.

4.2 Deployed Versions

We use the version negotiation results from ZMap scans but also versions directly indicated as ALPN value in HTTPS DNS RRs and the HTTP ALT-SVC Header. Figure 5 shows the distribution of version sets announced by servers in the version negotiation packet during the IPv4 ZMap scans. *Other* combines 46 sets with a visibility of less than 1 % each. Figure 6 shows the frequency of individual versions. Versions starting with Q and T indicate Google QUIC without and with TLS respectively and versions including *mvfst* are Facebook specific.

The sets solely consisting of IETF versions are primarily used by Cloudflare (AS13335). They are mainly responsible for the change of

Table 2: Top 5 providers hosting QUIC services (a mapping to AS numbers can be found in Appendix B, Table 7).

	ZMap			HTTPS DNS RR			ALT-SVC		
Rank	Provider[1]	#IPv4 Addr.	#Domains[2]	Provider[1]	#IPv4 Addr.	#Domains	Provider[1]	#IPv4 Addr.	#Domains
1	Cloudflare	676 483	23 843 989	Cloudflare	71 278	2 887 327	Cloudflare	78 033	19 286 420
2	Google	510 450	6 006 547	DigitalOcean	969	1256	OVH	14 011	1 691 721
3	Akamai	320 646	23 206	Google	719	1235	GTS Telecom	8160	234 149
4	Fastly	232 776	938 649	Amazon	709	814	A2 Hosting	8068	858 932
5	Cloudflare London	23 489	61 979	OVH	708	1034	DigitalOcean	6556	135 910
	Provider[1]	#IPv6 Addr.	#Domains[2]	Provider[1]	#IPv6 Addr.	#Domains	Provider[1]	#IPv6 Addr.	#Domains
1	Cloudflare	123 061	17 862 254	Cloudflare	68 963	2 735 390	Hostinger	195 023	195 049
2	Google	27 186	19 833	Amazon	263	48	Cloudflare	73 253	15 955 097
3	Akamai	23 997	12 745	DigitalOcean	56	65	PrivateSystems	5925	52 788
4	Cloudflare London	3443	25 763	Linode	49	73	EuroByte	1784	12 410
5	Jio	1441	153	1&1 IONOS	38	42	Synergy Wholesale	825	150 602

[1] Names are based on ASes announcing the prefix for each IP address (see Table 7 in Appendix B for a mapping to the respective AS number)
[2] Join with DNS scans

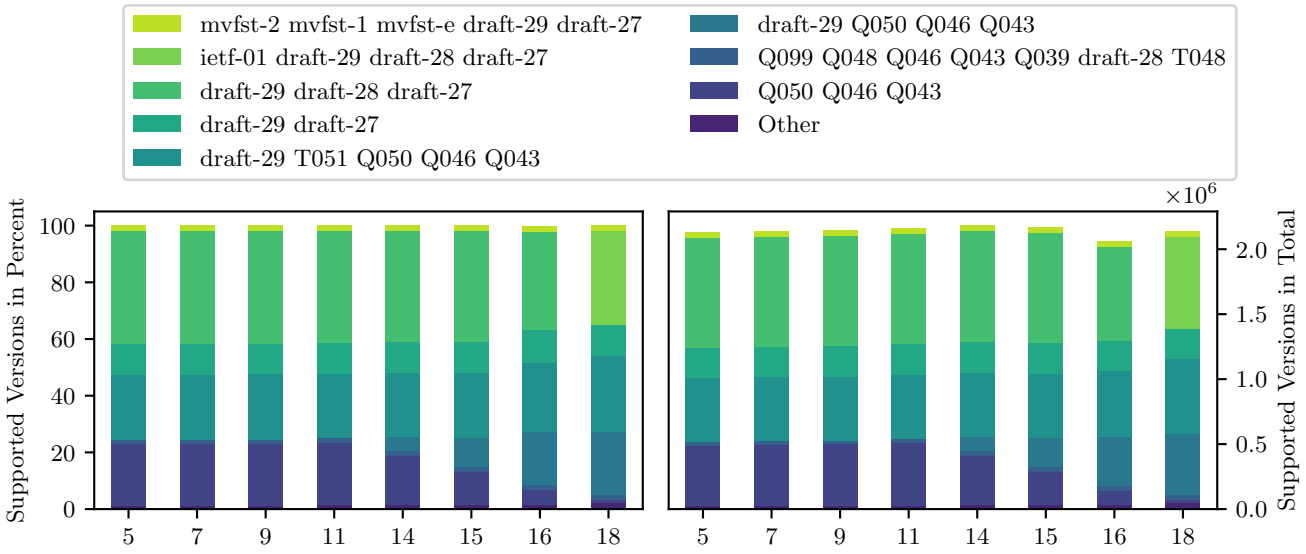

Figure 5: Supported QUIC version sets per IPv4 address from ZMap scans over several calendar weeks (x-axis) in 2021. *Other* combines all sets with an occurrence of less than 1 %.

sets in week 18 to a new set including IETF "Version 1". At the end of our scanning period, IETF "Version 1" was seen in 95 different ASes. While the version was mentioned in draft 34, it was labeled as "do not deploy" [22].

Google (AS15169) announces the set consisting of Google QUIC versions (including T051) but also IETF draft-29. We show in Section 5 that this set was often inconsistent to the actual server behavior observable as version mismatches in our stateful scan. The set with only Google QUIC versions was mainly used by Akamai (AS2094) at the beginning, but they started to include IETF draft-29 throughout our measurement period. Regarding individual versions, Figure 6 shows on one hand, that 50 % of found addresses still support Google QUIC. On the other hand, the frequency of IETF draft-29 increases form 80 % in February to 96 % in May 2021. IETF draft-27 is seen more often than draft-28 mainly due to Fastly (AS54113) announcing the set, draft-29 and draft-27.

Results from HTTPS DNS RRs and HTTP ALT-SVC Headers do not contain exact QUIC versions but HTTP ALPN values including the draft version, *e.g.*, *h3-29* (see Section 2). Furthermore, the ALPN can be different for domains even when they share the same IP address. Therefore, the following analysis is based on targets as combination of (Domain, IP address)-pairs.

