ISAE-SUPAERO

PROJET DE FIN D'ÉTUDES

Etude de QUIC sur les liens satellites géostationnaires

Auteur:
Ludovic THOMAS

Tuteur:
Nicolas KUHN
Encadrant école:
Emmanuel LOCHIN

Projet de fin d'études soutenu pour l'obtention du diplôme d'ingénieur ISAE-SUPAERO et effectué entre mai et octobre 2018

au

Service des télécommunications
Centre National d'Études Spatiales

November 16th, 2018

Remerciements

Pour ce projet, pour ce rapport et pour toutes les discussions passionnantes qui n'apparaîtront pas ici au nom de la sainte concision, merci à tous.

Merci tout d'abord à mon tuteur de stage, Nicolas KUHN, pour tous les échanges que nous avons eut. Ils m'ont ouvert la perspective de l'opérateur, le monde de l'IETF, des publications scientifiques et des jeux de pouvoirs dans Internet. Merci bien sûr pour le temps qu'il m'a consacré, pour son aide, pour sa confiance et pour son enthousiasme.

Merci à mon tuteur école, Emmanuel LOCHIN, qui fut avant tout mon professeur de réseaux. Merci pour sa motivation contagieuse, pour le suivi de mes travaux, les nombreux mails échangés et son aide pour la suite de mon parcours.

Un merci spécial à Ahlem MIFDAOUI pour sa confiance en moi et pour l'énergie qu'elle déploie pour la suite de mon parcours.

Merci à Emmanuel DUBOIS pour toutes les discussions que nous avons eut. Bien qu'aboutissant rarement à un consensus, elles furent passionnantes et pleines de perspectives. Merci pour le temps accordé notamment lors de la rédaction du papier. Merci également pour m'avoir présenté à son père.

Merci à Patrick GELARD pour son expertise, toutes ses connaissances en statistiques et les discussions sur des perspectives bien lointaines mais passionnantes de cryptographie quantique.

Merci à Santiago GARCIA GUILLEN pour son aide dans l'accès au banc d'essai. Merci aussi pour toutes les discussions que nous avons pu avoir autour d'un café.

Merci à Olivier MARTINEZ pour son expertise et son enthousiasme. Merci pour la disponibilité pour la mise en place du banc de test.

Merci à Lionel PICOT et Myriam ABLER pour leur aide lors de la mise en place du banc d'essai.

Merci à Denis BARON pour sa bonne humeur et cette passion contagieuse pour les nouvelles technologies.

Merci à Arnaud DERAMECOURT pour les discussions que nous avons eut et pour l'accès aux moyens CESARS.

Merci à Jérôme MOUEZA pour sa bonne humeur. J'ai adoré partager ma passion du spatial avec lui.

Merci à Pierre LACASSAGNE pour sa bonne humeur et toutes les discussions autour de CESARS.

Merci à Didier DUBOIS pour son temps et cette discussion. Elle fut vraiment intéressante et ouvrit de nombreuses perspectives.

Merci à Marlène MOST, pour toutes ces pauses café, Gemini, Natacha, TBT, ...

Merci à Mme PONCE pour son aide décisive.

Contents

List of Figures

List of Tables

Liste des sigles et acronymes

ACM *Association for Computing Machinery* (Association pour les systèmes d'informations)

ADSL *Asymmetric Digital Subscriber Line* (Ligne abonné numérique et assymétrique)

AEAD *Authenticated Encryption with Associated Data* (Chiffrement authentifié avec données associées)

AIMD *Additive Increase, Multiplicative Decrease* (Augmentation additive, retrait multiplicatif)

AP Applications

API *Application Programming Interface* (Interface de programmation)

BBR *Bottleneck Bandwidth and RTT* (Capacité du goulot d'étranglement et RTT)

BDP *Bandwidth-Delay Product* (Produit capacité-délai)

CC Contrôleur de Congestion

CDF *Cumulative Distribution Function* (Fonction de répartition cumulative)

CDN *Content Delivery Network* (Réseau de distribution du contenu)

CESARS Centre d'expertise et de support pour les usages en télécommunications par satellite

CNES Centre National d'Etudes Spatiales

CNRS Centre National de la Recherche Scientifique

DA Direction Adjointe

DDOS *Distributed Denial Of Service* (Attaque distribuée par deni de service)

DNO Direction du Numérique et des Opérations

DNS *Domain Name System* (Système de noms de domaine)

DPI *Deep Packet Inspection* (Inspection avancée des paquets)

DSO Direction des Systèmes Orbitaux

ECN *Explicit Congestion Notification* (Notification explicite de congestion)

ERN *Explicit Rate Notification* (Notification explicite de débit)

FTTH *Fiber To The Home* (Fibre optique jusqu'à la maison)

GQUIC Google QUIC

GEO *Geostationnary Earth Orbit* (Orbite géostationnaire terrestre)

GW *GateWay* (Passerelle)

HOL *Head-Of-Line blocking* (Blocage par tête de flux)

HTTP *Hypertext Transfert Protocol* (Protocole de tranfert de l'hypertexte)

HTTPS HTTP sécurisé (*i.e.*au dessus de SSL, TLS ou QUIC)

IAB *Internet Architecture Board* (Comité d'architecture d'Internet)

IESG *Internet Engineering Steering Group* (Groupe de pilotage de l'ingénierie pour Internet)

IETF *Internet Engineering Task Force* (Groupe d'ingénierie pour Internet)

IRTF *Internet Research Task Force* (Groupe de recherche pour Internet)

IJSCN *International Journal of Satellite Communications and Networking* (Journal international sur les communications et réseaux satellites)

INCOSE *International Council on Systems Engineering* (Conseil international sur l'ingénierie système)

IP *Internet Protocol* (Protocole Internet)

IRIT Institut de Recherche en Informatique de Toulouse

ISP *Internet service provider* (Fournisseur de services Internet)

LEO *Low Earth Orbit* (Orbite terrestre basse)

MAPRG *Measurement and Analysis for Protocols Research Group* (Groupe de recherche pour les mesures et analyses des protocoles)

MEO *Medium Earth Orbit* (Orbite terrestre de moyenne altitude)

METAR *METeorological Airport Report* (Rapport météorologique d'aéroport)

NAT *Network Address Translator* (Translateur d'adresse réseau)

NT Navigation et Télécommunications

OACI Organisation de l'Aviation Civile Internationale

OLT *Optical Line Terminator* (Terminaison de ligne optique).

ONT *Optical Network Terminator* (Terminaison de réseau optique).

OS *Operating System* (Système d'exploitation)

OTT *Over The Top* (Au dessus de la pile)

PEP *Performance-Enhancing Proxy* (Mandataire améliorant les performances)

PFE Projet de Fin d'Etudes

PLT *Page Load Time* (Temps de chargement de la page)

PR *Pull Request* (Requête d'intégration)

QoS *Quality of Service* (Qualité de service)

QUIC *Quick UDP Internet Connections* (Connexions Internet rapides sur UDP)

QUXA *QUIC User eXperience Assessment* (Evaluation de l'expérience de l'utilisateur QUIC)

RFC *Request for comments* (Requête de commentaires)

R&T Recherche & Technologies

RTT *Round-Trip Time* (Temps d'aller-retour)

SATCOM *SATtellite COMmunication system* (Système de communication par satellite)

SDN *Software defined network* (Réseau défini par logiciel)

SoS *System of Systems* (Système de systèmes)

SPDY *Speedy* ("Rapide")

SSL *Secure Sockets Layer* (Couche pour la sécurité des *sockets*)

ST *Satellite Terminal* (Terminal satellite).

ST Systèmes de Télécommunications

SWOT *Strengths, Weaknesses, Opportunities, Threats* (Forces, Faiblesses, Opportunités, Menaces)

TFO *TCP Fast Open* (Ouverture rapide de TCP)

TCP *Transport Control Protocol* (Protocole de contrôle du transport)

TLS *Transport Layer Security* (Sécurité de la couche de transport)

TTR *Time To* `reponseStart` (Temps écoulé jusqu'à `responseStart`)

UDP *User Datagram Protocol* (Protocole de datagramme utilisateur)

W3C *World Wide Web Consortium* (Consortium du WWW)

WG *Working Group* (Groupe de travail)

WWW *World Wide Web* (Toile Mondiale)

Introduction

Bien que représentant encore peu de clients, l'accès à Internet par satellite constitue une perspective particulièrement intéressante pour les régions du monde possédant de faibles infrastructures.

Les caractéristiques techniques et industrielles de l'accès à Internet par satellite en font un domaine vraiment à part dans le système Internet. En particulier, certaines technologies, telles que *Transport Control Protocol* (Protocole de contrôle du transport) (TCP), sont adaptées par les opérateurs satellites afin d'assurer une meilleure qualité de service pour les clients.

Or, depuis 2012, Google conçoit et déploie *Quick UDP Internet Connections* (Connexions Internet rapides sur UDP) (QUIC), un protocole visant à remplacer TCP et qui empêcherait les opérateurs satellites d'effectuer les mêmes optimisations. QUIC représente aujourd'hui jusqu'à **20% du trafic Internet mondial** et est appelé à se déployer toujours plus.

Dans ce contexte, les opérateurs satellites peuvent s'interroger sur les conséquences du déploiement de ce nouveau protocole sur la qualité d'expérience ressentie par leurs clients. Ils auront à se demander si des actions spécifiques doivent et peuvent être prises afin d'assurer une qualité d'expérience constante pour leurs clients.

Ce projet de fin d'études vise ainsi à :

- réaliser un état de l'art du protocole QUIC et de son utilisation,
- concevoir, mener et exploiter des tests permettant de caractériser les conséquences du déploiement de QUIC pour le client Internet par satellite,
- en déduire les limites du protocole et identifier les axes d'amélioration.

Le Chapitre 1 de ce rapport présente le contexte général d'Internet d'un point de vue industriel et technologique. Il aborde les différents types d'acteurs et les paradigmes technologiques importants. Le Chapitre 2 s'intéresse à l'accès Internet par satellite. L'accent est mis sur la comparaison des organisations industrielle et technologique de ce secteur avec celles d'Internet en général. Le Chapitre 3 présente le protocole QUIC. Toujours par comparaison avec le Chapitre 1, il explique les objectifs de QUIC et ses caractéristiques principales. Il amène ensuite le problème de la multiplicité des versions. Le Chapitre 4 présente les objectifs et le cadre de l'étude ainsi que ses risques. Il décrit ensuite les expériences menées, les mesures effectuées, etc. Enfin, le Chapitre 5 présente et analyse les résultats puis ouvre sur les discussions quant aux limites du protocole et aux potentiels axes d'amélioration.

Chapter 1

Contexte Internet

Ce chapitre n'a pas la prétention de présenter Internet ni de résumer des œuvres complètes et riches telles que [38]. Il va s'attacher à présenter quelques détails et principes importants qui nous seront utiles lorsque nous aborderons QUIC et les SATCOMs.

1.1 Contexte économique : le déploiement d'Internet

Commençons par quelques remarques sur les acteurs en jeu.

