A First Look at QUIC in the Wild

Jan Rüth[1], Ingmar Poese[2], Christoph Dietzel[3], and Oliver Hohlfeld[1]

[1] RWTH Aachen University {rueth,hohlfeld}@comsys.rwth-aachen.de
[2] Benocs GmbH ipoese@benocs.com
[3] TU Berlin / DE-CIX christoph@inet.tu-berlin.de

Abstract. For the first time since the establishment of TCP and UDP, the Internet transport layer is subject to a major change by the introduction of QUIC. Initiated by Google in 2012, QUIC provides a reliable, connection-oriented low-latency and fully encrypted transport. In this paper, we provide the first broad assessment of QUIC usage in the wild. We monitor the entire IPv4 address space since August 2016 and about 46% of the DNS namespace to detected QUIC-capable infrastructures. Our scans show that the number of QUIC-capable IPs has more than tripled since then to over 617.59 K. We find around 161K domains hosted on QUIC-enabled infrastructure, but only 15K of them present valid certificates over QUIC. Second, we analyze one year of traffic traces provided by MAWI, one day of a major European tier-1 ISP and from a large IXP to understand the dominance of QUIC in the Internet traffic mix. We find QUIC to account for 2.6% to 9.1% of the current Internet traffic, depending on the vantage point. This share is dominated by Google pushing up to 42.1% of its traffic via QUIC.

1 Introduction

Recent years have fostered the understanding that TCP as the de-facto default Internet transport layer protocol has become a technological bottleneck that is hard to update. This understanding is rooted in the fact that optimizing throughput is no longer a key concern in the Internet, but optimizing latency and providing encryption at the *transport* has become a concern. The focus on latency results from shifted demands (e.g., by interactive web applications) and is currently proposed to be addressed in part by TCP extensions at the protocol level, e.g., TCP Fast Open [15] or Multipath TCP [16]. While optimizing latency there is an additional demand to also provide an encrypted transport, typically realized by TLS on top of TCP. Since this additional encryption adds additional latency, further optimizations address this latency inflation, e.g., 0-RTT in the upcoming TLS 1.3 standard [17]. While these approaches present clear advantages, their deployment is currently challenged by middleboxes and legacy systems.

Google's Quick UDP Internet Connections (QUIC) protocol [10] aims to address these shortcomings in a new way. Like TCP, it provides a connection-oriented, reliable, and in-order byte stream. Yet unlike TCP, it enables stream multiplexing over a single connection while optimizing for latency. By fully encrypting already at the transport layer, QUIC provides security and excludes (interfering) middlebox optimizations; thereby paving the way for a rapidly evolving transport layer. By implementing QUIC in user space on top of UDP, its ability to rapidly update and customize a transport

The final publications is available at Springer via http://dx.doi.org/10.1007/978-3-319-76481-8_19

per application has yet unknown consequences and motives measurements. It was first introduced to Chromium in 2012 and has undergone rapid development and high update-rate since then—as we will partly show in our measurements. Since 2016, the IETF QUIC working group [2] is working on its standardization. Google widely enabled QUIC for *all* of its users in January 2017 [10, 18], motivating our study capturing its first 9 months of general deployment. Yet, in contrast to TCP and TLS, there is very limited tool support to analyze QUIC and the academic understanding is currently limited to protocol security [7, 8, 12] and performance [3, 5, 9, 10].

In this paper, we complement these works by providing the first large-scale analysis of the current QUIC *deployments* and its *traffic share*. To assess the QUIC deployment, we regularly probe the entire IPv4 space for QUIC support since August 2016. In our scans, we observe a growing adoption on QUIC reaching 617.59 K IPs supporting QUIC in October 2017, of which 53.53% (40.71%) are operated by Google (Akamai). We additionally probe the complete set of .com/.net/.org domains as well as the Alexa Top 1M list, i.e., around 46% of the domain name space [20]. To assess the traffic share that these deployments generate, we analyzed traffic traces from three vantage points: *i)* 9 months of traffic in 2017 on a transit link to an ISP (MAWI dataset [13]), *ii)* one day in August 2017 at a European tier-1 ISP, representing edge (DSL + cellular) and backbone traffic, and *iii)* one day in August 2017 at a large European IXP. In these networks, QUIC accounts for 2.6% – 9.1% of the monitored traffic. The observed traffic is largely contributed by Google (up to 98.1% in the ISP) and only marginally by Akamai (0.1% in the ISP and 59.9% in the IXP), despite having a large number of QUIC-capable IPs. Our contributions are as follows.

– We analyze the development and deployment of QUIC in the IPv4 Internet.
– We present the first view on QUIC deployment and traffic outside of Google's network from three different vantage points.
– We build and together with this paper publish tools to: Enumerate QUIC hosts and tools to massively grab and decode QUIC protocol parameters.
– We publish all our active measurement data and future scans on [1].