Figure 7 shows the distribution of ALPN sets retrieved from HTTP ALT-SVC Headers. *Other* combines all sets with an occurrence of less than 1 %. The remaining sets can be divided into three groups, *(i)* a set only consisting of IETF QUIC versions, *(ii)* sets including Google QUIC versions covering different ranges and *(iii)* a

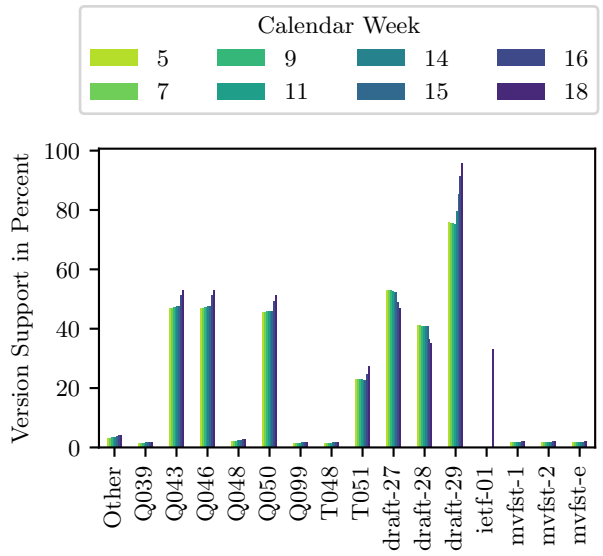

Figure 6: Supported individual QUIC versions from ZMap scans. *Other* combines versions with an occurrence of less than 1 %.

set only containing the string *quic*. The most common set only consists of IETF QUIC versions, namely *h3-27,h3-28,h3-29*. It is mainly used by Cloudflare, besides 269 additional ASes, and thus already covers a majority of domains. Interestingly, the deployment of IETF "Version 1" from Cloudflare seen in Figure 5 cannot be seen based on the HTTP ALT-SVC Header during our measurement period.

Even though Google itself is only the sixth common AS based on HTTP ALT-SVC Header data, the second most common set is *h3-25,h3-27,h3-Q043,h3-Q046,h3-Q050,quic*, containing Google QUIC versions and older IETF QUIC versions. It is used by 1.7 k ASes. However, over time a slight shift towards a new set including ALPN *h3-27,h3-29 and h3-34* besides Google QUIC versions can be seen for targets in 444 ASes. The set only consisting of *quic* was used more often at the beginning of our measurement period but mainly lost its share towards the end.

Regarding the ALPN sets retrieved from HTTPS DNS RRs, 99.9 % of the domains resolve to records including *h3-29, h3-28, h3-27*, the majorly used set from Cloudflare. As one of the driving forces behind the HTTPS DNS RR, Cloudflare is also dominating this dataset.

Key take-away. We find that based on announced versions, existing QUIC deployments were well-prepared for the final standardization of QUIC. Throughout our measurement period, the support for draft 29, the final draft supposed to be deployed [22], increased to 96 %. The activation of "Version 1" by Cloudflare but also other ASes even before the official conversion of the draft to an RFC shows that deployments were ready for the final standardization of IETF QUIC.

5 THE STATE OF QUIC DEPLOYMENTS

To analyze found QUIC deployments in more detail, we use the *QScanner* to complete QUIC handshakes with targets from ZMap, HTTP ALT-SVC Headers and HTTPS DNS RRs. We extract QUIC specific parameters, TLS configurations and HTTP headers.

For HTTP ALT-SVC Headers and HTTPS DNS RRs we only scan with SNI as we expect higher success rates and know at least one domain per IP address that indicates reachability using QUIC. As explained in Section 4, for 90 % of IPv4 and 38 % of IPv6 addresses, no domains were found by joining the data with our DNS scans. Therefore, we scan addresses from ZMap also without SNIs to test whether a QUIC connection can be established and to check the default behavior of targets without SNI.

To reduce the scan overhead, we only select targets, that announced a version compatible with the *QScanner*, namely draft 29, 32 and 34 (see Section 3.4). Similar to the results from Section 4 we use scans from calendar week 18 (May 3, 2021 until May 09, 2021). General results are shown in Table 3.

Table 3: Stateful scan results of combined sources.

	IPv4 (%)		IPv6 (%)	
	no SNI	SNI	no SNI	SNI
Success	7.25	76.06	27.66	90.70
Timeout	34.50	11.09	12.35	6.01
Crypto Error (0x128)	48.26	5.73	58.85	1.90
Version Mismatch	8.83	5.77	0.74	0.99
Other	1.16	1.35	0.40	0.39
Total Targets	2 M	17 M	210 k	14 M

NO SNI Scan. We scan 2 046 615 IPv4 (95.9 % of found IP addresses with ZMap) and 209 729 IPv6 addresses (99.4 %). For IPv4, 7.25 % or 148 281 connection attempts are successful, 34.5 % time out and 50 % result in the generic QUIC Alert 0x128, more precisely the generic TLS Alert 0x28 [43]. For IPv6, 27.7 % or 58 002 connection attempts are successful, only 12.3 % time out and 60 % result in the QUIC Alert `0x128`. Messages for alert `0x128` differ between scanned targets but do not reveal the exact reason why the handshake failed. We identify that the exact wording of the error message depends on their corresponding implementation, *e.g.*, we found the most prominent error message in the QUIC implementation from Cloudflare and the second most prominent string in the library from Google. We investigate this further in comparison to results from scans with SNI.

