HTTP/3 Explained

by Daniel Stenberg

Table des matières

HTTP/3 expliqué

Ce livre a été lancé en mars 2018. Il est prévu de documenter HTTP/3 et son protocole sous-jacent: QUIC. Pourquoi, comment fonctionnent-ils, les détails du protocole, les implémentations et plus.

Le livre est entièrement gratuit et se veut être un effort de collaboration impliquant tous ceux qui veulent aider.

Prérequis

Un lecteur de ce livre est censé avoir une compréhension de base du réseau TCP/IP, des principes fondamentaux de HTTP et du Web. Pour plus d'informations et de détails sur HTTP/2, nous vous recommandons tout d'abord de lire les détails dans http2 expliqué.

Auteur

Ce livre est créé et le travail est démarré par Daniel Stenberg. Daniel est le fondateur et le développeur principal de curl, le client HTTP le plus utilisé au monde. Daniel travaille avec et sur les protocoles HTTP et Internet depuis plus de deux décennies et est l'auteur de http2 expliqué.

Accueil

La page d'accueil de ce livre se trouve à l'adresse daniel.haxx.se/http3-explained.

Contribuer

Si vous trouvez des fautes, des omissions, des erreurs ou des mensonges flagrants dans ce document, veuillez nous envoyer une version mise à jour du paragraphe concerné et nous en ferons des versions corrigés. Nous allons donner des crédits appropriés à tous ceux qui aident. J'espère améliorer ce document avec le temps.

De préférence, vous soumettez des erreurs ou des demandes de tirage sur la page github du livre.

Licence

Pourquoi QUIC

QUIC est un nom, pas un acronyme. Il se prononce exactement comme le mot anglais "quick".

QUIC est à bien des égards ce qui pourrait être perçu comme un moyen de créer un nouveau protocole de transport fiable et sécurisé, adapté à un protocole comme HTTP et pouvant résoudre certains des inconvénients connus de HTTP/2 sur TCP et TLS. La prochaine étape logique dans l'évolution du transport Web.

QUIC n'est pas limité au seul transport de HTTP. La volonté de rendre le web et les données en général plus rapides aux utilisateurs finaux est probablement la raison principale et la poussée qui a initialement déclenché la création de ce nouveau protocole de transport.

Alors pourquoi créer un nouveau protocole de transport et pourquoi le faire par dessus UDP?

Souvenez-vous de HTTP/2 ?

La spécification HTTP/2 RFC 7540 a été publiée en mai 2015 et le protocole a depuis été mis en œuvre et déployé largement sur Internet et sur le World Wide Web.

Début 2018, près de 40% des 1 000 meilleurs sites Web utilisaient HTTP/2, environ 70% de toutes les demandes HTTPS de Firefox recevaient des réponses HTTP/2 et tous les principaux navigateurs, serveurs et proxies le prenaient en charge.

HTTP/2 corrige toute une série de lacunes presentes dans HTTP/1 et avec l'introduction de la deuxième version de HTTP, les utilisateurs peuvent cesser d'utiliser toute une série de solutions de contournement. Certaines sont assez pénibles pour les développeurs Web.

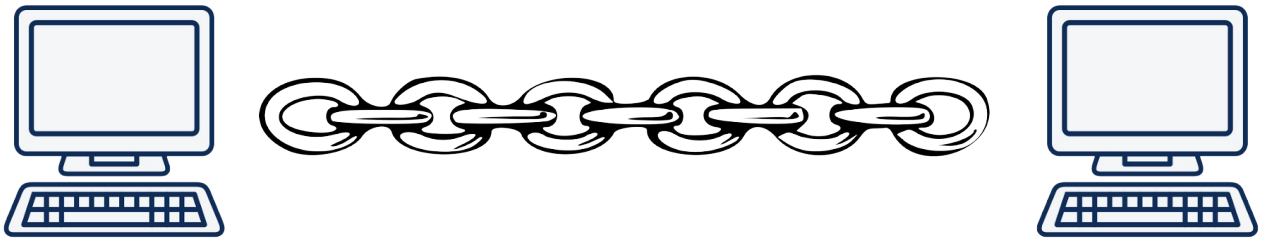
L'une des principales caractéristiques de HTTP/2 est qu'il utilise le multiplexage, de sorte que de nombreux flux logiques soient envoyés sur la même connexion TCP physique. Cela rend beaucoup de choses meilleures et plus rapides. Le contrôle de congestion fonctionne bien mieux, il permet aux utilisateurs d'utiliser bien mieux le protocole TCP et ainsi de saturer correctement la bande passante, de rendre les connexions TCP plus durables - ce qui est bien pour qu'ils atteignent la vitesse maximale plus souvent qu'avant. La compression d'en-tête lui fait utiliser moins de bande passante.

Avec HTTP/2, les navigateurs utilisent généralement *une* connexion TCP avec chaque hôte au lieu des précédents *six*. En fait, les techniques de fusion et de "désarchivage" des connexions utilisées avec HTTP/2 peuvent même réduire beaucoup plus que ça le nombre de connexions.

HTTP/2 a corrigé le problème de blocage de tête de ligne HTTP, dans lequel les clients devaient attendre la fin de la première requête en ligne avant que la suivante ne puisse être envoyée.

Blocage de tête de ligne TCP

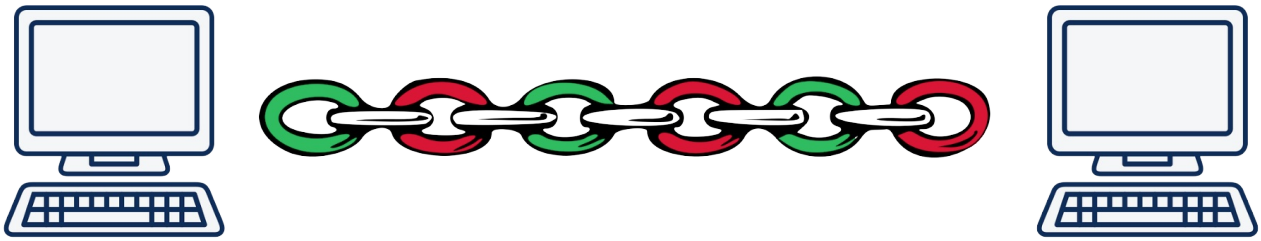
HTTP/2 est réalisé sur TCP et avec beaucoup moins de connexions TCP que lors de l'utilisation de versions HTTP antérieures. TCP est un protocole pour des transferts fiables et vous pouvez le considérer comme une chaîne imaginaire entre deux machines. Ce qui est mis sur le réseau d'un côté finira par se retrouver à l'autre bout, dans le même ordre - à terme. (Ou la connexion est rompue.)



Avec HTTP/2, les navigateurs classiques effectuent des dizaines, voire des centaines de transferts parallèles sur cette seule connexion TCP.

Si un seul paquet est abandonné ou perdu sur le réseau quelque part entre deux terminaisons qui parlent HTTP/2, cela signifie que toute la connexion TCP est interrompue et que le paquet perdu doit être retransmis et doit retrouver son chemin jusqu'à la destination. Puisque TCP est cette "chaîne", cela signifie que si un lien manque soudainement, tout ce qui viendrait après le lien perdu doit attendre.

