HTTP/3 Explained

by Daniel Stenberg

Inhaltsverzeichnis

- Introduction 1.1
- Warum QUIC 1.2
  - Erinnerst du dich an HTTP/2? 1.2.1
  - TCP Head-of-line blocking 1.2.2
  - TCP oder UDP 1.2.3
  - Ossifikation 1.2.4
  - Sicherheit 1.2.5
  - Reduzierte Latenz 1.2.6
- Prozess 1.3
  - IETF 1.3.1
  - Erfahrungen von HTTP/2 1.3.2
  - Status 1.3.3
- Eigenschaften des Protokolls 1.4
  - UDP 1.4.1
  - Zuverlässige Datenübertragung 1.4.2
  - Streams 1.4.3
  - In-Order Ausliefung 1.4.4
  - Schnelle Handshakes 1.4.5
  - TLS 1.3 1.4.6
  - Transport und Anwendung 1.4.7
  - HTTP/3 über QUIC 1.4.8
  - Nicht-HTTP über QUIC 1.4.9

HTTP/3 erklärt

Die Arbeit an diesem Buch wurde im März 2018 gestartet. Es dient der Dokumentation von HTTP/3 und dessen zugrundeliegendem Protokoll: QUIC. Warum, wie sie funktionieren, Protokolldetails, die Implementierung und mehr.

Dieses Buch ist zur Gänze frei zugänglich und wird als kollaborative Angstrengung von jedem getragen, der aushelfen will.

Voraussetzungen

Einer Leserin oder Leser wird ein Grundverständnis von TCP/IP, den Grundlagen von HTTP und dem Web vorausgesetzt. Für weitere Einblicke und Details über HTTP/2 empfehlen wir: http2 explained.

Autor

Dieses Buch wurde geschrieben von Daniel Stenberg. Daniel ist der Gründer und Lead Developer von curl, der meistgenutzten HTTP Client Software der Welt. Daniel hat über zwei Jahrzehnte lang mit und an HTTP sowie Internetprotokollen gearbeitet und ist der Autor von http2 explained.

Website

Die Website von diesem Buch kann hier gefunden werden: daniel.haxx.se/http3-explained.

Aushelfen

Solltest du Irrtümer, fehlende Details, Fehler oder offensichtliche Lügen in diesem Dokument finden, dann sende uns bitte eine ausgebesserte Version des betroffenen Paragrafen und wir werden diese als neue Version veröffentlichen. Natürlich führen wir alle an, die aushelfen. Ich hoffe, dass wir das Dokument im Laufe der Zeit verbessern können.

Fehler sollten als Issues oder als Pull Requests auf der GitHub Seite des Buches übermittelt werden.

Lizenz

Warum QUIC

QUIC ist ein Name, kein Akronym. Es wird genauso wie das englische Wort "quick" ausgesprochen.

Aus vielen verschiedenen Blickwinkeln kann QUIC als Lösung verstanden werden, wie ein neues, zuverlässiges und sicheres Transportprotokoll umgesetzt wird, das für ein Protokoll wie HTTP adäquat ist und die durch die Umsetzung von HTTP/2 über TCP und TLS entstandenen Mängel behebt. Der logische nächste Schritt in der Web-Transport Evolution.

QUIC ist nicht nur auf den Transport von HTTP limitiert. Der Wunsch nach einer schnelleren Auslieferung von Daten zum Endnutzer über das Web ist wahrscheinlich der größte Grund, der die Erschaffung dieses neuen Transportprotokolls angestoßen hat.

Warum also ein neues Transportprotokoll erschaffen und warum aufbauend auf UDP?

Erinnerst du dich an HTTP/2?

Die HTTP/2 Spezifikation RFC 7540 wurde im Mai 2015 veröffentlicht und seitdem flächendeckend im Internet und World Wide Web implementiert sowie ausgeliefert.

Anfang 2018 wurden fast 40% der Top 1000 Websites mit HTTP/2 ausgeliefert. Ungefähr 70% aller HTTPS Anfragen, die über Firefox verschickt wurden, bekamen eine HTTP/2 Antwort zurück. Die meistgenutzten Browser, Server und Proxies unterstützen die Spezifikation.

HTTP/2 behebt eine Fülle an Mängel von HTTP/1, weshalb HTTP-Nutzer nicht mehr zu Übergangslösungen greifen müssen - welche gerade Web Entwickler oft belastet haben.