Interestingly, the handshake failed for 180 k (9 %) IPv4 addresses and 1.5 k (0.7 %) IPv6 addresses due to a version mismatch. While these addresses indicated during the ZMap scan to support a version compatible to the *QScanner*, the stateful scan failed with a version mismatch, *i.e.*, none of the offered versions are supported. 99 % of these are part of AS15169 or AS396982, both operated by Google. We re-scanned a subset of these targets with the ZMap module and the *QScanner* and found that the mismatch between version negotiation and QUIC handshake was reproducible and constant over a period of time. However, in August 2021, the behavior changed, and version mismatches can not be seen anymore. We talked about our observation with Google and concluded that observed inconsistencies were most likely due to an iterative roll-out of IETF QUIC within the Google network. Due to the complete deployment of

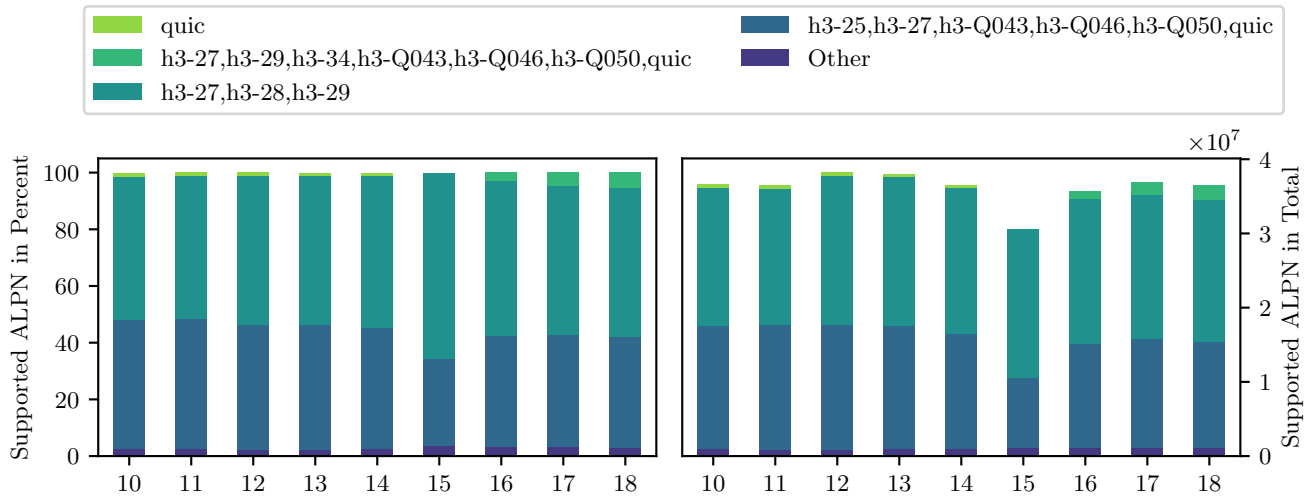

Figure 7: QUIC related ALPN values for domains from successful TLS-over-TCP IPv4 scans over several calendar weeks (x-axis) in 2021. *Other* combines all sets with an occurrence of less than 1 %.

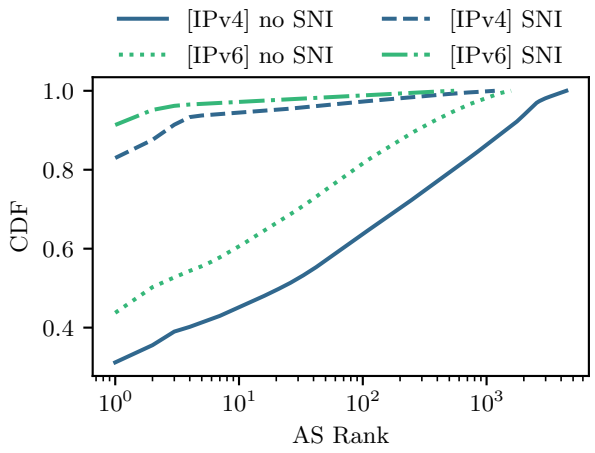

Figure 8: AS distribution of successfully scanned targets (Note the y-axis does not start at 0).

IETF QUIC by most Google services, these inconsistencies are not seen during new scans anymore.

As shown in Figure 8, IPv4 addresses with a successful scan still cover more than 4.4 k ASes (93.1 % of all seen ASes) even though the success rate is 7.25 %. Successful IPv6 address scans reach a similar coverage with at least one successful target in 92.6 % of the seen ASes.

SNI Scans. We combine found targets from all three sources for our scans with SNI. Besides filtering for targets with supported versions, we reduce the number of scanned domains per IP address to a maximum of 100 domains from each source for SNI scans as described in Appendix A. Therefore, we scanned 17 357 269 IPv4 targets consisting of 417 708 addresses in combination with 13 290 754 domains and 14 170 532 IPv6 targets consisting of 344 362 addresses and 10 176 968 domains.

Table 4: Individual success rate per input. Due to an overlap between sources, targets do not sum up to total scanned targets.

	IPv4		IPv6	
Source	Targets	Success	Targets	Success
ZMAP + DNS	14.4 M	85.6 %	14.1 M	85.3 %
ALT-SVC	14.1 M	85.2 %	13.7 M	84.9 %
HTTPS	6.2 M	77.6 %	6.0 M	77.0 %

The total IPv4 scan reaches a success rate of 76.1 % while 11.1 % of connection attempts time out, 5.7 % result in the QUIC alert `0x128` and another 5.8 % fail due to a version mismatch similar to the scan without SNI (see Table 3). The 13 M successful targets only account for 110 k addresses, 26.5 % of all scanned addresses. Furthermore, they are only part of 1.6 k ASes and 82.3 % are part of Cloudflare (AS13335).

Considering the IPv6 scan, the success rate is comparably high with 90.7 % but this only includes 90 k distinct addresses from 546 ASes. The most common errors are the QUIC alert `0x128`, a timeout and no compatible QUIC version in that order.

All three sources contribute to successful targets as shown in Table 4. We scanned 14 M million targets with IPv4 but also with IPv6 addresses each, from *(i)* ZMap joined with DNS and *(ii)* HTTP ALT-SVC Header with a respective success rate of 85 %. ZMap covers 105 k and HTTP ALT-SVC Headers cover 85 k distinct IPv4 addresses respectively. `HTTPS` DNS RR results in only 6.2 M targets based on IPv4 and IPv6 addresses respectively and reaches success rates of 77 %.

Key take-away. While many QUIC deployments can be found using stateless measures from Section 4, successful handshakes can

only be established to a subset of found hosts. The most unexpected error is the version mismatch for many targets from Google. While IETF versions are announced during the version negotiation, successful handshakes fail. After a discussion with Google, we were able to link the observed behavior, to an iterative roll-out of IETF QUIC throughout Google network services. While connection attempts have been impacted throughout the roll-out period, it was only temporary and a resolution of the error is visible with the deployment of the finally standardized version.