Structure. Section 2 introduces the QUIC handshake as a basis for our host enumeration. Section 3 presents our view on QUIC in IPv4 and in three large TLDs as well as the tools that drive our measurements. Section 4 shows how QUIC reshapes traffic in local and ISP/IXP networks. Section 5 discusses related works and Section 6 concludes the paper.

2 An Introduction to QUIC's Handshake

We first introduce the QUIC connection establishment phase that we utilize in our measurements for host enumeration and certificate grabbing. For a broader discussion of QUIC's features and design choices we refer to [10]. We focus on the QUIC early deployment draft as the IETF draft is not yet fully specified.

One of QUIC's main features is a fast connection establishment: In the ideal case, when cached information of a prior connection is available, it does not even take a single round-trip (0-RTT) to send encrypted application data. Yet, in the worst case (without prior connections as in our measurements), QUIC needs at least three round-trips as shown in Figure 1 and explained next.

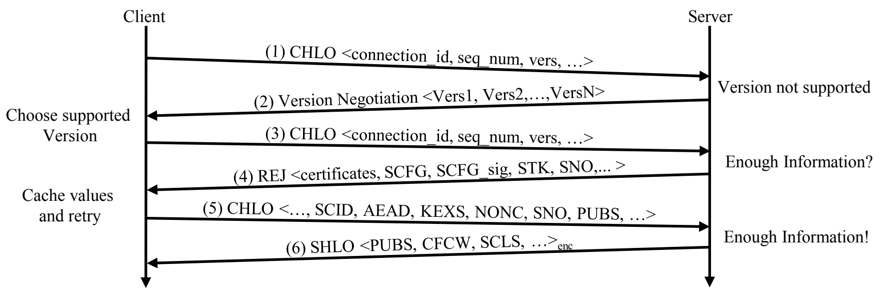
Client
Server
(1) CHLO <connection_id, seq_num, vers, ...>
Version not supported
(2) Version Negotiation <Vers1, Vers2,...,VersN>
Choose supported Version
(3) CHLO <connection_id, seq_num, vers, ...>
Enough Information?
(4) REJ <certificates, SCFG, SCFG_sig, STK, SNO,... >
Cache values and retry
(5) CHLO <..., SCID, AEAD, KEXS, NONC, SNO, PUBS, ...>
Enough Information!
(6) SHLO <PUBS, CFCW, SCLS, ...>enc

Fig. 1. A long QUIC handshake including version negotiation and caching of values.

Clients initiate a connection using a Client Hello (`CHLO`)(1) including the QUIC version it desires to use. In case the server does not support this version, it may send a version negotiation packet (2) enabling the client to choose from a list of supported versions for a second try. We will utilize packet (1) to quickly probe for QUIC-capable hosts with only a single packet exchange and analyze their supported versions provided in (2). Using a supported version, the client may advance in the handshake by sending another `CHLO` (3), without prior communication, it does not possess enough information about the server to establish a valid connection. The server supplies the necessary information (4), in one or multiple exchanges (i.e., (3) and (4) may be repeated until all required data is available). In these step(s), the client will be given a signed server config (`SCFG`) including supported ciphers, key exchange algorithms and their public values, and among other things the certificates authenticating the host. We will utilize these information to analyze the server-provided certificates. With this information, the client can issue another `CHLO` (5) including enough information to establish a connection, the client may even send encrypted data following the `CHLO` which depicts the optimal case for a 0-RTT connection establishment. Following the `CHLO`, the server acknowledges (6) the successful connection establishment with a Server Hello (`SHLO`), containing further key/value-pairs enabling to fully utilize the connection.

3 Availability: QUIC Server Infrastructures

We start by analyzing the availability of QUIC in the Internet, i.e., how many IPs, domains, and infrastructures support QUIC. If not stated otherwise, the results are based on scan data obtained in the first week of October 2017.

3.1 Enumerating QUIC IPv4 Hosts

IP Scan Methodology. To quickly probe the entire IPv4 space for QUIC capable hosts, we extend ZMap [6], which enables to rapidly enumerate IPv4 addresses. To identify QUIC hosts, we use QUIC's version negotiation feature (see Section 2). As QUIC is build to enable rapid protocol development and deployment, negotiation of a supported version (i.e., supported by client and server) is fundamental to its design. That is, the protocol requires to announce a version identifier in the initial packet sent from the client to the server. In case the version announced by the client is not supported by the

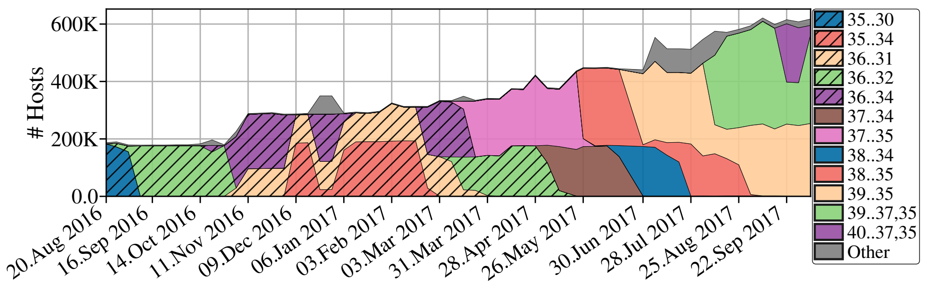

Fig. 2. Number of QUIC-capable IPs and support for sets of certain QUIC versions, here we display versions when there was support by at least 20000 hosts once. Versions that first appeared in 2016 are hatched.