1.1.1 Le SoS Internet

Internet est l'un des systèmes complexes par excellence. En effet l'*International Council on Systems Engineering* (Conseil international sur l'ingénierie système) (INCOSE) [31] décrit la complexité d'un système comme étant sa capacité à présenter des propriétés difficilement prévisibles et qui ne peuvent s'expliquer par l'étude distincte de ses composants. Ces propriétés sont alors qualifiées d'émergentes. Le développement des séries vidéos à la demande et le harcèlement en ligne sont deux des innombrables propriétés émergentes du système Internet.

Dans ses activités de taxonomie des systèmes, l'INCOSE a en particulier défini la catégorie des *System of Systems* (Système de systèmes) (SoS) : leurs composants possèdent une autonomie administrative[1] et opérationnelle[2]. Il n'y a donc pas d'autorité suprême chargée de l'ingénierie au niveau global. En revanche, une certaine organisation s'est formée d'elle-même parmi les acteurs d'Internet, les deux parties suivantes se proposent d'y jeter un œil.

L'organisation industrielle d'Internet

Traditionnellement, parmi les entreprises impliquées dans le développement d'Internet, on distingue :

- Les fournisseurs de contenus. Parfois aussi nommées entreprises *Over The Top* (Au dessus de la pile) (OTT)[3]. Ce sont elles qui mettent à disposition les données qui seront demandées par l'utilisateur final. Elles génèrent des applications, qui sont la raison d'être d'Internet [38, page 111]. Des entreprises comme Google ou Netflix en font partie.

[1] *I.e.*en termes d'objectifs économiques, de direction, ...
[2] *I.e.*en termes de cycle de vie, de procédures, de connexion ou nom à d'autres composants, ...
[3] La notion de pile est détaillée en section 1.2.1.

- Les *Internet service providers* (Fournisseurs de services Internet) (ISPs), aussi appelés opérateurs, sont en charge du déploiement des infrastructures permettant l'acheminement des données de l'OTT à l'utilisateur final et *vice versa*. Elles développent les moyens qui rendent possibles les applications. Orange en Europe ou Comcast outre-Atlantique en sont deux exemples.

Ces entreprises, aux objectifs distincts, sont néanmoins interdépendantes. En effet, les applications ne peuvent se développer sans un niveau de service suffisant des infrastructures et le niveau de service de ces dernières peut être affecté par le déploiement d'applications. On peut citer en exemple le conflit qui opposa Netflix et Comcast [60].

L'organisation technologique d'Internet

Comme on le verra plus en détails en section 1.2.1, les notions d'interface et de standard sont primordiales car c'est là que réside le travail d'ingénierie système. Or, en l'absence d'autorité supérieure, un standard nécessite un large consensus pour lui permettre d'être utilisé par de nombreux composants du SoS. Pour assurer ce consensus, les entreprises impliquées dans le développement d'Internet ont placé leur confiance dans certaines organisations indépendantes. Cette partie se propose ainsi de décrire sommairement le processus aboutissant à des standards Internet [24].

1/ Le besoin : Tout d'abord, l'idée, le besoin ou le problème pouvant mener au standard est identifié. Cela peut généralement provenir :

- De la contribution personnelle d'acteurs de l'Internet.
- En sortie de travaux menés par l'*Internet Research Task Force* (Groupe de recherche pour Internet) (IRTF). L'IRTF est un groupe chargé des recherches amont avec une visée long terme.
- Par l'identification d'un problème lors des travaux menés par l'IETF[4].

2/ Le WG : Ensuite, cette idée ou ce besoin fait l'objet d'un travail au sein de l'*Internet Engineering Task Force* (Groupe d'ingénierie pour Internet) (IETF). L'IETF est une organisation "faiblement auto-organisée" [24] dont les principaux objectifs sont de proposer des solutions aux besoins, de standardiser les protocoles et d'émuler la communauté *via* des événements d'échange. Chaque item de travail fait l'objet d'un *Working Group* (Groupe de travail) (WG). Tout le monde peut prendre part au travail d'un WG (rédaction des standards, implémentation des technologies, tests, commentaires,...). Les outils de travail des WG incluent des listes de diffusion (*mailing lists*), des systèmes de rédaction concurrentielle (serveurs `git`), des réunions, etc.

3/ La validation & la publication : Un WG finit son travail en soumettant le standard répondant au besoin à l'*Internet Engineering Steering Group* (Groupe de pilotage de l'ingénierie pour Internet) (IESG). Cet organisme est en charge de la gestion technique des activités de l'IETF. Il donne des directions et vérifie la justesse des standards produits par les WG de l'IETF. Les travaux sont organisés en domaines[5]. Ces

[4] Le rôle de l'IETF est plutôt de répondre aux besoins. Néanmoins ses travaux peuvent en générer de nouveaux.

[5] Au nombre de sept : Applications, Général, Internet, Applications et infrastructures temps-réel, Routage, Sécurité et Transport.

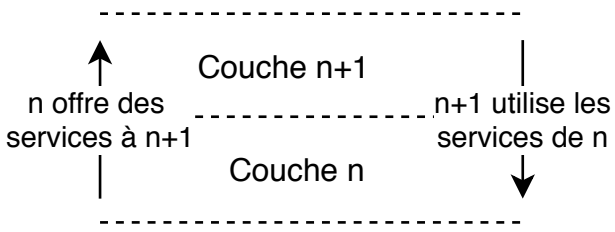


Figure 1.1: Principe d'une pile à deux couches : Les lignes en pointillés représentent les interfaces verticales.

domaines sont pilotés avec l'aide de l'*Internet Architecture Board* (Comité d'architecture d'Internet) (IAB) qui vérifie la cohérence d'ensemble parmi tous les travaux. Enfin, une fois validé, le standard est publié parmi les *Requests for comments* (Requêtes de commentaires) (RFCs)[6] [26, 28].

Si le processus décrit ici fait généralement consensus il faut bien noter qu'il n'est ni nécessaire ni suffisant. Comme pour tout SoS, la règle générale est la suivante : Pour qu'une technologie/un standard soit déployé, il faut :

- qu'une grosse majorité des composants du SoS nécessaires à son opération trouvent un intérêt personnel à l'implémenter et
- que les composants qui ont le pouvoir d'en limiter ou empêcher le déploiement s'en abstiennent.

1.2 Contexte technologique : le modèle en couche

La section précédente présentait quelques aspects du contexte industriel d'Internet. Ce chapitre propose de se pencher sur quelques aspects technologiques. Nous supposons ici que le lecteur a déjà la connaissance du modèle en cinq couches d'Internet[7]. On peut citer le chapitre 1.5 de [38] comme référence.

1.2.1 L'objectif initial : maîtriser la complexité

L'objectif principal des architectures en couches abstraites est de maîtriser la complexité en définissant des interfaces verticales segmentant les différentes étapes d'un processus de communication (Figure 1.1). L'espace entre deux interfaces successives est alors nommé "couche" et est défini indépendamment de son implémentation.

Si il est difficile de trouver une définition exacte d'une couche abstraite, la plupart des références la décrivent comme une généralisation d'un modèle conceptuel. Deux autres aspects sont souvent associés à cette définition [38, page 79]. (a) La notion de service : Chaque couche fournit un service à la couche supérieure et utilise les services de la couche inférieure (Figure 1.1). (b) La notion de responsabilité : Chaque couche est responsable de l'implémentation d'une partie de la communication.

Le modèle en couches d'Internet est sujet à de nombreux débats et parfois de confusions. Ces problèmes tournent souvent autour du critère utilisé pour séparer les couches. Dans la suite du document, nous reprenons la description faite dans [38] : les couches sont définies selon des critères topologiques, *i.e.*en fonction de leur

[6] Leur nom historique étant désormais trompeur : Une RFC est considérée comme un travail accompli et n'est jamais modifiée.

[7] Parfois aussi appelé "pile TCP/IP".

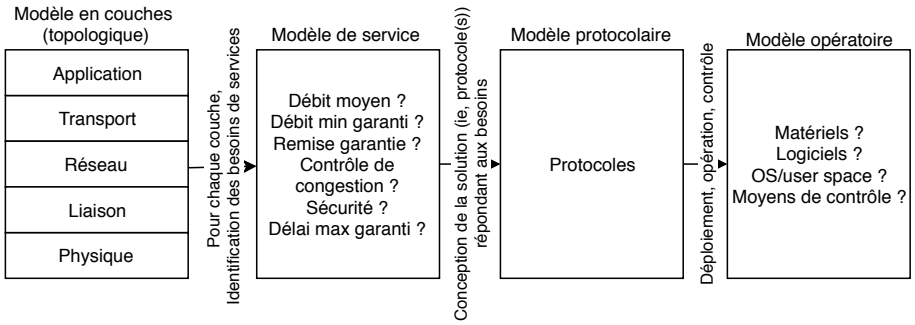


Figure 1.2: Modèles topologiques, de service, protocolaire et opératoires ainsi que leurs relations. Un choix technique réside dans chaque étape de l'association.

Modèle en couches (topologique)	Modèle de service	Modèle protocolaire	Modèle opératoire
Application	--	HTTP	user space
Transport	Remise garantie Contrôle de flux Contrôle congestion	TCP	kernel space algorithme de calcul de la congestion
Réseau	*Best effort* (aucune garantie)	IP	hardware network manag^{mt}

Figure 1.3: Modèles pour une pile HTTP classique. Ici, la couche applicative demande un service de fiabilité, or la couche réseau ne fournit pas ce service sur Internet : c'est donc un service dont l'implémentation incombe à la couche Transport.

rôle dans la communication et des acteurs qui y sont impliqués. Ainsi, la couche transport est définie par [38, page 79]:

"*The Internet's transport layer transports application-layer messages between application endpoints.*"

On a donc la description de son action (le transport), de la donnée transportée (*application-layer messages*) et des acteurs en jeu (les *endpoints*).

1.2.2 Relation avec d'autres types de modèles

Définir le modèle en couches par un critère topologique permet de bien le distinguer des autres modèles que nous décrivons ci-après (voir aussi Figure 1.2). Il est primordial de comprendre que l'application (le *mapping*) des modèles suivants sur le modèle en couche est un choix : d'autres systèmes de communications peuvent avoir des associations différentes.

Le modèle de service

Le modèle de service caractérise les propriétés des services proposés par une couche à la couche supérieure via son interface verticale haute. Parmi les propriétés le plus souvent considérées, on peut citer par exemple :

- La garantie de remise : La couche garantit à la supérieure que les données à transmettre arriveront *in fine* à destination.
- Le contrôle de congestion : La couche limite l'usage de la ressource à son utilisation équitable (*fairness*).

- La confidentialité : Les informations transmises ne seront accessibles qu'au(x) destinataire(s).
- La garantie de délai maximal : La couche garantit une remise de l'information à la couche haute du destinataire en un délai maximal fixé.
- ...

Voir sections 2.1.3 et 2.1.4 de [38] et la section 3 de [45] pour une description d'autres propriétés possibles.

La pile TCP/IP, majoritaire sur Internet, réalise ainsi un exemple d'association entre le modèle en couche et le modèle de service (Figure 1.3). Elle se base sur le principe d' "intelligence aux extrémités". La couche Réseau adopte un modèle de service "au mieux" (*Best Effort*) qui n'apporte aucune garantie aux services rendus. Ainsi la plupart des propriétés de service mentionnées ci dessus est implémentée par les couches transport ou application, qui, par topologie, mettent en œuvre des acteurs aux extrémités (*endpoints*).