Illustration utilisant la métaphore de la chaîne lors de l'envoi de deux flux sur cette connexion. Un flux rouge et un flux vert:



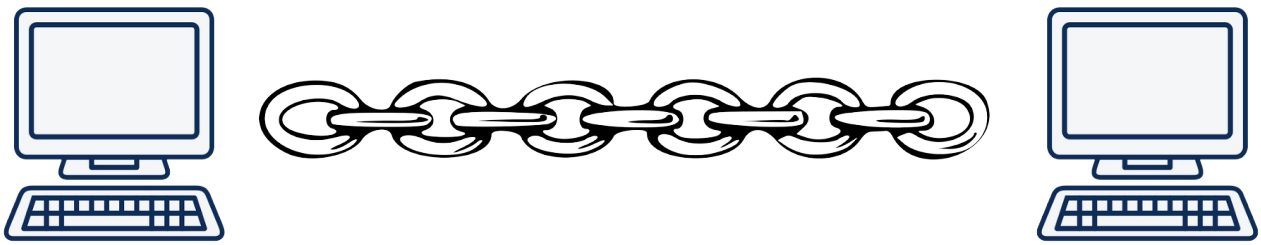
Cela devient un bloc de début de ligne basé sur TCP!

À mesure que le taux de perte de paquets augmente, HTTP/2 est de moins en moins performant. Avec 2% de perte de paquets (ce qui est une qualité de réseau épouvantable, remarquez-vous bien), des tests ont montré que les utilisateurs de HTTP/1 sont généralement mieux lotis - car ils disposent généralement de six connexions TCP pour répartir le paquet perdu donc pour chaque paquet perdu, les autres connexions sans perte peuvent toujours continuer.

Résoudre ce problème n’est pas facile, et si tout de même possible, à faire avec TCP.

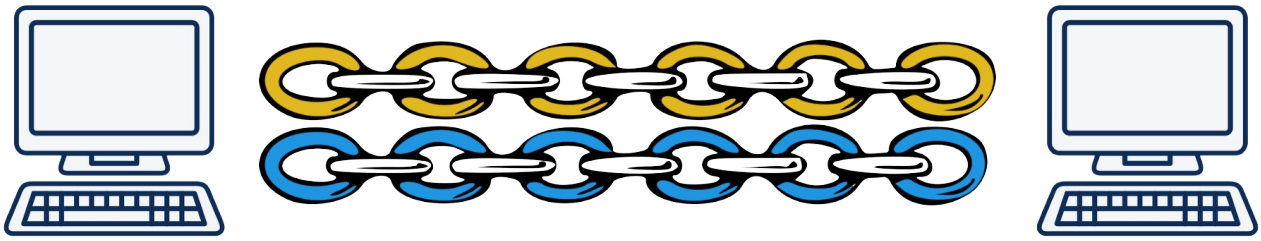
Les flux indépendants évitent le blocage

Avec QUIC, il existe toujours une connexion configurée entre les deux terminaisons qui sécurise la connexion et la livraison des données.



Lors de la configuration de deux flux différents sur cette connexion, ils sont traités indépendamment. Ainsi, si un lien manque à l'un des flux, seul ce flux, cette chaîne particulière, doit s'interrompre et attend que le lien manquant soit retransmis.

Illustré ici avec un flux jaune et un flux bleu envoyés entre deux terminaisons.



TCP ou UDP

Si nous ne pouvons pas résoudre le blocage de la tête de ligne dans TCP, nous devrions théoriquement pouvoir créer un nouveau protocole de transport à côté de UDP et TCP dans la stack réseau. Ou peut-être même utilisez-vous SCTP qui est un protocole de transport normalisé par l'IETF dans la RFC 4960 avec plusieurs des caractéristiques souhaitées.

Cependant, au cours des dernières années, les efforts pour créer de nouveaux protocoles de transport ont été presque complètement arrêtés en raison de la difficulté de les déployer sur Internet. Le déploiement de nouveaux protocoles est entravé par de nombreux pare-feu, NATs, routeurs et autres boîtes centrales qui autorisent uniquement le protocole TCP ou UDP déployés entre les utilisateurs et les serveurs qu'ils doivent atteindre. L'introduction d'un autre protocole de transport provoque l'échec de N% des connexions car elles sont bloquées par des boîtes ne la voyant pas comme étant UDP ou TCP et donc malfaisantes ou erronés d'une manière ou d'une autre. Le taux d'échec de N% est souvent jugé trop élevé pour en valoir la peine.

De plus, les modifications apportées à la couche de protocole de transport de la stack réseau impliquent généralement des protocoles implémentés par les noyaux de système d'exploitation. La mise à jour et le déploiement de nouveaux noyaux de système d'exploitation est un processus lent qui nécessite des efforts importants. De nombreuses améliorations TCP normalisées par l'IETF ne sont pas beaucoup déployées ou utilisées car elles ne sont pas énormément prises en charge.

Pourquoi pas SCTP sur UDP

SCTP est un protocole de transport fiable avec des flux, et pour WebRTC, il existe même des implémentations existantes qui l'utilisent via UDP.

Cela n'a pas été jugé suffisant comme alternative à QUIC pour plusieurs raisons, notamment:

- SCTP ne résout pas le problème de blocage de tête de ligne pour les flux
- SCTP exige que le nombre de flux soit déterminé lors de l'établissement de la connexion
- SCTP n'a pas un solide concept TLS/de sécurité
- SCTP a un handshake à 4 voies, QUIC offre 0-RTT
- QUIC est un flux par octet comme TCP, SCTP est basé sur des messages
- Les connexions QUIC peuvent migrer entre les adresses IP, mais SCTP ne peut pas

Pour plus de détails sur les différences, voir A Comparison between SCTP and QUIC.

Ossification

Internet est un réseau de réseaux. Il y a des équipements installés sur Internet dans de nombreux endroits pour assurer le fonctionnement de ce réseau de réseaux. Ces périphériques, les boîtiers distribués sur le réseau, sont ce que nous appelons parfois des boîtiers centraux. Les zones situées quelque part entre les points terminaisons sont l'une des deux parties principales impliquées dans un transfert de données réseau traditionnel.

Ces boîtes servent à de nombreuses fins spécifiques, mais je pense que nous pouvons dire qu'universellement, elles sont placées là parce que quelqu'un pense qu'elles doivent être là pour que les choses fonctionnent.

Les boîtiers centraux routent les paquets IP entre les réseaux, bloquent le trafic malveillant, effectuent la traduction d'adresses réseau (en anglais Network Address Translation (NAT)), améliorent les performances, tentent parfois d'espionner le trafic en passant, etc...

Afin de s'acquitter de leurs tâches, ces boîtiers doivent connaître la mise en réseau et les protocoles qu'ils surveillent ou modifient. Ils exécutent des logiciels à cette fin. Logiciels qui ne sont pas toujours mis à jour fréquemment.

Bien qu'ils soient des composants essentiels qui maintiennent Internet attaché, ils ne sont pas souvent en phase avec les dernières technologies. Le milieu du réseau ne se déplace généralement pas aussi vite que les bords, comme les clients et les serveurs du monde.

Tous les protocoles de réseau que ces boîtes pourraient vouloir inspecter et qui ont des idées sur ce qui est ok et ce qui ne l'est pas alors ont ce problème: ces boîtes ont été déployées il y a quelque temps, alors que les protocoles avaient un ensemble de fonctionnalités de cette époque. L'introduction de nouvelles fonctionnalités ou de changements de comportement inconnus auparavant risquerait d'être considérée comme mauvaise ou illégale par de telles boîtes. Ce trafic peut tout simplement être supprimé ou retardé dans la mesure où les utilisateurs ne souhaitent vraiment pas utiliser ces fonctionnalités.

C'est ce qu'on appelle "l'ossification du protocole".