server, it sends a version negotiation packet. This packet lists all supported versions by the server, enabling the client to find a common version that is used in a subsequent handshake. We leverage this feature and sent a valid handshake message containing a version that is likely to be *unsupported* by the other party, i.e., by including a version that is not reserved and does not follow the current pattern.In response, the server will not be able to continue the handshake as both versions do not match, thus, it will send a version negotiation packet containing a list of its supported versions. Using an invalid version has the advantage that we not only enumerate valid QUIC hosts but also gain further insights about the server, namely the list of its supported versions. We declare an IP as QUIC-capable, if we either receive a valid version negotiation packet or a QUIC public reset packet (comparable to a TCP RST). We build and publish [1] ZMap modules implementing this behavior enabling rapid enumeration of QUIC hosts in the IPv4 space.

QUIC Hosts. Figure 2 shows that the total number of QUIC-capable IPs (sum of stacked area) has more than tripled from 186.77 K IPs in August 2016 to 617.59 K IPs in October 2017. As of October, we find IPs in 3.04 K Autonomous Systems (ASs). To analyze who drives this trend, we attribute QUIC IPs to providers: we classify IPs by *i)* AS information, *ii)* per-IP X509 certificate data (e.g., who issued the certificate, who owns it), and *iii)* per-IP reverse DNS data (e.g., Akamai configures rDNS entries such as *.deploy.static.akamaitechnologies.com), using data available at Routeviews and scans.io. As of August 2016, we can already attribute 169.52 K IPs to Google. They have since doubled their QUIC-capable infrastructure to 330.62 K IPs as of October 2017, accounting for 53.53% of all QUIC-capable IPs. We identify Akamai as the second largest QUIC-enabler: they started to increasingly deploy QUIC on their servers in November 2016, while we find around 983 Akamai IPs in August, the number jumps to 44.47 K IPs in November 2016. Akamai has since then continued to deploy QUIC having 251.43 K IPs as of October 2017 accounting for 40.71% of all QUIC-enabled IPs.

To classify the remaining 35.54 K hosts, we executed TCP HTTP GET / on port 80 for these IPs. However, for 23.91 K IPs we could not get any data due to i/o timeouts. Apart from this, we find 7.34 K hosts announcing a *LiteSpeed* server string, a web server that added QUIC support in mid of July 2017 [11]. We find servers announcing *gws* (1.69 K) and *AkamaiGHost* (1.44 K), hinting at even more Google and Akamai installations. The fourth largest group of servers announces *Caddy* (356) as the server string, this server uses the quic-go [4] library and can also be used as a reverse proxy for other TCP-only servers.

Takeaway. *We observe a steady growth of QUIC-capable IPs, mainly driven by Google and Akamai. Few IPs already use third-party server implementations.*

QUIC Version Support. Since QUIC is under active development, it requires clients and servers to be regularly updated to support recent versions. To understand how the server infrastructure is updated, Figure 2 shows the number of hosts supporting a certain set of versions (recall: A host may support multiple versions!). The figure shows that many version combinations have a short lifespan in which old versions fade away and new versions appear. For example, hosts supporting version `Q035` down to version `Q030` switch to versions `Q036,...,Q032`, thus losing support for two versions. Yet, while some versions fade away, we also see that, e.g., version `Q035` is supported by almost all hosts over the course of our dataset. Even though, to the end of our observations support for version `Q036` is dropped. While this shows that some versions offer a long-term support, the figure also shows how vibrant the QUIC landscape is.

Given that some versions introduce radical protocol changes without backward compatibility, questions concerning the long-term stability of a QUIC-Internet are raised. On the one hand, the ability to easily update the protocol offers the possibility to quickly introduce new features and thereby to evolve the protocol. On the other hand, updating Internet systems is known to be notoriously hard. The vast amount of legacy systems raises the question of long-term compatibility—designing implementations to be easy to update is challenging.

Takeaway. *QUIC is currently subject to rapid development reflected in frequent version updates. Given its realization in user space at the application-layer, this property is likely to stay: future transports can be potentially updated as frequently as any other application. This motivates future measurements to assess the potentially highly dynamic future Internet transport landscape.*

3.2 Enumerating QUIC Domain Names

Methodology. We develop a second tool that finishes the handshake and enables to further classify previously identified hosts and infrastructures. To account for mandatory Server Name Indication (SNI), it can present a hostname that is necessary for the server to deliver correct certificates when hosting multiple sites on a single server. We base our tool [1] on the quic-go [4] library which we extended to enable tracing within the connection establishment to extract all handshake parameters (see Figure 1).

