A QUIC Look Back: Longitudinal Internet-Scale Deployment and Transport Parameter Evolution

Marcel Kempf
kempfm@net.in.tum.de
Technical University of Munich
Munich, Germany

Nikolas Gauder
gauder@net.in.tum.de
Technical University of Munich
Munich, Germany
NVIDIA Corporation
Roskilde, Denmark

Johannes Späth
spaethj@net.in.tum.de
Technical University of Munich
Munich, Germany

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

Johannes Zirngibl
jzirngib@mpi-inf.mpg.de
Max Planck Institute for Informatics
Saarbrücken, Germany

Abstract

The Internet Engineering Task Force (IETF) specified QUIC in 2021, marking a significant shift in the Internet transport landscape. With its user-space implementation and flexible transport parameters, QUIC promised to enable rapid deployment of new extensions and features, fostering transport protocol innovation. While early studies captured the initial surge of deployment, long-term evolution and maturation of the QUIC ecosystem remain largely unexplored.

In this paper, we present a comprehensive longitudinal study of QUIC based on a five-year dataset of weekly Internet-wide scans from April 2021 to February 2026. We use fingerprinting to identify 22 distinct libraries and track their market share over time. Further, we analyze the diversity and evolution of transport parameter configurations and the adoption of new extensions.

Our analysis reveals a steady growth in QUIC adoption across both IPv4 and IPv6. Additionally, we show that while the ecosystem is changing, a significant "long tail" of outdated drafts and default configurations persists. This work provides the first multi-year overview of the QUIC landscape, offering insights into its volatility, the speed of protocol updates in user space, and the use of QUIC's flexible transport parameters on the Internet.

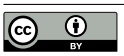
CCS Concepts

• **Networks** → **Transport protocols**; **Public Internet**.

Keywords

QUIC, Libraries, Internet Measurement, Transport Parameters

ACM Reference Format:
Marcel Kempf, Nikolas Gauder, Johannes Späth, Georg Carle, and Johannes Zirngibl. 2026. A QUIC Look Back: Longitudinal Internet-Scale Deployment and Transport Parameter Evolution. In *Proceedings of the 2026 ACM Internet Measurement Conference (IMC '26), October 12–16, 2026, Karlsruhe, Germany.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3777912.3839803



IMC '26, Karlsruhe, Germany

ACM ISBN 979-8-4007-2327-8/26/10
https://doi.org/10.1145/3777912.3839803

1 Introduction

QUIC was finally specified by the Internet Engineering Task Force (IETF) in 2021 as RFC 9000 [16]. Related work has shown a widespread deployment during the draft phase [29] and during the release of the RFC [38]. QUIC builds upon UDP and can thus be implemented in user space. As of 2021, more than 20 implementations already existed, and Zirngibl et al. [39] showed in 2023 that the majority of these implementations are also visible on the Internet. Besides the diversity in implementations, the user-space property allows for fast updates if new features are drafted. Widespread deployment is possible by simple software updates without waiting for kernel updates. Besides the potential diversity through available implementations or software versions, QUIC further enhances agility by introducing transport parameters that can be configured to change the behavior of an endpoint. An example is the maximum UDP payload size a server is willing to accept.

Previous work has evaluated individual features, *e.g.*, the adoption of Explicit Congestion Notifications (ECNs) [30] or the spin bit [19], or have shown the heterogeneity of configurations based on a single snapshot [38]. However, to the best of our knowledge, no work has yet evaluated the development of QUIC deployments over time, the adoption of different libraries, new features and drafts, and the changing multiplicity of configurations.

Knowledge about the actual diversity of the QUIC landscape on the Internet and its volatility is important to several groups. Client and library implementers must work with what operators deploy, not with what the RFCs specify: most deployments that announce acknowledgment frequency still use a draft identifier the IETF replaced in 2023 (see Section 7.2), so a client that implements only the current one loses the extension for most peers. Operators can compare their own settings against the rest of the Internet, as transport parameters decide how many resources a server commits per connection, and we find flow control limits and idle timeouts far beyond any library default. Draft authors learn how long a new parameter takes to reach deployment and how long an old identifier survives, and studies that measure or emulate QUIC learn which library defaults still hold. In contrast to a snapshot, a longitudinal view separates a feature that spreads quickly from one that merely exists. QUIC makes updates and configuration changes easy, but

we do not know whether operators use this freedom or stay with the defaults and features of the initial RFC.

Therefore, this work answers the question: *How did the QUIC ecosystem and its diversity develop over the last five years?* More specifically, we ask:

- **RQ1** How did the deployment of QUIC servers evolve over the last five years and which libraries are used?
- **RQ2** How diverse are QUIC deployments with respect to their transport parameter configurations?
- **RQ3** How are new (draft) transport parameters adopted and by whom?

Our key contributions in this work are:

(i) We use a five-year dataset of weekly Internet-wide QUIC scans to evaluate the development of QUIC and the use of different libraries. We show an increase of visible QUIC targets by 5× for IPv4 and 50× for IPv6. The distribution across libraries changes slightly over time with NGINX and quic-go experiencing the highest growth in the recent months.

(ii) Based on transport parameters shared by servers, we evaluate the diversity of QUIC deployments, the visibility of different configurations, and changes over time. We show that while only around 50 different configurations were visible in 2021, 1162 configurations are visible in 2026. The `initial_max_data` transport parameter is the most diverse one with almost 500 different values.

(iii) We evaluate the adoption of new transport features based on new drafts and RFCs. While we demonstrate that some features are adopted quickly, we provide empirical evidence of protocol "lag", showing that outdated draft versions can persist in the wild for up to four years.

We introduce required background in Section 2 and discuss related work in Section 3. Section 4 introduces the used dataset. To answer our research questions, we evaluate the growth of QUIC and used libraries in Section 5. Section 6 evaluates the use of different QUIC transport parameters and configurations, while Section 7 evaluates the adoption of new (draft) transport parameters. We discuss our findings in Section 8 and conclude in Section 9.

2 Background

Standardized as RFC 9000 [16] by the IETF in 2021, QUIC combines transport functionality and security while increasing flexibility.

Transport Parameters: Transport parameters allow both endpoints to transmit and authenticate connection specific parameters during the QUIC handshake [16]. These parameters are not negotiated but announced unilaterally. Therefore, the transport parameters sent by a scanner (as client) should not influence the response of the server. We verify this independence empirically in Section 4.4, confirming that varying the transport parameters sent by the scanner does not affect server responses. Transport parameters unknown to an endpoint must be ignored, which allows the introduction of new parameters without breaking backward compatibility. RFC 9000 [16] initially specified 17 QUIC transport parameters, covering essential functions such as flow control. As of April 2026, the Internet Assigned Numbers Authority (IANA) [14] lists four additional permanent transport parameters, *i.e.*, `version_information` [32], `max_datagram_frame_size` [28], `initial_max_path_id` [21], and `grease_quic_bit` [36]. Furthermore, 10 provisional parameters are listed.

While mainly used by endpoints to expose capabilities and constraints, transport parameters have different purposes. Most configure connection parameters, *e.g.*, `max_udp_payload_size` or `max_idle_timeout`. Others provide information for future connections, *e.g.*, `preferred_address`. In contrast, `original_destination_connection_id` is used by a server to mirror the initial destination connection ID selected by a client. This value must be used by the client to verify that the connection attempt was not tampered with. Similarly, `retry_source_connection_id` is used to verify the connection ID used in a Retry packet. However, they do not impact the configuration of the established connection. Finally, specific identifiers ($31 * n + 27$) are reserved and often used to exercise the other endpoint's ability to ignore unknown parameters and thus prevent ossification of the protocol. This technique is referred to as GREASE [2].

TLS: QUIC strictly relies on TLS to authenticate a connection partner and establish a secure channel during the handshake. The TLS handshake is directly included in the QUIC handshake. Thus, the client sends all required information to set up a secure channel in the QUIC Initial packet, but can also add information about the requested services, *i.e.*, its name as Server Name Indication (SNI) or the requested application layer protocol as Application-Layer Protocol Negotiation (ALPN) value. The server uses the latter information to check if it can authenticate itself for this service with a valid certificate. In case of failure, it can send an error message or time out. If no SNI is present, the server can use its default certificate (if configured), fail, or time out. Therefore, successful handshakes with QUIC servers often require a valid domain as SNI value, as shown by Zirngibl et al. [38, 39]. The data used for this paper covers both scans with and without SNI values (see Section 4).

The integration of TLS into the handshake also allows the server to switch to the encrypted channel quickly. Only its Server Hello with the key material is passively observable. All remaining information, *e.g.*, the certificates and further extensions including the transport parameters by the server, are encrypted. Therefore, active scans are required to collect server information and evaluate configuration changes in detail.

3 Related Work

Rüth et al. [29] evaluated QUIC deployments before the release of RFC 9000, mostly focusing on Google QUIC. They analyzed early deployments throughout one year (2016–2017) and found that initial deployments were driven by large Content Delivery Networks (CDNs). While they showed a quick adoption of new draft versions, they did not evaluate configurations or changes. Zirngibl et al. [38] used a similar approach in 2021 to evaluate the state of QUIC deployments shortly before the release of RFC 9000. They found increased deployment numbers compared to Rüth et al. [29] and identified 45 different transport parameter configurations. In 2023, Zirngibl et al. [39] extended their work on QUIC scans and developed a methodology to identify the used QUIC library of a server based on the order of received transport parameters. They conducted an Internet-wide measurement study to evaluate the state of used libraries, but did not evaluate configured parameters

or changes over time. This work relies on data collected by Zirngibl et al. [38, 39] but focuses on the longitudinal development of the QUIC ecosystem and the increasing diversity of configurations.

Besides these broad scanning studies, related work focused on specific features of QUIC. Marx et al. [23] compared different libraries and highlighted differences. Gbur and Tschorsch [10] evaluated QUIC libraries focusing on security aspects in local tests. However, both studies focus on the libraries themselves and not on deployments seen on the Internet. Nawrocki et al. [26] evaluated whether QUIC libraries adhere to the amplification limit (3×) of QUIC [16] during the handshake. They showed that different libraries and different operators do not necessarily adhere to the limit but send too large certificates before validating the client. Kunze et al. [19] specifically focused on the spin bit [16, 20], a passively observable bit that allows RTT estimates. Sander et al. [30] focused on ECN. Both studies evaluate data covering around a year but focus specifically on these two features. Buchet and Pelsser [6] analyzed whether servers offer successful connection migration. Luvizotto Cesar et al. [22] focused on services offered via anycast deployments, including QUIC endpoints. While all of these studies evaluate different QUIC features, they do not investigate deployments over larger periods of time and do not focus on the diversity of configurations and transport parameters.

Mücke et al. [25] used a passive approach to evaluate the QUIC ecosystem based on backscatter in a telescope and passive network traces. They showed differences, *e.g.*, the usage of connection IDs and deployment sizes. While they crosschecked findings based on active scans, their work focuses on passive data. Thus, they could not evaluate transport parameters.

Besides the evaluation of QUIC, related work looked at the adoption of TLS over time [12, 18]. They showed that changes in the ecosystem can be fast, *e.g.*, in reaction to Heartbleed [18], and are often driven by hypergiants [12]. In this work, we focus on QUIC and its use of TLS.

4 Dataset

For this work, we use a dataset of weekly QUIC scans collected over five years. The scans were initially set up for a previous study [38]. In this section, we briefly describe how the data is collected and provide an overview of available data. For a detailed description of the scans and the used tools we refer to the original work [38]. We discuss limitations of this dataset in Section 8 and ethical considerations in Appendix A.

4.1 Data Collection

Zirngibl et al. [38] set up a weekly QUIC scan running since April 2021. Each scan runs in three phases. *(i)* A ZMap scan sends a version negotiation probe to every target and thus finds addresses that support QUIC and answer on UDP port 443. *(ii)* The stateful QScanner completes a full QUIC handshake with each responsive address, first without an SNI value. *(iii)* The scanner joins every address that supports QUIC with up to 100 domains that resolve to it and repeats the handshake with each domain as SNI value. All phases target UDP port 443, the default port of HTTP/3, and the scanner uses `h3` as ALPN value.