We argue that the large number of timeouts is either due to load balancers or due to the high duration of ZMap scans (see Section 3.1) and thus the large interval between version negotiation and stateful handshake for some targets. If `HTTPS` DNS RRs are deployed more widely in the future, it offers a reliable method to quickly detect QUIC service endpoints in the future and reduce the overall scan overhead for further studies drastically.

5.1 QUIC TLS Behavior compared to TLS over TCP

TLS is an intrinsic part of QUIC. Furthermore, due to changed requirements, *e.g.*, the new Transport Parameters extension and the necessity to use TLS 1.3, many QUIC implementations rely on custom TLS libraries. As a consequence, services reachable using QUIC and TLS over TCP might use different TLS stacks and configurations. Therefore, we evaluate the deployment of TLS as part of QUIC and compare it to TLS over TCP measurements for the same targets. As reported in Table 3, the stateful no SNI QUIC scans exhibit a low success rate. Our TLS over TCP scans can perform a successful TLS handshake for 43 % IPv4 and 50 % of the IPv6 targets. We analyzed this substantial difference and found that it is caused by only a handful providers. Google, Akamai, Cloudflare, and Fastly are nearly evenly responsible for more than 80 % (600 k) of cases where the TLS over TCP scan succeeds but the QUIC scan fails. We evaluated the QUIC scan errors in order to get more insight into parallel deployments and to rule out any error in our stateful QUIC scanner. While the Google errors are caused by a version mismatch as already described before, Akamai and Fastly run into a timeout using QUIC. We assume these timeouts are due to a similar reason Google returns IETF QUIC versions in the version negotiation but cannot complete a handshake afterwards. The same behavior could be observed for the same targets in multiple scans. In contrast, Cloudflare returns the TLS alert `0x128` reporting a handshake failure in these cases. As we are able to perform a successful QUIC scan with other Cloudflare IP addresses, we assume this is an issue on Cloudflare's side. To rule out any CDN specific issues with the no SNI scans we joined the failing IP addresses with any targets on the same address in our SNI scan. We found 42.5 k IP addresses in our SNI scan. We performed a successful QUIC handshake to only 2.8 k of these. Therefore, it seems most of these addresses are not actively used to deploy QUIC but are rather an artifact of CDN architectures. This enforces our idea of an early middlebox answering the QUIC version negotiation with the end hosts not being able to actually complete such a handshake.

We find a small number (less than 0.5 %) of targets successfully completing a QUIC scan but resulting in an error on the TLS over TCP scan. As this does not represent a relevant share and could also be caused by the two scans not running in parallel, we do not investigate it further.

Table 5: Share of hosts (%) using the same TLS properties on TLS over TCP and QUIC. All properties after the TLS version are made on targets where the TLS over TCP scan also performed a TLS 1.3 handshake.

	IPv4		IPv6	
	no SNI	SNI	no SNI	SNI
Certificate	31.7	98.1	17.7	98.2
TLS Version	99.6	99.7	99.8	99.7
Key Exchange Group	100.0	100.0	100.0	100.0
Cipher	99.2	100.0	100.0	100.0
Extensions	67.3	99.9	56.4	99.9

In Table 5 we compare the TLS over TCP scans with the QUIC results. The comparison of returned certificates can provide insights into the deployment strategy. We evaluate whether the certificate collected by the QUIC scan is used for the same set of domains as with the TLS over TCP scan. With the SNI scan we find more than 98 % of all targets returning the same certificate for QUIC but also TLS over TCP. Some certificates differ due to the delay between QUIC and TLS over TCP scans. However, we see a different picture for the no SNI scans. Only 31.7 % and 17.7 % of the targets for IPv4 and IPv6 respectively return the same certificates. Our evaluation of this artifact revealed that Google returns a self signed certificate with the common name indicating an error due to the SNI missing on TLS over TCP. When a QUIC handshake is performed to the same target, it returns a valid certificate. Moreover, we see the effect of Google rolling its certificates about weekly [6] which produces further certificate mismatches between QUIC and TLS over TCP.

We find few IP addresses which only offer TLS 1.2 over TCP while QUIC uses at least TLS 1.3 [43]. As QUIC implementations need a modified TLS library they often include it into their software while traditional HTTPS over TCP web servers usually allow the configuration of a TLS library. Therefore, the versions and exact deployment configurations can differ. We find that 99.7 % of all targets use the same TLS version (*i.e.*, TLS 1.3). The single most contributor to differing TLS versions is Cloudflare. We investigated this and found that using Cloudflare, it is possible to disable TLS 1.3 but enable QUIC. Moreover, we found that Cloudflare enables QUIC by default, which might be a reason for this behavior. To our knowledge there is no other reason to disable TLS 1.3 but enable QUIC.

As QUIC requires TLS 1.3, a comparison of TLS ciphers, key exchange groups and extensions is only useful if the chosen TLS over TCP version is the same. In order to have comparable results we made sure to send the same TLS *Client Hello* with our QUIC and TLS over TCP scanner. We offer the X25519 key exchange group which is accepted by close to all targets (*e.g.*, 206 of IPv4 SNI targets chose other curves). Only among those who did not chose X25519 there is in parts a discrepancy between QUIC and TLS over TCP. Although the vast majority uses the offered key exchange group, this shows on a small scale the effect of two different TLS deployments.

We got a similar result for the chosen cipher. Currently, TLS 1.3 has five allowed cipher suites, limited to four by the QUIC draft [43] and only three specified as required to be supported, hence it provides less choice compared to TLS 1.2. Most servers chose `TLS_AES_128_GCM_SHA256` in both scans.

QUIC requires a new TLS extension in order to transmit its transport parameters. We exclude that from our comparison to TLS over TCP to guarantee comparability. Furthermore, the latest draft of QUIC also requires using ALPN for application protocol negotiation unless there is another mechanism [44]. ALPN is also the single most significant extension which is missing in TLS over TCP no SNI scans while it is present in the QUIC scan. Again, Google and its edge deployments are the root cause for this observation.

The mismatch for the SNI scans is caused by a missing SNI extension on the TLS over TCP scans. According to RFC6066 [1] the server is required to return the extension if it used the name for certificate selection. The RFC does not actively forbid to send the extension if the value from the client is not used. As it is unlikely that a target on TCP 443 only serves a single domain but multiple on QUIC, we assume this uncritical gap in the standard leads to this observation.