IP-based Certificate Scan. In a first step, we cluster all QUIC-enabled IPs discovered in Section 3.1 by their X509 certificate hash. This step enables to better understand QUIC-enabled infrastructures. Since the server's hostname is unknown at the request time when enumerating the IPv4 address space, we present dummy domains (e.g., foo.com) to each IP and retrieve the X509 certificate. The retrieved certificate provides information on the domain names for which the certificate is valid, which can indicate the hosting infrastructure. We remark that this approach yields the *default* website that is configured at a server and will not identify different sites in the presence of SNI. In fact, we find that 216.64 K hosts require SNI and do not deliver a certificate (for which we account for when scanning domain zones later). Figure 3 shows that we only observe 320

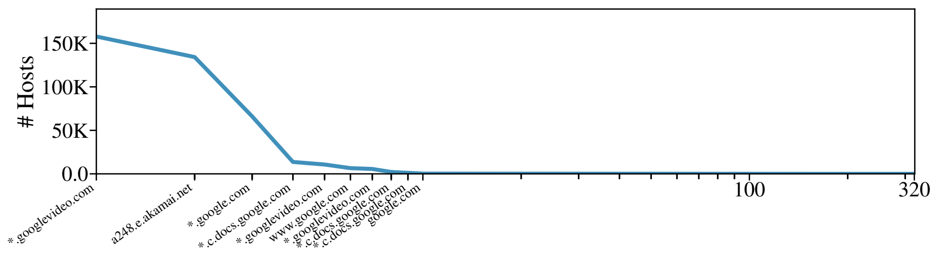

Fig. 3. Number of hosts giving out the same certificate on the y-axis. First listed common names for the 10 certificates with the highest coverage shown on the log x-axis.

different certificates for the probed 617.59 K QUIC IPs. The heavy-tailed distribution shows the top-five (ten) certificates already represent 95.41% (99.28%) of the IPs, most prominently Google and Akamai. We validated that these IPs actually belong to both companies by requesting content via TCP and HTTP on port 80 on the same hosts. We next assess QUIC support among domain names.

Probing complete domain lists. Presenting a non-existing SNI name in our previous measurement will miss any server that enforces to present a valid hostname, thus we next assess the QUIC support by probing complete domain name lists. That is, we probe all domains in the .com/.net/.org zone files and in the Alexa Top 1M list. These zones are available at Verisign [19] (.com/.net) and PIR [14] (.org). Together they contain more than 150M domains, i.e., about 46 % of the domain space [20]. We use zDNS to resolve the domains and for each successful resolution, we use our tool to check for QUIC support and to grab all parameters from the connection establishment. The whole process takes roughly 15 h and is thus feasible to run on a daily basis. Yet, as QUIC CHLO packets are padded to nearly fill the MTU, the scan easily saturates a 1Gbit link.

Table 1 shows the QUIC-support in the .com/.net/.org zones as well as in the Alexa Top 1M list. We define *QUIC-enabled* domains as being able to initiate a QUIC handshake. A domain is tagged as *Timeout* when we received no response to our initial QUIC `CHLO` within 12 seconds, e.g., in the absence of QUIC support. We furthermore show some specific errors as well as *DNS-failures*.

Overall QUIC-support is very low. Depending on the zone, 0.06%—0.1% domains are hosted on QUIC-enabled hosts. Only 1.6% – 2.44% of these domains present a valid X509 certificate. This questions how many domains actually deliver content via QUIC.

	06. Oct 2017		03. Oct 2017		04. Oct 2017		08. Oct 2017	
	.com		**.net**		**.org**		**Alexa 1M**	
# Domains	129.36 M	(100.0%)	14.75 M	(100.0%)	10.37 M	(100.0%)	999.94 K	(100.0%)
QUIC-enabled	133.63 K	(0.1%)	8.73 K	(0.06%)	6.51 K	(0.06%)	11.97 K	(1.2%)
Valid Certificate	2.14 K	(0.0%)	181	(0.0%)	159	(0.0%)	342	(0.03%)
Timeout	114.63 M	(88.61%)	10.80 M	(73.23%)	8.09 M	(78.06%)	826.67 K	(82.67%)
Version-failed	29	(0.0%)	6	(0.0%)	1	(0.0%)	5	(0.0%)
Protocol-error	606	(0.0%)	222	(0.0%)	0	(0.0%)	1	(0.0%)
Invalid-IP	322.24 K	(0.25%)	59.24 K	(0.4%)	40.15 K	(0.39%)	15.42 K	(1.54%)
DNS-failure	13.76 M	(10.64%)	2.40 M	(16.26%)	1.18 M	(11.41%)	49.34 K	(4.93%)

Table 1. QUIC support in different TLDs and in the Alexa Top 1M list. Weekly data is available at `https://quic.comsys.rwth-aachen.de`.