Les modifications apportées au protocole TCP souffrent également d’ossification: certaines boîtes entre un client et le serveur distant détectent de nouvelles options TCP inconnues et bloquent ces connexions car ils ne savent pas quelles sont les options. S'ils sont autorisés à détecter les détails de protocole, les systèmes apprennent le comportement typique des protocoles et, avec le temps, il devient impossible de les modifier.

Le seul moyen véritablement efficace de "combattre" l'ossification consiste à chiffrer le plus possible la communication afin d'empêcher les boîtes moyennes de voir beaucoup du protocole la traversant.

Sécurisé

QUIC est toujours sécurisé. Il n'y a pas de version en texte clair clair du protocole, donc négocier une connexion QUIC signifie faire de la cryptographie et de la sécurité avec TLS 1.3. Comme mentionné ci-dessus, cela empêche l'ossification ainsi que d'autres types de blocages et traitements spéciaux, et garantit que QUIC possède toutes les propriétés sécurisées de HTTPS auxquelles les utilisateurs Web s'attendent et souhaitent.

Il n’y a que quelques paquets handshake initiaux qui sont envoyés en clair, avant que les protocoles de cryptage aient été négociés.

Données antérieures

QUIC offre à la fois des handshakes 0-RTT et 1-RTT qui réduisent le temps nécessaire pour négocier et établir une nouvelle connexion. Comparez avec le handshake à 3 voies du TCP.

En plus de ça, QUIC offre une prise en charge des "données antérieures" dès le départ, ce qui est fait pour autoriser plus de données et est utilisé plus facilement que TCP Fast Open.

Avec le concept de flux, vous pouvez établir une autre connexion logique avec le même hôte sans avoir à d'abord attendre la fin de la connexion existante.

TCP Fast Open est problématique

TCP Fast Open a été publié en décembre 2014 en tant que la RFC 7413 et cette spécification explique comment les applications peuvent transmettre des données au serveur afin qu'elles soient déjà livrées dans le premier paquet TCP SYN.

La prise en charge effective de cette fonctionnalité a pris du temps et pose encore de nombreux problèmes, même aujourd'hui en 2018. Les responsables de la mise en œuvre de la stack TCP ont rencontré des problèmes, tout comme les applications qui ont essayé de tirer parti de cette fonctionnalité, en sachant dans quelle version d'OS essayer pour l'activer mais également pour savoir comment revenir en arrière avec élégance et régler les problèmes qui surviennent. Plusieurs réseaux ont été identifiés pour interférer avec le trafic TFO et ont donc activement ruiné de telles handshakes TCP.

La procédure

Le protocole QUIC originel a été conçu par Jim Roskind chez Google et a été initialement mis en œuvre en 2012, puis annoncé publiquement au monde en 2013 lorsque l'expérimentation de Google s'est élargie.

À l'époque, il était toujours prétendu que QUIC était l'acronyme de "Quick UDP Internet Connections", mais il a été abandonné depuis.

Google a mis en œuvre le protocole et l'a ensuite déployé à la fois dans son navigateur beaucoup utilisé (Chrome) et dans ses services côté serveur beaucoup utilisés (Google search, gmail, youtube, etc...). Ils ont répété les versions du protocole assez rapidement et, avec le temps, ils ont prouvé que le concept fonctionnait de manière fiable pour une grande partie des utilisateurs.

En juin 2015, le premier brouillon Internet pour QUIC a été envoyé à l'IETF pour une standardisation, mais il a fallu attendre la fin de l'année 2016 pour qu'un groupe de travail QUIC soit approuvé et lancé. Mais ensuite, il a immédiatement décollé avec beaucoup d'intérêt de la part de nombreux partis.

En 2017, des chiffres cités par les ingénieurs de QUIC chez Google ont indiqué qu'environ 7% de *tout* le trafic Internet utilisait déjà ce protocole. La version de Google du protocole.

IETF

Le groupe de travail QUIC mis en place pour standardiser le protocole au sein de l'IETF a rapidement décidé que le protocole QUIC devait pouvoir transférer des protocoles autres que "simplement" HTTP. Google-QUIC ne transportait jamais que HTTP - en pratique, il transportait ce qui était en réalité des trames HTTP/2, en utilisant la syntaxe HTTP/2.

Il a également été indiqué que l'IETF-QUIC devrait baser son cryptage et sa sécurité sur TLS 1.3 au lieu de l'approche "personnalisée" utilisée par Google-QUIC.

Afin de satisfaire la demande d'envoi-plus-que-HTTP, l'architecture de protocole IETF QUIC a été scindée en deux couches distinctes: la couche de transport QUIC et la couche "HTTP sur QUIC" (cette dernière parfois appelée "hq").

Cette division de couche, bien que cela puisse paraître inoffensif, a entraîné une grande différence entre l'IETF-QUIC et l'original de Google-QUIC.

Cependant, le groupe de travail a rapidement décidé que pour se concentrer correctement et délivrer la version 1 de QUIC à temps, il se concentrerait sur la livraison de HTTP, laissant les transports non-HTTP à un travail ultérieur.

En mars 2018, lorsque nous avons commencé à travailler sur ce livre, le plan était d'expédier la spécification finale de la version 1 de QUIC en novembre 2018; cela a ensuite été reporté à juillet 2019.

Alors que les travaux sur l'IETF-QUIC ont progressé, l'équipe de Google a incorporé les détails de la version de l'IETF et a commencé à faire progresser lentement sa version du protocole vers ce que pourrait devenir la version de l'IETF. Google a continué d'utiliser sa version de QUIC dans son navigateur et ses services.

La plupart des nouvelles implémentations en cours de développement ont décidé de se concentrer sur la version de l'IETF et ne sont pas compatibles avec la version de Google.

Expérience depuis HTTP/2

La spécification HTTP/2 RFC 7540 a été publiée en mai 2015, juste un mois avant que QUIC ne soit présenté pour la première fois à l'IETF.

Avec HTTP/2, les bases de la modification de HTTP sur le réseau ont été aménagées et le groupe de travail qui a créé HTTP/2 était déjà convaincu que cela aiderait à effectuer une itération sur de nouvelles versions HTTP plus rapidement qu’il a fallu pour passer de la version 1 à la version 2 (environ 16 ans).

Avec HTTP/2, les utilisateurs et les stacks logiciels se sont habitués à l'idée que le protocole HTTP ne peut plus être supposé être exécuté en série avec un protocole texte.

HTTP-over-QUIC a été renommé HTTP/3 en novembre 2018.

Statut

Le groupe de travail de QUIC a travaillé d'arrache-pied depuis fin 2016 pour spécifier les protocoles et le plan est maintenant de le faire d'ici juillet 2019.

En novembre 2018, il n'y avait toujours pas de tests d'interopérabilité plus grands avec HTTP/3 - uniquement avec les deux implémentations existantes et aucun d'entre eux n'est effectué par un navigateur ou un logiciel populaire de serveur ouvert.

Il y a une quinzaine de différentes implémentations de QUIC répertoriées dans les pages de wiki des groupes de travail de QUIC, mais toutes ne peuvent pas interagir sur les dernières revisions des spécifications.

L'implémentation de QUIC n'est pas facile et le protocole a continué à évoluer, même à cette date.

Les serveurs

Le support NGINX pour QUIC et HTTP/3 est en cours de développement. Il est prévu qu'il soit déployé durant le cycle de développement NGINX 1.17.

Il n'y a eu aucune déclaration publique en termes de support QUIC d'Apache.

Les clients

Aucun des plus gros éditeurs de navigateurs n’a encore fourni de version, quel que soit son état, pouvant exécuter la version IETF de QUIC ou de HTTP/3.