Key take-away. We find that the TLS deployments on QUIC enabled hosts are very similar to the ones over TCP even though QUIC deployments are often based on new, dedicated implementations or forked TLS libraries. For a majority of QUIC deployments, the respective TLS over TCP deployments use TLS 1.3 with similar configurations. All major differences can be explained with new requirements by QUIC. As QUIC requires TLS 1.3 and most QUIC deployments are by large providers the overall state seems solid. We expect more diversity in deployments when adoption increases.

5.2 QUIC Configurations and Setups

We evaluate the previously omitted and newly specified TLS extension to transmit transport parameters [43] in more detail individually. While some transport parameters are session specific, *e.g.*, *stateless reset token*, others are implementation and / or configuration specific [43] and can be used to analyze deployments, thus we ignore options which contain tokens or connection IDs. In total, we see 45 different configurations, combining all parameters. All configurations are published with our results (see Section 1). Figure 9 depicts the usage of each configuration based on targets and ASes.

Most of the scanned domains reside within Cloudflare (AS13335), which all share the same parameter configuration (0 in Figure 9). The configuration is used by targets in 15 ASes in total. It mostly consists of the default values as specified in draft 34 [22], an *initial stream data* of 1 048 576 B and *initial max data* of a magnitude larger.

20 configurations are only used by a single AS each and in 50 % of the seen ASes, we only see a single configuration. Analyzing specific values reveals, that while for the parameters *active conn id limit*, *max ack delay* and *ack delay exponent* default values are mostly used, some parameters differ widely. For example, in the QUIC specification *max udp payload size* is by default set to the maximum UDP payload size (65 527 B). This value is used by 12 configurations, however 12 further configurations use 1500 B and overall, 10 different values can be seen. Furthermore, data transmission related parameters vary within multiple orders of magnitudes. While some deployments only promote 8192 B of *inital max data*, others support up to 16 777 216 B. For initial stream data, values still range in between 32 k and 10 M. However, these values can be updated throughout the connection.

Table 6: Top 5 HTTP Server values by the number of ASes with at least one target returning the value. #Parameters displays the number of distinct transport parameter configurations seen in combination with the Server value.

Server Value	#ASes	#Targets	#Parameters
proxygen-bolt	2224	46 421	4
gvs 1.0	1537	5664	1
LiteSpeed	238	23 846	2
nginx	156	10 526	16
Caddy	105	1526	1

Edge Point of Presences (POPs). Interestingly, targets in 42.2 % of the ASes use three configurations. To better understand these results, we include results from HTTP additionally collected by the *QScanner*. The HTTP HEAD request is successful for:

- IPv4 with SNI: 12.6 M (95.8 %)
- IPv4 without SNI: 104 k (70.4 %)
- IPv6 with SNI: 12.3 M (96.1 %)
- IPv6 without SNI: 36 k (62.2 %)

In total, we collect more than 8 k different HTTP Headers but focus on the HTTP Server header in the following. Even though it is often recommended to reduce information included in the Server header, it can contain hints about implementations. Besides the most frequent value, *Cloudflare*, some values can be seen from targets located in a wide variety of ASes.

As shown in Table 6, we see the value *proxygen-bolt* from targets in 2244 ASes. It indicates usage of the Facebook HTTP Library *Proxygen* [12] which provides QUIC based on the *mvfst* [11] implementation. The value is from successful connections with 50 k IP addresses, out of which we were able to associate 7.5 k addresses with a domain and scan with SNI. 95 % of domains contain either *fbcdn.net* or *cdninstagram.com*. Furthermore, they share four combinations of QUIC transport parameters not seen in combination with other HTTP server values. Two configurations are only used by targets located in the Facebook AS (AS32934). They allow a relatively high initial value for all stream data parameters with 10 485 760. They differ only in the `max_udp_payload_size` parameter with 1500 B and 1404 B respectively.

The remaining two configurations are mostly responsible for two out of three configurations seen in 42.2 % ASes as mentioned earlier. They respectively announce both payload sizes as well but further differ in the initial value for all stream data parameters with 67 584. We assume, the latter are edge POPs part of the Facebook CDN providing content close to the user [27]. Therefore, while these deployments are not hosted on Facebook ASes directly, they are most likely set up by Facebook as large provider.

The value *gvs 1.0* behaves similarly with 8.5 k IP addresses in 1.7 k ASes. While in most cases, we cannot associate a domain with

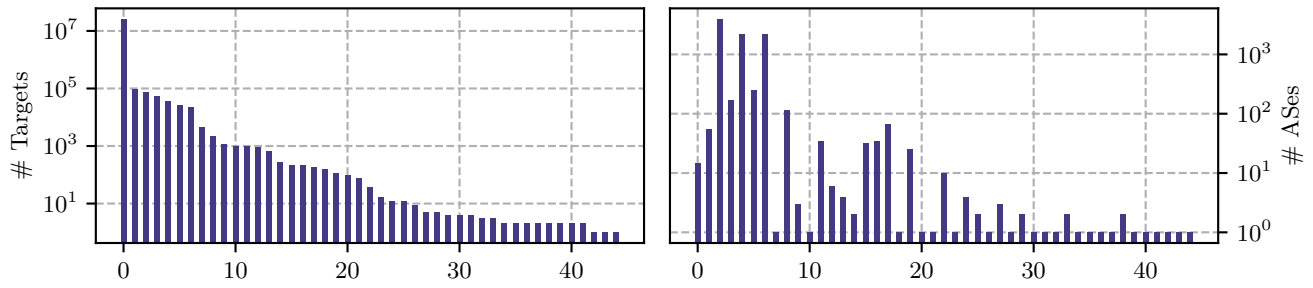

Figure 9: Distribution of transport parameter configurations ranked by number of targets.

these IP addresses, they all use the same set of transport parameters not seen with any other HTTP Server header values. Furthermore, this value is the third configuration, seen besides the values above for many ASes. 14 % of IP addresses are part of Google (AS15169) indicating a similar deployment with edge POPs.