Landing Page Content. Websites can utilize different server configuration and even different server implementations for different protocols. The successful establishment of QUIC connections does thus not imply that meaningful content is being served. To assess how many QUIC-capable domains deliver content similar to their HTTP 1.1/2 counterparts, we instruct Google's QUIC test client (part of the Chromium source) to download their landing page via QUIC. We then compare their content to their HTTP 1.1/2 counterparts which should be similar if these QUIC-capable domains are properly set up. We disabled certificate checks to probe all capable domains. Out of the probed 161K domains, 16K (9.8%) return no data and 33K (20.7%) $>$ 1kB via QUIC. In case of the latter, 33K domains (22K served by Akamai) do deliver content similar to their HTTP 1.1/2 counterparts. We define similarity by structural HTML similarity (e.g., in the number of tags, links, images, scripts, ...) and require $>$ 3 metrics to agree to define a web page to be similar. Domains delivering similar content over QUIC are thus in principle ready to be served by a QUIC-capable browser. To be discovered by a Chrome browser, they, however, need to present an alternative service (`alt_srv`) header via TCP-based HTTP(S) pointing to their QUIC counterpart. 11K domains present this header via HTTPS (5K hosted by Google and 0 by Akamai) and only 7 via HTTP. Thus a large share of the domains would not be contacted by a Chrome browser even though QUIC support is in principle available. The header further specifies the QUIC versions supported by the server, of which at measurement time Chrome requires QUIC version 39. Only 5K domains present this version in their `alt_srv` header, all hosted by Google. We remark that our content analysis only regards *landing* pages and does not account for additional assets (e.g., images or videos). Particularly CDNs offer dedicated products for media delivery, whose QUIC support can differ. Assessing their QUIC support in detail thus provides an interesting angle for future work.
Takeaway. *The limited number of X509 certificates retrieved in our IP-based scan hints at the small number of different providers currently using or experimenting with QUIC. Furthermore, only a small fraction of the monitored domains are hosted on QUIC-capable infrastructures–an even smaller fraction can actually deliver valid certificates for the requested domains. Regardless, of the certificate, many QUIC-enabled domains do deliver their pages via QUIC. Yet in our measurements, many would not be contacted by a Chrome browser, either because of a non-present `alt_srv` header or insufficient version support. There is thus a big potential to increase QUIC support. We next study how this QUIC-support is reflected in actual traffic shares.*

4 Usage: QUIC Traffic Share

We quantify the QUIC traffic share by analyzing three traces representing different vantage points: *i)* 9 months of traffic in 2017 on a transit link to an upstream ISP (MAWI dataset [13]), *ii)* one day in August 2017 at a European tier-1 ISP, representing edge (DSL + cellular) and backbone traffic, and *iii)* the same day at a large European IXP.
Traffic Classification. We use protocol and port information to classify HTTPS (TCP port 443), HTTP (TCP port 80), and QUIC (UDP port 443). We chose this classification since it is applicable to all of our traces: MAWI (PCAP header traces) and ISP + IXP (Netflow traces without protocol headers). We remark that this classification can *i)* miss

	Overall				Operator's share			Share in Protocol		
	HTTP	HTTPS	QUIC		HTTP	HTTPS	QUIC	HTTP	HTTPS	QUIC
MAWI	28.0%	44.9%	6.7%	-		-			-	
ISP	37.7%	40.1%	7.8%	Akamai	67.9%	32.1%	0.1%	27.2%	12.6%	0.1%
				Google	1.4%	59.5%	39.1%	0.7%	28.8%	98.1%
Mobile ISP	24.8%	55.4%	9.1%	Akamai	57.7%	42.3%	0.0%	28.5%	9.6%	0.1%
				Google	1.6%	64.4%	34.0%	1.8%	29.5%	96.9%
IXP	32.2%	30.9%	2.6%	Akamai	33.3%	33.3%	33.3%	5.0%	5.2%	59.9%
				Google	3.1%	70.0%	26.9%	0.3%	7.2%	33.1%

Table 2. Average traffic shares (overall), among the operators, and among the protocol. Operator's share is e.g., from all of Google's traffic the share of the QUIC traffic at a vantage point. Share in Protocols denotes the traffic share of a protocol at a vantage point, e.g., the amount of Google QUIC traffic from all other QUIC traffic.

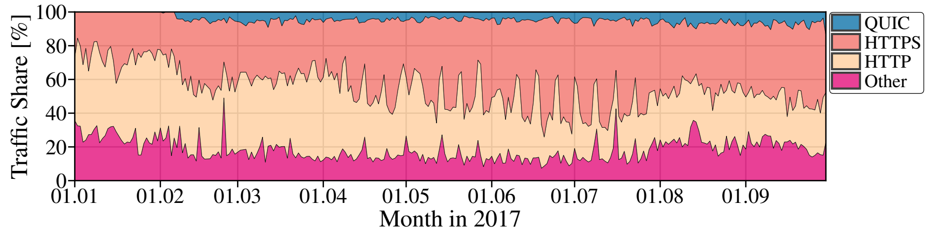

Fig. 4. Traffic share of QUIC compared to HTTP and HTTPS in the MAWI trace.

protocol traffic on non-standard ports and can *ii)* wrongly attribute other traffic on the monitored ports. However, it still enables to report an upper bound on the protocol usage on standard ports. We show the per-trace traffic shares in Table 2 which we discuss next.