Google Chrome est livré avec une implémentation fonctionnelle de la version QUIC de Google depuis de nombreuses années, mais cela n'interagit pas avec le protocole QUIC officiel et son implémentation HTTP est différente de HTTP/3.

Mozilla est en train de développer Neqo - une implémentation de QUIC et HTTP/3 écrit en Rust. Neqo est prévu d'être intégré dans Necko (qui est une biliothèque réseau utilisé dans plein d'applications clientes basés sur Mozilla - Firefox inclus).

Obstacles d'Implémentation

QUIC a décidé d'utiliser TLS 1.3 comme base pour la couche de chiffrement et de sécurité pour éviter d'inventer quelque chose de nouveau et plutôt s'appuyer sur un protocole fiable et existant. Cependant, ce faisant, le groupe de travail a également décidé que, pour rationaliser réellement l'utilisation de TLS dans QUIC, il ne devrait utiliser que des "messages TLS" et non des "enregistrements TLS" pour le protocole.

Cela peut sembler être un changement anodin, mais cela a en fait créé un obstacle considérable pour de nombreux développeurs de stack QUIC. Les bibliothèques TLS existantes qui prennent en charge TLS 1.3 n'ont tout simplement pas assez d'API pour exposer cette fonctionnalité et permettre à QUIC d'y accéder. Bien que plusieurs développeurs QUIC proviennent de grandes organisations qui travaillent sur leur propre stack TLS en parallèle, cela n’est pas vrai pour tout le monde.

OpenSSL, le poids lourd open source dominant, par exemple, n’a pas d’API et n’a manifesté aucun désir de la fournir dans les meilleurs délais (à partir de novembre 2018).

Cela entraînera éventuellement des obstacles de déploiement puisque les stacks QUIC devront se baser sur d'autres bibliothèques TLS, utiliser une version OpenSSL corrigée ou nécessiter une mise à jour d'une version future d'OpenSSL.

Noyaux et charge du processeur

Google et Facebook ont tous deux mentionné que leurs déploiements à grande échelle de QUIC requièrent environ deux fois plus de ressources processeur que la même charge de trafic lorsque HTTP/2 est utilisé via TLS.

Quelques explications à cela incluent

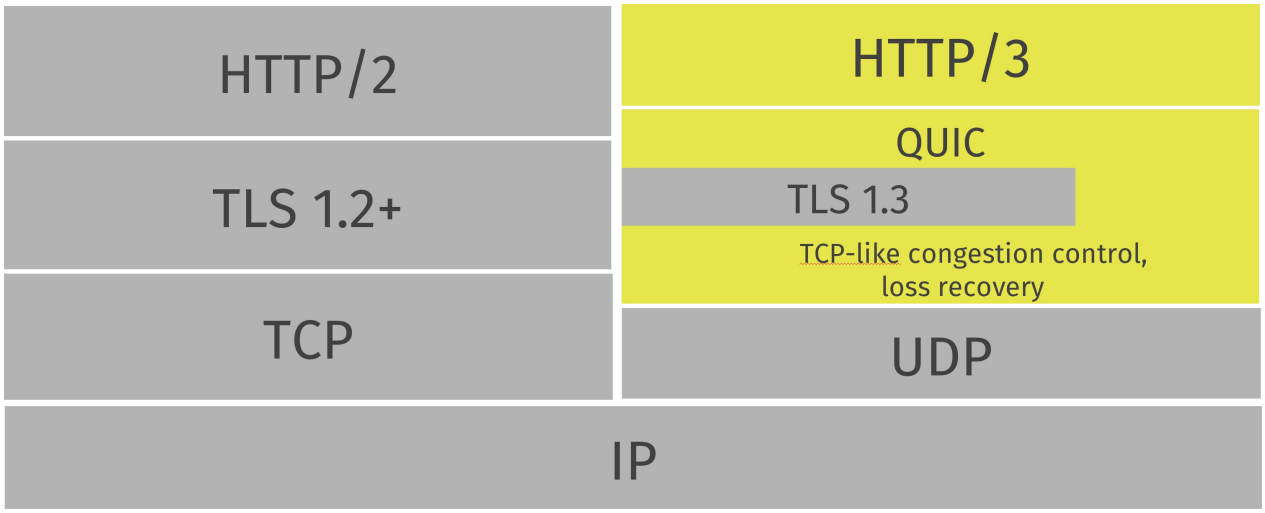
- Les composants UDP, principalement sous Linux, ne sont pas du tout aussi optimisés que la stack TCP, car elle n'a pas été utilisée traditionnellement pour des transferts à grande vitesse comme celui-ci.
- Le déchargement TCP et TLS sur le matériel existe, mais cela est beaucoup plus rare pour UDP et pratiquement inexistant pour QUIC.

Il y a des raisons de croire que les performances et les exigences en matière de processeur s'amélioreront avec le temps.

Fonctionnalités du protocole

Le protocole QUIC d'un niveau élevé.

Illustré ci-dessous, la stack réseau HTTP/2 à gauche et la stack réseau QUIC à droite, quand utilisées comme transport HTTP.



Protocole de transfert sur UDP

QUIC est un protocole de transfert implémenté au-dessus d'UDP. Si vous surveillez votre trafic réseau par hasard, vous verrez QUIC apparaître sous forme de paquets UDP.

Basé sur UDP, il utilise également les numéros de port UDP pour identifier des serveurs spécifiques sur une machine donnée.

Toutes les implémentations QUIC connues se trouvent actuellement dans l'espace utilisateur, ce qui permet une évolution plus rapide que ne permettent généralement pas les implémentations noyau.

Est-ce que ça va fonctionner ?

Certaines entreprises et autres configurations réseau bloquent le trafic UDP sur des ports autres que 53 (utilisé pour DNS). D'autres limitent ces données de manière à rendre QUIC moins performant que les protocoles basés sur TCP. Il n'y a pas de fin à ce que certains opérateurs peuvent faire.

Dans un avenir prévisible, toute utilisation de transports basés sur QUIC devra probablement être en mesure de faire appel à une autre alternative (basée sur TCP). Les ingénieurs de Google ont précédemment mentionné les taux d'échec mesurés dans de faibles pourcentages à un chiffre.

Cela va-t-il s'améliorer ?

Il est fort probable que si QUIC s'avère être un atout précieux au monde d'Internet, les utilisateurs voudront l'utiliser et le feront fonctionner dans leurs réseaux, ce qui permettra aux entreprises de reconsidérer leurs obstacles. Au fil des années, le développement de QUIC a progressé, le taux de réussite de l’établissement et de l’utilisation de connexions QUIC sur Internet a augmenté.

Transferts de données fiables

Bien qu'UDP ne soit pas un transport fiable, QUIC ajoute une couche au-dessus d'UDP qui introduit la fiabilité. Il offre la retransmission de paquets, le contrôle de congestion, la stimulation et les autres fonctionnalités présentes par ailleurs dans TCP.

Les données envoyées sur QUIC depuis un point de terminaison apparaîtront dans l'autre tôt ou tard, tant que la connexion est maintenue.

Plusieurs flux au sein de connexions

Semblable à SCTP, SSH et HTTP/2, QUIC propose des flux logiques séparés au sein des connexions physiques. Un certain nombre de flux parallèles pouvant transférer des données simultanément sur une seule connexion sans affecter les autres flux.

Une connexion est une configuration négociée entre deux points de terminaison, similaire au fonctionnement d'une connexion TCP. Une connexion QUIC est établie sur port UDP et une adresse IP, mais une fois établie, la connexion est associée à son "ID de connexion".