This shows, that results about the deployment state of QUIC as shown in Section 4 can be misleading if further information about deployments is neglected. Combining the unique information collected by the *QScanner* allows to investigate QUIC in more detail throughout the early stages of the protocol. These additional insights show that the current deployment status is mainly dominated by large providers, not only in their own networks, but also due to POPs in external networks.

Diversity within single ASes. Single ASes do not necessarily need to provide a unique setup, *e.g.*, from cloud providers, a variety of configurations could be visible due to individual setups from customers. The highest number of different configurations in a single AS is 11 seen at Google (AS15169), Amazon (AS16509) and DigitalOcean (AS14061). All three offer cloud computing services allowing setups from customers. The configurations differ in most values and single configurations dominate for most of the hosts while the remaining ones are rarely seen. Considering HTTP Server header values reveals a large diversity, with 44 different values at Google, including *e.g.*, different NGINX versions or *Python/3.7 aiohttp/3.7.2*. For Amazon, only 12 values can be seen and 9 for DigitialOcean. While some of these deployments and corresponding server values are most likely set up by the provider themselves, we argue that others imply individual setups inside the cloud computing services.

Additional HTTP Server Values. Besides the earlier mentioned values, the third most common value is *LiteSpeed* (see Table 6) indicating a deployment based on LSQUIC [30]. It is used by 24 k domains in combination with 1.3 k IP addresses and 240 ASes. While most targets share the same configuration, no relation between domains or ASes can be found. Thus, in our data it is the most seen implementation not deployed by a single large provider.

The value *nginx* is present for 15 k targets but also as sub-string of the HTTP Server header value for 16 k targets on 7.8 k scanned IP addresses in combination with 17 different transport parameter combinations. Besides only *nginx*, *yunjiasu-nginx* is used by 15 k targets and the remaining values include a variety of different versions between 1.13.12 and 1.20.0. This reflects, that besides the official QUIC branch from NGINX starting after the release of version 1.17.8 [35], others have based HTTP3 implementations on NGINX forks, *e.g.*, Cloudflare [15].

Searching for further implementations listed by the QUIC working group [17] only reveals a few hits, *e.g.*, *h2o* used by 12 targets in five ASes including different commit hashes. Most implementations are either infrequently used or not revealed by the header value.

Key take-away. Using the QUIC specific *Transport Parameter* TLS extension allows analyzing and identifying deployments in more detail. Due to the variety of used configurations and individual parameters, we are able to identify deployments located in a multitude of ASes as edge POPs of large providers similar to the work from Gigis et al. [16] but based on a differing methodology. Taking these edge POPs into account, reveals that the deployment state of QUIC is even more focused towards large providers than shown in Section 4, solely based on originating ASes.

Furthermore, we argue that advertised transport parameters can be used to analyze deployments and their differences in more detail in the future. The availability of server preferences of relevant parameters regarding connection properties allows analyzing setups and the impact of different parameters on QUIC connections.

6 RELATED WORK

After the initial announcement of QUIC by Google in 2013 [23] and their initial study regarding the Internet-wide deployment by Google [29] little has been done to analyze the deployment of QUIC in its early stages. To our knowledge, the only large scale study, that investigates the deployment of QUIC on the Internet is from Rüth et al. [39]. However, they focused on the deployment and usage of Google QUIC versions in 2016 and early 2017. The IETF drafts were only in an early stage and not considered. They discover a steady growth of IP addresses with gQUIC support and an increasing traffic share mainly deployed by Google and Akamai as driving forces. Our work is similar in parts of our approach and overall goals but we set our self apart, by focusing on IETF QUIC versions, including IPv6 and investigating additional sources, namely HTTPS DNS RR and HTTP ALT-SVC Header, to discover QUIC capable targets. The inclusion of the latter two sources reveals additional QUIC deployments not found by the implemented ZMap module. Furthermore, our stateful approach allows the analysis of deployment characteristics like QUIC transport parameters and TLS configurations.

A subsequent study from Piraux et al. [38] introduces a test suite which scans targets on the Internet and evaluates specification conformity of QUIC implementations. They determine that 10 % to 20 % of the responses contain errors, which is similar to our SNI stateful scans but do not further present the distribution of error messages. In comparison, we investigate deployments in more detail, analyzing their current state independent of used implementations.

A recent study from Trevisan et al. [45] investigates the HTTP/3 adoption based on the HTTP ALT-SVC Header extracted from the open-source HTTPArchive Dataset but finds only 14 k websites with HTTP/3 support in December 2020. The remaining study focuses on a performance comparison between HTTP versions of found sites. Deploying active scans, we find that the current QUIC and HTTP/3 support is larger by multiple order of magnitudes.

A longitudinal analysis of the general TLS deployment can be found in Kotzias et al. [28]. They show the overall reaction to known high profile attacks (*e.g.*, Heartbleed) and that the ecosystem can react quickly to such. The deployment of TLS 1.3 throughout its standardization and early years was analyzed by Holz et al. [19]. Similar to our work, they find that new transport protocols are deployed quickly but mainly due to large providers activating server side support by default, *e.g.*, Cloudflare but also client side support, *e.g.*, Google and Mozilla. Based on their results, 75 % of domains with TLS 1.3 support are hosted at Cloudflare. They solely focus on TLS 1.3 over TCP and do not investigate QUIC. Our *QScanner* can be used to perform similar analyses for QUIC.

Gigis et al. [16] recently reported about hypergiants' off-nets and their development over several years. They used TLS certificates and HTTP header information from Internet-wide scans to identify edge POPs of large providers. They report extensive usage of edge POPs by large providers, *e.g.*, Google and Facebook. Using QUIC transport parameters in combination with HTTP Server header values, we independently identify similar deployments for QUIC.

Additional related work covering QUIC mainly focuses on security aspects [13, 31], the QUIC diversity due to the rapidly involving draft [38] and variety of available implementations [33] and performance analyses including comparisons to TLS and / or TCP [25, 26, 34, 38, 46].