MAWI Backbone Trace. We start by analyzing traffic on a trans-Pacific WIDE backbone link provided by the MAWI working group [13]. We analyze anonymized header traces available at the MAWI repository (*samplepoint F*). The monitored link is a transit link connecting the WIDE backbone to an upstream ISP. The traces involve 15 minutes of traffic captured at 14h on each day. Each packet is caped to the first 96 bytes.

We begin to analyze traffic on January 1st 2017, since Google enabled QUIC for all of its Chrome and Google-developed Android App users in January 2017 [10]. Figure 4 shows the traffic volume until end of September 2017. The trace shows that the QUIC traffic share is 0.0% in January. This is in contrast to the Google report of having widely enabled QUIC in January, suggesting that the monitored user-base is not using Google products (e.g., Chrome) at the time, QUIC has not been enabled for this network or that traffic is routed differently. We observe the first QUIC traffic in February where the QUIC traffic share is at 3.9%. It continues to increase to 5.2% in March and reaches 6.7% in September. QUIC offers an alternative to TCP+TLS, which is the foundation of legacy HTTPS, its share is at around 44.9%, even the unencrypted version HTTP is still at around 28.0%. As the provided trace anonymizes destination and source addresses, we cannot attribute this traffic to infrastructures (e.g., Google or Akamai) or services (e.g., YouTube). We leave this analysis to the ISP trace for which we have AS-level information available.

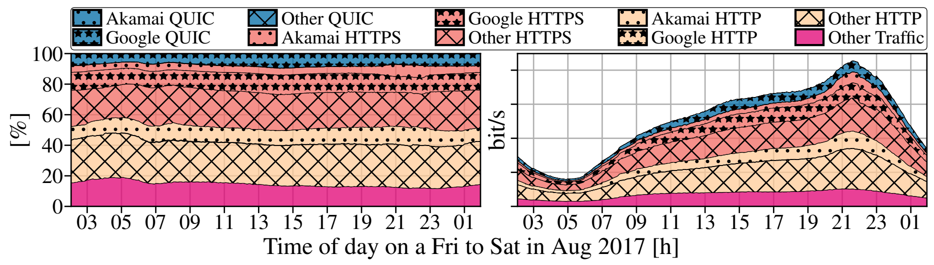

Fig. 5. QUIC traffic share in a major European ISP (up- and downstream). Left, relative share of QUIC. Right, total traffic compared to HTTP(S), y-axis has been anonymized at the request of the ISP. Nearly all QUIC traffic is served by Google.

Takeaway. *Within nine months after its general activation by Google, QUIC already accounts for a non-negligible traffic share, demonstrating its ability to evolve Internet transport.*

European Tier-1 ISP. We obtained anonymized and aggregated Netflow traces from all border routers of a large European ISP for one day in August 2017. The Netflow traces were aggregated to 5-minute bins and all IP addresses were replaced by AS numbers before they were made available to us. Thus the traces do not reveal the behavior of individual users. The captured traffic contains *i)* edge traffic by DSL, *ii)* cellular customers, and *iii)* transit backbone traffic.

Figure 5 shows the traffic volume (up- and downstream) over the course of 24 h by protocol and prominent infrastructures (the traffic volume (y-axis) has been removed at the request of the ISP). As our previous host-based analysis (see Section 3) showed that QUIC is mainly supported by Google and Akamai servers, we also show their traffic shares (according to their AS numbers). At first, we observe that QUIC traffic follows the same daily pattern as HTTP and HTTPS. On average QUIC accounts for 7.8% of the traffic with a standard deviation of σ: 1.0%. This deviation is similar to HTTP (σ: 1.2%) and HTTPS (σ: 1.4%) which account for 37.7% and 40.1% of the traffic, respectively.

The observed QUIC traffic is almost exclusively contributed by Google: They account for 98.1% of the overall observed QUIC traffic. Among all of Google's traffic, 39.1% is using QUIC (σ: 2.3%), peaking at 42.1%. This is a larger share than a global average of 32 % reported by Google in November 2016 [10]. Currently, QUIC is mainly supported by Google-developed applications (e.g., Chrome or the Youtube Android app). In the absence of QUIC libraries, third-party support is low (e.g., Opera has optional QUIC and Firefox no QUIC support). The availability of QUIC libraries thus has the potential to drastically improve client support and therefore increase QUIC's traffic share.