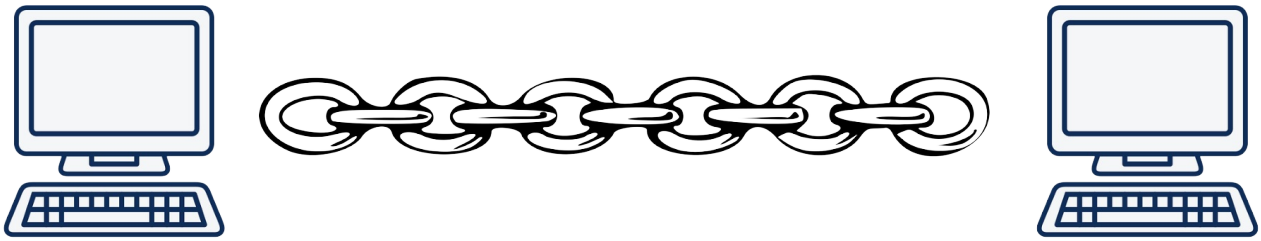
Eines der Hauptmerkmale von HTTP/2 ist Multiplexing, die Möglichkeit viele logische Streams über die gleiche physische TCP Verbindung zu schicken. Das macht viele Dinge besser und schneller; es verbessert die Überlastungskontrolle wesentlich, es lässt User TCP besser nutzen um die Bandbreite besser auszunutzen, die TCP Verbindungen sind langlebiger - was gut ist, weil Verbindungen öfter die volle Geschwindigkeit aufbauen können. Header-Komprimierung nutzt dabei weniger Bandbreite.

Mit HTTP/2 nutzen Browser typischerweise *eine* TCP Verbindung zu jedem Host, anstatt der vorherigen *sechs*. Tatsächlich verringert sich die Anzahl der Verbindungen wegen der Verbindungsvereinigung und "Desharding" Techniken von HTTP/2 noch viel wesentlicher.

HTTP/2 löst das Problem des sogenannten HTTP Head-of-line blockings, bei dem Clients so lange warten müssen, bis eine gestellte Anfrage fertig ist, bevor die nächste gestellt werden kann.

TCP Head-of-line blocking

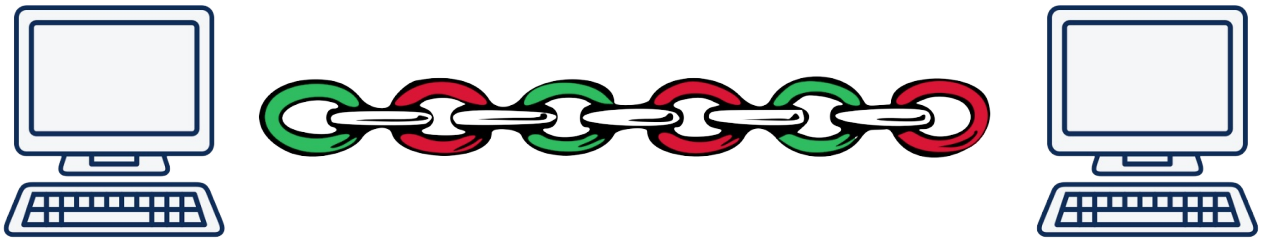
HTTP/2 nutzt TCP, wenngleich es im Gegensatz zu früheren HTTP Versionen weniger TCP Verbindungen benötigt. TCP ist ein Protokoll für eine zuverlässige Übertragung und kann als imaginäre Kette zwischen zwei Computern gesehen werden. Was vom ersten Computer über das Netzwerk übermittelt wird, wird schließlich beim zweiten Computer in der gleichen Reihenfolge ankommen (oder die Verbindung ist unterbrochen).



Mit HTTP/2 machen Browser zehn oder hunderte Übertragungen gleichzeitig über eine einzige TCP Verbindung.

Wenn ein einziges Paket fallen gelassen wird oder im Netzwerk irgendwo zwischen den beiden Endpunkten - welche über HTTP/2 kommunizieren - verloren geht, dann wird die gesamte TCP Verbindung solange gestoppt, bis das verlorene Paket wieder übertragen wurde und ihren Weg zum Ziel gefunden hat. Weil TCP diese "Kette" darstellt, muss alles - sollte eine Verbindung plötzlich fehlen - was nach der verlorenen Verbindung kommen würde, warten.

Eine Illustration der Ketten-Metapher, wenn zwei Streams über diese Verbindung versendet werden. Ein roter und ein grüner Stream:



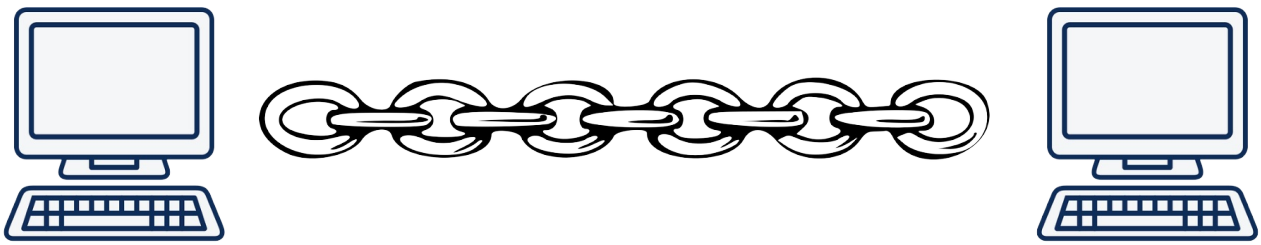
Es resultiert ein TCP basierender Head-of-line block!