The two address families use different input sets for the first phase. For IPv4, a full /0 ZMap scan covers the whole address space. The IPv6 scans rely on the *IPv6 Hitlist* [9, 40] and on IPv6 addresses from `AAAA` records. The *IPv6 Hitlist* filters aliased prefixes, so these are generally excluded. Single addresses inside such prefixes can still enter the target set through the Domain Name System (DNS) input, namely when a domain resolves to individual addresses specifically, as suggested by Zirngibl et al. [40]. The IPv6 results therefore depend on this input set, and a change over time can follow from a changing input set instead of a changing deployment. We discuss which results this affects in Section 5.2 and Section 8.

To obtain domains and IPv6 addresses for the SNI scans, the scan pipeline resolves more than 400 M domains regularly and joins the `A` and `AAAA` records with the ZMap results. These domains come from zone files of the Centralized Zone Data Service (CZDS), from Certificate Transparency Logs (CT logs), and from top lists. The pipeline resolves them from the same vantage point that runs the scans, so a deployment that a DNS answer hides from this location stays invisible to us.

Our analysis uses only scans that end in a successful handshake, as only these reveal the transport parameters of a server and thus provide insights into the actual deployment and configuration of QUIC servers. An address that answers the version negotiation probe but never completes a handshake, *e.g.*, because it requires an SNI value that the domain set does not contain, does not appear in our results. All numbers in this paper are therefore lower bounds.

4.2 Library Classification

We rely on the methodology proposed by Zirngibl et al. [39] to identify the used QUIC library of successfully scanned targets. Zirngibl et al. have shown that the order of transport parameters is library specific and can therefore be used for library identification. They have shown that the fingerprint for each library is stable, at least for one year from 2023 until 2024. We use their test environment based on docker containers from library maintainers to verify that current versions have the same fingerprint. Our dataset, however, covers five years, and we can only test the versions that maintainers still provide, so we cannot rule out that a library changed its fingerprint in the earlier years. Such a change would show up as a target that switches library, and we find that more than 99.97 % of the targets seen at least twice never do (see Section 5.3), which bounds how often this can have happened. During this analysis, we observed a second fingerprint for XQUIC, which is not distinguishable from Akamai QUIC. Looking at custom transport parameter identifiers, we expect 1786 unique IPv4 addresses and 1317 unique IPv6 addresses to be running XQUIC instead of Akamai QUIC. Furthermore, we add new fingerprints for OpenSSL, TQUIC, and Kwik. This allows us to identify 22 QUIC libraries. Since Cloudflare and Google use the same name for their QUIC libraries, we call the Cloudflare implementation "quiche" and the Google implementation "gQUICHE". If we cannot unambiguously identify the library, we use a slash between the two library names. However, those collisions are rare, and we typically focus only on the top N libraries.

In this work, we do not rely on the fingerprinting methodology based on error messages with respect to an invalid ALPN value. While this methodology was proposed by Zirngibl et al. [39] besides

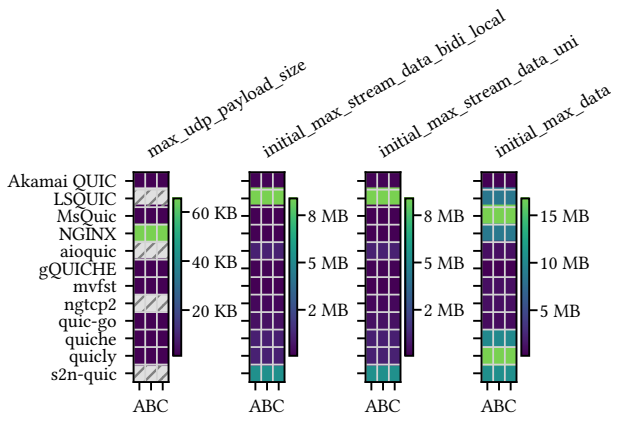


Figure 1: Median server-advertised transport parameter value per library and client configuration (A = low, B = mid, C = high client hints). Uniform color within a row confirms that server values do not adapt to client-advertised transport parameters. Hatched cells indicate unavailable data for a library/parameter combination.

the order of transport parameters, required data is not available for the same period of time.

4.3 Data Overview

The dataset available to us spans approximately five years, from the first scan on April 16, 2021 to the most recent scan on February 21, 2026. In total, the dataset includes 239 weeks of active measurements. More than 12 billion QUIC targets were scanned and more than six billion successful handshake records were collected. This massive volume of data allows us to observe not just snapshots, but the evolution of protocol adoption and configuration changes.

In the IPv4 dataset, about 75 % of the IP addresses need an SNI value for a successful handshake. Servers often require this value to select the correct service and certificate to authenticate themselves. While the dataset includes up to 100 targets per IP address, we focus on unique IP addresses for our analysis to avoid skewing results. The distribution of the number of targets per IP address is shown in Appendix Figure 15. The number of domains per IP address is impacted by the set of domains used as input and load balancing behavior of CDNs. In Section 5.1, we show that library and configuration rarely differ between targets of the same IP address, which supports our focus on unique IP addresses.

4.4 Transport Parameter Independence

RFC 9000 [16] permits an endpoint to choose arbitrary values for each transport parameter within allowed ranges. While parameters should have unilateral, connection-independent values, a server could adapt its behavior to the client, *e.g.*, mirror or scale its transport parameter values to adapt its flow control windows, based on the client's advertised limits. To eliminate any impact of the scanner on the results, we test the behavior of different libraries. The available data is based on a static scanner with a single client configuration. For a small target sample, we test three QUIC handshakes advertising deliberately different client-side transport parameter values: a configuration with *(i)* low values (config A), *(ii)* medium values (config B), and *(iii)* large/extreme values (config C). The exact configurations are given in Appendix Table 4. We examine whether the server's advertised transport parameters differ across the three distinct probes.

We scan a stratified sample of approximately 100 endpoints per library, yielding 881 targets for which all three configurations resulted in successful handshakes (we exclude eight targets that only had partial completions). For every target and parameter, we check whether the server returns the same value for all three probes. Figure 1 visualizes the result as a heatmap of median server-advertised values per library and client configuration. The uniform color across configurations within each panel confirms that server values are entirely invariant to client input. The independence rate is 100 % across all tested parameters and libraries: not a single server adapted any of its advertised transport parameters in response to the client's parameter value choices.

These results show the independence of server parameters from client behavior and suggest that our later findings are generalizable to the current QUIC ecosystem.

5 The Rise of QUIC

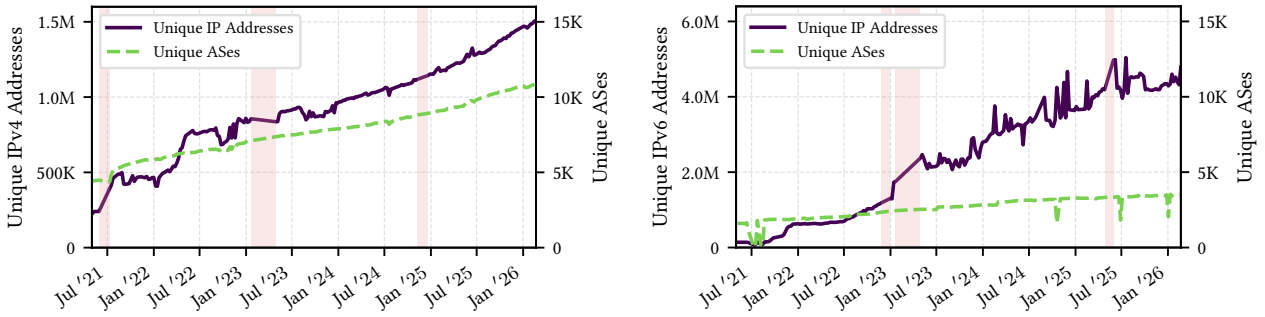
The available data allows us to evaluate the development of QUIC deployments and potential changes in used libraries over a five-year period. In this section, we analyze the longitudinal growth of QUIC adoption and the evolving landscape of its implementations.

5.1 Consistency of Results per IP Address

The scans cover each IP address without an SNI value and with up to 100 domains, while we analyze IP addresses and Autonomous Systems (ASes) rather than individual domains. To check that this does not blur differences between targets of the same address, we measure how often the library or configuration differs within one address. We find that the share of IP addresses exhibiting multiple libraries is less than 0.01 % for more than 99 % of the scans in both the IPv4 and IPv6 datasets. The actual share is likely lower, as randomized parameter ordering and library transitions during the scan period inflate this value. The same applies for the transport parameter configurations, where we find that the share of IP addresses with multiple transport parameter configurations is less than 0.7 % for more than 99 % of the scans in both datasets. The slightly higher variance has two reasons. *(i)* quiche targets in Cloudflare's AS13335 inconsistently set `disable_active_migration`, potentially due to load balancing. *(ii)* Two scan weeks exhibit large shares of differences. We assume that deployment changes during these scans hit randomized targets.

5.2 Longitudinal Deployment Identification

Figure 2 illustrates the growth of QUIC deployments over time for IPv4 and IPv6. Intervals marked in red represent temporary outages of the scanning infrastructure that affected more than one scan. These were mainly caused by maintenance work and upgrades of the scanning environment. Figure 2a shows that the growth of unique IPv4 addresses and ASes is relatively uniform over time. While successful QUIC handshakes were possible with 246 111 addresses located in 4426 ASes in April 2021, the numbers increased



(a) IPv4 **(b) IPv6**

Figure 2: Growth of unique addresses and Autonomous Systems (ASes) scanned successfully over time. Outages of the scanning infrastructure are marked in red.

to 1 509 735 addresses in 10 898 ASes in February 2026. For IPv6 (see Figure 2b), a steeper growth is visible from 89 605 addresses (540 ASes) to 4 785 067 addresses (3494 ASes). While handshakes are successful with a higher number of distinct IPv6 addresses, they are only distributed across a fraction of the ASes. On one hand, the IPv6 scans rely on target lists instead of a /0 scan (see Section 4), so deployments might be missed. On the other hand, the IPv6 data is more biased towards some CDNs. Parts of the evaluation therefore use only the IPv4 dataset, which is more representative of the overall QUIC landscape. The growth in Figure 2b mixes new deployments with a growing input list, so the factor of 50 is an upper bound for the growth in deployments. The library shares in Figure 4b and the configuration counts in Figure 6 follow the two ASes that hold more than 90 % of the IPv6 targets. Libraries of hypergiants such as gQUICHE and quicly therefore appear small in the IPv6 shares, which says more about the composition of the target set than about their deployments. An IPv6 share also depends on how an operator uses addresses, not only on how much it deploys: Amazon and Hostinger seem to spread services across many rotating addresses, while others serve the same content from few stable ones.

To quantify the impact of CDN deployments, we analyze the cumulative distribution of unique IP addresses supporting QUIC per AS per year in Figure 3. The x-axis ranks the ASes logarithmically by their number of unique IP addresses in 2026, and the y-axis accumulates the share of addresses they hold. An early and steeply rising curve indicates that targets are concentrated in few ASes. For each year, we only took data from the first scan to avoid biasing the results with the target lifetime differences between larger and smaller ASes. Looking at the IPv4 dataset, we can see that the top 10 ASes account for more than 50 % of the unique IP addresses in every year. The largest AS alone accounts for more than 20 % of the unique IPv4 addresses each year. See Appendix Table 2 for a list of ASes, targets, and their libraries. These findings underscore the significant influence of a few large CDNs on the QUIC landscape, as they manage a substantial portion of the Internet's traffic and, as shown, also the majority of QUIC-capable IPv4 addresses.