In contrast, Akamai only serves 0.1% of its traffic via QUIC—despite contributing a large portion of the overall QUIC-capable IPs (40.71%, see Section 3.1). This discrepancy between the number of IPs and the traffic share suggests that QUIC is not yet widely activated among all customers/products. Yet on average, Akamai accounts for 10.3% (HTTP) and 5.1% (HTTPS) of all traffic and thus, together with the fact that they already have a QUIC-capable infrastructure, has the potential to shift more traffic towards QUIC. A higher QUIC share has several implications, while QUIC and TCP are generally similar in nature, subtle differences in the protocols may influence the performance of

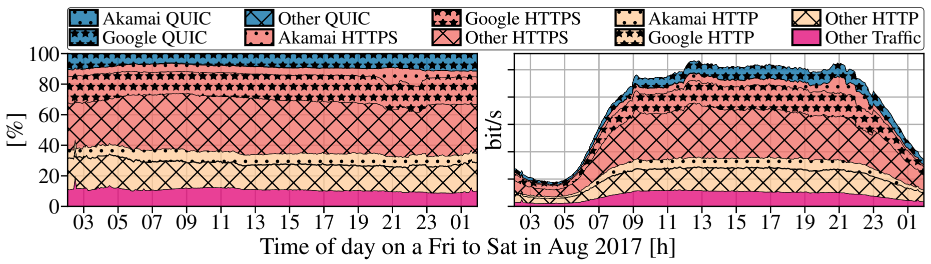

Fig. 6. Mobile network traffic share of QUIC in a major European ISP. Left, relative share of QUIC traffic. Right, absolute traffic share compared to HTTP(S), y-axis has been anonymized at the request of the ISP.

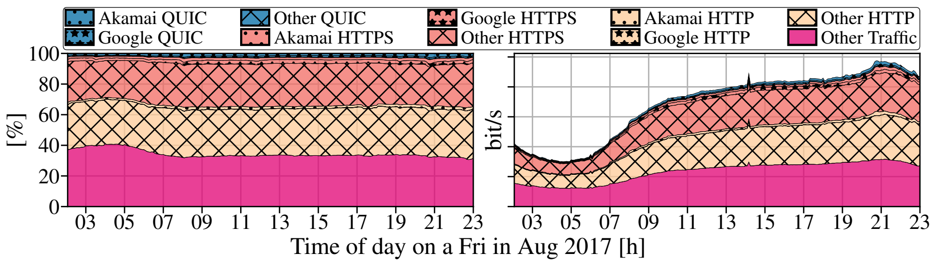

Fig. 7. QUIC traffic share at a large European IXP. Left, relative share of QUIC traffic. Right, absolute traffic share compared to HTTP(S), y-axis has been anonymized at the request of the IXP.

whole networks, e.g., by default QUIC uses larger initial congestion windows than those standardized for TCP by IETF and demands pacing for smoothing the traffic.

Mobile ISP. The ISP supplied us with information which traffic is for their mobile (cellular) customers, which we show in Figure 6. Please note that the reported mobile traffic is also contained in Figure 5. In contrast to the entire network of the ISP, the mobile traffic shows a different traffic pattern: while its throughput also decreases over night, mobile traffic rapidly increases in the morning and stays rather constant over the course of the day. Apart from this, the average QUIC share in the mobile network of 9.1% (σ: 1.4%), the highest share among all traces (see Table 2). In contrast, among the entire mobile Google traffic, only 34.0% (σ: 2.6%) is served via QUIC, lower than overall for the ISP. Also for mobile traffic, Akamai only serves a negligible share of its traffic via QUIC and thus has the potential increase the QUIC traffic share.

Takeaway. *QUIC traffic shares do (yet) not reflect server support. While Akamai operates a comparably large infrastructure in the number of QUIC-capable IPs, QUIC traffic is (still) almost entirely served by Google: this is likely to change.*

European IXP. We obtained sampled flow data of a large (European) IXP for the same day in August as for the ISP and show its traffic share in Figure 7. We classify Google and Akamai traffic by customer port information—since both peer at the IXP—and plot their HTTP(S) and QUIC traffic shares similar to the ISP. On average QUIC accounts for 2.6% (σ: 1.0%) of the traffic, which is lowest share among all traces (see Tables 2). Unlike the ISP, the largest portion is now contributed by Akamai (59.9%) and we observe a lower share of Google traffic (33.1%)—recall that Google contributed 98.1% of the QUIC traffic at the ISP.

Takeaway. *(Per-CDN) traffic shares largely depend on the chosen vantage point.*

Discussion. We observe different QUIC traffic shares at the ISP/IXP and particularly different shares of the QUIC traffic by Google / Akamai (relative to the overall traffic of each vantage point). These vantage point dependent differences are likely caused by different traffic engineering strategies since both providers peer at both vantage points. These differences highlight that observed traffic shares are in general highly vantage point dependent. Understanding the incentives for these different traffic engineering strategies is an interesting starting point for future research.