Notons pour finir que la majorité des services profite *in fine* à la couche la plus haute (*i.e.*, Applications)[8] et qu'ainsi c'est elle qui dirige le choix de la répartition des services dans les couches. Certaines garanties de service peuvent être obtenues dans une couche quelque soit les services rendus par les couches inférieures (ex: la confidentialité) d'autres nécessitent que toutes les couches y participent (ex: la garantie de délai maximal).

Le modèle protocolaire

Le modèle protocolaire est la solution technologique du modèle de service pour le modèle en couches. A chaque couche, on associe un ou plusieurs protocoles répondant aux besoins du modèle de service.

Le modèle opératoire

Le modèle opératoire décrit la manière dont est opéré chaque couche. D'une part, cela décrit les technologies d'implémentation de la couche (matérielles, logicielles) : on appelle cela le plan de données (*Data Plane*). Et d'autre part le modèle décrit la manière dont sont calculés et générés les paramètres de contrôle du protocole : c'est le plan de contrôle (*Control Plane*)[9].

1.2.3 La récursivité du modèle en couches

Un autre aspect à conserver à l'esprit est la récursivité du modèle en couches : selon le niveau du système considéré, certaines couches peuvent se présenter comme de nouveaux réseaux à part entière et se décomposer à leur tour en plusieurs couches. C'est particulièrement le cas lorsqu'on interconnecte plusieurs réseaux de technologies différentes pour créer un réseau plus grand, comme dans les réseaux embarqués par exemple. Enfin, l'histoire d'Internet lui-même, conçu pour interconnecter des réseaux ("*a network of networks*" [38, page 90]) reflète cet aspect de récursivité.

[8] On pourrait citer en exception le contrôle de congestion. D'après [38, page 220] : "*Congestion control is not so much a service provided to the invoking application as it is a service for the Internet as a whole, a service for the general good*".

[9] Par exemple le routage, c'est à dire la constitution des tables de transferts.

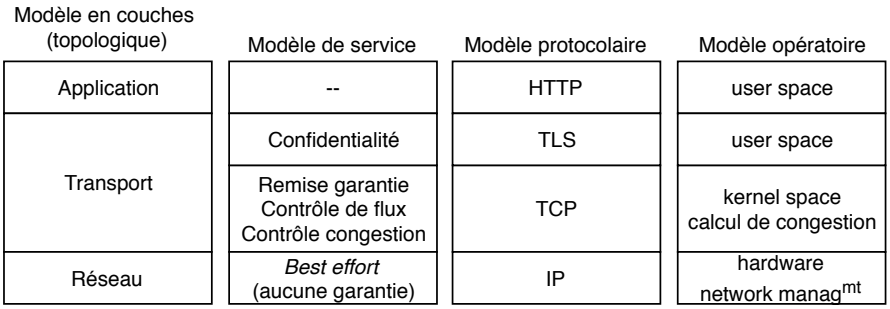


Figure 1.4: Modèles pour une pile HTTPS classique. Bien que situé entre TCP et HTTP, TLS et le service qu'il fourni appartiennent à la couche Transport.

1.2.4 Les limites du modèle en couches

Le modèle en couches reste toutefois un modèle d'abstraction qui n'est pas nécessairement toujours applicable. L'histoire d'Internet est peuplée de débats et remises en cause de cette abstraction [13]. Cette section se propose de présenter quelques débats qui montrent les limites des paradigmes d'Internet et amènent les problématiques de QUIC sur les SATCOMs.

La "couche" de sécurité

Le *World Wide Web* (Toile Mondiale) (WWW) est considéré comme la *killer* application d'Internet à partir des années 1990 [38, page 111] et a propulsé la pile HTTP/TCP/IP à la première place du trafic mondial. En parallèle de ce modèle protocolaire, les modèles opératoires ont été conçus et déployés. *Transport Control Protocol* (Protocole de contrôle du transport) (TCP) fut implémenté dans le cœur (*kernel*) des *Operating Systems* (Systèmes d'exploitation) (OS), les TCP *sockets* ont été créées pour fournir les *Application Programming Interfaces* (Interfaces de programmation) (APIs).

Ce n'est qu'à partir de 1994 [59] que les besoins de sécurité (confidentialité, intégrité, authentification) ont commencé à se manifester. Pour répondre à cette nouvelle exigence de service, *Secure Sockets Layer* (Couche pour la sécurité des *sockets*) (SSL) fut déployé à partir de février 1995. Ce protocole intermédiaire est implémenté entre TCP et *Hypertext Transfert Protocol* (Protocole de tranfert de l'hypertexte) (HTTP). Par sa conception privée et son origine tardive par rapport au déploiement de TCP dans l'espace *kernel*, SSL fut implémenté en *user space*.

Beaucoup d'ambiguïté en découla : tantôt SSL est considéré comme appartenant à la couche "Applications"[10], tantôt il est considéré être un protocole de transport[11], tantôt, ainsi que son nom le laisse supposer, il est considéré comme appartenant à une nouvelle couche[12]. Les travaux de l'IETF avec *Transport Layer Security* (Sécurité de la couche de transport) (TLS) ont toutefois levé une bonne partie des ambiguïtés en confirmant la supression du paradigme "1 couche = 1 protocole"[13] (Figure 1.4).

[10] Suivant le paradigme que l'*user* space est l'espace des applications.

[11] Légende de la figure 8.24 de [38, page 660].

[12] Une couche (*layer*) de "Sécurité des *sockets*", entre Transport et Applications.

[13] TLS étant ainsi considéré comme un protocole assurant la sécurité de la couche Transport, tandis que TCP a en charge les aspects de fiabilité, de contrôle de congestion, etc.

Toutefois, dans la suite du document et sauf mention contraire, la notion de "paramètres de transport" fera référence à des paramètres de contrôle des fonctions de transport à l'exception de celles associées à la sécurité. Lorsqu'on parlera des paramètres des fonctions de sécurité, nous préciserons "paramètres cryptographiques" ou "paramètres de sécurité". L'objectif est ici de rester consistant vis à vis de la documentation dans ce domaine.

Les intermédiaires de couches hautes

La paradigme du réseau Internet *Best Effort* avec l'intelligence aux extrémités est aussi à l'origine de remises en cause du modèle en couches. Une nouvelle fois, des besoins, souvent liés à la sécurité, ont été identifiés tardivement. C'est ainsi que sont apparus les pare-feu, les *Network Address Translators* (Translateurs d'adresse réseau) (NATs), les proxy, les analyseurs *Deep Packet Inspection* (Inspection avancée des paquets) (DPI), etc. Ces éléments, situés dans le réseau et non à ses extrémités, opèrent sur les champs des protocoles associés aux couches Transport voir Applications. Voilà pourquoi on considère parfois qu'ils violent le modèle en couches. Avec le déploiement des *Software defined networks* (Réseaux définis par logiciel) (SDNs), ils sont génériquement regroupés sous le terme d'intermédiaires de couches hautes, ou *middle-boxes*.

Chapter 2

Description d'un SATCOM

Le chapitre 1 détaillait l'organisation industrielle et technologique d'Internet en général. Dans ce chapitre, nous nous concentrons sur l'accès Internet par les *SATtellite COMmunication systems* (Systèmes de communication par satellite) (SATCOMs).

2.1 Le lien d'accès

L'architecture opératoire d'Internet est par exemple décrite en section 1.1 de [38]. Pour notre propos, nous nous concentrons simplement sur le fait qu'Internet peut être décomposé en son cœur (*backbone*) et ses bords (*edges*). Un utilisateur accède à Internet via un **lien d'accès** qui le connecte au réseau interne de son ISP, lequel est connecté au cœur de réseau.

2.1.1 Le lien d'accès terrestre

La Figure 2.1 présente un accès typique par fibre optique : le client installe chez lui une *box* intégrant un *Optical Network Terminator* (Terminaison de réseau optique) (ONT). Ce dispositif adapte les échanges au canal de transmission utilisé (ici, une fibre optique). De l'autre côté, un *Optical Line Terminator* (Terminaison de ligne optique) (OLT) effectue la même action, mais en étant connecté au cœur de réseau *via* le réseau de l'ISP. Pour limiter les coûts d'infrastructure, la ressource canal (ici, la fibre optique) est généralement divisée à proximité de l'utilisateur pour distribuer plusieurs clients.

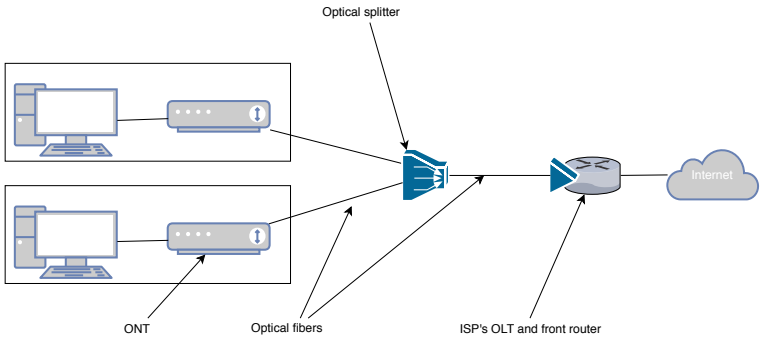


Figure 2.1: Principe d'accès à Internet par lien optique : Les utilisateurs à gauche sont reliés au cœur de réseau *via* le réseau de leur ISP et un lien optique partagé.

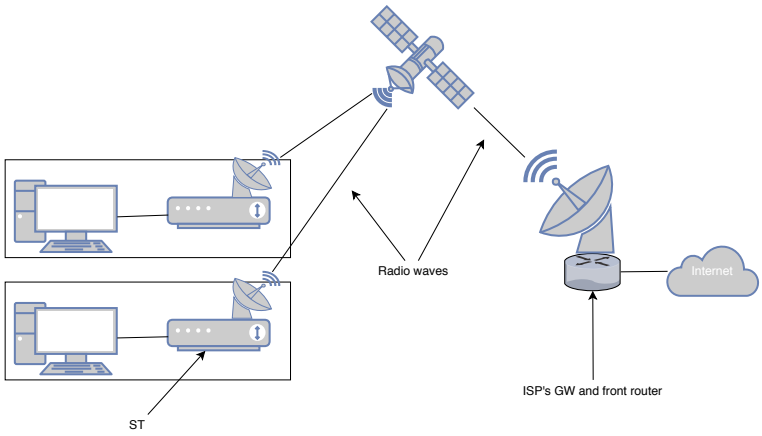


Figure 2.2: Principe d'accès à Internet par satellite : Les utilisateurs à gauche sont reliés au réseau de leur ISP via des ondes radios et un (ou plusieurs) satellite(s).

Dans la majorité des cas, le lien d'accès est le plus gros contributeur de la latence[1] et est le goulot d'étranglement de la capacité[2]. En revanche, du point de vue protocolaire, il n'est pas différent d'un lien très haut débit reliant deux routeurs dans le cœur de réseau.