Sur une connexion établie, chaque côté peut créer des flux et envoyer des données à l'autre terminaison. Les flux sont livrés dans l'ordre et ils sont fiables, mais différents flux peuvent être livrés dans le désordre.

QUIC offre un contrôle de flux sur la connexion et les flux.

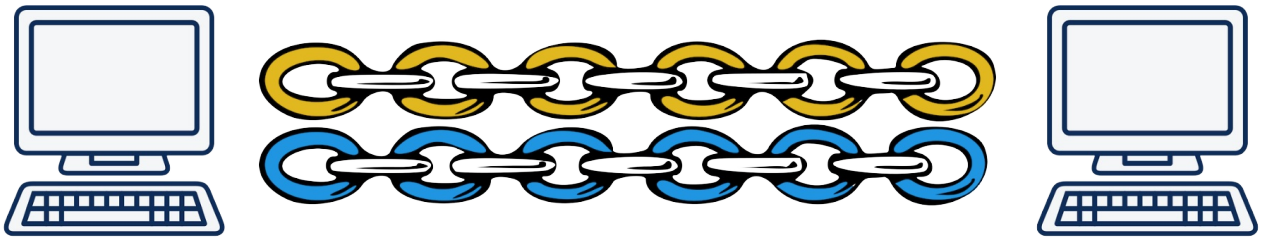
Voir plus de détails dans les sections connexion et flux

Livraison dans l'ordre

QUIC garantit la livraison des flux dans l'ordre, mais pas entre les flux. Cela signifie que chaque flux enverra des données et maintiendra l’ordre des données, mais chaque flux pourra atteindre la destination dans un ordre différent de celui que l’application a envoyé!

Par exemple: les flux A et B sont transférés d'un serveur à un client. Le flux A est démarré en premier, puis ensuite le flux B. Dans QUIC, un paquet perdu affecte uniquement le flux auquel appartient le paquet perdu. Si le flux A perd un paquet mais pas le flux B, le flux B peut poursuivre ses transferts et se terminer pendant que le paquet perdu du flux A est retransmis. Ce n'était pas possible avec HTTP/2.

Illustré ici avec un flux jaune et un flux bleu envoyés entre deux points de terminaison QUIC sur une seule connexion. Ils sont indépendants et peuvent arriver dans un ordre différent, mais chaque flux est livré de manière fiable, dans l'ordre, à l'application.



Handshakes rapides

QUIC offre à la fois des configurations de connexion 0-RTT et 1-RTT, ce qui signifie qu'au mieux, QUIC ne nécessitera aucun aller-retour supplémentaire lors de la configuration d'une nouvelle connexion. Le plus rapide des deux, la négociation 0-RTT, ne fonctionne que si une connexion précédente a été établie avec un hôte et qu'un secret de cette connexion a été mis en cache.

Données précoces

QUIC permet à un client d'inclure des données déjà dans le handshake 0-RTT. Cette fonctionnalité permet à un client de transmettre les données à l'homologue aussi rapidement que possible, ce qui permet bien entendu au serveur de répondre et de renvoyer les données encore plus tôt.

TLS 1.3

La sécurité de transport utilisée dans QUIC utilise TLS 1.3 (RFC 8446) et il n'y a jamais de connexions QUIC non chiffrées.

TLS 1.3 présente plusieurs avantages par rapport aux anciennes versions de TLS, mais l'une des principales raisons de son utilisation dans QUIC est que la 1.3 a modifié le handshake pour exiger moins d'allers-retours. Cela réduit la latence du protocole.

L'ancienne version de QUIC de Google utilisait une crypto personnalisée.

Niveau de transport et d'application

Le protocole IETF QUIC est un protocole de transport sur lequel d'autres protocoles d'application peuvent être utilisés. Le protocole de couche d'application initial est HTTP/3 (h3).

La couche de transport prend en charge les connexions et les flux.

L'ancienne version de QUIC de Google regroupait le transport et le protocole HTTP dans un même fait-tout et constituait un protocole send-http/2-frames-over-udp plus spécifique.

HTTP/3 over QUIc

La couche HTTP effectue des transports de style HTTP, y compris la compression d'en-tête HTTP à l'aide de QPACK - ce qui est similaire à la compression HTTP/2 appelée HPACK.

L'algorithme HPACK dépend d'une livraison *ordonnée* de flux, il n'a donc pas été possible de le réutiliser pour HTTP/3 sans modifications depuis que QUIC propose des flux pouvant être livrés dans le désordre. QPACK peut être considéré comme la version de HPACK adaptée à QUIC.

Non-HTTP sur QUIC

Les travaux d’envoi de protocoles autres que HTTP sur QUIC ont été reportés après la livraison de la version 1 de QUIC.

Comment QUIC fonctionne

Sans expliquer les bits et les octets exacts sur le réseau, cette section décrit le fonctionnement des blocs de construction fondamentaux du protocole de transport QUIC. Si vous souhaitez implémenter votre propre stack QUIC, cette description doit vous donner une compréhension générale, mais pour tous les détails, référez-vous aux actuel brouillons internets et RFCs de l'IETF.

1. Configurez une connexion
2. ... ce qui inclut la sécurité TLS
3. Puis utilisez les flux

Connexions

Une connexion QUIC est une conversation unique entre deux terminaisons QUIC. L’établissement de la connexion QUIC associe la négociation de la version à la négociation cryptographique et du handshake de transfert afin de réduire le temps d’attente de l’établissement de la connexion.

Pour envoyer des données via une telle connexion, un ou plusieurs flux doivent être créés et utilisés.

ID de connexion

Chaque connexion possède un ensemble d'identifiants de connexion, ou d'IDs de connection, chacun d'entre eux pouvant être utilisé pour identifier la connexion. Les IDs de connexion sont sélectionnés indépendamment par les terminaisons; chaque terminaison sélectionne les identifiants de connexion que son pair utilise.

La fonction principale de ces IDs de connexion est de garantir que les changements d’adressage au niveau des couches inférieures du protocole (UDP, IP et au-dessous) n’entraînent pas la transmission des paquets d’une connexion QUIC à la mauvaise terminaison.

En tirant parti de l'ID de connexion, les connexions peuvent ainsi migrer entre les adresses IP et les interfaces réseau d'une manière que TCP n'aurais jamais pu. Par exemple, la migration permet à un téléchargement en cours de passer d'une connexion réseau cellulaire à une connexion wifi plus rapide lorsque l'utilisateur déplace son appareil dans un emplacement proposant le wifi. De même, le téléchargement peut s'effectuer par la connexion cellulaire si le wifi devient indisponible.

Numéro de port

QUIC est construit sur UDP, donc un champ de numéro de port de 16 bits est utilisé pour différencier les connexions entrantes.

Négociation de version

Une demande de connexion QUIC émanant d'un client indiquera au serveur la version du protocole QUIC qu'il souhaite utiliser, et le serveur répondra avec une liste des versions prises en charge que le client pourra sélectionner.

Connections use TLS

Immédiatement après le paquet initial établissant une connexion, l'initiateur envoie une trame de cryptage qui commence à établir le handshake de couche sécurisée. La couche de sécurité utilise la sécurité TLS 1.3.

Il n'y a aucun moyen de vous désabonner ou d'éviter d'utiliser TLS pour une connexion QUIC. Le protocole est conçu pour être difficile à altérer par des boîtes intermédiaires, afin de prévenir l'ossification du protocole.

Flux

Les flux dans QUIC fournissent une abstraction légère, ordonnée et ordonnée.