For IPv6, the most prominent AS is AS47583 (Hostinger), closely followed by AS16509 (Amazon). Together, they account for more than 90 % of the targets in each scan. Both ASes already dominate

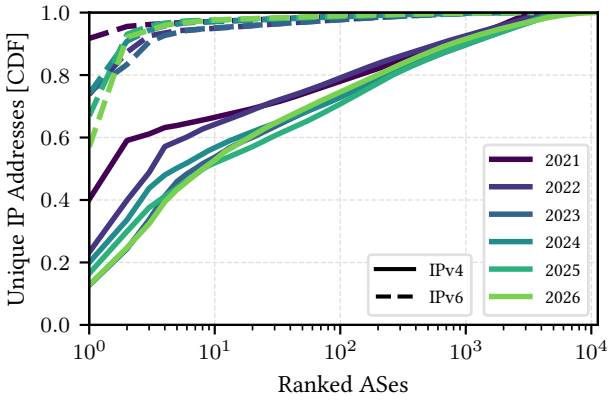
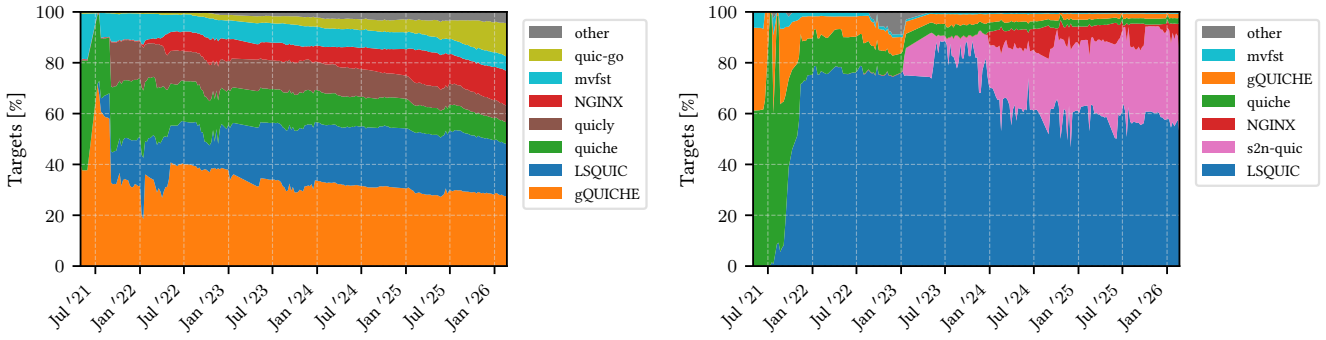


Figure 3: Cumulative distribution of unique IP addresses per AS per year. ASes on the x-axis are sorted by the number of visible QUIC deployments in 2026. For each year, the first scan is shown.

the probed addresses and not only the responsive ones. In the latest scan with SNI values, they hold 83 % and 14 % of the probed addresses and 62 % and 32 % of the responsive ones. Hostinger thus loses weight, as only 26 % of its addresses answer against 35 % on average, while Amazon gains weight with 78 %. Together with the third largest AS, they cover 98 % of all probed IPv6 addresses. Their overall share of targets across all scans is even above 99 %, as more than 50 % of the targets in each AS are only observed in one single scan and thus are extremely short-lived. In comparison, the median lifetime of targets outside these two ASes in the IPv6 dataset is 70 days. DNS resolutions for domains hosted by Amazon and Hostinger frequently return different, random-looking IPv6 addresses. We speculate that Amazon and Hostinger are load balancing incoming traffic into subnets with many responsive addresses, encoding short-lived information into the target IP address.

The targets per scan do not yet show how stable the target set is. We therefore measure how long each target stays in the dataset and find that both datasets peak at the two extremes, targets seen in a single scan and targets seen throughout (see Appendix Figures 20



(a) IPv4 **(b) IPv6**

Figure 4: Share of libraries observed over time for IPv4 and IPv6. For better visibility, libraries with small shares are grouped into "other".

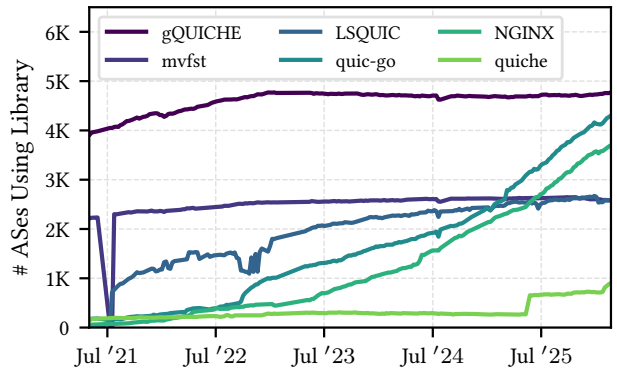


Figure 5: The expansion of selected libraries in ASes over time. Only data from the IPv4 dataset is shown.

and 21). The address families differ strongly, as 13.2 % of the IPv4 targets but 56.3 % of the IPv6 targets appear in a single scan only. These numbers describe addresses, not hosts. The DNS behavior of Amazon and Hostinger described above suggests that most short-lived IPv6 targets are stable services behind rotating addresses rather than deployments that vanish.

5.3 Used Libraries

While the market shares of QUIC libraries differ for the IPv4 and IPv6 datasets, we observe a set of just four libraries (LSQUIC, quiche, gQUICHE, and s2n-quic) that make up more than 95 % of the successful handshakes in both datasets. Figure 4 shows the share of visible libraries over time for IPv4 and IPv6. The earliest scans are dominated by gQUICHE, quiche, and mvfst for both datasets, indicating the early adoption by Google, Cloudflare, and Meta. Note that the distribution slightly differs from the results by Zirngibl et al. [39], as we focus on successful handshakes and cannot use the identification based on error messages. However, the share of these three libraries in the IPv6 dataset decreases heavily in the first year and steadily continues to decrease below 5 % each until the end of the dataset, while they remain significant in the IPv4 dataset.

For IPv4, Figure 4a illustrates how libraries such as quic-go and NGINX gain more market share over time. Figure 5 shows the number of ASes with at least one deployment for the respective library. While other libraries just gain more targets in ASes they are already present in, quic-go and NGINX expand heavily into new ASes over time. gQUICHE was constantly present in around 4700 ASes and mvfst in around 2600 ASes over the last four years. In contrast, quic-go and NGINX were both present in less than 500 ASes four years ago, but have expanded to 4300 and 3700 in the latest scans. Since the beginning of 2026, more than 50 % of the newly added targets per scan in the IPv4 dataset are identified as quic-go or NGINX. In Section 6.1, we show that also the number of unique transport parameter configurations is increasing similarly for those two libraries. We assume that one reason for these observations is that NGINX is currently the most popular web server, being used by more than 32 % of the web servers according to W3techs [37] as of April 2026. In addition, quic-go is integrated into numerous web server applications and reverse proxies [33].

To further understand where this growth is happening, we join the latest IPv4 scan with the network type in PeeringDB, which covers 57 % of the ASes we observe. As expected, most individual targets (82 %) are located in networks labeled as content networks. However, if results are evaluated on the granularity of ASes, only 21 % of the ASes containing QUIC targets are content networks and access networks are the largest class with 42 % (see Appendix Figure 17). quic-go and NGINX are the two libraries that follow this AS population most closely, as no single AS holds more than 15 % of their targets, and the median AS runs two of them. The libraries of hypergiants sit at the other extreme, with 93 % of the quiche targets in AS13335 (Cloudflare) and almost all s2n-quic targets in AS16509 (Amazon). The growth of NGINX and quic-go targets concentrates in hosting and virtual server providers, which confirms that the adoption of QUIC spreads through self-managed deployments outside the large CDNs. However, the overall footprint of these libraries remains biased towards early adopters. While the number of ASes is increasing, ASes that started using these libraries within the last year account for less than 20 % each of the newly added targets.

For IPv6, LSQUIC has the largest market share, mostly due to AS47583 (Hostinger) accounting for more than 2.4 million unique addresses in the latest scans, which is more than 50 % of the targets.

More than 97 % of the targets appearing at least twice in the IPv4 dataset are identified as using the same library in all scans. However, we also observe targets that seem to change their library at some point in time. Assuming that a transition from a uniquely identified library to another library with a fingerprint colliding with the previous one is very unlikely, we can even assume that more than 99.97 % of the targets that appear at least twice do not change their library during the scanning period. Filtering for short time intervals where more than 10 000 targets change their library, we find a significant shift in June 2025. We detect 25 166 targets changing from NGINX to gQUICHE in less than a month. All targets are located in 49 different ASes belonging to Chinese entities including China Mobile, China Unicom, and ChinaNet. A deeper dive into the data reveals that these targets are related to the video sharing platform Bilibili and their implementations based on NGINX [4] and gQUICHE [5].

The share of a library matters because libraries differ in more than their fingerprint. Related work shows differences in performance [17], in defaults and design decisions [23], and in the congestion control they ship [24]. A property of a widely deployed library thus becomes a property of a large part of the Internet: the four libraries behind more than 95 % of the successful handshakes decide how most QUIC connections react to loss, how they share a bottleneck, and which extensions are available to clients. While we cannot say which library serves an application best, the stable spread across many libraries keeps the ecosystem from ossifying around the behavior of a single implementation.

5.4 Key Takeaways

QUIC keeps growing in addresses and ASes, but the growth no longer comes from the same place. The libraries of hypergiants gain addresses inside the ASes where they already ran in 2021, while quic-go and NGINX reach 25 to 60 times more ASes than five years ago. The QUIC that grows today is thus self-managed: a web server or reverse proxy that somebody installs and updates, not a fleet that one operator rolls out centrally. This decides how fast the ecosystem can change. Cloudflare moved 50 000 targets to a new flow control value almost at once (see Section 6), and Bilibili swapped the library of 25 166 targets in less than a month. A new default in NGINX or quic-go, in contrast, spreads only as fast as operators upgrade, which Section 7 shows can take years. Finally, no library holds a majority, and more than 99.97 % of the targets keep their library once deployed. Client implementers therefore face a wide and stable set of peers: testing against the largest two or three servers covers only part of what they meet on the Internet.

6 Increasing Diversity through Configurations

To answer RQ2 and shed light on the diversity and stability of QUIC deployments, this section analyzes the evolution of transport parameters over time. We first examine the overall evolution of transport parameter configurations, followed by an analysis of their stability and the diversity of their actual values. We also explore the use of GREASE for transport parameters.

Table 1: Transport parameters carrying integer values.

ID	Name	Unit	Default	Range
0x01	max_idle_timeout	ms	0	
0x03	max_udp_payload_size	B	65 527	≥ 1200
0x04	initial_max_data	B	0	$< 2^{62\dagger}$
0x05	initial_max_stream_data_bidi_local	B	0	$< 2^{62\dagger}$
0x06	initial_max_stream_data_bidi_remote	B	0	$< 2^{62\dagger}$
0x07	initial_max_stream_data_uni	B	0	$< 2^{62\dagger}$
0x08	initial_max_streams_uni		0	$\leq 2^{60}$
0x09	initial_max_streams_bidi		0	$\leq 2^{60}$
0x0a	ack_delay_exponent		3	≤ 20
0x0b	max_ack_delay	ms	25	$\leq 2^{14}$
0x0e	active_connection_id_limit		2	≥ 2
0x20	max_datagram_frame_size	B	0‡	

† Max Var-Int value; ‡ Disables datagrams

Before the analysis, we summarize what the transport parameters in Table 1 control. All of them are sent by a server to announce values to a client. Four set flow control limits: `initial_max_data` caps the data a client may send during a connection, and the three `initial_max_stream_data_*` parameters cap it per stream, depending on which side opens the stream and whether it is bidirectional. A server picking a low limit throttles a fast client until it raises the limit during the connection, while a large value gives up protection against a client that sends more than the server can handle. A useful limit follows the bandwidth-delay product of the expected clients [16]. The number of streams a client may open is limited by `initial_max_streams_bidi` and `initial_max_streams_uni`. `max_idle_timeout` decides how long an endpoint keeps an idle connection and trades server memory against a new handshake for the client. `max_udp_payload_size` announces the largest packet a server accepts, and `max_datagram_frame_size` does the same for unreliable datagrams, where zero disables the extension. `ack_delay_exponent` sets how an endpoint encodes the delay it reports in its acknowledgments, while `max_ack_delay` bounds how long it may wait before sending one. Both influence the round-trip time (RTT) estimate and the loss detection timers of the peer. `active_connection_id_limit` announces how many connection IDs an endpoint stores from its peer, which matters for connection migration.