Steigt die Verlustrate von Paketen, liefert HTTP/2 eine immer schlechtere Performance. Bei einer 2%-igen Verlustrate (was wohlgemerkt eine schlechte Netzwerkqualität ist) haben Tests gezeigt, dass HTTP/1 User normalerweise besser aussteigen - weil diese bis zu sechs TCP-Verbindungen haben, auf welche verlorene Pakete aufgeteilt werden können. Das bedeutet, dass für jedes verlorene Paket eine andere Verbindung weiterarbeiten kann.

Dieses Problem zu beheben ist nicht einfach - wenn überhaupt mit TCP möglich.

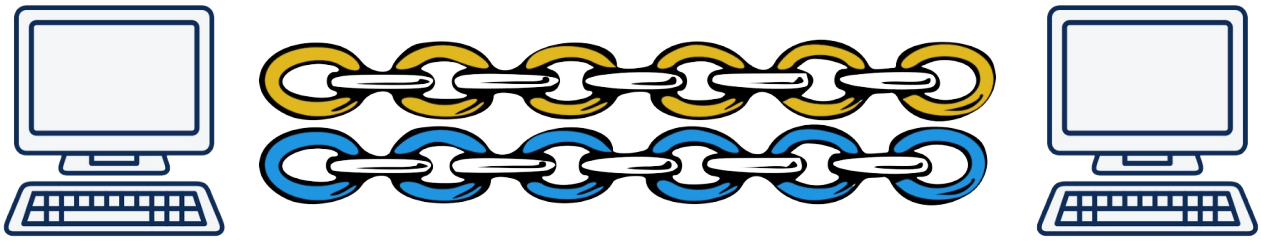
Unabhängige Streams vermeiden den Block

Mit QUIC gibt es noch immer einen Verbindungsaufbau zwischen den beiden Endpunkten, welcher die Verbindung sicher und den Datentransport zuverlässig macht.



Wenn zwei unterschiedliche Streams über diese Verbindung aufgebaut werden, werden diese unabhängig voneinander behandelt, sodass - sollte eine Verbindung für einen der Streams verloren gehen - nur der eine Stream, diese eine Kette pausieren und darauf warten muss, dass die fehlende Verbindung wieder versendet wird.

Hier wird dies mit einem gelben und einem blauen Stream illustriert, die zwischen zwei Endpunkten versendet werden.



TCP oder UDP

Wenn wir Head-of-line blocking innerhalb von TCP nicht reparieren können, dann sollten wir theoretisch ein neues Transportprotokoll neben UDP und TCP im Netzwerkstack erstellen können. Oder vielleicht sogar SCTP nutzen; ein Transportprotokoll mit etlichen der von uns gewünschten Charakteristiken, welches von IETF in RFC 4960 standardisiert wurde.

Jedoch ist die Entwicklung von neuen Transportprotokollen in den letzten Jahren nahezu vollständig eingestellt worden, weil diese nur unter schwierigen Umständen im Internet eingesetzt werden können. Der Einsatz neuer Protokolle wird durch viele Firewalls, NATs, Router und andere Netzwerk-Appliances verhindert, welche lediglich TCP oder UDP in der Kommunikation zwischen User und Server zulassen. Ein weiteres Transportprotokoll einzuführen hat die Auswirkung, dass N% der Verbindungen fehlschlagen, weil diese durch Appliances geblockt werden, welche die Verbindungen wegen der Nutzung eines anderen Protokolls als UDP oder TCP als böswillig einstufen. Die N% Fehlerrate wird oft als zu hoch erachtet, um den Aufwand zu rechtfertigen.

Werden Änderungen auf der Ebene des Transportprotokolls im Netzwerkstack vorgenommen, bedeutet das, dass Protokolle über den Kernel des Betriebssystems implementiert werden. Kernel des Betriebssystems zu aktualisieren und auszuliefern ist ein langsamer Prozess, der signifikanten Aufwand mit sich bringt. Viele Verbesserungen von TCP, welche durch IETF standardisiert wurden, sind nicht flächendeckend eingesetzt, weil sie nicht umfassend unterstützt werden.

Warum nicht SCTP-over-UDP

SCTP ist ein zuverlässiges Transportprotokoll mit Streams. Für WebRTC gibt es sogar existierende Implementierungen von SCTP über UDP.

Aus folgenden Gründen wurde SCTP-over-UDP als nicht gut genug befunden, um QUIC zu ersetzen:

- SCTP löst das Head-of-line-blocking Problem für Streams nicht
- SCTP hat keine solide TLS/Security Geschichte
- SCTP hat einen 4-way handshake, QUIC bietet 0-RTT
- QUIC ist ein Bytestream wie TCP, SCTP basiert auf Nachrichten
- QUIC Verbindungen können zwischen IP Adressen migrieren, was SCTP nicht kann

  Für weitere Details zu den Unterschieden empfiehlt sich folgende Übersicht: A Comparison between SCTP and QUIC.