Il existe deux types de flux de base dans QUIC:

- Les flux unidirectionnels transportent des données dans un sens: de l'initiateur du flux à son homologue.
- Les flux bidirectionnels permettent l'envoi de données dans les deux sens.

L'un ou l'autre type de flux peut être créé par l'un ou l'autre des points de terminaison, peut simultanément envoyer des données entrelacées à d'autres flux et peut être annulé.

Pour envoyer des données via une connexion QUIC, un ou plusieurs flux sont utilisés.

Contrôle de flux

Streams are individually flow controlled, allowing an endpoint to limit memory commitment and to apply back pressure. The creation of streams is also flow controlled, with each peer declaring the maximum stream ID it is willing to accept at a given time.

Les flux sont contrôlés individuellement, ce qui permet à un point de terminaison de limiter l’engagement de la mémoire et d’appliquer une contre-pression. La création de flux est également contrôlée en flux, chaque pair déclarant l’ID de flux maximum qu’il est prêt à accepter à un moment donné.

Identificateurs de flux

Les flux sont identifiés par un entier non signé de 62 bits, appelé ID de flux. Les deux bits les moins significatifs de l'ID de flux sont utilisés pour identifier le type de flux (unidirectionnel ou bidirectionnel) et l'initiateur du flux.

Le bit le moins significatif (0x1) de l'ID de flux identifie l'initiateur du flux. Les clients initient des flux pairs (ceux dont le bit le moins significatif est défini sur 0); les serveurs lancent des flux impairs (avec le bit mis à 1).

Le deuxième bit le moins significatif (0x2) de l'ID de flux différencie les flux unidirectionnels des flux bidirectionnels. Les flux unidirectionnels ont toujours ce bit défini sur 1 et les flux bidirectionnels ont ce bit défini sur 0.

Flux simultané

QUIC permet à un nombre arbitraire de flux de fonctionner simultanément. Une terminaison limite le nombre de flux entrants actifs simultanément en limitant l'ID de flux maximal.

L'ID de flux maximal est spécifique à chaque terminaison et s'applique uniquement à l'homologue qui reçoit le paramètre.

Envoi et réception de données

Les terminaisons utilisent des flux pour envoyer et recevoir des données. C'est après tout leur but ultime. Les flux sont une abstraction ordonnée de flux d'octets. Les flux séparés ne sont toutefois pas nécessairement livrés dans l'ordre d'origine.

Priorité des flux

Le multiplexage de flux a un effet significatif sur les performances des applications si les ressources allouées aux flux sont correctement hiérarchisées. L'expérience acquise avec d'autres protocoles multiplexés, tels que HTTP/2, montre que des stratégies efficaces de hiérarchisation ont un impact positif significatif sur les performances.

QUIC lui-même ne fournit pas de frames pour l'échange d'informations sur la priorisation. Au lieu de cela, il repose sur la réception d'informations de priorité de l'application qui utilise QUIC. Les protocoles qui utilisent QUIC peuvent définir n’importe quel schéma de priorisation adapté à la sémantique de leurs applications.

Lorsque HTTP/3 est utilisé sur QUIC, la hiérarchisation est effectuée dans la couche HTTP.

0-RTT

Pour réduire le temps nécessaire à l'établissement d'une nouvelle connexion, un client déjà connecté à un serveur peut mettre en cache certains paramètres de cette connexion et établir une connexion **0-RTT** avec le serveur. Cela permet au client d'envoyer des données immédiatement, sans attendre la fin d'un handshake.

Spin Bit

Peut-être l'une des discussions de conception les plus longues au sein du groupe de travail QUIC, qui a fait l'objet de plusieurs centaines de mails et d'heures de discussions, concerne un seul bit: le Spin Bit.

Les partisans de ce bit insistent sur le fait qu'il est nécessaire que les opérateurs et les personnes se trouvant entre deux terminaisons QUIC puissent mesurer le temps de latence.

Les opposants à cette fonctionnalité n'aiment pas la potentielle fuite d'informations.

Faire tourner un bit

Les deux points de terminaison, le client et le serveur, conservent une valeur de rotation, 0 ou 1, pour chaque connexion QUIC, et définissent le bit de rotation sur les paquets qu'il envoie pour cette connexion sur la valeur appropriée.

Les deux côtés envoient ensuite des paquets avec ce bit de rotation défini sur la même valeur pendant toute la durée d'un aller-retour, puis la valeur est modifiée. L'effet est alors une impulsion de uns et de zéros dans ce champ binaire que les observateurs peuvent surveiller.

Cette mesure ne fonctionne que lorsque l'expéditeur n'est ni limité par l'application ni par le contrôle de flux, la réorganisation des paquets sur le réseau peut également rendre les données tumultueuses.

Espace utilisateur

L'implémentation d'un protocole de transport dans l'espace utilisateur permet une itération rapide du protocole, car il est relativement facile de faire évoluer le protocole sans nécessiter que les clients et les serveurs mettent à jour le noyau de leur système d'exploitation pour déployer de nouvelles versions.

Rien d’inhérent dans QUIC ne l’empêche d’être implémenté et proposé ultérieurement par les noyaux de systèmes d’exploitation, si quelqu'un trouve ça une bonne idée.

De nombreuses implémentations

Un des effets évidents de l’implémentation d’un nouveau protocole de transport dans l’espace utilisateur est que nous pouvons nous attendre à de nombreuses implémentations indépendantes.

Différentes applications sont susceptibles d'inclure (ou de superposer) différentes implémentations de HTTP/3 et de QUIC dans un avenir prévisible.

API

L’un des facteurs de succès du TCP classique et des programmes qui l’utilisent est l’API de socket standardisé. Il a des fonctionnalités bien définies et à l'aide de cette API, vous pouvez déplacer des programmes entre de nombreux systèmes d'exploitation différents, car TCP fonctionne de la même manière.

QUIC n'est pas là. Il n'y a pas d'API standard pour QUIC.

Avec QUIC, vous devez choisir l’une des implémentations de bibliothèque existantes et s'en tenir à son API. Cela rend les applications "verrouillées" à une seule bibliothèque dans une certaine mesure. Le passage à une autre bibliothèque signifie une autre API, ce qui peut nécessiter beaucoup de travail.

De plus, étant donné que QUIC est généralement implémenté dans l'espace utilisateur, il ne peut pas simplement enrichir l'API de socket ou sembler similaire à la fonctionnalité existante des protocoles TCP et UDP. L'utilisation de QUIC signifie l'utilisation d'une autre API que l'API socket.

HTTP/3

Comme mentionné précédemment, le premier et principal protocole pour transporter sur QUIC est HTTP.

Tout comme HTTP/2 a été introduit pour transporter HTTP sur le réseau d’une manière totalement nouvelle, HTTP/3 introduit encore une fois une nouvelle façon d’envoyer HTTP sur le réseau.

HTTP maintient toujours les mêmes paradigmes et concepts qu'avant. Il y a des en-têtes et un corps, il y a une demande et une réponse. Il y a les méthodes, les cookies et la mise en cache. Ce qui change principalement avec HTTP/3, c'est la manière dont les bits sont envoyés de l'autre côté de la communication.

Pour pouvoir utiliser HTTP sur QUIC, des modifications étaient nécessaires et le résultat de ceci est ce que nous appelons désormais HTTP/3. Ces modifications étaient nécessaires en raison de la nature différente fournie par QUIC par opposition à TCP. Ces changements incluent:

- Dans QUIC, les flux sont fournis par le transport lui-même, tandis que dans HTTP/2, les flux étaient créés dans la couche HTTP.
- Les flux étant indépendants les uns des autres, le protocole de compression d'en-tête utilisé pour HTTP/2 ne pourrait pas être utilisé sans provoquer une situation de tête de bloc.
- Les flux QUIC sont légèrement différents des flux HTTP/2. La section HTTP/3 le détaillera un peu.