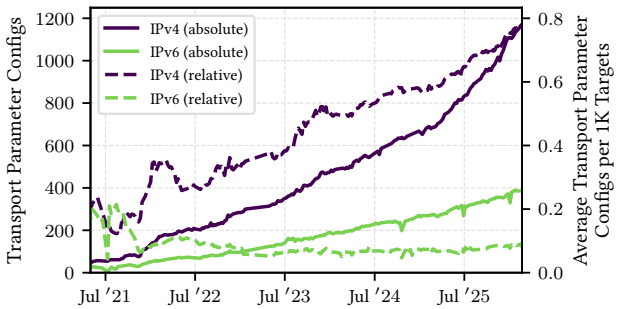


Figure 6: Number of transport parameter configurations. The solid lines are absolute values (left y-axis) and the dashed lines are relative values (per 1K targets, right y-axis).

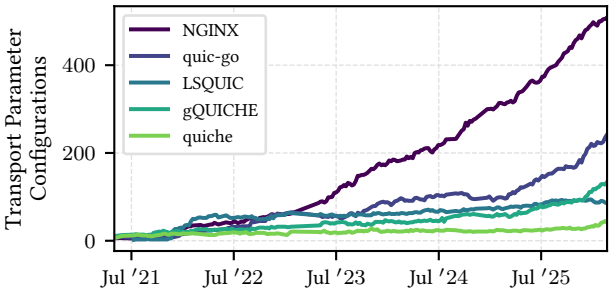


Figure 7: Number of transport parameter configurations per library over time. Only the five libraries with the most distinct configurations are shown.

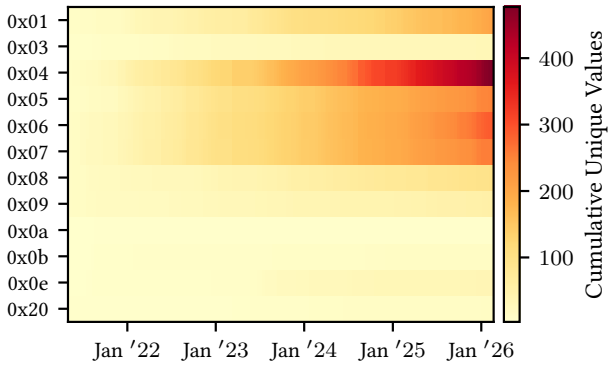


Figure 8: Cumulative discovery of unique transport parameter integer values over time.

6.1 Evolution of Transport Parameters

In the IPv4 dataset, we observe a total of 56 unique transport parameter identifiers. While the first scans contain only around 20 unique identifiers, the number increased almost linearly with two small steps. Both steps are caused by XQUIC deployments introducing either multiple transport parameters for Forward Error Correction (FEC) or other experimental transport parameters not related to a public draft (see Section 7 for an evaluation of QUIC extensions). To understand the evolution of transport parameters, we first exclude certain parameters and their values to ensure a meaningful analysis. This includes all session-specific and regularly changing parameters, such as all parameters containing connection IDs, the `stateless_reset_token`, and the `preferred_address`. In addition, identifiers reserved for use with GREASE and parameters containing randomized values, such as `version_information`, are also excluded, as they intentionally contain randomized identifiers or values. As some libraries randomize the order of transport parameters, as shown by Zirngibl et al. [39], we sort the resulting set to mitigate this effect.

Figure 6 shows the development of the number of transport parameter configurations relative to the number of unique IP addresses over time. While the targets in the IPv6 dataset show an almost linear increase in configurations, we observe more of an exponential increase for IPv4, especially in the last two years. The first scans in April 2021 contain around 25 configurations in the IPv6 dataset and around 50 configurations in the IPv4 dataset. In the most recent scans in February 2026, we observe around 380 configurations in the IPv6 dataset and around 1160 configurations in the IPv4 dataset. Both address families together hold 1162 distinct configurations, as nearly every IPv6 configuration also occurs in IPv4. With the target distribution among ASes from Figure 3 in mind, this difference can be explained by the fact that larger ASes operated by CDNs likely introduce lots of targets using the same configuration (especially for IPv6).

Looking into the five libraries with the most distinct transport parameter configurations in Figure 7, we notice that NGINX and quic-go are primarily responsible for the increase in transport parameter configurations. This observation matches the findings from the previous section, where we observed that these two libraries are increasingly deployed in various ASes. We can also assume here that the increase in transport parameter configurations is primarily caused by changes in the values of existing parameters.

6.2 Configuration Stability

To measure the stability of transport parameter configurations, we analyze how many targets change their configuration over time and how significant these changes are. In the IPv4 dataset, we find that 56.3 % of all targets have the same transport parameter configuration in all scans they are observed in. If we also exclude the `disable_active_migration` transport parameter, which is partially inconsistently set as mentioned in Section 5.1, this share increases to 58 %. Breaking down visible changes in configurations for the remaining targets shows that 7.9 % of all targets only change the values of the used parameters, while the remaining 35.9 % also change the used parameters. Custom transport parameters, *e.g.*, used by XQUIC and mvfst, as well as `google_version` turn out to be the most volatile ones. They are changed at least once for 18 % to 79 % of all targets using them, while the other parameters introduced in RFC 9000 [16] are changed by 0 % to 14 % of the targets. The most volatile of those parameters are `0x06` to `0x08`. Therefore, changes in visible configuration are not only due to manual changes in values by operators but also updates by libraries, functionality, or defaults.

6.3 Diversity of Parameter Values

While transport parameters can carry values of arbitrary types, most of them carry a variable length integer as of 2026. In contrast to boolean or string values, integers are suitable for an analysis of the diversity. We pick all 12 transport parameters carrying integers that are specified in an RFC and analyze their values over time. These transport parameters are listed in Table 1.

Figure 8 shows the cumulative number of observed unique values for the 12 transport parameters over time. It can be observed immediately that the diversity of initial flow control windows, *i.e.*, the values of the `initial_max_data` and `initial_max_stream_data_*` parameters, has increased steadily and significantly over time. The values of all four transport parameters are in units of bytes and range from a few kilobytes to hundreds of terabytes. Besides that,

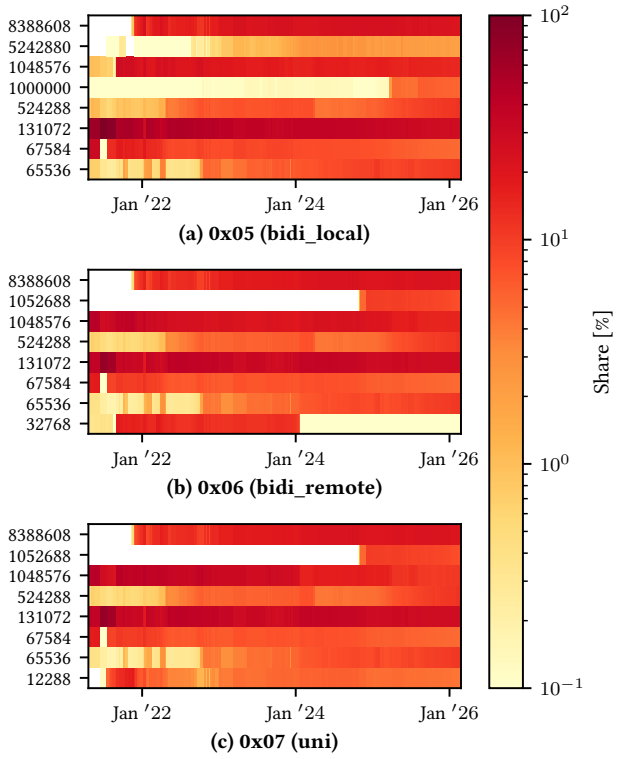


Figure 9: Share of the eight most popular values for the three `initial_max_stream_data_*` transport parameters. Each plot shows the share of each value among all seen values for the transport parameter.

the `max_idle_timeout` parameter, which is in units of milliseconds, also shows a significant increase in diversity, with values ranging from a few milliseconds to several years. A substantial portion of the observed values are obviously outliers, suggesting misconfigurations or intentionally exaggerated values.

While those parameters exhibit a wide range of values, the most frequently observed ones are clustered. Figure 9 shows the share of the eight most popular values for the three `initial_max_stream_data_*` parameters over time. These values are used by more than 90 % of all targets. The range decreased to only two orders of magnitude with the values being mostly multiples of 1000 and 1024. Taking a closer look, it can be observed that the value of 1 MB got very popular for `0x05` almost instantly in March 2025. By looking at the involved ASes and libraries, we can attribute this increase to roughly 50 000 targets running quiche in AS13335 (Cloudflare). The targets were not newly added but did not advertise `0x05` before and set `0x06` and `0x07` to 1 048 576 before. Further details on the distribution of values for the three parameters can be found in Appendix Figure 18.

To understand the dynamics of the observed values, we analyze the monthly changes of values using the example of `max_idle_timeout`, visualized in Figure 10. The plot differentiates between newly introduced values (green), the return of known values (orange), and the disappearance of values (red). The steady change of observed values indicates a dynamic ecosystem where operators

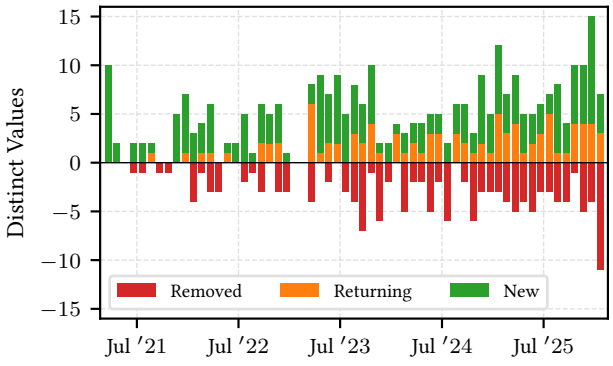


Figure 10: Monthly changes of transport parameter values for `max_idle_timeout`. The plot decomposes changes into the introduction of new values (green), the return of known values (orange), and the disappearance of values (red).

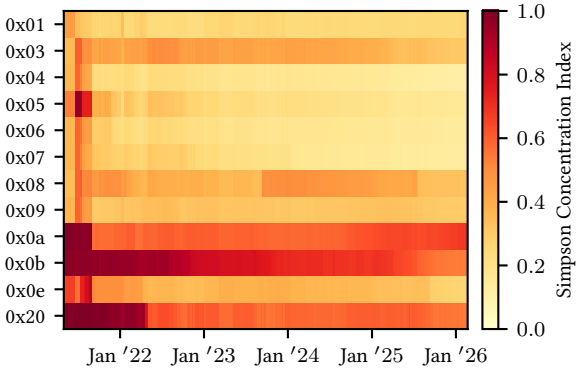


Figure 11: Concentration of transport parameter values measured by the Simpson Index. High values (darker) indicate a parameter dominated by a handful of implementation defaults, while low values indicate a more balanced/diverse configuration space.

adjust their configurations. The increase of configurations is a continuous process. Some configurations seem to disappear and are not re-added to the set of visible configurations.

Finally, Figure 11 shows how concentrated the values of the 12 transport parameters are over time, with one row per identifier of Table 1. We use the Simpson Index, which measures the probability that two random targets carry the same value. A dark row therefore marks a parameter that a few defaults dominate, a light row one with diverse values. As the multiplicity of values in Figure 8 already suggests, the parameters with more unique values are mostly also more balanced. For the transport parameters related to acknowledgment delay, the values are less balanced. We assume that libraries tend not to expose settings for these parameters to operators. Overall, the parameters become more balanced over time, likely due to the growing number of observed values.