Ossifikation

Das Internet ist ein Netzwerk der Netzwerke. Um sicherzustellen, dass dieses Netzwerk der Netzwerke so funktioniert, wie es soll, gibt es Infrastruktur an vielen verschiedenen Plätzen. Diese Geräte - jene Boxen, die über das Netzwerk verteilt sind - bezeichnen wir manchmal als "Middle-Boxes". Boxen, die irgendwo zwischen den beiden Endpunkten sitzen und die primären Beteiligten in einem traditionellen Netzwerk-Datentransfer sind.

Diese Boxen dienen vielen verschiedenen spezifischen Zwecken - aber ich glaube sagen zu können, dass sie dort genutzt werden, weil jemand glaubt, dass diese dort sein müssen, damit alles funktioniert.

Middle-Boxes leiten IP-Pakete zwischen Netzwerken, sie blockieren böswilligen Traffic, führen die NAT (Network Address Translation, zu Deutsch "Netzwerkadressübersetzung") durch, verbessern die Leistung, manche versuchen den Traffic auszuspionieren und vieles mehr.

Um ihren Pflichten gerecht zu werden, müssen diese Boxen Networking und jene Protokolle kennen, die sie überwachen oder modifizieren. Dazu nutzen sie Software - die aber nicht oft aufgerüstet wird.

Auch wenn sie wie Klebstoff das Internet zusammenhalten, halten sie oft nicht mit der neuesten Technologie mit. Die Mitte eines Netzwerks verändert sich nicht so schnell wie die Enden, wie die Clients und Server dieser Welt.

Die Netzwerkprotokolle, die diese Boxen inspizieren wollen und kennen, haben dann folgendes Problem: die Boxen wurden vor einiger Zeit ausgeliefert, als die Protokolle gewissen Features hatten. Neue Features oder Veränderungen in der Verhaltensweise riskieren, dass Boxen diese als schlecht oder illegal interpretieren. Solcher Traffic könnte fallen gelassen oder verzögert werden - bis hin zu einem Ausmaß, dass User diese Features nicht nutzen wollen.

Das wird "Protokoll-Ossifikation" genannt.

Änderung an TCP leiden auch an Ossifikation: einige Boxen zwischen einem Client und einem entfernten Server erkennen neue TCP Optionen und blockieren solche Verbindungen, weil diese die Optionen nicht kennen. Wenn sie Protokolldetails erkennen dürfen, werden Systeme lernen, wie sich Protokolle normalerweise verhalten und im Laufe der Zeit wird es unmöglich sie zu ändern.

Der einzig effektive Weg um Ossifikation zu bekämpfen, ist die Verschlüsselung von so viel Kommunikation wie möglich. Damit werden Middle-Boxes daran gehindert, von durchgeleiteten Protokollen viel einsehen zu können.

Sicherheit

QUIC ist immer sicher. Es gibt keine Klartext-Version des Protokolls, daher involviert der Verbindungsaufbau einer QUIC Verbindung Kryptographie und Sicherheit mit TLS 1.3. Wie vorhergehend angemerkt, verhindert dies eine Ossifikation sowie andere Blockierungen und spezielle Handhabung - und sorgt dafür, dass QUIC alle sicheren Eigenschaften von HTTPS hat, die User des Webs erwarten und sich wünschen.

Es gibt nur wenige initiale Handshake-Pakete, die im Klaren geschickt werden, bevor die Verschlüsselungsprotokolle verhandelt wurden.

Daten im Voraus

QUIC bietet 0-RTT und 1-RTT Handshakes, welche die Zeit des Verhandelns sowie den Aufbau einer neuen Verbindung verkürzen. Vergleiche diesen Prozess mit dem 3-Schritt Handshake von TCP.

Zusätzlich bietet QUIC "Daten im Voraus" Support, welcher größeren Datenfluss erlaubt und im Vergleich zu TCP Fast Open benutzerfreundlicher ist.

Mit dem Stream-Konzept kann sofort eine weitere logische Verbindung zum gleichen Host aufgebaut werden, ohne auf das Ende einer bereits bestehenden Verbindung warten zu müssen.

TCP Fast Open ist problematisch

TCP Fast Open wurde als RFC 7413 im Dezember 2014 veröffentlicht. Diese Spezifikation beschreibt, wie Applikationen Daten zum Server bereits im ersten TCP SYN Paket übermitteln können.

Der tatsächliche Support für dieses Feature hat lange gedauert und ist bis ins heutige Jahr 2018 mit Problemen gespickt. Diejenigen, die den TCP-Stack implementiert haben, hatten Probleme und somit auch Applikationen, die die Vorteile dieses Features nutzen wollten. Dies gilt sowohl für das Verständnis der zu aktivierenden Betriebssystemversionen als auch für das Lösen von Problemen im Zusammenhang mit der mangelnden Clientseitigen Unterstützung (backdown). Viele Netzwerke konnten identifiziert werden, die TFO-Traffic stören und daher den reibungslosen Ablauf von TCP-Handshakes beeinträchtigen.