2.1.2 Le lien d'accès par satellite

En particulier la technologie sous-jacente à un tel lien est interchangeable. La figure 2.2 représente la même situation, avec un lien d'accès par satellite. Ici, client et ISP se dotent respectivement d'un *Satellite Terminal* (Terminal satellite) (ST) et d'une *GateWay* (Passerelle) (GW) pour effectuer l'adaptation de la transmission au canal (ici, des ondes radios via un ou plusieurs satellite(s)).

2.2 Les architectures satellitaires

L'accès à Internet par satellite ne nécessite pas le déploiement d'infrastructures terrestres à l'exception de celles nécessaires pour le fonctionnement des GW (regroupées dans des téléports). Cet avantage sur les autres technologies permet aux réseaux SATCOM d'être déployés dans les zones peu peuplées ou difficilement accessibles, de fournir un accès Internet en mobilité (avion au dessus d'un océan) ou après une catastrophe naturelle (Figure 2.3).

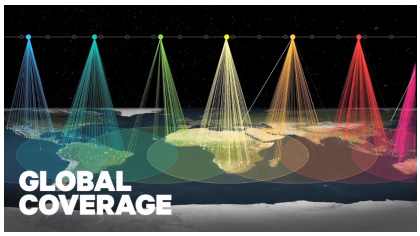
Les récentes annonces sur le déploiement de constellations pour un accès global à Internet[3] justifient que l'on aborde rapidement les types d'architectures satellitaires des SATCOMs.

2.2.1 Les constellations

Les constellations de communications opèrent généralement des satellites en *Low Earth Orbit* (Orbite terrestre basse) (LEO)[4] disposés de manière à couvrir l'ensemble

[1] Latence : Temps mis par les paquets pour traverser le réseau.
[2] Capacité : Débit maximal atteignable entre deux *endpoints* du réseau.
[3] Oneweb [46] et Starlink [58] par exemple.
[4] Parfois plus rarement en *Medium Earth Orbit* (Orbite terrestre de moyenne altitude) (MEO), comme la constellation O3b.

Figure 2.3: Image d'une parabole mobile déployée lors d'un exercice de secours. Couplée à un groupe électrogène, elle permet un accès à Internet et une coordination des services, même en l'absence de toute infrastructure. `Source : lafibre.info`



(a) Sans connexion inter-satellite : Chaque satellite couvre une zone mobile mais ne communique pas avec ses voisins. `Source : O3b Networks.`

(b) Avec connexion inter-satellite : Chaque satellite est connecté à 4 voisins, créant ainsi un véritable maillage. `Source : Iridium communications.`

Figure 2.4: Vues d'artistes de deux catégories de constellations SATCOM.

du globe terrestre à tout instant. Leur faible altitude (en général environ 800km) permet une certaine proximité avec les clients donc des latences réduites. On en distingue deux types :

Sans routage inter-satellite : Dans ce type de constellation, les satellites ne se contentent que de retransmettre directement du client au téléport régional. C'est l'infrastructure terrestre qui se charge alors de router le paquet vers sa destination, si nécessaire en repassant par un téléport (Figure 2.4a).

Avec routage inter-satellite : Dans ce type de constellation, les satellites sont des composants à part entière du réseau, capables d'effectuer des actions de transfert à d'autres satellites (Figure 2.4b). Le segment sol nécessaire à l'opération de telles constellations est particulièrement allégé, les paquets pouvant transiter directement d'un terminal satellite à un autre sans passer par le téléport de l'opérateur. En revanche, la complexité technologique en fait généralement des systèmes peu performants et prohibitifs à l'usage.

Table 2.1: Caractéristiques typiques de certaines méthodes d'accès: Le SATCOM présente une meilleure capacité que l'ADSL mais une latence bien plus élevée. La fibre optique surpasse ces deux technologies.

Accès	Capacité descendante typique (Mbits/s)	Capacité montante typique (Mbits/s)	Double latence (RTT) typique (ms)
SATCOM	20-30	5-8	500-800
ADSL	5-20	2-5	10-100
FTTH	100-1000	100-500	1-10

2.2.2 Le satellite géostationnaire

Les architectures à base d'un satellite *Geostationnary Earth Orbit* (Orbite géostationnaire terrestre) (GEO) reposent sur un unique système spatial, immobile dans le référentiel terrestre et agissant uniquement dans la couche physique avec de l'amplification de signal et de la transposition de fréquence[5]. L'avantage décisif de tels systèmes se base sur l'immobilité apparente pour l'utilisateur du satellite : une antenne parabolique grand gain fixe peut être déployée côté client, assurant ainsi des débits élevés pour une complexité limitée du terminal. C'est pourquoi cette solution constitue toujours actuellement le moyen d'accès privilégié à Internet par satellite et c'est sur cette architecture que se portera l'objet de l'étude. Dans la suite du document, sauf mention contraire, l'acronyme SATCOM désignera les systèmes d'accès par satellite géostationnaire.

2.3 Les SATCOMs GEO

2.3.1 Comparaison entre l'accès SATCOM et les accès terrestres

La Table 2.1 présente les métriques typiques de certains liens d'accès à Internet. Mettons de côté la *Fiber To The Home* (Fibre optique jusqu'à la maison) (FTTH) qui, tout en surpassant les deux autres technologies, nécessite le déploiement d'infrastructures terrestres complexes. Le SATCOM présente ainsi des capacités de débits supérieures à l'*Asymmetric Digital Subscriber Line* (Ligne abonné numérique et assymétrique) (ADSL) mais une latence jusqu'à 80 fois plus élevée en raison de la distance parcourue par les ondes et ainsi de leur délai de propagation. Ces caractéristiques font donc de ce type de lien d'accès un secteur vraiment à part dans l'environnement Internet, avec des contraintes dans les solutions qui ne sont pas nécessairement les mêmes que pour les contextes terrestres.

2.3.2 Les acteurs du SATCOM

Les caractéristiques techniques étant très différentes des liens terrestres, le secteur des SATCOM présente aussi une organisation industrielle différente.

[5]Connu sous le nom de mode *Bent Pipe* (tuyau courbé).

Les opérateurs d'Internet par satellite

Tout d'abord il faut noter que l'accès Internet par SATCOM constitue un marché de niche. En effet, les SATCOM ne représentent qu'une part marginale du trafic Internet mondial[6].

Les gros opérateurs terrestres sont donc généralement absents de ce secteur :

- Dans les territoires développés, les opérateurs terrestres sont concentrés sur le déploiement de technologies plus coûteuses mais aussi plus performantes (FTTH et 4G)[7].
- Et les gros opérateurs sont absents des territoires en développement, là où le satellite a un avantage décisif.

Les **opérateurs** de service Internet par SATCOM sont donc généralement des opérateurs satellites possédant déjà l'expertise de la diffusion (télévision, radio) par satellite. L'activité SATCOM représente alors généralement une part négligeable de leur activité[8].

Les **distributeurs** sont des sociétés tierces qui assurent l'intermédiaire entre les opérateurs et les clients finaux en achetant de la capacité en gros et en gérant les aspects commerciaux, équipements clients, etc.

Le rôle du CNES

Dans ce marché très spécifique possédant une capacité limitée en Recherche & Technologies (R&T), le Centre National d'Etudes Spatiales (CNES) se place comme un acteur tiers d'expertise technologique. Ses compétences et ses moyens en systèmes de télécommunications et en systèmes spatiaux lui permettent :

- De mener des études et des activités de R&T. Les résultats peuvent ensuite être présentés à la communauté Internet.
- D'assister les opérateurs dans leurs choix technologiques.
- De mettre à disposition ses moyens pour permettre aux collectivités et entreprises en ayant le besoin de réaliser des tests et/ou une partie de leurs activités.

Au sein du CNES, cette mission d'expertise et de mise à disposition de moyens est assurée par deux services :

DSO/NT/ST : Le service Systèmes de Télécommunications (ST) de la sous-direction Navigation et Télécommunications (NT) de la Direction des Systèmes Orbitaux (DSO) apporte son expertise technique. La majeure partie des travaux menés concerne les couches basses (Physique, Liaison, Réseau) au service des constructeurs de satellites et des opérateurs mais d'autres travaux peuvent aussi concerner les couches Transport et Application.

[6] Il est assez difficile de trouver une étude quantifiant la part du trafic d'origine SATCOM dans le trafic total en raison de la complexité à définir et obtenir une telle métrique. A défaut de données sur le lien d'accès, on peut citer [48] qui note que moins de 0.2% du trafic intercontinental passe par satellite.

[7] "*Nous n'avons pas fait le choix du satellite dans nos projections futures*" Fabienne Dulac, directrice Orange France [42].

[8] En 2015, dans un document destiné à ses investisseurs [2], Eutelsat (opérateur de satellites de communications) indiquait que ses activités d'accès Internet haut débit ne représentaient que 5% de ses revenus.

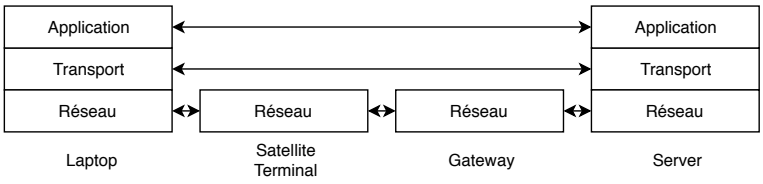


Figure 2.5: Architecture en couches attendue pour l'accès SATCOM : Le *Satellite Terminal* (Terminal satellite) (ST) et la *GateWay* (Passerelle) (GW) commutent les paquets au niveau Réseau.

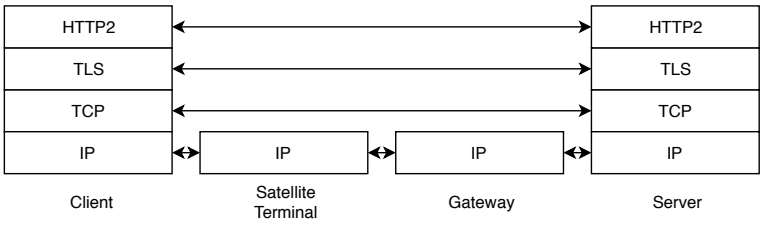


Figure 2.6: Architecture protocolaire attendue pour l'accès SATCOM: Le ST et la GW agissent comme des routeurs IP.

DNO/DA/AP : Le service Applications (AP) de la Direction Adjointe (DA) de la Direction du Numérique et des Opérations (DNO) contrôle les moyens du CNES, gère leur planning d'utilisation et leur opération. Ces moyens incluent des offres Internet par satellite publiques[9] ainsi que des moyens propres via le satellite Athena-Fidus. Il met aussi en œuvre des démonstrations de cas d'application pour promouvoir l'usage du satellite auprès des professionnels, services de secours et particuliers.

Mon projet de fin d'études a ainsi été mené au sein du service DSO/NT/ST avec le soutien de DNO/DA/AP pour la partie banc de test.