The `preferred_address` transport parameter is a mechanism for servers to steer clients to a different IP address or port after the

handshake completes, enabling topology-aware load balancing and failover without establishing a new connection. The specification motivates this with clients that arrive on an address shared by many servers but are better served by a unicast address, and it allows a server to announce an address for either address family so that the client can pick. In our whole dataset, 3530 endpoints advertise a `preferred_address`, representing only 0.23 % of all targets. All those endpoints belong to Amazon, with 3174 hosts in AS16509 (AMAZON-02) and 356 hosts in AS14618 (AMAZON-AES). Fingerprinting matches none of the known libraries, but the randomized ordering and the parameter set match gQUICHE, with `preferred_address` (`0x0d`) as the only structural difference. This suggests a modified or internal build of gQUICHE. In every case, the preferred address is an IPv4 address on port 443.

The distribution of SNIs uncovers an Apple CDN context: `api.smoot.apple.cn` occurs 109 times and `api.smoot.apple.com` 80 times, with related Apple API endpoints dominating. These hostnames correspond to Apple's search and suggestion services, including Siri, Spotlight, Lookup, and Safari, as documented in Apple's support pages [1]. Notably, as the SNI distribution confirms, the served content is Apple's, meaning the entire deployment with `preferred_address` is not operated on Apple's own ASes but on third-party Amazon infrastructure. The underlying infrastructure presumably is Amazon's QUIC-enabled Network Load Balancer, announced in November 2025, which uses `preferred_address` to steer clients to a suitable load balancer after the initial handshake completes [11]. The AWS announcement cites an early adopter running a latency-sensitive search service at 145 TB/d and 200 000 peak connections per second, which matches the scale of the Apple services identified by the hostnames in our dataset. No target outside these ASes announces a preferred address, so production use of this standardized mechanism stays concentrated in one place. The 59 distinct preferred IP addresses extracted from these 3530 endpoints suggest that a relatively small pool of redirect targets underlies the entire deployment. The observed use matches the motivation of the specification closely, as clients reach a shared entry point and continue with one of these 59 specific addresses. The choice between address families stays unused, as every announcement carries only an IPv4 address.

6.4 Transport Parameter Greasing

While RFC 9000 [16] does not explicitly require endpoints to do so, some QUIC implementations employ GREASE for transport parameter identifiers, as described in Section 2. In recent scans, roughly 60 % of all targets are affected, with the share varying significantly across implementations. For all libraries, we observe one of the three following behaviors: *(i)* no use of reserved identifiers, *(ii)* almost all targets use them, or *(iii)* almost no target uses them. Specifically, some popular libraries including quiche, mvfst, XQUIC, and s2n-quic do not employ GREASE for transport parameters, while other popular implementations like LSQUIC, gQUICHE, quic-go, and Akamai QUIC likely enable this feature by default. Others, including quicly, MsQuic, and NGINX, appear to keep it disabled by default, as we only observe this behavior in a small subset of targets. We infer these defaults from the share of targets per library, not from source code, as a handshake never reveals which version of a library a deployment runs. Looking not only at the recent scans, we notice that before 2022, almost all targets using MsQuic and quicly were using GREASE, while this behavior has almost completely disappeared by mid-2022. Furthermore, the selection of identifiers varies significantly across implementations: while quic-go limits its selection to 256 different identifiers by using an 8-bit random input, gQUICHE utilizes all possible values in the 62-bit space available with variable-length integer encoding.

6.5 Key Takeaways

The visible configurations of both address families combined grow from around 50 in 2021 to 1162 in 2026, and the integer parameters gain values at the same pace. Besides changes in supported parameters or features, different integer-based parameters see an exponential increase in value heterogeneity. The ecosystem concentrates in code and spreads in configuration (see Section 5.3). A client meets one of four implementations in more than 95 % of its handshakes, but more than a thousand different configurations. Loss recovery and congestion control therefore change only when a maintainer acts, while the announced limits might change with every deployment a client connects to. Operators who tune their deployments mostly pick round values close to a default, which is why values are mostly clustered around a small set. Flow control limits of several terabytes and idle timeouts of several years also exist and leave a client with no useful limit at all, so a client should bound selected transport parameter values itself instead of trusting the server. Besides the increase of useful values, the initially visible diversity is increased through an extensive use of GREASE by some libraries. Finally, we uncover a potential Apple deployment relying on Amazon and the preferred address feature for a uniquely visible load balancing configuration.

7 Adoption of New Transport Parameters

Besides the increased diversity of initially specified QUIC parameters, new extensions to the protocol are specified and the user-space nature of QUIC potentially allows for fast adoption and deployment. In this section, we examine the adoption of new transport parameters introduced in recent drafts, such as those related to acknowledgment frequency and datagram support. The extensions we observe share a purpose. Nearly all of them address performance or measurement, *e.g.*, acknowledgment frequency, timestamps, loss bits, FEC, path MTU discovery, and multipath, while unreliable datagrams add a capability that streams cannot express.

7.1 Unreliable Datagrams

In March 2022, the `max_datagram_frame_size` transport parameter was specified in RFC 9221 [28]. It allows endpoints to indicate that they support unreliable datagrams besides reliable streams. They are used, *e.g.*, by MASQUE proxies via connect-udp [31]. A value of zero indicates that datagrams are not supported for this connection while larger values inform about support and the maximum datagram size an endpoint is willing to accept (including frame type, length, and payload).

Already in the first scan in the IPv4 dataset, we observe that more than 36 % of the targets support this extension by sending a non-zero value for this transport parameter. While around 45 000 targets

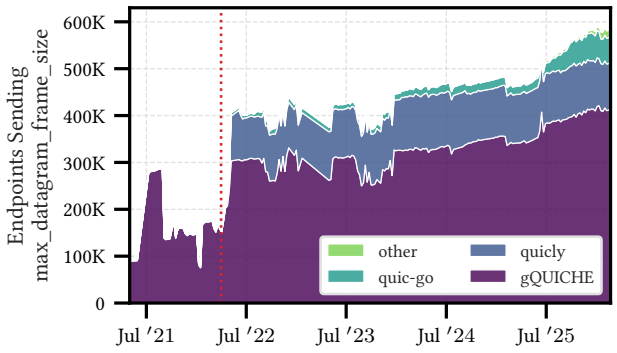


Figure 12: Targets with datagram support per library. The red line indicates the release of RFC 9221 [28].

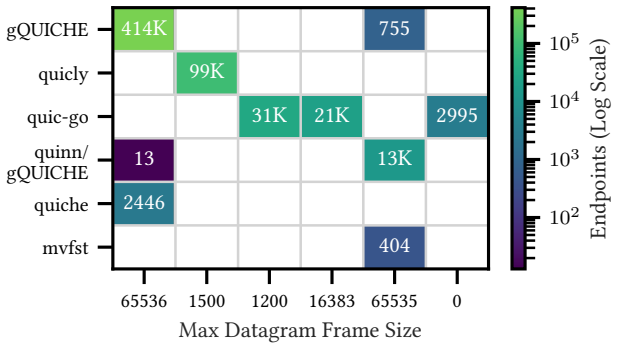


Figure 13: Values of the `max_datagram_frame_size` transport parameter observed per library. Only data from the latest scan and only the six most popular libraries and values are included.

are located in AS15169 (Google), we also see around 45 000 targets distributed across almost 4000 ASes running gQUICHE (most likely off-net deployments). Besides gQUICHE, other libraries support datagrams already in the first scan. These include quiche, Akamai QUIC, and quicly with less than five targets each, and 175 targets using quic-go. This can also be seen in Figure 12, showing the share of libraries among targets sending the `max_datagram_frame_size` transport parameter. Shortly after the release of RFC 9221 [28] (indicated by the red line), targets using quicly and quic-go started to support this extension as well, leading to a significant increase in the overall adoption of datagram support. Even though the adoption at that point in time was above 50 % and the number of targets supporting datagrams is still rising, the adoption has slowly decreased since then, reaching around 40 % in the most recent scans, as the total number of targets is increasing faster.

Looking at the values in Figure 13, we see that the majority of targets use the default value of the libraries, which is either close to a typical network MTU or significantly higher. In the latter case, the actual size is limited by the maximum of `max_udp_payload_size` or the path MTU. Therefore, setting this parameter to an unrealistically high value does not lead to issues in practice. We also observe in the latest scan that more than 94 % of the targets with datagrams

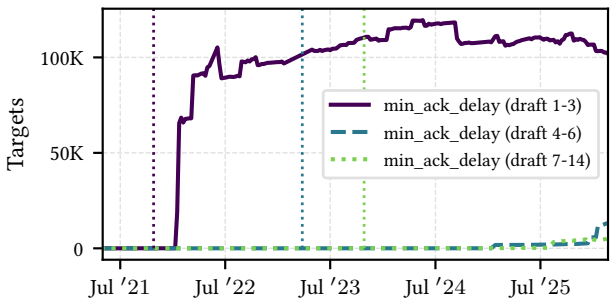


Figure 14: Number of IPv4 targets using the respective identifier of the `min_ack_delay` transport parameter. Vertical lines indicate the introduction of identifiers.

enabled use a smaller value for the `max_datagram_frame_size` than for the `max_udp_payload_size`. For quic-go, the captured values are most diverse, being either 0, 1200, or 16 383. A quick look at the source code reveals that quic-go changed the default from 1200 to 16 383 in December 2023.[1] We assume that many targets returning 1200 are outdated versions of quic-go using the old default instead of manual re-configurations. The value of zero, indicating that datagrams are not supported, is only observed for targets using quic-go and only by a share of around 1 % of all targets.

7.2 Acknowledgment Frequency

The "QUIC Acknowledgment Frequency" draft [15] is a current effort of the IETF to specify a QUIC extension that enables endpoints to signal their desired acknowledgment frequency. A new transport parameter, `min_ack_delay`, is used to indicate support for the extension to the other endpoint. The identifier of this transport parameter has changed three times since the first version of the draft was published in July 2021, with the current identifier being `0xff04de1b`. Figure 14 shows the visibility of the new parameter over time in the IPv4 dataset. We observe the first use of this transport parameter in December 2021. The usage increased to roughly 90 000 unique IPv4 addresses in only three months. This increase is driven by deployments of the quicly library within AS54113, belonging to Fastly. One of the main draft authors worked for Fastly at that time. Until the most recent scan, these deployments make up more than 80 % of the unique IPv4 addresses using this transport parameter. Besides quicly, we also observe early MsQuic deployments, both using the identifier for the `min_ack_delay` transport parameter introduced in the second version of the draft. Even though this identifier has changed again in March 2023, both libraries still mainly use the old identifier in all scans until the most recent one. A look at the repositories of both libraries reveals that the implementation of the "QUIC Acknowledgment Frequency" draft has been updated in the default branches for both libraries to a draft version using the latest identifier (quicly in September 2025[2] and MsQuic in January 2025[3]). Even in the latest scans, only 6.2 % of the MsQuic targets and only 0.7 % of the quicly targets use the new identifier,

[1] https://github.com/quic-go/quic-go/commit/7b9d21fb
[2] https://github.com/h2o/quicly/commit/7d85b207
[3] https://github.com/microsoft/msquic/commit/5eb2b6a6

revealing a remarkably large number of targets using an old version of the libraries. The second ID, introduced in draft version 4, is supported by Quinn, while the newest ID is mostly supported by gQUICHE and Akamai QUIC. Interestingly, gQUICHE deployments sending this parameter seem to be unrelated to Google itself, but independent deployments using the library (based on domains and seen HTTP server headers). While gQUICHE supports the draft, it is deactivated by default as of April 2026.[4]

The split between the identifiers costs a feature, not a connection. QUIC endpoints must ignore transport parameters they do not know, so a client that implements only the current identifier reads the old one as unknown and keeps the acknowledgment rules of RFC 9000 [16]. It thereby gives up savings in acknowledgment traffic and client processing, which matter most on asymmetric or high-rate links [15]. As both endpoints have to announce the parameter, and gQUICHE, the library behind Google Chrome, keeps the extension disabled by default, many connections to the deployments in Figure 14 never use it regardless of the identifier.