5 Related Work

QUIC Security. A first security analysis QUIC's key exchange is presented in [7], followed by a later analysis of the complete protocol [12]. These works on the security analysis are complemented by presenting an attack vector in which the server config can be computed offline to impersonate the server [8].
QUIC Performance. An early performance comparison of QUIC and HTTP1 [3] indicates that QUIC can provide better goodput and lower page loading times as traditional HTTP1 over TCP. A more extensive evaluation in [5] also involves the comparison to HTTP2 and shows that QUIC can outperform HTTP2 over TCP/TLS, a finding that is supported by extensive evaluations in [9]. The reported performance experience by Google [10] shows that QUIC lowers the Google search latency by 3.6–8% and reduces YouTube rebuffering by 15–18%.

We complement these works by providing the first broad assessment of QUIC usage in the wild and outside Google's network. We study both the QUIC-enabled infrastructures and its traffic shares from three vantage points.

6 Discussion and Conclusion

This paper presents the first broad assessment of QUIC, nine months after the general activation by Google for all of its users. We study both the available *infrastructure* in terms of the number of QUIC-capable IPs and domains and their *traffic share* at three vantage points. By probing the entire IPv4 address space, we find a steadily growing number of QUIC-enabled IPs which has tripled since August 2016 and reached 617.59 K in October 2017. This growth is mainly driven by Google and Akamai, which account for 53.53% and 40.71% of these IPs. When regularly probing $\approx$ 150M domains for QUIC support, we observe 161K capable domains of which 33K serve content similar to their HTTP1/2 counterparts and only 15K present valid certificates. Many (of the non-Google hosted) domains would not be contacted by a Chrome browser, either because of a non-present alternative service headers in HTTP(S) or insufficient version support. This infrastructure size does, however, not reflect their traffic share: depending on the vantage point, Google accounts for 98.1% (ISP) of the QUIC traffic and Akamai contributes 0.1% (ISP) to 59.9% (IXP), despite operating a similarly large number of QUIC-capable IPs. Given the factors that impede QUIC support, the QUIC traffic share is likely to increase in the future when being largely enabled at a wide range of infrastructures.

Realized as user space application-layer protocol, QUIC paves the way towards a rapidly evolving transport that can be updated as easily and as frequently as any application. This aspect is manifested in the short lifetime of QUIC versions observed in our measurements while the protocol is still under development. In light of these findings we expect a highly dynamic future Internet transport landscape to be studied and observed by future work.

Acknowledgments

This work has been funded by the DFG as part of the CRC 1053 MAKI and SPP 1914 REFLEXES, and by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960). We would like to thank the network operators at RWTH Aachen University, especially Jens Hektor and Bernd Kohler. We further thank our shepherd Tobias Flach and the anonymous reviewers. Furthermore, we would like to thank Konrad Wolsing for maintaining our changes to the quic-go implementation.

References

1. Active measurements and tools. `https://quic.comsys.rwth-aachen.de`.
2. IETF QUIC working group. `https://datatracker.ietf.org/wg/quic/about/`.
3. G. Carlucci, et al. HTTP over UDP: An Experimental Investigation of QUIC. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015.
4. L. Clemente. quic-go. `https://github.com/lucas-clemente/quic-go`.
5. S. Cook, et al. QUIC: Better For What and For Whom? In *IEEE ICC*, 2017.
6. Z. Durumeric, et al. Zmap: Fast internet-wide scanning and its security applications. In *USENIX Security*, 2013.
7. M. Fischlin and F. Günther. Multi-Stage Key Exchange and the Case of Google's QUIC Protocol. In *ACM CCS*, 2014.
8. T. Jager, et al. On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 V1.5 Encryption. In *ACM CCS*, 2015.
9. A. M. Kakhki, et al. Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols. In *ACM IMC*, 2017.
10. A. Langley, et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *ACM SIGCOMM*, 2017.
11. LiteSpeed Technologies Inc. LiteSpeed — Release Log. `https://www.litespeedtech.com/products/litespeed-web-server/release-log`.
12. R. Lychev, et al. How Secure and Quick is QUIC? Provable Security and Performance Analyses. In *IEEE Symposium on Security and Privacy*, 2015.
13. MAWI Working Group Traffic Archive. `http://mawi.nezu.wide.ad.jp/mawi/`.
14. Public Interest Registry. Zone File Access. `http://pir.org/`.
15. S. Radhakrishnan, et al. TCP Fast Open. In *ACM CoNEXT*, 2011.
16. C. Raiciu, et al. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *NSDI*, 2012.
17. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-tls13-21, Internet Engineering Task Force, 2017. WiP.
18. I. Swett. QUIC - Deployment Experience @Google. `https://www.ietf.org/proceedings/96/slides/slides-96-quic-3.pdf`.
19. Verisign. Zone Files For Top-Level Domains (TLDs). `verisign.com`.
20. Verisign. The verisign domain name industry brief. `https://www.verisign.com/assets/domain-name-report-Q22017.pdf`, Sept. 2017.