2.3.3 Architecture réseau d'un SATCOM

Architecture réseau théorique

La Figure 2.2 présentait le principe d'accès à Internet par satellite. Dans le cas de notre étude (SATCOM GEO), nous indiquions en section 2.2.2 que le satellite en lui-même peut être considéré comme appartenant à un simple lien d'accès. Intéressons-nous ainsi à l'architecture du point de vue réseau, c'est à dire en appliquant les principes vus en section 1.2.2. Le *Satellite Terminal* (Terminal satellite) (ST) et la *GateWay* (Passerelle) (GW) peuvent être considérés comme des passerelles entre plusieurs sous-réseaux[10] : ce sont donc des commutateurs de niveau Réseau. Ainsi, en ne gardant que les couches supérieures, on s'attend à observer l'architecture telle qu'en Figure 2.5 : Le ST et la GW sont des intermédiaires dans le réseau, ils ne participent ainsi ni au transport ni à l'application. Si maintenant on réalise l'association des protocoles telle que décrite en Figure 1.4, nous obtenons l'architecture attendue présentée en Figure 2.6 dans laquelle il apparaît clairement que ST et GW ne travaillent qu'au niveau *Internet Protocol* (Protocole Internet) (IP).

[9] Louées comme elles le seraient par de simples particuliers et utilisées/instrumentées pour des tests.

[10] Entre le réseau domestique local du client et le réseau satellite d'une part et entre le réseau satellite et le réseau interne de l'ISP d'autre part.

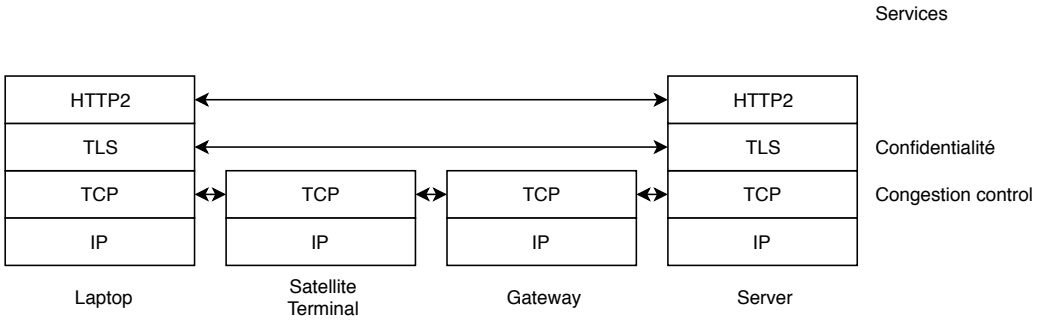


Figure 2.7: Architecture protocolaire réelle dans un accès SATCOM: Le ST et la GW intègrent des *Performance-Enhancing Proxies* (Mandataires améliorant les performances) (PEPs) qui coupent les connexions TCP.

Architecture réseau en pratique

En pratique, dans la majorité des accès par satellite, le ST, la GW ou les deux implémente une fonctionnalité dite *Performance-Enhancing Proxy* (Mandataire améliorant les performances) (PEP) séparant en plusieurs morceaux les connexions TCP[11] (voir Figure 2.7). Le PEP est ainsi qualifié de mandataire car au niveau TCP, le *end-point* a l'impression de communiquer avec le serveur[12]. Le mandataire (ou proxy) assure donc à la place du serveur une partie des services TCP : transport fiable, contrôle de congestion, etc.

La section 7.8 de [43] présente la manière dont les caractéristiques d'un lien satellite influent sur les performances de TCP et justifie ainsi la nécessité des PEPs : mécanismes spécifiques de niveau TCP dans le réseau de l'opérateur SATCOM. Les RFC [20, 51] précisent ces mécanismes. Dans la suite, nous nous concentrons sur le bénéfice apporté par ces PEPs sur le contrôle de congestion.

Le contrôle de congestion d'un protocole de transport fiable

Le contrôle de congestion a été introduit à la fin des années 1980 pour contrer plusieurs effondrements des performances d'Internet ("*congestion collapses*") observés à partir de 1986 [33]. L'origine de tels effondrement était liée aux principes même du transport fiable : le débit en sortie d'une interface d'un *end point* utilisant TCP est le débit fourni par l'application[13] auquel s'ajoute le débit lié aux retransmissions des segments perdus. Or, en l'absence de contrôle de congestion, plus il y a de paquets perdus, plus le débit de retransmission augmente, *i.e.*plus le débit total en sortie d'interface augmente et donc plus la congestion est importante et ainsi plus il y a de paquets perdus. Ce cercle vicieux entraîne un effondrement du débit utile (*goodput*) délivré à l'application destinatrice (voir notamment la Section 3.6 et la Figure 3.48 de [38]).

Depuis, le contrôle de congestion est devenu le sujet de nombreux travaux de recherche. Plusieurs algorithmes ont été développés et déployés. Les plus connus sont TCP New Reno [4], CUBIC [22] et *Bottleneck Bandwidth and RTT* (Capacité du

[11] On se concentre ici sur le mécanisme le plus répandu.

[12] On parle alors aussi de mandataire transparent (*transparent proxy*).

[13] Auquel s'applique, le cas échéant, une limitation de contrôle de flux pour éviter le débordement du buffer de réception du destinataire.

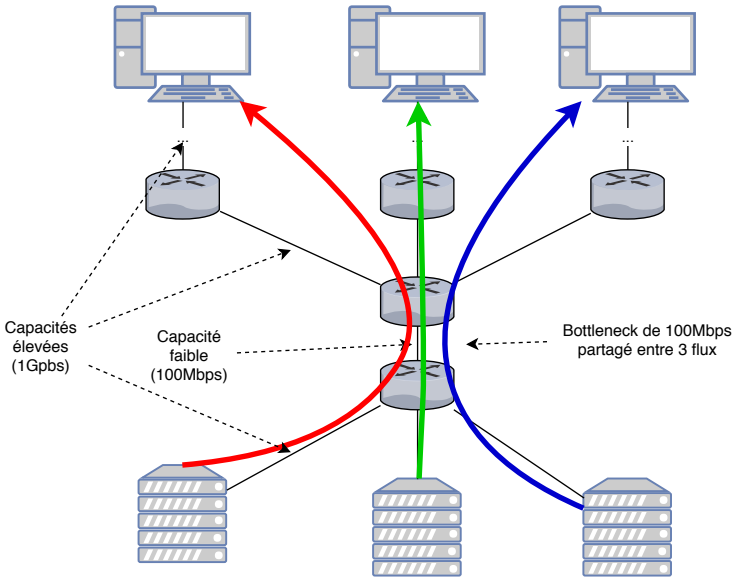


Figure 2.8: Exemple de situation avec un goulot d'étranglement : Un lien faible débit est partagé entre trois flux de téléchargement, tous les autres liens ayant une capacité très élevée par rapport à l'étranglement.

goulot d'étranglement et RTT) (BBR) [8]. Tous se basent sur les quelques principes suivants :

- Un émetteur et un récepteur communiquent via un chemin dans le réseau. Le lien qui, dans ce chemin, a la capacité minimale $\mathcal{B}$ (*bottleneck bandwidth*), est le goulot d'étranglement de la connexion (Figure 2.8).
- Si N connexions se partagent ce goulot d'étranglement, chaque connexion peut utiliser une capacité équitable $\mathcal{B}_f = \mathcal{B}/N$ (*fairness*).
- L'objectif du contrôle de congestion est donc d'utiliser au mieux cette capacité équitable $\mathcal{B}_f$. Avec un débit plus faible, la ressource est sous-utilisée et l'expérience utilisateur est impactée. Avec un débit plus élevé, de la congestion peut survenir et réduire ainsi le débit utile (*goodput*), réduisant aussi l'expérience de l'utilisateur.

Les algorithmes de contrôle de congestion sont donc notamment évalués selon les performances suivantes :

- Leur équité (*fairness*), c'est à dire leur capacité à partager la capacité d'étranglement.
- Leur vitesse de convergence : caractérise la rapidité du Contrôleur de Congestion (CC) à atteindre le débit équitable cible soit parce-que le flux vient de s'initialiser, soit parce qu'un nouveau flux souhaite partager le goulot d'étranglement.

Les contrôleurs de congestion d'Internet sont tous de types "bout en bout" *end-to-end*, ce qui signifie que la couche Réseau ne fournit pas d'information de congestion au CC[14]. Ce dernier doit donc utiliser les pertes, les mesures de *Round-Trip Time* (Temps d'aller-retour) (RTT) et de débit pour détecter la congestion [38, page 296].

[14]L'exception remarquable est ici le mécanisme *Explicit Congestion Notification* (Notification explicite de congestion) (ECN) [18], mais c'est hors de notre étude.

En particulier, lorsque la connexion se crée, le CC n'a pas la connaissance *à priori* de la capacité équitable, ni même de son ordre de grandeur[15]. Le temps nécessaire au CC pour converger vers ce débit est ainsi un facteur déterminant pour l'expérience de l'utilisateur, en particulier lorsqu'il télécharge des objets de taille moyenne.

Le *slow start* : Pour l'obtention de l'ordre de grandeur de la capacité équitable, tous les CCs répandus utilisent un mécanisme appelé *slow start* (démarrage lent), parfois aussi *binary search* (recherche binaire). Le principe se base sur une augmentation exponentielle du débit, conditionnée à chaque pas à la réception d'acquittements, jusqu'à ce qu'une heuristique détecte que le débit s'approche de la capacité équitable.

Le CC contrôle ainsi la taille d'une fenêtre de congestion $\mathcal{W}$ qui représente le nombre maximal de données que l'émetteur peut envoyer sans avoir reçu d'acquittement (voir Figure 3.23 de [38, page 255]). On parle alors de paquets ou d'octets "en vol" (envoyés mais dont la réception n'est pas confirmée). $\mathcal{W}$ est initialisée à une valeur initiale $\mathcal{I}$. Lors du *slow start*, à chaque fois qu'un segment est acquitté, $\mathcal{W}$ est augmentée d'un segment ce qui se traduit par un doublement de sa valeur à chaque RTT. Lorsque $\mathcal{W}$ dépasse la valeur du *Bandwidth-Delay Product* (Produit capacité-délai) (BDP), les buffers des routeurs commencent à se remplir : la congestion apparaît. En effet le BDP $\mathcal{B}$ correspond au produit de la capacité équitable $\mathcal{B}_f$ par le temps d'aller ($\approx$ RTT/2) et caractérise ainsi la quantité de données qui peut être "en route" dans le chemin (envoyées mais pas encore reçues), sans remplir les buffers des routeurs. Le lecteur pourra retrouver plus d'informations sur ces notions à la section 3.7 de [38].

Le *slow start* dure jusqu'à ce que l'heuristique du CC détecte une congestion. Cette heuristique varie selon le contrôle de congestion[16]. En règle générale, elle ne se déclenche qu'une fois que la fenêtre de congestion atteint au moins un BDP. Pour cela, $\mathcal{W}$ aura dû être doublée $\ln_2\left(\frac{\text{BDP}}{\mathcal{I}}\right)$ fois. L'ordre de grandeur de la durée du *slow start* vaut donc :

$$\begin{aligned} t(\mathcal{R}, \mathcal{B}_f) &= \ln_2\left(\frac{\text{BDP}}{\mathcal{I}}\right)\mathcal{R} \\ &= \ln_2\left(\frac{\mathcal{B}_f\mathcal{R}}{2\mathcal{I}}\right)\mathcal{R} \end{aligned} \tag{2.1}$$

Avec $\mathcal{R}$ le RTT, $\mathcal{B}_f$ la capacité équitable du goulot d'étranglement et $\mathcal{I}$ la taille initiale de la fenêtre de congestion. Comme on peut ainsi le voir, plus la capacité et plus le RTT sont importants, plus le CC prend du temps pour atteindre le débit cible.