7 DISCUSSION AND CONCLUSION

As a foundation for future research regarding the newly standardized, fundamental network protocol QUIC, our work provides a versatile tool set, to identify QUIC capable hosts and their properties. We presented an extensive analysis of different methodologies to detect the QUIC deployment state on the Internet shortly before the standardization. We verified that IETF QUIC already gained relevant traction before its final standardization, and we showed widespread QUIC deployment. Based on ZMap scans, HTTPS DNS RRs and HTTP ALT-SVC Header, we find deployments in more than 4.7 k ASes and are able to conduct successful QUIC handshakes with more than 26 M targets using the *QScanner*. We argue, that QUIC has the potential to change the Internet ecosystem drastically and highlight its importance to future Internet studies due to the extensive deployment by large network providers.

Stateful scans with the *QScanner* reveal that TLS as an integral part is similarly configured between QUIC and TLS over TCP stacks for the same target. In contrast, different implementations and configurations, with 45 transport parameter sets can be found on the Internet. A thorough analysis of these differences and their impact on network communications and especially user experience in the future is necessary to improve the Internet, support long-term deployment of QUIC and allow the evaluation of design decisions from the protocol specification.

The Dominance of CDNs. Similar to related work from Rüth et al. [39], a small group of providers dominates the deployment of QUIC. While their research reported in 2018, that a majority of found deployments could be associated to Google, our work shows, that the current state of IETF QUIC is mainly dominated by Cloudflare. Google is still highly involved in the development of QUIC but deploys its own version of QUIC in parallel. The dominance of large providers during the deployment of IETF drafts in their early years has also been shown by Holz et al. [19]. Mainly Cloudflare, but also Google, Akamai and Mozilla are the driving forces behind the quick deployment of TLS 1.3 on the Internet. While our work shows that QUIC capable hosts can be found in more than 4.7 k ASes and successful connections can be established with targets in 4.4 k ASes, the analysis of transport parameters and HTTP Server Header values indicates, that many of these are orchestrated by large CDNs as edge POPs similar to the work from Gigis et al. [16].

We argue that it has to be considered carefully by research in the future and leads to substantial centralization. Measurement studies are easily biased towards these providers. Operators cannot solely be identified based on ASes but might be responsible for distributed deployments. Nevertheless, it can be seen that QUIC as new protocol is used by individuals even before the QUIC draft is finally standardized and even though prominent HTTP servers, *e.g.*, NGINX [35], only provide QUIC support on specific branches. With the standardization of QUIC and increased deployment, its status needs to be further evaluated in the future.

Fingerprinting QUIC. Based on presented results, we argue that the combination of functionality from multiple layers of the network stack into a single protocol increases the possibility to fingerprint specific implementations. As long as many QUIC stacks implement transport functionality, necessary TLS adaptations and HTTP servers on top individually, the number of parameters pointing towards a specific implementation is comparably higher than for traditional HTTP servers with exchangeable TLS libraries built on top of an independent TCP stack. As shown in Section 5 we find 45 sets of QUIC parameters, out of which, some are closely related to specific providers. Further adding TLS properties and HTTP results allowed us to identify edge POP deployments of specific providers. Whether this persists in the future or whether the standardization leads to a separation of functionality, *e.g.*, with TLS specific libraries adapting to new requirements should be evaluated.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and our shepherd Steve Uhlig for their valuable feedback. This work was partially funded by the German Federal Ministry of Education and Research under the project PRIMEnet, grant 16KIS1370 and the German-French Academy for the Industry of the Future.

REFERENCES

[1] Donald E. Eastlake 3rd. 2011. *Transport Layer Security (TLS) Extensions: Extension Definitions.* RFC 6066.

[2] Alexa. 2021. Top 1M sites. https://www.alexa.com/topsites

[3] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. 2017. Mission Accomplished? HTTPS Security after Diginotar. In *Proceedings of the 2017 Internet Measurement Conference (IMC '17)*. Association for Computing Machinery, New York, NY, USA, 325–340.

[4] Mike Bishop. 2021. *Hypertext Transfer Protocol Version 3 (HTTP/3).* Internet-Draft draft-ietf-quic-http-34. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34 Work in Progress.

[5] Cisco. 2021. Umbrella Top 1M List. https://umbrella.cisco.com/blog/cisco-umbrella-1-million

[6] crt.sh. 2021. Certificates for Google Video in CT Log. Retrieved 2021-05-27 from https://crt.sh/?q=googlevideo.com https://web.archive.org/web/20210526164544/https://crt.sh/?q=googlevideo.com.

[7] Dragana Damjanovic. 2021. QUIC and HTTP/3 Support now in Firefox Nightly and Beta. Retrieved 2021-09-27 from https://hacks.mozilla.org/2021/04/quic-and-http-3-support-now-in-firefox-nightly-and-beta/

[8] David Schinazi and Fan Yang and Ian Swett. 2020. Chrome is deploying HTTP/3 and IETF QUIC. Retrieved 2021-09-27 from https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html

[9] David Dittrich, Erin Kenneally, et al. 2012. The Menlo Report: Ethical principles guiding information and communication technology research. *US Department of Homeland Security* (2012).

[10] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *Proc. USENIX Security Symposium*. Washington, D.C., USA.

[11] Facbook. 2021. mvfst. https://github.com/facebookincubator/mvfst

[12] Facbook. 2021. Proxygen: Facebook's C++ HTTP Libraries. https://github.com/facebook/proxygen

[13] Marc Fischlin and Felix Günther. 2014. Multi-Stage Key Exchange and the Case of Google's QUIC Protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1193–1204.

[14] Oliver Gasser, Quirin Scheitle, Pawel Foremski, Qasim Lone, Maciej Korczyński, Stephen D. Strowes, Luuk Hendriks, and Georg Carle. 2018. Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. Association for Computing Machinery, New York, NY, USA, 364–378.

[15] Ghedini Alessandro. October 17, 2019. Experiment with HTTP/3 using NGINX and quiche. https://blog.cloudflare.com/experiment-with-http-3-using-nginx-and-quiche/

[16] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven Years in the Life of Hypergiants' off-Nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 516–533.

[17] IETF QUIC Working Group. 2021. Implementations. https://github.com/quicwg/base-drafts/wiki/Implementations

[18] IETF QUIC Working Group. 2021. QUIC Versions. https://github.com/quicwg/base-drafts/wiki/QUIC-Versions

[19] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. 2020. Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization. *SIGCOMM Comput. Commun. Rev.* 50, 3 (July 2020), 3–15.