In our data, the values of the `min_ack_delay` transport parameter are not diverse: we only see four distinct values (1K, 5K, 16K, and 25K) across all scans.

7.3 Other Parameters

In this section, we evaluate remaining parameters, *e.g.*, without configurable values or from personal IETF drafts. The parameters and identifiers are listed in Appendix Table 3.

`version_information`, defined for QUIC version negotiation and standardized in May 2023 in RFC 9368 [32], is present in around 25 % of the targets in February 2026. Deployment is highly recent: prior to early 2025, the corresponding transport parameter (`0x11`) was observed in fewer than 1 % of deployments, indicating a substantial lag between standardization and operational adoption. Figure 19 illustrates this longitudinal evolution. `google_version` was observed in over 30 % of targets until early 2026, after which its prevalence dropped to 3.9 %. This pattern indicates that Google's earlier proprietary version mechanism likely informed the design of RFC 9368 [32], after which it was replaced by the standardized parameter in gQUICHE in December 2024.[5]

Since mid-2025, the use of `google_connection_options` has been steadily increasing. It now appears in 2.8 % of the targets. We also observed an unknown transport parameter with identifier `0xe101` appearing in 20 gQUICHE deployments starting in January 2026, all associated with `dbankcloud.cn` (Huawei).

Another prevalent pair of extensions is `loss_bits` from "The QUIC Loss Bits" draft [8] and `enable_timestamp` from the "Quic Timestamps For Measuring One-Way Delays" draft [13], each appearing in approximately 20 % of all targets in early 2026. These two parameters are always co-present, identifying the LSQUIC implementation: LSQUIC enables both draft extensions by default.[6] The `enable_timestamp` transport parameter is also observed only in LSQUIC deployments in our dataset.

Meta's mvfst advertises several proprietary transport parameters. `ack_receive_timestamps_enabled` appears in all mvfst deployments. `receive_timestamps_exponent` and `max_receive_timestamps_per_ack`, both defined in the first version of the "QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps" draft [35], are observed in approximately 93 % of mvfst deployments. An additional unknown transport parameter with identifier `0x5178` appears in approximately 91 % of mvfst deployments as of February 2026. The `facebook_partial_reliability` parameter was already observed in the earliest measurements in April 2021 but disappeared entirely after October 2022, with a maximum deployment of 66 targets. Together, these parameters indicate that mvfst consistently advertises a set of proprietary extensions across all observed deployments. A few transport parameters, *e.g.*, `facebook_partial_reliability` are only found in a limited subset of targets, suggesting they are used selectively for testing or experimental purposes.

quic-go exhibits a small set of implementation-specific transport parameters. `reset_stream_at`, defined in the "Reliable QUIC Stream Resets" draft [34], is only observed in quic-go deployments.

XQUIC also exposes three proprietary transport parameters related to FEC functionality. These have been observed since September 2024, with a steady increase to approximately 150 targets by February 2026. `pmtud_options` has been observed since early 2025 and is currently present in approximately 150 XQUIC targets as of February 2026. We further see four unknown transport parameter identifiers (`0xbaba`, `0xbabc`, `0xbac0`, `0xbac1`) in fewer than 20 XQUIC deployments. These first appeared in March 2022 and disappeared by the end of the same year.

Regarding the "Multipath Extension for QUIC" draft [21], we identify deployments that implement different draft revisions, with `enable_multipath` and `initial_max_path_id` appearing in multiple versions. Early-stage multipath QUIC deployments are already visible: the `enable_multipath` parameter from draft-00 appears in 622 targets in February 2026, while `initial_max_path_id` from draft-09 is observed in 106 targets. While the counts remain small as of early 2026, their presence shows that multipath QUIC is being experimentally deployed outside of controlled environments.

7.4 Key Takeaways

QUIC libraries adopt new drafts quickly, and some extensions reach the Internet long before they are standardized: multipath is visible in 622 targets in February 2026 while the draft is still open. The deployments, however, do not follow the libraries. quicly and MsQuic both moved to the current `min_ack_delay` identifier in their default branch, but only 0.7 % and 6.2 % of their targets use it (see Section 7.2). The user-space promise of QUIC thus holds for the library and breaks at the operator: the code exists, the running deployment does not use it. Every change of a code point can split the ecosystem for years, so a working group should expect a long overlap, and a client that wants an extension to work with most peers has to accept the old identifiers as well. Some libraries also ship parameters without a public draft, *e.g.*, mvfst, XQUIC, and gQUICHE, so outside these companies it is hard to tell what the parameters do.

[4] https://github.com/google/quiche/blob/1ff5238/quiche/common/quiche_feature_flags_list.h#L62
[5] https://github.com/google/quiche/commit/f4690db
[6] https://github.com/litespeedtech/lsquic/blob/c1ca798/include/lsquic.h

8 Discussion

QUIC Heterogeneity: Our work shows a large and increasing heterogeneity in the QUIC ecosystem in terms of deployments, libraries, but also configurations. This development can be double-edged. On one side, it shows that QUIC is easily usable, extensible, and flexible. Thus, it is deployed and will serve as a good foundation for future developments. On the other side, it increases the complexity of the ecosystem and the network. While libraries follow the same standards, differences exist and can have major impact, *e.g.*, related work has shown that major performance differences exist and congestion control algorithms are diverging [17, 24]. Furthermore, available configurations can have immediate impact on endpoints (*e.g.*, massive flow control values or idle timeouts). We already communicated initial findings of this work with library implementers and plan to intensify this exchange in the near future.

Fast Adoption but Long Tail: Our findings show that QUIC deployments can quickly adapt to new features due to the user-space nature. Early draft versions and new extensions are adopted early by libraries and deployed widely. This is not only driven by hypergiants but also libraries used in common proxies or web servers, *e.g.*, quic-go and LSQUIC (see Section 7). However, deployments are not always updated and the early adoption of features can lead to widespread but potentially outdated features or configurations. The `max_datagram_frame_size` value of 1200 seen with many quic-go deployments is one example. 1200 was the default value of the parameter until December 2023, but was changed to 16 383 afterwards. While 21 235 deployments with the new value are visible in the most recent scan, the old value still dominates with 30 850 deployments. This parameter has only performance implications but it shows how old values can persist and future developments need to be treated carefully. A change motivated by security or privacy rather than performance would face the same lag, so a library that changes a default has to stay interoperable with the old value.

Ecosystem Considerations: Three parts of the ecosystem are exposed to fragmentation, and each one suggests a different action. The first is the identifier of a transport parameter that is still part of a draft. `min_ack_delay` carried four identifiers in five years and the deployments did not follow during our scanning period (see Section 7.2), so every renumbering leaves a split ecosystem behind for years. A working group should expect this overlap, and an implementation should keep parsing the identifiers it once shipped. The second is the default value of a library. A default reaches thousands of ASes and stays visible long after the library has changed it, so a maintainer who picks a default configures a large part of the Internet, and changing one deserves the care of a protocol change. The third is the uneven use of GREASE. Roughly 60 % of the targets exercise the ability of their peers to ignore unknown parameters, while quiche, mvfst, XQUIC, and s2n-quic never do. The protection against ossification thus rests on one part of the ecosystem, and a library that does not GREASE lets peers ossify.

Lessons Beyond QUIC: The five years also show where the agility of a user-space transport protocol ends. Moving a transport protocol out of the kernel removes the kernel from the update path, but not the operator, and the slowest updater sets the pace of the ecosystem. The update path can even grow longer than the kernel's, as a library bundled into a proxy or a container image reaches an operator only after every vendor in between updates. A protocol that expects this should keep old code points readable and let outdated deployments degrade gracefully, as QUIC does by requiring endpoints to ignore unknown transport parameters.

Limitations: For scalability reasons and due to ethical considerations, the available dataset has some limitations. First, all scans are searching for QUIC deployments on the standard HTTP/3 port 443. There are no scans on alternative ports or ports used by other protocols on top of QUIC, *e.g.*, DNS over QUIC (DoQ). However, DoQ deployments are rare as of 2026 [3]. Furthermore, the scans limit the number of tested SNI values per IP address to reduce the overall impact on single targets per week. Section 4.4 shows that the server transport parameter configuration is independent of the client parameters at least from a single vantage point. Furthermore, as shown in Section 5.1, more than 99 % of targets show the same behavior for all domains. Thus, the impact is minimal, but some outliers might be missed. Furthermore, all scans are conducted from a single vantage point. Therefore, we might miss regional differences. However, our evaluation provides a lower bound of the diversity of the QUIC ecosystem.

Besides the scan, the mapping to libraries can lead to false labels. The introduced methodology by Zirngibl et al. [39] relies on a fixed set and order of transport parameters. They extracted these fingerprints from handshakes with QUIC example servers provided by the library implementers. We have verified that current versions of the libraries have the same fingerprint or updated the fingerprint if necessary, *e.g.*, for XQUIC, and extended the list with additional libraries, *e.g.*, OpenSSL (see Section 4). However, most libraries are open source and could be changed by operators for their use case.

Future Work: Future work could evaluate the impact of this diversity on the QUIC ecosystem, *e.g.*, how different parameters affect clients and their performance, or the fairness between flows. It could also tell library defaults and operator choices apart in general, which our view manages only where a value dominates the targets of a library and its source is public.

9 Conclusion

Specified in 2021, QUIC offers interesting features and a high extensibility due to its design and user-space approach. In this paper, we use a five-year dataset of weekly QUIC scans to provide a unique, longitudinal view on the development of the QUIC ecosystem. We show an increasing heterogeneity of the ecosystem in three dimensions: an increasing, widespread deployment of QUIC and libraries (RQ1), an increasing number of transport parameter configurations (RQ2), and diverse adoption to new features (RQ3). We show that the number of deployments is increasing and especially quic-go and NGINX are used in at least 25 times more ASes compared to 2021. Furthermore, the visible QUIC configurations based on transport parameters are increasing by orders of magnitude from only 50 to 1162. However, not only do configurations based on initial parameters diversify; different libraries also adopt various new extensions and drafts throughout our observation period. The current QUIC ecosystem is a fast-lived, diverse landscape that provides the foundation for future protocols and applications, but needs to be carefully monitored.

Acknowledgment

We thank the anonymous reviewers and our shepherd for their valuable feedback. This work was funded by the Horizon Europe programme, projects GreenDIGIT (101131207), SLICES-IP (101287692), and FLECON-6G (101192462), by the German Federal Ministry of Research, Technology and Space (BMFTR), projects ALL-HANDS (02K25A026) and 6G-life (16KIS2414), and by the German Research Foundation (DFG), project SLICES-SUSTAINABILITY (566292327).

References

[1] Apple Inc. 2026. *Use Apple products on enterprise networks.* https://support.apple.com/en-us/101555#aisirisearch Accessed: 2026-04-29.

[2] D. Benjamin. 2020. *Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility.* RFC 8701. IETF. https://www.rfc-editor.org/rfc/rfc8701.txt

[3] Philipp Bielefeld, Felix Hoffmann, Steffen Sassalla, Vasilis Ververis, and Vaibhav Bajpai. 2026. The Future of DNS Privacy: A Comparison of DNS over QUIC andDNS over HTTP/3. In *Proc. Passive and Active Measurement (PAM).* doi:10.1007/978-3-032-18268-5_10

[4] Bilibili. 2026. *nginx-quic-stack.* https://github.com/bilibili/nginx_quic_stack Accessed: 2026-04-29.

[5] Bilibili. 2026. *quiche.* https://github.com/bilibili/quiche Accessed: 2026-04-29.