L'"accélération" des connexions TCP

Or c'est justement sur ce temps de *slow start* que notre étude va porter. Plus il est court, plus le CC utilise rapidement la ressource au meilleur de ses capacités et plus

[15] Les ordres de grandeur des capacités pouvant être très divers sur Internet. Extrait de [8] (document de définition de BBR): *"To handle Internet link bandwidths spanning 12 orders of magnitude, Startup implements a binary search*".

[16] On peut citer :

- Pour TCP Reno, c'est la perte d'un segment qui déclenche la sortie.
- Pour Hystart [21] (utilisée notamment avec CUBIC), l'heuristique se base sur l'évolution des RTT.
- Pour BBR, l'heuristique se déclenche lorsque le débit n'augmente plus significativement.

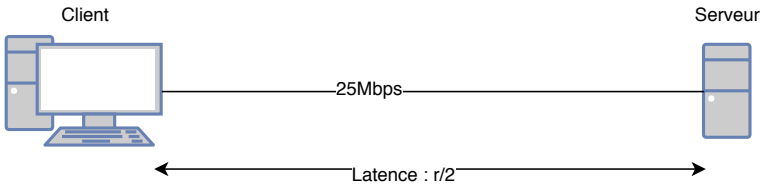


Figure 2.9: Exemple d'une situation simple sans PEP : Un client est connecté à un serveur via un lien de grand RTT r.

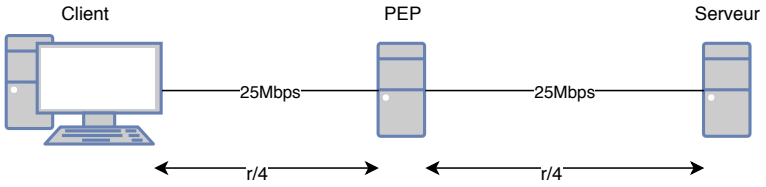


Figure 2.10: Exemple d'une situation simple avec PEP : Un *Performance-Enhancing Proxy* (Mandataire améliorant les performances) (PEP) est placé au centre du chemin entre le serveur et le client.

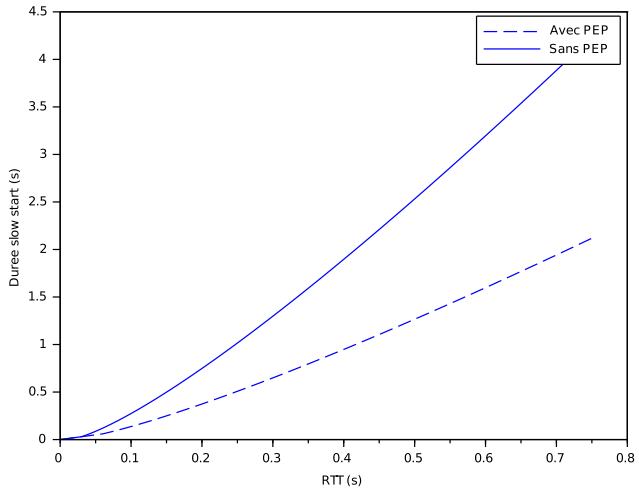


Figure 2.11: Temps nécessaire pour sonder le lien en *slow start* en fonction du RTT, avec ou sans PEP placé à mi-distance. Capacité cible: 25 Mbps. Avec un PEP, on gagne un facteur 2.

les téléchargements sont rapides sans déclencher de congestion. Dans cette section, nous présentons le mécanisme qui permet aux PEPs de réduire la durée de ce *slow start*.

Supposons la situation telle que présentée en Figure 2.9 : un client est connecté à un serveur via un lien de RTT r et de capacité équitable fixée à 25Mbps[17]. Dans cette architecture, la connexion TCP n'est pas coupée. Supposons la fenêtre de congestion initiale $\mathcal{I}$ fixée à 32 segments TCP[18]. La courbe pleine de la figure 2.11 présente la durée nécessaire au serveur pour réaliser son *slow start* en fonction du RTT de la connexion, calculée avec l'Équation 2.1. On observe bien que plus le RTT est long, plus le temps nécessaire pour tester le lien est élevé, avec jusqu'à 3 à 4 secondes lorsqu'on s'approche des RTT typiques d'un lien satellite (Table 2.1).

Ajoutons maintenant un PEP au milieu du chemin entre le serveur et le client (Figure 2.10). On obtient alors deux liens de RTT égal à $r/2$ pour chacun. L'un de ces deux liens contient le goulot d'étranglement et présente donc une capacité équitable de 25Mbps. On peut cependant travailler comme si l'autre lien présentait aussi une capacité équitable de 25Mbps. En effet le PEP ne peut pas supporter de différence de débit entre les deux liens car il ne peut ni envoyer des données qu'il n'a pas reçu ni stocker indéfiniment des données qu'il ne peut envoyer. Son contrôle de flux impose ainsi un débit macroscopiquement égal sur les deux liens.

Dans la situation de la figure 2.10, dès que le PEP reçoit un segment du serveur, il l'acquitte et peut immédiatement utiliser ce même segment pour sonder son propre lien vers le client. Tout se passe comme si serveur et PEP sondaient leur lien respectif indépendamment. Le temps nécessaire pour converger sur le débit de bout en bout devient ainsi le maximum des durées des *slow start* pour le PEP et le serveur[19]. Dans le cas d'un PEP placé à mi-distance, le temps global est présenté par la courbe en pointillés en Figure 2.11. Comme on pouvait s'y attendre avec l'Équation 2.1, le temps nécessaire est à peu près divisé par deux. On dit ainsi que le PEP **accélère** la connexion TCP. L'utilité d'une telle accélération est d'une importance non négligeable pour les liens à gros RTT en particulier les SATCOMs puisqu'on est alors sur des ordres de grandeurs totalement perceptibles par l'homme (avec un gain jusqu'à plusieurs secondes).

Cet exemple simple permet ainsi de comprendre le bénéfice de couper les connexions TCP. Comme expliqué dans la Figure 2.7, les PEP sont en général déployés selon des schémas plus complexes qui permettent de meilleures performances encore.

Notons pour finir que cette accélération est possible car les fonctions de sécurité et de contrôle de congestion sont séparées d'une interface verticale dans la pile traditionnelle TCP/TLS (Figures 1.4 et 2.7). En effet, l'utilisation du chiffrement et des certificats SSL empêche tout intermédiaire, en particulier un PEP, de couper les connexions TLS. Ici, TCP peut être coupé sans couper TLS.

[17] Capacité typique observable sur un lien SATCOM, voir Table 2.1.

[18] Cela correspond aux valeurs typiques des implémentations modernes, comme dans le code source de QUIC sur `Chromium` [19].

[19] Une petite subtilité, ici ignorée, réside dans le fait que le CC du PEP est en retard d'un quart de RTT par rapport à celui du serveur.

Chapter 3

QUIC

Dans ce chapitre, nous introduisons *Quick UDP Internet Connections* (Connexions Internet rapides sur UDP) (QUIC), un protocole de transport en cours de développement.

Grâce au Chapitre 1, nous expliquerons les origines, les raisons et le positionnement de ce protocole vis à vis de la structure actuelle d'Internet et en particulier vis à vis de TCP, le protocole de transport historique.

Le Chapitre 2 a mis en lumière les adaptations du protocole de transport TCP pour le contexte spécifique des *SATtellite COMmunication systems* (Systèmes de communication par satellite) (SATCOMs). Nous aborderons donc les relations entre ces optimisations et QUIC ainsi que les conséquences que peut avoir le déploiement de ce protocole pour l'opérateur SATCOM.

3.1 QUIC, un protocole de transport

QUIC est un protocole de niveau transport. Rappelons-nous (Section 1.2.1) que cela signifie simplement qu'il est intégré dans la couche topologique de transport et qu'il participe donc à transmettre des messages applicatifs d'un *end-point* à l'autre.

Dans cette section, nous commencerons par détailler les objectifs qui ont menés à la création de QUIC, puis nous verrons certaines des caractéristiques principales de QUIC.

3.1.1 Les objectifs de QUIC

L'étape d'identification des besoins est particulièrement importante pour les protocoles engagés dans des SoS du fait de la complexité du contexte (Section 1.1.1). QUIC a pu bénéficier de dizaines d'années de retour d'expérience sur les protocoles TCP et TLS. Ainsi, si QUIC reprend une bonne partie des exigences de ces deux protocoles (transport fiable, contrôle de congestion, confidentialité,...), nous listons ici les objectifs issus de l'expression d'un manque sur TCP/TLS.

Réduire la latence des connexions

La Section 1.4.1 de [38, page 63] rappelle que le délai de transfert d'un paquet dépend du temps de transmission (fonction de la capacité) et du temps de propagation (fonction du RTT).

Dès ses premières lignes dans *QUIC: Design Document and Specification Rationale* [50], le document originel de QUIC (2012), Jim Roskind fait le constat suivant:

"Over time, bandwidth throughout the world will grow, but round trip times, governed by the speed of light, will not diminish. We need a protocol to move requests, responses, and

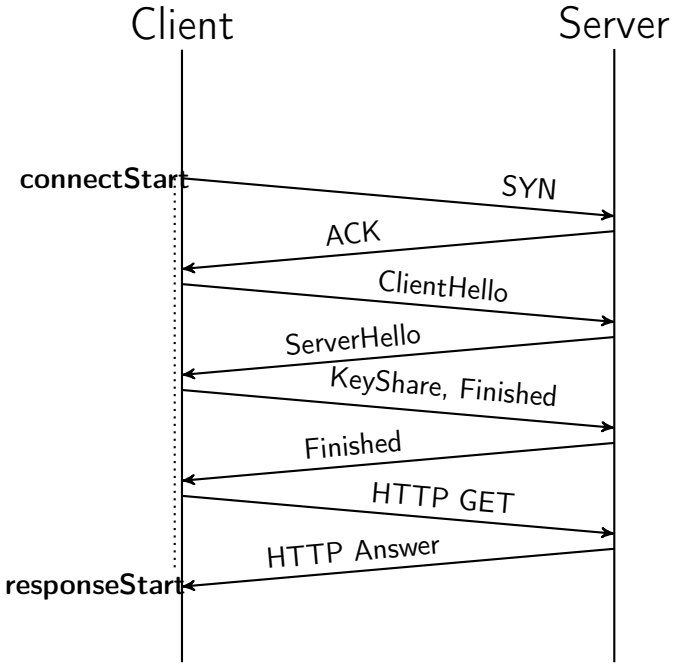


Figure 3.1: Poignée de main TCP/TLS1.2 : `connectStart` représente le début de la connexion au serveur, `responseStart` représente le début de la réponse du serveur. On observe qu'avec TCP/TLS1.2, 4 RTT sont nécessaires entre le moment où l'utilisateur demande la ressource et le moment où elle commence à atteindre son ordinateur.