URLs HTTPS://

HTTP/3 sera exécuté en utilisant des URLs `HTTPS://` . Le monde est rempli de ces URLs et il a été jugé peu pratique et tout à fait déraisonnable d'introduire un autre schéma d'URL pour le nouveau protocole. Tout comme HTTP/2 n'a pas besoin d'un nouveau schéma, HTTP/3 non plus.

La complexité supplémentaire de la situation de HTTP/3 réside toutefois dans le fait que, lorsque HTTP/2 était un nouveau moyen de transporter HTTP sur le réseau, il était toujours basé sur TLS et TCP comme l'était HTTP/1. Le fait que HTTP/3 soit effectué sur QUIC change les choses en quelques aspects importants.

Les anciennes, en texte clair, URLs `HTTP://` , seront laissées telles quelles et, au fur et à mesure que nous avançons dans le futur avec des transferts plus sécurisés, elles seront probablement de moins en moins utilisées. Les requêtes adressées à ces URLs ne seront tout simplement pas mises à niveau pour utiliser HTTP/3. En réalité, ils passent rarement à HTTP/2 non plus, mais pour d'autres raisons.

Connexion initiale

La première connexion à un hôte récent, non visité précédemment pour une URL HTTPS:// devra probablement être établie via TCP (éventuellement en plus d'une tentative parallèle de connexion via QUIC). L'hôte peut être un ancien serveur sans prise en charge de QUIC ou il peut y avoir une boîte intermédiaire entre les obstacles empêchant le succès d'une connexion QUIC.

Un client et un serveur modernes négocieraient probablement HTTP/2 lors du premier handshake. Lorsque la connexion a été configurée et que le serveur répond à une requête HTTP du client, le serveur peut informer le client de sa prise en charge et de sa préférence pour HTTP/3.

Alt-svc

L'en-tête du service de remplacement (Alt-svc:) et sa trame `ALT-SVC` HTTP/2 correspondante ne sont pas créés spécifiquement pour QUIC ou HTTP/3. Ils font partie d'un mécanisme déjà conçu et créé pour qu'un serveur indique à un client: "Regardez, je lance le même service sur CET HÔTE en utilisant CE PROTOCOLE sur CE PORT" *. Voir les détails dans la RFC 7838.

Un client qui reçoit une telle réponse Alt-svc est ensuite invité, s'il le prend en charge et le souhaite, à se connecter en arrière-plan en parallèle à cet hôte donné - à l'aide du protocole spécifié - et s'il réussit à basculer ses opérations sur cela au lieu de la connexion initiale.

Si la connexion initiale utilise HTTP/2 ou même HTTP/1, le serveur peut répondre et indiquer au client qu'il peut se reconnecter et essayer HTTP/3. Cela pourrait être vers même hôte ou à un autre qui sait comment servir cette origine. Les informations fournies dans une telle réponse Alt-svc ont un temporisateur d'expiration, permettant aux clients de diriger les connexions et demandes ultérieures directement vers l'hôte alternatif à l'aide du protocole alternatif suggéré, pendant une certaine période.

Exemple

Un serveur HTTP incluant une en-tête `Alt-Svc:` dans sa réponse:

```
Alt-Svc: h3=":50781"
```

Cela indique que HTTP/3 est disponible sur le port UDP 50781 avec le même nom d'hôte que celui utilisé pour obtenir cette réponse.

Un client peut ensuite essayer de configurer une connexion QUIC avec cette destination et, en cas de succès, continuer à communiquer avec l'origine comme cela au lieu de la version HTTP initiale.

Flux QUIC et HTTP/3

HTTP/3 étant conçu pour QUIC, il tire pleinement parti des flux de QUIC, où HTTP/2 devait concevoir l'ensemble de son concept de flux et de multiplexage au-dessus de TCP.

Les requêtes HTTP effectuées via HTTP/3 utilisent un ensemble spécifique de flux.

Trames HTTP/3

HTTP/3 signifie la configuration de flux QUIC et l'envoi d'un ensemble de trames à l'autre extrémité. Il n'y a qu'un petit nombre fixe (huit!) de trames connues dans HTTP/3. Les plus importants sont probablement:

- HEADERS, qui envoie des en-têtes HTTP compressés
- DATA, envoie le contenu des données binaires
- GOAWAY, veuillez arrêter cette connexion

Requête HTTP

Le client envoie sa requête HTTP sur un flux QUIC *bidirectionnel* initié par le client.

Une requête consiste en une seule trame HEADERS et peut éventuellement être suivie d'une ou deux autres trames: une série de trames DATA et éventuellement d'une trame HEADERS finale pour terminer.

Après avoir envoyé une requête, un client ferme le flux pour l'envoyer.

Réponse HTTP

Le serveur renvoie sa réponse HTTP sur le flux bidirectionnel. Une trame HEADERS, une série de trames DATA et éventuellement une dernière trame HEADERS.

En-têtes QPACK

Les trames HEADERS contiennent des en-têtes HTTP compressés à l'aide de l'algorithme QPACK, QPACK est styliquement similaire à celui de la compression HTTP/2 appelée HPACK (RFC 7541), mais modifiée pour fonctionner avec des flux livrés dans le désordre.

QPACK lui-même utilise deux flux QUIC unidirectionnels supplémentaires entre les deux terminaisons. Ils sont utilisés pour transporter des informations de table dynamique dans les deux sens.

Priorisation de HTTP/3

Une des trames de flux HTTP/3 s'appelle `PRIORITY`. Elle est utilisée pour définir la priorité et la dépendance à un flux d'une manière similaire à celle de HTTP/2.

La trame peut définir un flux spécifique pour dépendre d'un autre flux spécifique et définir un "poids" sur un flux donné.

Des ressources dépendantes doivent se voir allouer des ressources que si tous les flux dont il dépend sont fermés ou s'il est impossible de progresser sur ces flux.

Un poids de flux est compris entre 1 et 256 et il est spécifié que les flux avec le même parent **devraient** se voir allouer des ressources proportionnellement en fonction de leur poids.

HTTP/3 push serveur

Un serveur push HTTP/3 est similaire à ce qui est décrit dans HTTP/2 (RFC 7540), mais utilise des mécanismes différents.

Un serveur push est en réalité la réponse à une requête que le client n'a jamais envoyée!

Les push serveur ne sont autorisés que si le côté client les a acceptés. Dans HTTP/3, le client définit même une limite pour le nombre de push qu'il accepte en informant le serveur de l'ID de flux de push maximal. Dépasser cette limite entraînera une erreur de connexion.

Si le serveur estime probable que le client souhaite une ressource supplémentaire qu'il n'a pas demandée mais qu'il devrait avoir de toute façon, il peut envoyer une trame `PUSH_PROMISE` (sur le flux de la requête) indiquant à quoi ressemble la requête dont la réponse est destinée, puis envoyer cette réponse réelle sur un nouveau flux.

Même si les envois ont au préalable été déclarés acceptables par le client, chaque flux envoyé individuellement peut toujours être annulé à tout moment si le client le juge approprié. Il envoie ensuite une trame `CANCEL_PUSH` au serveur.

Problématique

Depuis que cette fonctionnalité a été abordée pour la première fois dans le développement de HTTP/2 et ensuite plus tard, après que le protocole ai été livré et déployé sur Internet, cette fonctionnalité a été discutée, détestée et perfectionnée de nombreuses différentes manières afin de la rendre utile.