[6] Aurélien Buchet and Cristel Pelsser. 2025. An Analysis of QUIC Connection Migration in the Wild. *ACM SIGCOMM Computer Communication Review* (2025). doi:10.1145/3727063.3727066

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

[8] Alexandre Ferrieux, Isabelle Hamchaoui, Igor Lubashev, and Dmitri Tikhonov. 2020. *Packet Loss Signaling for Encrypted Protocols.* Internet-Draft draft-ferrieuxhamchaoui-quic-lossbits-03. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ferrieuxhamchaoui-quic-lossbits/03/ Work in Progress.

[9] Oliver Gasser, Quirin Scheitle, Pawel Foremski, Qasim Lone, Maciej Korczynski, Stephen D. Strowes, Luuk Hendriks, and Georg Carle. 2018. Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists. In *Proc. ACM Int. Measurement Conference (IMC).* doi:10.1145/3278532.3278564

[10] Yuri Gbur and Florian Tschorsch. 2023. QUICforge: Client-side Request Forgery in QUIC. In *Proc. Network and Distributed System Security Symposium (NDSS).* doi:10.14722/ndss.2023.23072

[11] Gray, Andrew and Kulkarni, Milind. 2025. *Introducing QUIC Protocol Support for Network Load Balancer: Accelerating Mobile-First Applications.* https://aws.amazon.com/blogs/networking-and-content-delivery/introducing-quic-protocol-support-for-network-load-balancer-accelerating-mobile-first-applications/ Accessed: 2026-04-29.

[12] 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. *ACM SIGCOMM Computer Communication Review* (2020). doi:10.1145/3411740.3411742

[13] Christian Huitema. 2022. *Quic Timestamps For Measuring One-Way Delays.* Internet-Draft draft-huitema-quic-ts-08. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-huitema-quic-ts/08/ Work in Progress.

[14] Internet Assigned Numbers Authority. 2026. *QUIC.* Retrieved 2026-03-10 from https://www.iana.org/assignments/quic/quic.xhtml

[15] Jana Iyengar, Ian Swett, and Mirja Kühlewind. 2026. *QUIC Acknowledgment Frequency.* Internet-Draft draft-ietf-quic-ack-frequency-14. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-ack-frequency/14/ Work in Progress.

[16] J. Iyengar and M. Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* RFC 9000. IETF. https://www.rfc-editor.org/rfc/rfc9000.txt

[17] Marcel Kempf, Benedikt Jaeger, Johannes Zirngibl, Kevin Ploch, and Georg Carle. 2024. QUIC on the Fast Lane: Extending Performance Evaluations on High-rate Links. *Computer Communications* 223 (2024), 90–100. doi:10.1016/j.comcom.2024.04.038

[18] 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 *Proc. ACM Int. Measurement Conference (IMC).* doi:10.1145/3278532.3278568

[19] Ike Kunze, Constantin Sander, and Klaus Wehrle. 2023. Does It Spin? On the Adoption and Use of QUIC's Spin Bit. In *Proc. ACM Int. Measurement Conference (IMC).* doi:10.1145/3618257.3624844

[20] M. Kühlewind and B. Trammell. 2022. *Manageability of the QUIC Transport Protocol.* RFC 9312. IETF. https://www.rfc-editor.org/rfc/rfc9312.txt

[21] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, and Mirja Kühlewind. 2026. *Managing multiple paths for a QUIC connection.* Internet-Draft draft-ietf-quic-multipath-21. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/21/ Work in Progress.

[22] Gustavo Luvizotto Cesar, Remi Hendriks, Martin Angelov, Mattijs Jonker, Roland Martijn van Rijswijk - Deij, Johannes Zirngibl, and Ralph Holz. 2026. A First Look at Deployment Strategies of Services Provisioned using Anycast. In *Proc. Network Traffic Measurement and Analysis Conference (TMA).*

[23] 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.* doi:10.1145/3405796.3405828

[24] Ayush Mishra and Ben Leong. 2023. Containing the Cambrian Explosion in QUIC Congestion Control. In *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC 2023, Montreal, QC, Canada, October 24-26, 2023.* ACM, 526–539. doi:10.1145/3618257.3624811

[25] Jonas Mücke, Marcin Nawrocki, Raphael Hiesgen, Patrick Sattler, Johannes Zirngibl, Georg Carle, Jan Luxemburk, Thomas C. Schmidt, and Matthias Wählisch. 2025. Waiting for QUIC: Passive Measurements to Understand QUIC Deployments. *Proc. ACM Netw.* (2025). doi:10.1145/3768988

[26] Marcin Nawrocki, Pouyan Fotouhi Tehrani, Raphael Hiesgen, Jonas Mücke, Thomas C. Schmidt, and Matthias Wählisch. 2022. On the Interplay between TLS Certificates and QUIC Performance. In *Proc. ACM Int. Conference on emerging Networking EXperiments and Technologies (CoNEXT).* doi:10.1145/3555050.3569123

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

[28] T. Pauly, E. Kinnear, and D. Schinazi. 2022. *An Unreliable Datagram Extension to QUIC.* RFC 9221. IETF. https://www.rfc-editor.org/rfc/rfc9221.txt

[29] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. 2018. A First Look at QUIC in the Wild. In *Proc. Passive and Active Measurement (PAM).*

[30] Constantin Sander, Ike Kunze, Leo Blöcher, Mike Kosek, and Klaus Wehrle. 2023. ECN with QUIC: Challenges in the Wild. In *Proc. ACM Int. Measurement Conference (IMC).* doi:10.1145/3618257.3624821

[31] D. Schinazi. 2022. *Proxying UDP in HTTP.* RFC 9298. IETF. https://www.rfc-editor.org/rfc/rfc9298.txt

[32] D. Schinazi and E. Rescorla. 2023. *Compatible Version Negotiation for QUIC.* RFC 9368. IETF. https://www.rfc-editor.org/rfc/rfc9368.txt

[33] Marten Seemann and Lucas Clemente. 2026. *quic-go.* https://github.com/quic-go/quic-go Accessed: 2026-04-29.

[34] Marten Seemann and Kazuho Oku. 2025. *QUIC Stream Resets with Partial Delivery.* Internet-Draft draft-ietf-quic-reliable-stream-reset-07. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-reliable-stream-reset/07/ Work in Progress.

[35] Ian Swett and Joseph Beshay. 2026. *QUIC Extended Acknowledgement for Reporting Packet Receive Timestamps.* Internet-Draft draft-ietf-quic-receive-ts-02. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-quic-receive-ts/02/ Work in Progress.

[36] M. Thomson. 2022. *Greasing the QUIC Bit.* RFC 9287. IETF. https://www.rfc-editor.org/rfc/rfc9287.txt

[37] W3techs. 2026. *Usage statistics and market shares of web servers.* https://w3techs.com/technologies/overview/web_server Accessed: 2026-04-29.

[38] 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 *Proc. ACM Int. Measurement Conference (IMC)* (Virtual Event, USA).

[39] Johannes Zirngibl, Florian Gebauer, Patrick Sattler, Markus Sosnowski, and Georg Carle. 2024. QUIC Hunter: Finding QUIC Deployments and Identifying Server Libraries Across the Internet. In *Proc. Passive and Active Measurement (PAM).* doi:10.1007/978-3-031-56252-5_13

[40] Johannes Zirngibl, Lion Steger, Patrick Sattler, Oliver Gasser, and Georg Carle. 2022. Rusty Clusters? Dusting an IPv6 Research Foundation. In *Proc. ACM Int. Measurement Conference (IMC).* doi:10.1145/3517745.3561440

A Ethics

This work relies on data collected by Zirngibl et al. [38]. They provided us access to the data for this study. They follow ethical measures based on community standards for all conducted scans [7, 27]. They scan with a small rate to reduce the impact on networks. They apply a list of IP prefixes and domains that should be excluded from scans based on operator requests. They provide contact information via WHOIS, reverse Domain Name System (rDNS) records and web servers on scan machines.

B Additional Figures & Tables

In this section, we provide additional figures that yield further insights into the dataset.

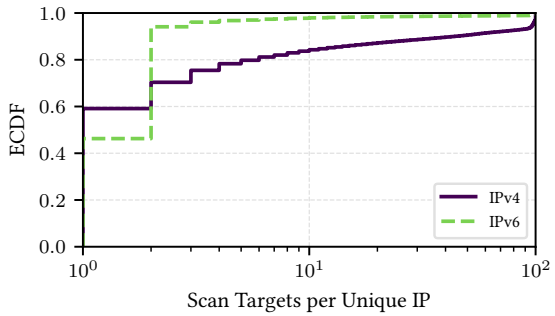


Figure 15: Distribution of the number of scan targets per IP address.

Figure 15 illustrates the distribution of the number of scan targets per unique IP address. In this work, we consider a scan target to be a unique combination of an IP address and a hostname while only considering the first scan target per IP address as target for our analysis. For IPv4, approximately 60 % of addresses are associated with only a single scan target, although there is also a notable fraction of addresses with multiple scan targets. In contrast, for IPv6, about 45 % of addresses correspond to a single target, while a substantial proportion of addresses are linked to exactly two targets.

Figure 17 shows the network types the 10 most popular libraries of Section 5.3 are deployed in. We take the self-reported `info_type` of the AS from the PeeringDB data of the day of the last scan and fall back to the most common type among the networks of the same organization when the type is not set. Per target, 82 % of the QUIC deployments sit in content networks, reflecting the presence of a few large ASes. Per AS, content networks account for 21 % and access networks for 42 %, indicating where QUIC is deployed across the Internet. However, two limits apply. First, PeeringDB is self-reported and covers only 57 % of the ASes and 88 % of the

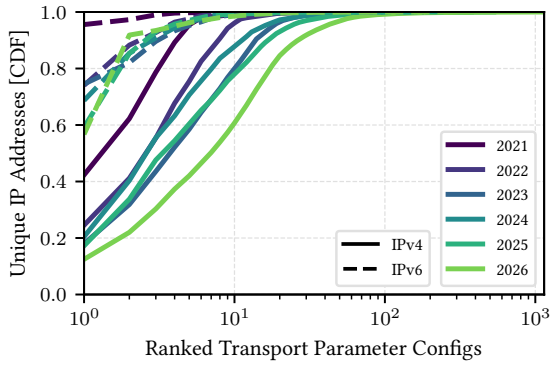


Figure 16: Cumulative distribution of unique IP addresses per transport parameter configuration per year. Configurations on the x-axis are sorted by the number of visible QUIC deployments in 2026. For each year, the first scan is shown.

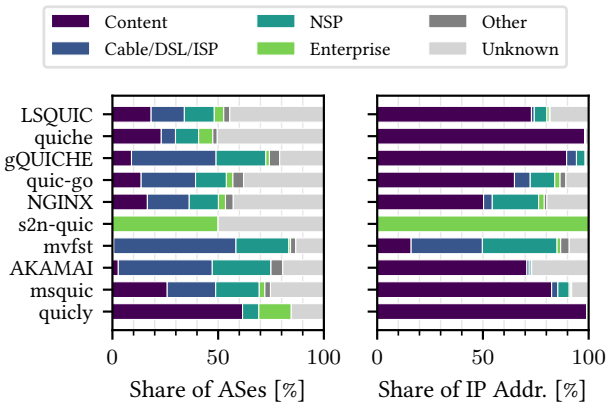


Figure 17: Network types the libraries are deployed in, weighted by ASes (left) and by targets (right), for the last IPv4 scan.

targets we observe. Second, we only use one snapshot and do not track the types over time.

Figure 16 illustrates the cumulative distribution of unique IP addresses per transport parameter configuration per year. Similar to Figure 3, the configurations on the x-axis are sorted in descending order by the number of unique IP addresses in 2026 and only the first scan of every year is considered. For both IPv4 and IPv6, the configurations are getting more diverse over the years. For IPv6, the share of IP addresses using the same config is significantly higher, with around 60 % in 2026 compared to around 15 % for IPv4.