.

interactions through the internet with less latency along with fewer time-consuming retransmits [...]"

En effet la couche Physique[1] peut réduire le temps de transmission grâce aux évolutions technologiques qui augmentent les capacités. En revanche, seules les couches supérieures (en particulier la couche Transport[2]) peuvent réduire l'influence du temps de propagation sur la latence en réduisant le nombre d'aller-retours nécessaires au traitement d'une requête, chaque RTT ayant un limite basse incompressible (liée à la vitesse de la lumière).

Or l'assemblage des couches TCP et TLS1.2 [14] nécessite un total de 4 RTT entre le moment où le client envoie son premier paquet et le moment où il reçoit les premiers bits de la ressource demandée : 1 RTT pour établir la connexion TCP, 2 RTT pour TLS1.2 et enfin 1 RTT pour la requête/réception de la ressource. Sur un lien SATCOM de latence $\sim$ 750ms, cela représente ainsi 3 secondes !

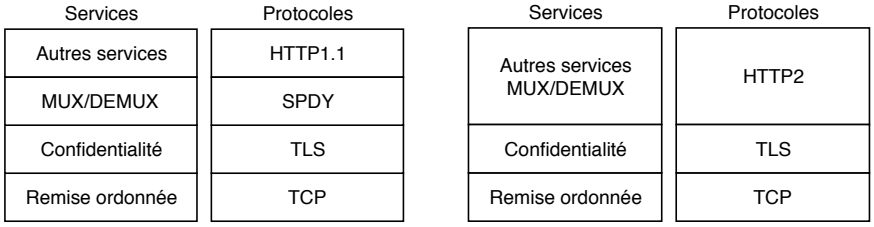
QUIC vise à réduire significativement le nombre de RTT nécessaires pour se connecter à un serveur et y récupérer une ressource.

Multiplexer efficacement les flux

Le multiplexage de plusieurs flux applicatifs dans une même connexion de transport est un besoin poussé principalement par :

- La limitation du nombre de tuples adresse IP/numéro de port.

[1] Couche qui présente, avec "Applications", une évolution rapide.
[2] Plus rigide que la couche Physique.



(a) *Speedy* ("Rapide") (SPDY) multiplexe plusieurs flux HTTP dans une même connexion de transport.

(b) Avec HTTP2, c'est la couche HTTP qui s'occupe directement de multiplexer ses requêtes.

Figure 3.2: Pile protocolaire et services associés pour SPDY et HTTP2.

- La nécessité d'améliorer le chargement des pages web en chargeant plusieurs ressources en parallèle, en permettant au serveur de pousser des ressources en anticipation, etc.

Ainsi, Google a développé à partir de 2009 un protocole nommé *Speedy* ("Rapide") (SPDY) [55] permettant de multiplexer plusieurs requêtes et réponses HTTP dans une même connexion TCP (Figure 3.2a). Ce protocole s'intercalait ainsi entre HTTP et TLS, puis ses principes ont été intégrés dans le nouveau standard HTTP2 [3] (Figure 3.2b).

Cependant, plusieurs problèmes affectent les technologies de SPDY et HTTP2. Nous en présentons ici deux :

Le HOL : Le *Head-Of-Line blocking* (Blocage par tête de flux) (HOL) est un phénomène qui apparaît lorsque, dans une pile de service, le service de remise ordonnée (*in-order-delivery*) est rendu à destination avant le service de démultiplexage. C'est bien ici le cas avec SPDY et HTTP2 (Figures 3.2a et 3.2b) puisque la remise ordonnée est offerte par TCP, en dessous[3] du démultiplexage offert par SPDY ou HTTP2. Avec le HOL, les retards affectant un flux (en particulier en cas de perte/retransmission) peuvent affecter les autres flux, rendant ainsi caduc tout objectif de chargement parallèle et indépendant des ressources d'une page web. L'annexe A présente plus en détails le problème du HOL.

Avec QUIC, l'objectif est de créer dès le début les moyens de multiplexer plusieurs flux en inversant l'ordre entre ces deux services pour ne pas avoir de HOL.

Le désavantage du Contrôleur de Congestion : Rappelons-nous de la section 2.3.3 que lorsque N connexions se partagent une ressource d'une capacité $\mathcal{B}$, l'objectif des CCs est que chaque connexion utilise une part équitable $\mathcal{B}/N$ de cette capacité. Avant l'arrivée de SPDY ou de HTTP2, lorsqu'un navigateur souhaitait charger plusieurs ressources en parallèles, il ouvrait k connexions TCP[4] avec le serveur. Si l'on note n le nombre de connexions utilisant déjà la ressource[5], le navigateur disposait ainsi d'une capacité "équitable" de $\frac{k}{k+n}\mathcal{B}$ pour charger les ressources. Si maintenant ces k flux applicatifs sont regroupés sous une même connexion TCP, le navigateur ne disposera plus que de la capacité $\frac{1}{1+n}\mathcal{B}$ pour charger les mêmes ressources. Ici par exemple on suppose que n ne bouge pas car les navigateurs concurrents ne multiplexent pas leurs flux applicatifs. On voit donc que les empilements (*stacks*)

[3] Donc temporellement avant, au niveau du destinataire.

[4] k pouvant valoir jusqu'à 6 en général.

[5] *I.e.* sans compter celles du navigateur.

utilisant SPDY et HTTP2 sont désavantagés par rapport aux systèmes utilisant des empilements plus vieux.

Comme tous les vieux systèmes ne peuvent pas être mis à jour, l'objectif avec QUIC est d'obtenir un CC plus agressif afin qu'il simule k connexions pour récupérer sa part équitable face à d'autres implémentations. Les paramètres *Additive Increase, Multiplicative Decrease* (Augmentation additive, retrait multiplicatif) (AIMD) du CC TCP étant difficilement sélectionnables par les couches supérieures, il faudra se passer de TCP pour accomplir cet objectif.

Combattre l'ossification d'Internet

La définition de la notion d'ossification d'Internet est complexe à trouver. La plupart des papiers qui la mentionne renvoient à [1] qui décrit une "impasse" d'Internet. L'idée générale de l'ossification est qu'Internet a perdu de sa flexibilité, il devient de plus en plus compliqué de déployer de nouveaux protocoles, en particulier ceux qui concernent le cœur de réseau (IP) ou le transport. Une des sources communément identifiée de cette ossification est la suivante :

"The inability to adapt to new pressures and requirements has led to an increasing number of adhoc workarounds, many of which violate the Internet's canonical architecture."[1]

Cette violation de l'architecture Internet renvoie au non-respect du modèle en couches, en particulier par les *middle-box* (intermédiaires de couches hautes) que nous mentionnions en Section 1.2.4 ou encore par les PEPs mentionnés en Section 2.3.3. Ils utilisent des champs du protocole de transport pour effectuer des mesures passives ou actives, voir prendre des décisions de *Quality of Service* (Qualité de service) (QoS) ou de sécurité (décision de *drop* un paquet s'il présente un risque de sécurité). Ces mécanismes implémentés par les ISPs interfèrent avec le déploiement de nouveaux protocoles de bout en bout, rendant Internet rigide.

Pour QUIC, l'objectif est double :

- D'abord ne pas subir l'ossification afin d'assurer un large déploiement du protocole.
- Une fois déployé, empêcher les *middle-box* de solidifier QUIC.

Pour le second point, Roskind note dans [50] :

"Experience with SPDY development has taught us that the only way to prevent middleboxes from maligning a new protocol built atop UDP or TCP (e.g., misconstruing it for a "known" protocol, and making "less than helpful" changes), is to encrypt as much of the payload and control structure as feasible."

Notons par ailleurs le champ lexical qui traduit bien la position des développeurs de QUIC (poussé par des entreprises OTT) vis à vis de ces intermédiaires de couches hautes (mis en œuvre par les ISP). Cela souligne, si besoin est, les tensions possibles entre les acteurs d'Internet.

Assurer un large déploiement du protocole

Cet objectif est principal dans la conception de QUIC. En section 1.1.1 nous indiquions que pour qu'un protocole soit déployé, il faut que acteurs nécessaires à son opération l'implémentent et que les acteurs ayant le pouvoir d'en empêcher l'opération ne le bloquent pas.

QUIC étant un protocole de transport, ce double objectif se résume donc à :

Table 3.1: Objectifs de QUIC et solutions.

Objectif	Solution(s)
Réduire la latence des connexions	Échange simultané des paramètres de transport et de sécurité Poignées de main 0RTT
Multiplexer efficacement les flux	Frames de flux
Combattre l'ossification	Chiffrement étendu
Assurer le déploiement	Implémentation au-dessus de UDP
Permettre l'évolution	Implémentation en *user space* Négociation de version

- Permettre un déploiement rapide et facile au niveau des *endpoints*.
- S'assurer le non-blocage par les intermédiaires de réseau.

Dans [50] les objectifs suivants étaient relevés :

"*Widespread deployability in today's internet (i.e., makes it through middle-boxes; runs on common user client machines without kernel changes, or elevated privileges)*"

[...]

"*The number one goal of viability today is clearly a major driver for this protocol development. With the understanding that middleboxes and firewalls would typically block or dramatically degrade any transport based on formats other than TCP or UDP, we will not even consider revolutionary protocols.*"

On retrouve bien cet objectif double : déploiement simplifié sans privilèges particuliers sur le *end-point* et encapsulation au dessus de TCP ou *User Datagram Protocol* (Protocole de datagramme utilisateur) (UDP) pour assurer le passage dans le cœur de réseau.

Permettre l'évolution du protocole

Au fur et à mesure que nous parcourons ses objectifs, on s'aperçoit bien que le design de QUIC se base sur des retours d'expérience.

Il est donc normal de s'apercevoir que dans son esprit initial, QUIC est présenté comme un protocole expérimental, appelé à évoluer en fonction des expériences rencontrées. L'évolutivité du protocole est ainsi un objectif principal, basé aussi sur l'expérience de la rigidité de TCP.

3.1.2 Principales caractéristiques de QUIC

La section précédente introduisait les objectifs de QUIC et leur justification. Ils sont repris en Table 3.1 et nous permettent ainsi de plonger plus en détails dans les caractéristiques principales de QUIC.

Nous précisons que, sauf mention contraire, les propriétés détaillées ci-après présentent un caractère général. D'après notre compréhension de l'évolution de QUIC, elles devraient être conservées dans les versions futures. Elles ne sont en revanche pas nécessairement standardisées comme invariantes. Pour être consistant, nous citons volontairement peu de sources de spécifications mais le lecteur pourra se reporter à la Section 3.2 qui explique les travaux de standardisation et renvoie aux documents en cours de production.