Un envoi n’est jamais "gratuit", car même s’il enregistre un demi aller-retour, il utilise toujours de la bande passante. Il est souvent difficile ou impossible pour le côté serveur de savoir avec un niveau élevé de certitude si une ressource doit être envoyée ou non.

HTTP/3 comparé à HTTP/2

HTTP/3 est conçu pour QUIC, qui est un protocole de transport qui gère les flux par lui-même.

HTTP/2 est conçu pour TCP et gère donc les flux dans la couche HTTP.

Similitudes

Les deux protocoles offrent aux clients des ensembles de fonctionnalités pratiquement identiques.

- Les deux protocoles offrent des flux
- Les deux protocoles offrent un support push serveur
- Les deux protocoles ont une compression d'en-tête, et QPACK et HPACK ont une conception similaire.
- Les deux protocoles offrent le multiplexage sur une seule connexion utilisant des flux
- Les deux protocoles établissent des priorités sur les flux

Differences

Les différences sont dans les détails et principalement là grâce à l'utilisation de QUIC par HTTP/3:

- HTTP/3 a plus de chances de fonctionner plus tôt grâce aux handshakes 0-RTT de QUIC, alors que TCP Fast Open et TLS envoient généralement moins de données et rencontrent fréquemment des problèmes.
- HTTP/3 a des handshakes beaucoup plus rapides grâce à QUIC vs TCP + TLS.
- HTTP/3 n'existe pas dans une version non sécurisée ou non chiffrée. HTTP/2 peut être implémenté et utilisé sans HTTPS - même si c'est rare sur Internet.
- HTTP/2 peut être négocié directement dans un handshake TLS avec l'extension ALPN, alors que HTTP/3 est sur QUIC donc nécessite une réponse d'en-tête `Alt-Svc:` pour informer le client de ce fait.

Critique

UDP ne fonctionnera jamais

Beaucoup d'entreprises, d'opérateurs et d'organisations bloquent ou limitent le débit du traffic UDP en dehors du port 53 (utilisé pour DNS) depuis qu'il a été depuis ces derniers jours principalement abusé pour des attaques. En particulier, certains des protocoles UDP existants et leur implémentations populaires pour serveur ont été vulnérables aux attaques par amplification dans lesquelles un attaquant peut générer une quantité considérable de trafic sortant afin de cibler des victimes innocentes.

QUIC dispose d'une atténuation intégrée contre les attaques d'amplification en exigeant que le paquet initial soit au minimum de 1200 octets et par une restriction dans le protocole qui stipule qu'un serveur NE DOIT PAS envoyer plus de trois fois ça en réponse sans recevoir un paquet du client en réponse.

UDP est lent dans les noyaux

Cela semble être la vérité, du moins aujourd'hui en 2018. Nous ne pouvons bien sûr pas dire comment cela va évoluer et à quel point cela est simplement le résultat des performances de transfert UDP qui ne sont plus au cœur des préoccupations des développeurs depuis de nombreuses années.

Pour la plupart des clients, cette "lenteur" n'est probablement même jamais perceptible.

QUIC prend trop de processeur

Semblable à remarque "UDP est lent" ci-dessus, c'est en partie du fait que l'utilisation du protocole TCP et TLS dans le monde a pris plus de temps pour se développer, s'améliorer et obtenir une assistance matérielle.

Il y a des raisons de s'attendre à ce que cela s'améliore avec le temps. La question est de savoir de combien et combien cette utilisation supplémentaire du processeur va faire mal aux déployeurs.

C'est juste Google

Non ça ne l'est pas. Google a communiqué les spécifications initiales à l'IETF après avoir prouvé, à grande échelle à l'échelle de l'Internet, que le déploiement de ce style de protocole sur UDP fonctionne actuellement et correctement.

Depuis lors, des membres d'un grand nombre d'entreprises et d'organisations ont travaillé au sein de l'organisation indépendante du fournisseur, l'IETF, pour élaborer un protocole de transport standard. Les employés de Google y ont bien sûr participé, tout comme les employés d'un grand nombre d'autres entreprises intéressées par l'état des protocoles de transport sur Internet, notamment Mozilla, Fastly, Cloudflare, Akamai, Microsoft, Facebook et Microsoft. Apple.

C'est une trop petite amélioration

Ce n'est pas vraiment une critique mais une opinion. Peut-être est-ce le cas, et c'est peut-être une amélioration trop minime, trop proche dans le temps depuis la publication de HTTP/2.

HTTP/3 fonctionnera probablement beaucoup mieux dans les réseaux saturés de pertes de paquets, il offre des handshakes plus rapide, ce qui améliore la latence réelle et perçue. Mais est-ce assez d'avantages pour motiver les gens à déployer la prise en charge HTTP/3 sur leurs serveurs et pour leurs services? Le temps et les mesures de performance futures nous le diront sûrement!

Les spécifications

Voici un recueil des dernières versions officielles des différentes parties et composants de QUIC et de HTTP/3.

Invariants

Version-Independent Properties of QUIC

Transport

QUIC: A UDP-Based Multiplexed and Secure Transport

Récupération

QUIC Loss Detection and Congestion Control

TLS

Using Transport Layer Security (TLS) to Secure QUIC

HTTP

Hypertext Transfer Protocol (HTTP) over QUIC

QPACK

QPACK: Header Compression for HTTP over QUIC

QUIC v2

Afin de se concentrer au mieux sur le coeur du protocole QUIC et de pouvoir le livrer à temps, plusieurs fonctionnalités qui étaient prévues à l'origine pour faire partie du protocole principal ont été reportées et sont maintenant prévues pour être remplacées dans une prochaine version de QUIC. QUIC version 2 ou au-delà.

Cependant, l'auteur de ce document a une boule de cristal plutôt défectueuse, nous ne pouvons donc pas savoir exactement quelles fonctionnalités apparaîtront ou ne figureront pas dans la version 2. Nous pouvons toutefois mentionner certaines des fonctionnalités et éléments explicitement supprimés du travail de la v1 pour être "travaillé plus tard" et pourraient alors éventuellement apparaître dans une version 2.

Forward Error Correction

La Correction d'Erreur Directe (en englais Forward Error Correction (FEC)) est une méthode d'obtention du contrôle d'erreur dans la transmission de données dans laquelle l'émetteur envoie des données redondantes et le récepteur ne reconnaît que la partie des données qui ne contient aucune erreur apparente.

FEC a été expérimenté par Google dans leur travail original sur QUIC, mais il a été retiré à nouveau car les expériences ne se sont pas bien déroulées. Cette fonctionnalité est un sujet de discussion pour QUIC v2, mais il faut probablement que quelqu'un prouve qu'elle peut en fait être un ajout utile sans pénalité excessive.

Multitrajet

Multitrajet (en anglais multipath) signifie que le transport peut lui-même utiliser plusieurs chemins d'accès réseau pour optimiser l'utilisation des ressources et augmenter la redondance.

Les partisans du SCTP dans le monde aiment mentionner que le SCTP est déjà multitrajet, tout comme le TCP moderne.

Données non fiables

Il a été envisagé de proposer comme option des flux "non fiables", qui permettraient alors à QUIC de remplacer également les applications de type UDP.

Adaptations non-HTTP

DNS sur QUIC est l'un des premiers protocoles non HTTP mentionnés qui pourrait attirer l'attention une fois que QUIC v1 et HTTP/3 seront disponibles. Mais avec un nouveau moyen de transport comme celui-ci ayant été apporté au monde, je ne peux pas imaginer que cela s'arrêtera là.