Figure 18 gives more information on the distribution of values for the three `initial_max_stream_data_*` transport parameters. The mean values experience a significant jump above 10^{13} in early 2024, caused by a small number of targets setting those parameters to very high values, while the medians remain stable.

In Figure 19, we depict the evolution of the parameter `version_information` over time. Initially in 2021, up to 70 % of hosts used the `google_version` parameter, but this share steadily declined from mid-2022 to the end of 2025. During the standardization of QUIC version negotiation in RFC 9368 [32], the draft codepoint of `version_information` (draft-04) appeared and largely mirrored

Table 2: Top 10 ASes with the most unique IP addresses in the IPv4 dataset for the latest scan (February 2026). Libraries with less than 2 % share are grouped as "other".

#	ASN	Targets	Library Distribution	AS Name
1	396982	186 836	gQUICHE: 183 733, other: 3103	GOOGLE-CLOUD-PLATFORM, US
2	47583	183 191	LSQUIC: 151 410, quic-go: 22 220, NGINX: 9474, other: 87	AS-HOSTINGER, CY
3	13335	113 058	quiche: 113 051, other: 7	CLOUDFLARENET, US
4	54113	98 790	quicly: 98 790	FASTLY, US
5	15169	59 286	gQUICHE: 59 127, other: 159	GOOGLE, US
6	14061	37 769	quic-go: 24 596, LSQUIC: 8890, NGINX: 3348, other: 935	DIGITALOCEAN-ASN, US
7	16509	36 553	s2n-quic: 26 023, quic-go: 2920, gQUICHE: 2247, NGINX: 1317, other: 4046	AMAZON-02, US
8	20473	27 913	NGINX: 19 092, LSQUIC: 4808, quic-go: 3742, other: 271	AS-VULTR, US
9	24940	24 521	quic-go: 18 103, LSQUIC: 2892, NGINX: 2764, other: 762	HETZNER-AS, DE
10	16276	23 373	quic-go: 10 542, LSQUIC: 8083, NGINX: 3695, other: 1053	OVH, FR

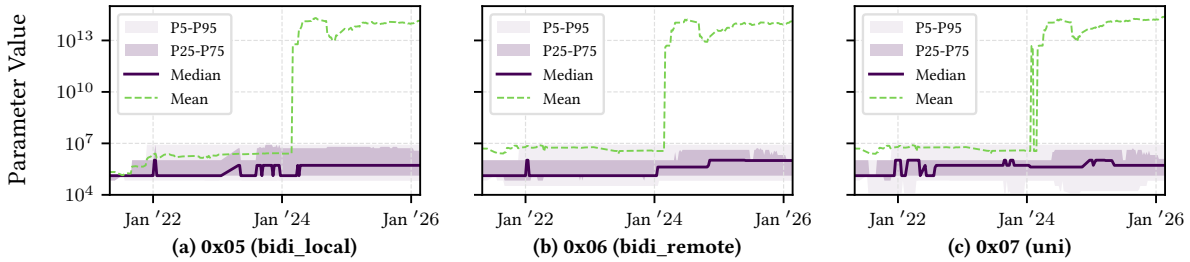


Figure 18: Temporal evolution of the median and mean of the three `initial_max_stream_data_*` transport parameter values.

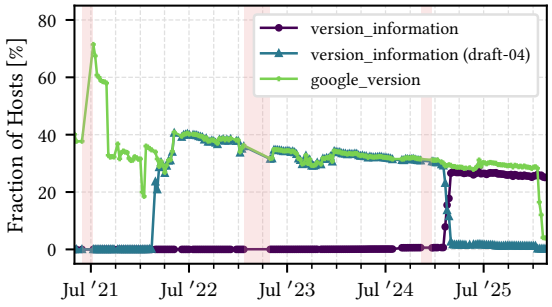


Figure 19: Temporal evolution of the `version_information` transport parameter in IPv4 scans, as discussed in Section 7.3.

this decline. Around March 2025, adoption of the draft codepoint dropped to nearly zero in favor of the standardized `version_information` parameter (`0x11`), which rose to almost 30 % of hosts. At the turn of 2025/2026, the use of `google_version` also dropped to nearly zero.

Section 5.2 summarizes how long targets stay in the dataset. We normalize the lifetime of each target to its maximum possible value and exclude the last six months of scans to avoid a bias towards short lifetimes. Both datasets then show peaks at the two extremes. In the IPv4 dataset, 13.2 % of the targets appear in a single scan, while 34.9 % appear in all scans after their first appearance. In the IPv6 dataset, the single-scan targets dominate with an average of 56.3 % and a share above 80 % in early 2026, while the share of targets with the maximum lifetime stays around 5 % throughout.

Figure 20 and Figure 21 give insights into the lifetime of targets in the IPv4 and IPv6 datasets, respectively. The x-axis shows the first scan in which a target was observed, while the y-axis indicates the lifetime of the target in days. The color intensity represents the number of targets with a given first-seen scan and lifetime. The heatmaps reveal that a significant portion of targets have either a very short lifetime of just one scan or are observed across all remaining scans, indicating a mix of transient and persistent QUIC deployments. These plots support the analysis in Section 5.2.

Table 3: Draft transport parameters and their corresponding identifiers discussed in Section 7.3.

ID	Name
`0x3e`	`initial_max_path_id`
`0x1057`	`loss_bits`
`0x4752`	`google_version`
`0x7158`	`enable_timestamp`
`0xbabf`	`enable_multipath`
`0xff00`	`facebook_partial_reliability`
`0xff0a001`	`ack_receive_timestamps_enabled`
`0xff0a002`	`max_receive_timestamps_per_ack`
`0xff0a003`	`receive_timestamps_exponent`
`0xfec001`	`fec_max_symbol_size`
`0xfec002`	`fec_version_02`
`0xfecb01`	`fec_version`
`0x17f7586d2cb571`	`reset_stream_at`
`0xf739bbc1b666d04`	`enable_multipath`
`0xf739bbc1b666d05`	`enable_multipath`
`0xf739bbc1b666d06`	`enable_multipath`
`0xf739bbc1b666d09`	`initial_max_path_id`
`0x0e08a234ff112300`	`pmtud_options`

C Transport Parameter Configuration Tests

Table 4 lists the three client configurations we use in Section 4.4 to evaluate whether server transport parameters are independent and not influenced by the client configuration.

Table 4: Client configurations used for the independence test in Section 4.4. The medium values match typical values servers use based on our dataset.

	Configuration		
Transport Parameter	**Low**	**Medium**	**High**
`max_idle_timeout`	5 s	30 s	600 s
`initial_max_data`	4 KiB	128 KiB	10 MiB
`initial_max_stream_data_bidi_local`	4 KiB	128 KiB	10 MiB
`initial_max_stream_data_uni`	4 KiB	128 KiB	10 MiB
`ack_delay_exponent`	0	3	20
`max_ack_delay`	0 ms	25 ms	16 383 ms

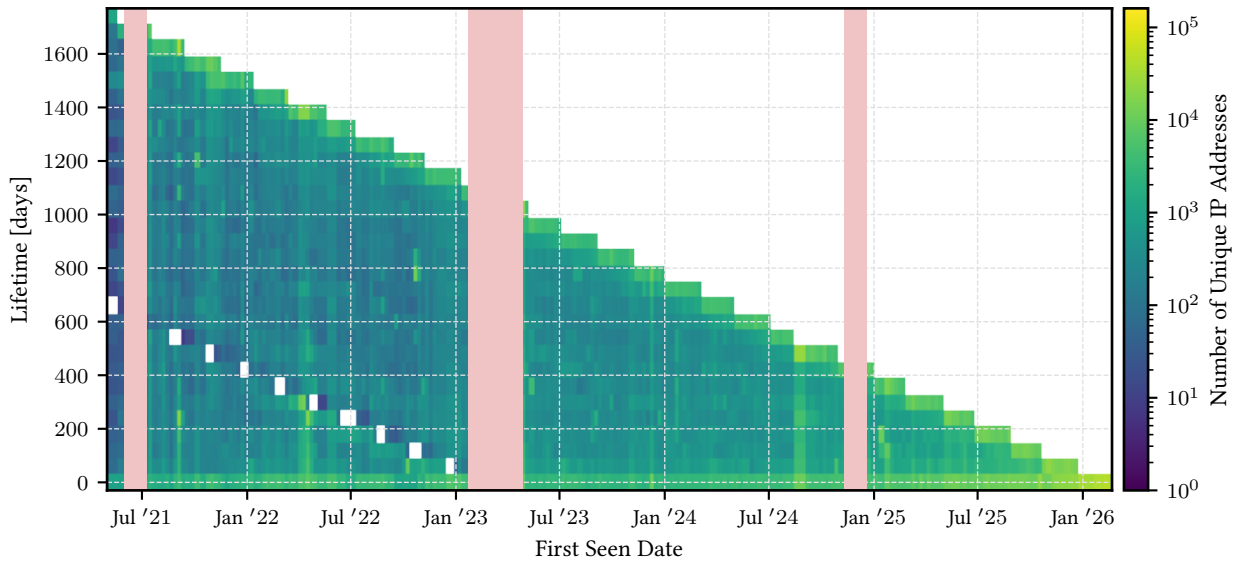


Figure 20: Lifetime of targets in the IPv4 dataset. The color indicates the number of targets, while the x-axis shows the first scan in which a target was observed, and the y-axis shows lifetime in days.

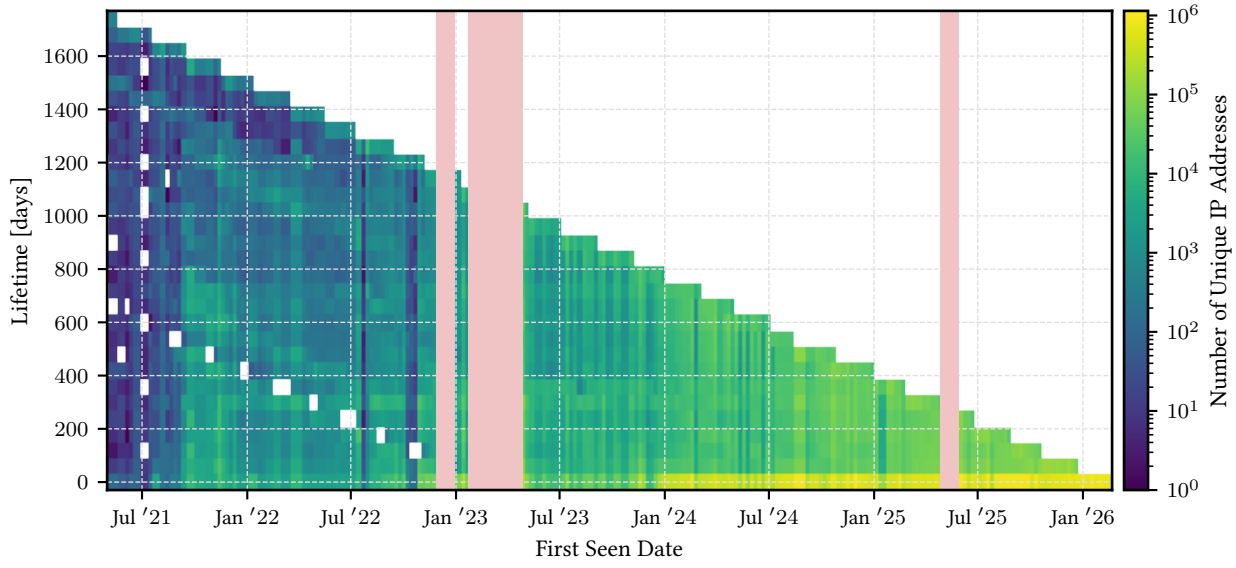


Figure 21: Lifetime of targets in the IPv6 dataset. The color indicates the number of targets, while the x-axis shows the first scan in which a target was observed, and the y-axis shows lifetime in days.