Prozess

Das ursprüngliche QUIC Protokoll wurde von Jim Roskind bei Google entworfen, im Jahr 2012 erstmals implementiert und der Welt 2013 vorgestellt, als Google's Experiment erweitert wurde.

Damals galt QUIC noch als Akronym für "Quick UDP Internet Connections", aber das wurde seitdem eingestellt.

Google implementierte das Protokoll und nutzte es anschließend sowohl in seinem weit verbreiteten Browser (Chrome) als auch in seinen weit verbreiteten serverseitigen Diensten (Google-Suche, Gmail, YouTube und mehr). Sie haben ziemlich schnell zwischen Protokollversionen iteriert und im Laufe der Zeit bewiesen, dass das Konzept für einen großen Teil der Nutzer zuverlässig funktioniert.

Im Juni 2015 wurde der erste Internetentwurf von QUIC zur Standardisierung an die IETF übermittelt. Es dauerte jedoch bis Ende 2016, bis eine QUIC-Arbeitsgruppe genehmigt und gestartet wurde. Aber dann ist es mit großem Interesse von vielen Parteien sofort losgegangen.

Im Jahr 2017 gaben die QUIC-Ingenieuren bei Google an, dass rund 7% des *gesamten* Internetverkehrs bereits das Protokoll verwenden. Die Google-Version des Protokolls.

IETF

Die zur Standardisierung des Protokolls innerhalb der IETF eingerichteten QUIC-Arbeitsgruppe entschied bald, dass das QUIC-Protokoll auch andere Protokolle als "nur" HTTP übertragen können soll. Google-QUIC hatte lediglich HTTP transportiert - in der Praxis wurden HTTP/2 Frames mithilfe des HTTP/2 Frame Syntax transportiert.

Es wurde auch angegeben, dass IETF-QUIC seine Verschlüsselung und Sicherheit auf TLS 1.3 anstelle des für Google-QUIC eigens entwickelten Ansatzes stützen sollte.

Um die Anforderung zu erfüllen, mehr als nur HTTP zu transportieren, wurde die IETF-QUIC-Protokollarchitektur in zwei separate Schichten aufgeteilt: die Transport-QUIC-Schicht und die "HTTP-over-QUIC"-Schicht (letztere wird manchmal als "hq" bezeichnet).

Diese Teilung in Schichten - auch wenn sie harmlos klingt - hat dazu geführt, dass sich IETF-QUIC erheblich vom ursprünglichen Google-QUIC unterscheidet.

Die Arbeitsgruppe hat jedoch bald entschieden, dass sie sich auf die Bereitstellung von HTTP konzentriert, um eine rechtzeitige Auslieferung der ersten QUIC Version gewährleisten zu können. Damit wurde die Implementierung des Nicht-HTTP-Transports auf später verschoben.

Als wir im März 2018 mit der Arbeit an diesem Buch begonnen haben, war geplant, die endgültige Spezifikation für die erste QUIC Version im November 2018 auszuliefern. Dies wurde später auf Juli 2019 verschoben.

Während die Arbeit an IETF-QUIC vorangeschritten ist, hat das Google-Team Details aus der IETF-Version übernommen und langsam damit begonnen, ihre Version des Protokolls dahingehend weiterzuentwickeln, dass diese wie die IETF-Version aussieht. Google hat die eigene QUIC-Version weiterhin in ihrem Browser und ihren Diensten im Einsatz.

[Die meisten neuen Implementierungen, die sich aktuell in Entwicklung befinden,] (https://github.com/quicwg/base-drafts/wiki/Implementations) haben beschlossen, sich auf die IETF-Version zu konzentrieren und sind nicht mit der Google-Version kompatibel.

Erfahrungen von HTTP/2

Die HTTP/2 Spezifikation RFC 7540 wurde im Mai 2015 veröffentlicht, nur einen Monat bevor QUIC zum ersten Mal bei der IETF eingebracht wurde.

Mit HTTP/2 wurde die Grundlage für zukünftige Änderungen von HTTP gelegt. Die Arbeitsgruppe, die HTTP/2 kreiert hatte, war der Ansicht, dass diese Grundlage Iterationen zu neuen HTTP Versionen beschleunigen wird - viel schneller als der Wechsel von Version 1 zu Version 2 (ca. 16 Jahre).

Mit HTTP/2 haben Nutzer und Software-Stacks erkannt, dass es nicht mehr zeitgemäß ist, HTTP mit einem textbasierten Protokoll seriell abzuwickeln.

HTTP-over-QUIC wurde im November 2018 in HTTP/3 umbenannt.

Status

Die QUIC-Arbeitsgruppe hat seit Ende 2016 intensiv an der Festlegung der Protokolle gearbeitet. Es ist geplant, die Spezifikation bis Juli 2019 zu finalisieren.