[20] ICAN. 2021. Centralized Zone Data Service. https://czds.icann.org/home

[21] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. https://doi.org/10.17487/RFC9000

[22] Jana Iyengar and Martin Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft draft-ietf-quic-transport-34. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34 Work in Progress.

[23] Jim Roskind. June 27, 2013. Experimenting with QUIC. https://blog.chromium.org/2013/06/experimenting-with-quic.html

[24] Matt Joras and Yang Chi. 2020. How Facebook is bringing QUIC to billions. Retrieved 2021-09-27 from https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/

[25] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols. In *Proceedings of the 2017 Internet Measurement Conference (IMC '17)*. Association for Computing Machinery, New York, NY, USA, 290–303.

[26] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2019. Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols. *Commun. ACM* 62, 7 (June 2019), 86–94.

[27] Kim, Hyojeong and Zeng, James Hongyi. August 21, 2017. Steering oceans of content to the world. https://research.fb.com/steering-oceans-of-content-to-the-world/

[28] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *Proceedings of the Internet Measurement Conference 2018 (IMC '18)*. Association for Computing Machinery, New York, NY, USA, 415–428.

[29] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196.

[30] LiteSpeed QUIC Team. 2021. LiteSpeed QUIC (LSQUIC) Library. https://github.com/litespeedtech/lsquic

[31] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. 2015. How Secure and Quick is QUIC? Provable Security and Performance Analyses. In *2015 IEEE Symposium on Security and Privacy*. 214–231.

[32] Majestic. 2021. The Majestic Million. https://majestic.com/reports/majestic-million/

[33] Robin Marx, Joris Herbots, Wim Lamotte, and Peter Quax. 2020. Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ '20)*. Association for Computing Machinery, New York, NY, USA, 14–20.

[34] Késsia Nepomuceno, Igor Nogueira de Oliveira, Rafael Roque Aschoff, Daniel Bezerra, Maria Silvia Ito, Wesley Melo, Djamel Sadok, and Géza Szabó. 2018. QUIC and TCP: A Performance Evaluation. In *2018 IEEE Symposium on Computers and Communications (ISCC)*. 00045–00051.

[35] NGINX QUIC. 2021. Welcome to the demo site for nginx-quic. https://quic.nginx.org/

[36] Mark Nottingham, Patrick McManus, and Julian Reschke. 2016. *HTTP Alternative Services.* RFC 7838.

[37] Craig Partridge and Mark Allman. 2016. Addressing Ethical Considerations in Network Measurement Papers. *Commun. ACM* 59, 10 (Oct. 2016).

[38] Maxime Piraux, Quentin De Coninck, and Olivier Bonaventure. 2018. Observing the Evolution of QUIC Implementations. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ'18)*. Association for Computing Machinery, New York, NY, USA, 8–14.

[39] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. 2018. A First Look at QUIC in the Wild. In *Passive and Active Measurement*. Springer International Publishing, 255–268.

[40] David Schinazi and Eric Rescorla. 2021. *Compatible Version Negotiation for QUIC.* Internet-Draft draft-ietf-quic-version-negotiation-03. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-version-negotiation-03 Work in Progress.

[41] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. 2021. *Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs).* Internet-Draft draft-ietf-dnsop-svcb-https-05. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-svcb-https-05 Work in Progress.

[42] Marten Seemann and Jana Iyengar. 2020. Automating QUIC Interoperability Testing. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ '20)*. Association for Computing Machinery, New York, NY, USA, 8–13.

[43] Martin Thomson and Sean Turner. 2021. Using TLS to Secure QUIC. RFC 9001. https://doi.org/10.17487/RFC9001

[44] Martin Thomson and Sean Turner. 2021. *Using TLS to Secure QUIC.* Internet-Draft draft-ietf-quic-tls-34. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-ietf-quic-tls-34 Work in Progress.

[45] Martino Trevisan, Danilo Giordano, Idilio Drago, and Ali Safari Khatouni. 2021. Measuring HTTP/3: Adoption and Performance. *CoRR* abs/2102.12358 (2021). arXiv:2102.12358 https://arxiv.org/abs/2102.12358

[46] Konrad Wolsing, Jan Rüth, Klaus Wehrle, and Oliver Hohlfeld. 2019. A Performance Perspective on Web Optimized Protocol Stacks: TCP+TLS+HTTP/2 vs. QUIC. In *Proceedings of the Applied Networking Research Workshop (ANRW '19)*. Association for Computing Machinery, New York, NY, USA, 1–7.

A ETHICS

All our scans are set up based on a set of ethical measures we follow strictly. These are mainly based on informed consent [9] and well known best practices [37]. Our study does not involve users, their information or sensitive data but focuses on publicly reachable and available services. To not cause harm to any infrastructure, we apply measures described by Durumeric et al. [10]. We limit the rate of our scans and use a collective blocklist based on requests to be excluded from our scans. We are directly registered as abuse contact for our scan infrastructure and react quickly to all requests. Furthermore, we host websites on all IP addresses used for scanning to inform about our research and provide contact information for further details or scan exclusion.

As explained in Section 2, *Initial* packets need to be at least 1200 B. This increases the overall traffic from our scans in comparison to most TCP ZMap scans but mainly impacts our own uplink to the Internet. Due to the randomization of scanned targets, we argue that the impact on servers is still small. Furthermore, we limit the number of scanned domains per IP address to reduce the load on hosting services and providers.

B IMPORTANT AS NAMES

Table 7 provides a summary of important ASes and their according names.

Table 7: Important ASes and according names.

AS	Name
AS5606	GTS Telecom SRL
AS8560	1&1 IONOS SE
AS13335	Cloudflare, Inc.
AS14061	DigitalOcean, LLC
AS15169	Google LLC
AS16276	OVH SAS
AS16509	Amazon.com, Inc.
AS20940	Akamai International B.V.
AS45638	SYNERGY WHOLESALE PTY LTD
AS47583	Hostinger International Limited
AS54113	Fastly
AS55293	A2 Hosting, Inc.
AS55836	Reliance Jio Infocomm Limited
AS63410	PrivateSystems Networks
AS63949	Linode, LLC
AS209242	Cloudflare London, LLC
AS210079	EuroByte LLC