Bis November 2018 gab es noch keine größeren Interoperabilitätstests mit HTTP/3 - nur mit den beiden vorhandenen Implementierungen, nicht aber mit einem Browser oder einer beliebten Open-Server Lösungen.

In den Wiki-Seiten der QUIC-Arbeitsgruppe werden rund fünfzehn verschiedene QUIC Implementierungen aufgelistet - jedoch sind bei weitem nicht alle dieser Implementierungen mit den letzten Änderungen der Entwurfsspezifikation kompatibel.

QUIC zu implementieren ist nicht einfach. Das Protokoll hat sich bis zu diesem Zeitpunkt ständig weiterentwickelt und verändert.

Server

Es wurden keine öffentlichen Erklärungen hinsichtlich der Unterstützung von QUIC durch Apache oder Nginx abgegeben.

Clients

Keiner der größeren Browser-Anbieter hat bisher eine Version ausgeliefert, mit der die IETF-Version von QUIC oder HTTP/3 ausgeführt werden kann.

Google Chrome wird seit vielen Jahren mit einer funktionsfähigen Implementierung von Googles eigener QUIC-Version ausgeliefert, die jedoch nicht mit dem IETF-QUIC-Protokoll kompatibel ist und deren HTTP-Implementierung sich von HTTP/3 unterscheidet.

Implementierungshindernisse

QUIC hat sich dazu entschlossen, auf TLS 1.3 als Grundlage der Krypto- und Sicherheitsebene zu setzen, um sich auf ein zuverlässiges und bestehendes Protokoll stützen zu können. Jedoch hat sich die Arbeitsgruppe auch dazu entschieden, TLS in QUIC wirkungsvoller zu gestalten: es sollen lediglich "TLS messages" und nicht "TLS records" genutzt werden.

Dies mag nach einer harmlosen Änderung klingen, hat jedoch für viele QUIC-Stack-Implementierer eine erhebliche Hürde bedeutet. Bestehende TLS-Bibliotheken, die TLS 1.3 unterstützen, verfügen einfach nicht über ausreichende APIs, um diese Funktionalität verfügbar zu machen und QUIC den Zugriff darauf zu ermöglichen. Während mehrere QUIC-Implementierer von größeren Organisationen stammen, die parallel an ihrem eigenen TLS-Stack arbeiten, gilt dies nicht für alle.

Das dominierende Open-Source-Schwergewicht OpenSSL zum Beispiel, hat keine API dafür - und hat bisher auch keinen Wunsch geäußert, eine solche bald zur Verfügung zu stellen (Stand November 2018).

Dies wird letztendlich auch zu Hindernissen bei der Bereitstellung führen, da sich QUIC-Stacks entweder auf andere TLS-Bibliotheken stützen müssen, einen separat gepatchten OpenSSL-Build verwenden müssen oder ein Update auf eine zukünftige OpenSSL-Version benötigen.

Kernel und CPU-Last

Sowohl Google als auch Facebook haben angemerkt, dass die Bereitstellung von QUIC in großem Umfang ungefähr die doppelte Menge an CPU-Rechenleistung erfordert als wenn die gleiche Traffic-Last via HTTP/2 über TLS abgewickelt würde.

Einige Erklärungen hierfür sind:

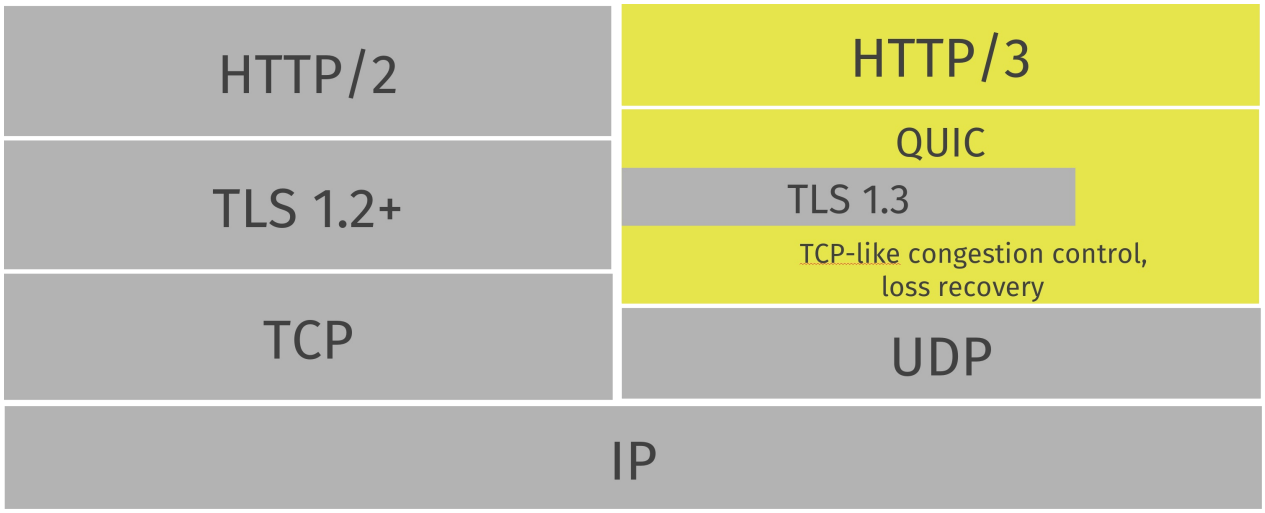
- Teile von UDP sind vor allem unter Linux nicht so optimiert wie der TCP-Stack, da UDP traditionell nicht für solche Hochgeschwindigkeitsübertragungen verwendet wurde.
- TCP- und TLS-Offloading auf Hardware ist vorhanden, bei UDP jedoch viel seltener und bei QUIC im Grunde nicht vorhanden.

Daher gibt es Gründe zur Annahme, dass sich die Leistung und die CPU-Anforderungen im Laufe der Zeit verbessern werden.

Eigenschaften des Protokolls

Das QUIC-Protokoll aus der Ferne betrachtet.

Folgend ist der HTTP/2-Netzwerkstack links und der QUIC-Netzwerkstack rechts dargestellt, wenn als HTTP-Transport verwendet.



Transportprotokoll über UDP

QUIC ist ein Transportprotokoll, das basierend auf UDP implementiert wurde. Wenn Sie Ihren Netzwerkverkehr gelegentlich beobachten, wird QUIC als UDP-Pakete angezeigt.

Basierend auf UDP werden dann auch UDP-Portnummern verwendet, um bestimmte Netzwerkdienste unter einer bestimmten IP-Adresse zu identifizieren.

Alle bekannten QUIC-Implementierungen befinden sich derzeit im User-Space, was eine schnellere Entwicklung ermöglicht, als dies bei Kernel-Space-Implementierungen normalerweise möglich ist.

Wird es funktionieren?

Es gibt Unternehmen und andere Netzwerkkonfigurationen, die den UDP-Verkehr an anderen Ports als 53 blockieren (für DNS verwendet). Andere drosseln solche Daten auf eine Weise, die die Leistung von QUIC schlechter macht als jene von TCP-basierten Protokollen. Es gibt kein Ende für das, was manche Betreiber tun könnten.

Auf absehbare Zeit muss jeder Einsatz von QUIC-basierten Transporten möglicherweise auf eine andere (TCP-basierte) Alternative zurückgreifen können. Google-Ingenieure haben zuvor gemessene Fehlerraten im niedrigen einstelligen Prozentbereich angegeben.

Wird es besser?

Wenn sich QUIC als eine wertvolle Bereicherung für das Internet herausstellt, möchten Leute es nutzen und das es in ihren Netzwerken funktioniert - dann können Unternehmen damit beginnen, ihre Hindernisse zu überdenken. Im Laufe der Jahre hat die Entwicklung von QUIC Fortschritte gemacht und die Erfolgsquote beim Aufbau sowie Nutzung von QUIC-Verbindungen über das Internet ist gestiegen.

Zuverlässige Datenübertragung

Während UDP kein zuverlässiger Transport ist, fügt QUIC über UDP eine Schicht hinzu, die Zuverlässigkeit einführt. Es bietet Neuübertragungen von Paketen, Überlastungskontrolle, Pacing und andere Funktionen, die sonst in TCP vorhanden sind.

Daten, die von einem Endpunkt über QUIC gesendet werden, erscheinen früher oder später am anderen Ende - solange die Verbindung besteht.

Mehrere Streams innerhalb von Verbindungen

Ähnlich wie SCTP, SSH und HTTP/2 verfügt QUIC über separate logische Streams innerhalb von physischen Verbindungen. Eine Anzahl paralleler Streams, die Daten gleichzeitig über eine einzelne Verbindung übertragen können, ohne die anderen Streams zu beeinträchtigen.

Eine Verbindung ist ein ausgehandelter Aufbau zwischen zwei Endpunkten, ähnlich wie eine TCP-Verbindung funktioniert. Eine QUIC-Verbindung wird zu einem UDP-Port und einer IP-Adresse hergestellt, aber sobald die Verbindung hergestellt ist, wird sie durch ihre "Verbindungs-ID" verknüpft.

Über eine bestehende Verbindung kann jede Seite Streams erstellen und Daten an das andere Ende senden. Streams werden in richtiger Reihenfolge (in-order) und zuverlässig geliefert - andere Streams können aber in nicht richtiger Reihenfolge (out-of-order) geliefert werden.

QUIC bietet eine Datenflusssteuerung sowohl für Verbindungen als auch für Streams.

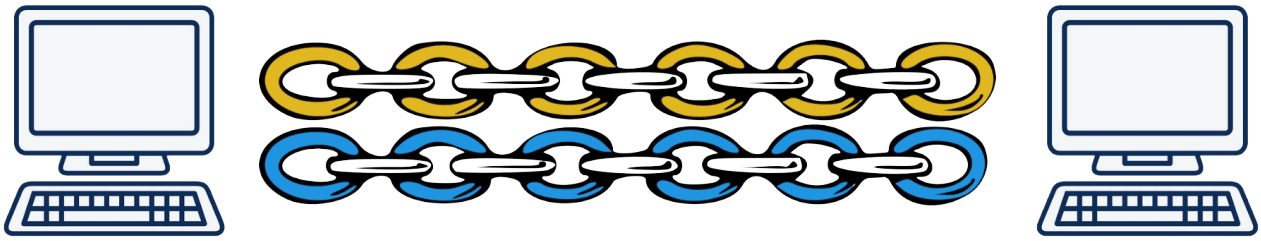
Weitere Details findest du in den Bereichen Verbindungen und Streams.

In-Order Ausliefung

QUIC garantiert die richtige Reihenfolge (in-order) der Zustellung von Streams, jedoch nicht zwischen Streams. Dies bedeutet, dass jeder Stream Daten sendet und die Datenreihenfolge beibehält, aber die Streams das Ziel möglicherweise in einer anderen Reihenfolge erreichen, als wie die Anwendung diese gesendet hat.

Beispiel: Stream A und B werden von einem Server zu einem Client übertragen. Zuerst wird Stream A und dann Stream B gestartet. In QUIC wirkt sich ein verlorenes Paket nur auf jenen Stream aus, zu dem das verlorene Paket gehört. Wenn Stream A ein Paket verliert, Stream B jedoch nicht, setzt Stream B die Übertragung möglicherweise fort und wird abgeschlossen, während das verlorene Paket von Stream A erneut übertragen wird. Dies war unter HTTP/2 nicht möglich.

Hier dargestellt mit einem gelben und einem blauen Stream, die über eine einzige Verbindung zwischen zwei QUIC-Endpunkten gesendet werden. Sie sind unabhängig und können in einer anderen Reihenfolge eintreffen, aber jeder Stream wird zuverlässig und in der richtigen Reihenfolge (in-order) an die Anwendung geliefert.



Schnelle Handshakes

QUIC bietet einen Verbindungsaufbau sowohl mit 0-RTT als auch 1-RTT. Das bedeutet, dass QUIC beim Aufbau einer neuen Verbindung bestenfalls keine zusätzlichen Roundtrips benötigt. Der Schnellere von beiden, der 0-RTT-Handshake, funktioniert nur, wenn zuvor eine Verbindung zu einem Host hergestellt und ein Geheimnis dieser Verbindung zwischengespeichert wurde.

Früher Daten versenden

Mit QUIC kann ein Client Daten bereits zu 0-RTT-Handshakes hinzufügen. Diese Funktion ermöglicht es einem Client, Daten so schnell wie möglich an den Peer zu übermitteln. Auf diese Weise kann der Server natürlich noch früher antworten und Daten zurücksenden.

TLS 1.3

QUIC verwendete TLS 1.3 (RFC 8446) zur Transportsicherheit und es gibt unter keinen Umständen unverschlüsselte QUIC-Verbindungen.

TLS 1.3 hat einige Vorteile gegenüber älteren TLS-Versionen - ein Hauptgrund für die Verwendung in QUIC ist, dass 1.3 den Handshake so geändert hat, dass weniger Roundtrips erforderlich sind. Dieses Vorgehen reduziert die Protokolllatenz.

In der Google-Version von QUIC wurde eine eigens entwickelte Lösung verwendet.

Transport- und Applikationsschicht

Das IETF-QUIC-Protokoll ist ein Transportprotokoll, über welches andere Anwendungsprotokolle genutzt werden können. Das erste Protokoll der Anwendungsschicht ist HTTP/3 (h3).

Die Transportschicht unterstützt Verbindungen und Streams.

In der Google-Version von QUIC waren Transport und HTTP zu einem Allheilmittel gebündelt worden und war eher ein spezielles Sende-http/2-Frames-über-UDP-Protokoll.

HTTP/3 über QUIC

Die HTTP-Schicht namens HTTP/3 führt Transporte im HTTP-Stil durch, einschließlich HTTP-Header-Komprimierung mit QPACK - ähnlich der HTTP/2-Komprimierung HPACK.

Der HPACK-Algorithmus hängt von einer *geordneten* Übermittlung von Streams ab, weshalb es nicht möglich war, ihn ohne Änderungen in HTTP/3 wiederzuverwenden; vor allem, weil QUIC Streams anbietet, die nicht in der richtigen Reihenfolge übermittelt werden können. QPACK kann als die QUIC-angepasste Version von HPACK gesehen werden.

Nicht-HTTP über QUIC

Die Arbeit am Senden anderer Protokolle als HTTP über QUIC wurde solange verschoben, bis die erste Version von QUIC ausgeliefert wurde.