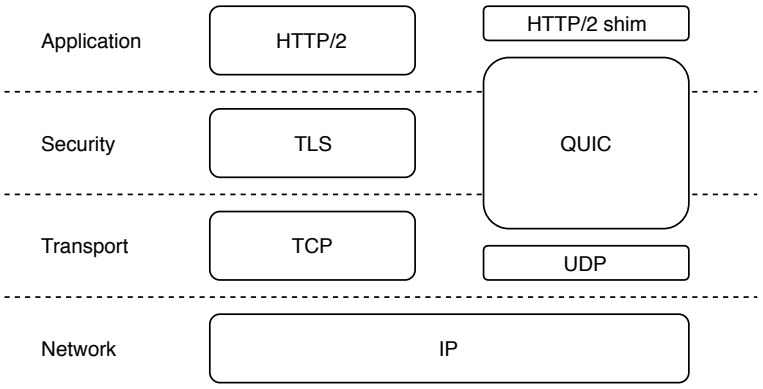


Figure 3.3: Pile QUIC comparée à la pile HTTPS : *Quick UDP Internet Connections* (Connexions Internet rapides sur UDP) (QUIC) est implémenté au dessus de UDP et se présente comme un protocole de transport unifiant les services de fiabilité (TCP), de sécurité (TLS) et de multiplexage (HTTP2). `Ré-éditée d'après la Figure 1 de [41].`

Positionnement protocolaire

La Figure 3.3 compare l'empilement imaginé par les concepteurs de QUIC [41] avec la traditionnelle pile HTTPS.

On observe que QUIC est implémenté au-dessus de UDP. Bien qu'UDP n'apporte que peu de service à la pile, ce choix permet de traverser la majorité des *middle-boxes* comme voulu (Table 3.1). L'association de UDP avec QUIC peut être comparée à TCP puisque QUIC apporte la majorité des services de transports comparables à TCP.

QUIC remplace ensuite entièrement TLS. En fait, le protocole utilisait initialement sa propre suite cryptographique[6]. Avec la sortie récente de TLS1.3 [49], les nouvelles versions de QUIC interagissent intimement avec TLS1.3 pour gérer les aspects de sécurité.

Enfin, on remarque que QUIC accomplit une partie des services normalement à la charge de HTTP2. C'est par exemple le cas pour le multiplexage des flux : avant géré par HTTP2, il sera implémenté par QUIC pour corriger les problèmes identifiés en Section 3.1.1. C'est donc une version "allégée" de HTTP2 qui est présente au-dessus de QUIC.

Finissons avec la Figure 3.3 par quelques remarques sur les choix d'association de QUIC avec les couches topologiques par les auteurs de [41]. Notons qu'en fait ils reprennent l'association traditionnellement réalisée pour HTTP sécurisé (*i.e.*au dessus de SSL, TLS ou QUIC) (HTTPS). Ainsi, comme on l'a déjà dit, la sécurité n'est pas vraiment une couche topologique mais un service. De plus, bien qu'il implémente des services de HTTP2, QUIC n'est pas un protocole de la couche "Application", mais bien un protocole de transport.

Constitution d'un paquet QUIC

Pour comprendre les concepts de QUIC, plongeons-nous dans la constitution d'un paquet.

Supposons par exemple qu'un client a ouvert une connexion QUIC avec un serveur et que 3 flux applicatifs souhaitent utiliser cette connexion du serveur vers

[6]QUIC Crypto, [40].

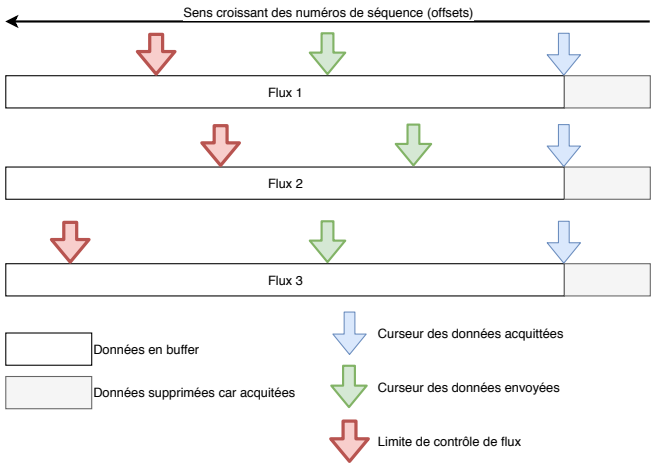


Figure 3.4: Exemple de trois flux à multiplexer dans une même connexion QUIC : Chaque flux possède son propre contrôle de flux.

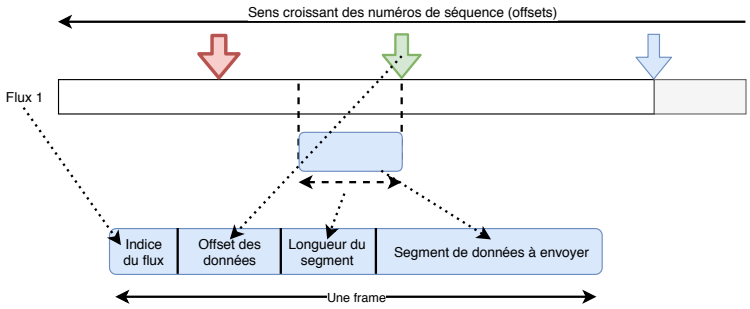


Figure 3.5: Création d'une frame de flux QUIC : Un segment de données du flux est extrait. Sa longueur, son offset et son indice de flux sont associés pour former une frame de flux.

le client (Figure 3.4). Nous représentons chaque flux sous la forme d'une bande de donnée continue dans le temps. Remarquons que chaque flux peut être assimilé à un flux TCP indépendant : ils possèdent tous des curseurs d'offset pour le dernier bit acquitté, le dernier envoyé ainsi qu'un contrôle de flux indépendant. Cette indépendance permet leur multiplexage sans *Head-Of-Line blocking* (Blocage par tête de flux).

Prenons le Flux 1 par exemple (Figure 3.5). Comme pour TCP, le contrôleur QUIC va extraire un segment de données parmi celles non encore envoyées tout en respectant la limite de flux. Toujours comme pour TCP, il va y associer la valeur de l'offset et de la longueur du segment. Sauf qu'ici, puisqu'on multiplexe plusieurs flux, on ajoute aussi l'index du flux. Cet ensemble élémentaire est alors appelé "frame".

Il y existe deux types de frames : les frames de flux, que l'on vient de voir et les frames spéciales. Elle sont dans un format spécifique et permettent de contrôler la connexion, en échangeant des paramètres. Parmi les frames spéciales on peut citer par exemple :

- Les frames d'acquittement : Elles informent de la réception de paquets.
- Les frames de contrôle de flux : Elles informent de la progression (indépendante) des curseur de limite de flux.

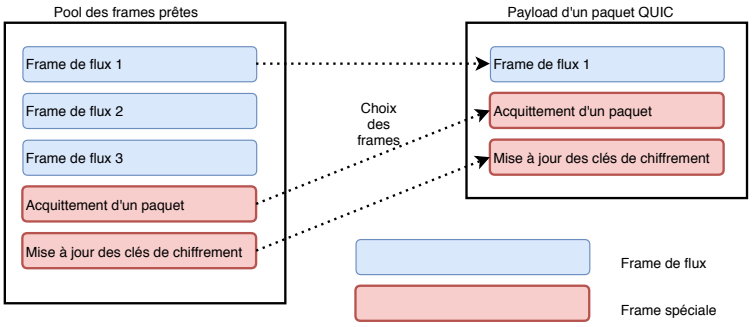


Figure 3.6: Sélection des frames pour constituer un paquet QUIC : Parmi les frames de flux et les frames spéciales disponibles, le contrôleur en choisit un ensemble pour constituer la charge utile d'un paquet QUIC.

- Les frames cryptographiques : Elles échangent des informations sur l'environement cryptographique de la connexion (changement de clés, etc.).

Le contrôleur QUIC dispose ainsi d'une piscine (*pool*) de frames prêtes à l'envoi (Figure 3.6). Il va alors les sélectionner pour former des paquets en se basant :

- Sur des règles de priorités qui privilégient par exemple les frames spéciales aux frames de flux.
- Sur un système de contrôle de flux global, distinct du contrôle par flux, qui évite le débordement du buffer aggrégé UDP.
- Et sur le contrôle de congestion qui opère au niveau du paquet.

La charge utile (*payload*) d'un paquet QUIC est ainsi un ensemble de frames.
On ajoute ensuite à cet ensemble l'en-tête (*header*) QUIC (Figure 3.7). D'une manière simplifiée, celui-ci est constitué :

- D'un numéro de paquet unique qui ne sera jamais réutilisé au cours d'une connexion. Ce numéro ne sert qu'au contrôle au niveau paquet/connexion, il n'est pas utilisé par le niveau frame. En particulier ce n'est pas un offset des segments de données, son incrément est donc de 1 entre deux paquets[7]. Dans les nouvelles versions de QUIC, il commence à 0.
- D'un identifiant de connexion. Il permet d'identifier la connexion sans dépendre du tuple adresse ip/port, ce qui permet ainsi à la connexion de changer de chemin.
- D'un certain nombre de flags permettant l'interprétation du header.

Une fois le paquet composé, il subit un chiffrement dit *Authenticated Encryption with Associated Data* (Chiffrement authentifié avec données associées) (AEAD) en utilisant les clés échangées lors de la poignée de main. La charge utile (*payload*) est chiffrée puis une signature de l'ensemble du paquet, y compris le header (données associées) est calculée à l'aide de la clé de chiffrement (Figure 3.7).

Ainsi, le contenu de la charge utile est confidentiel et modifier un bit du paquet dans la *payload* ou dans le *header* invalide la signature ce qui mène au rejet du paquet côté client.

[7] En fait l'incrément peut être supérieur à 1 pour des raisons de sécurité. Voir Section 12.3 de [32, page 111].

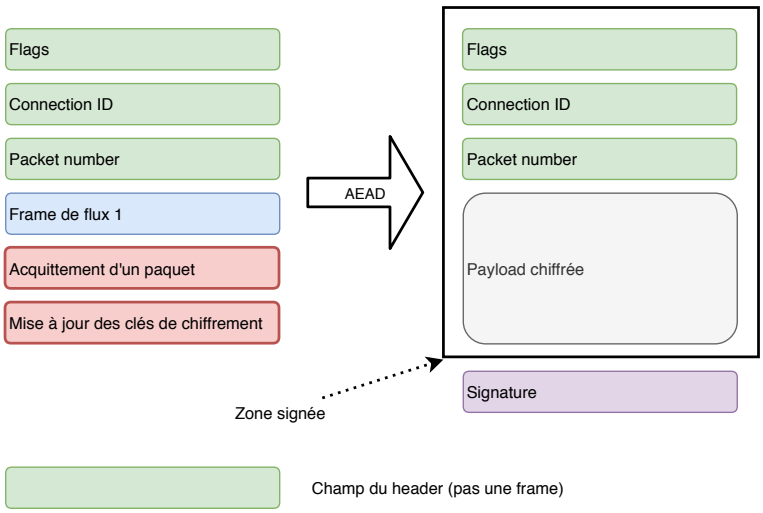


Figure 3.7: Ajout de l'entête QUIC et chiffrement : les champs de *header* sont ajoutés, la charge utile est chiffré et le paquet entier avec l'en-tête est signé.

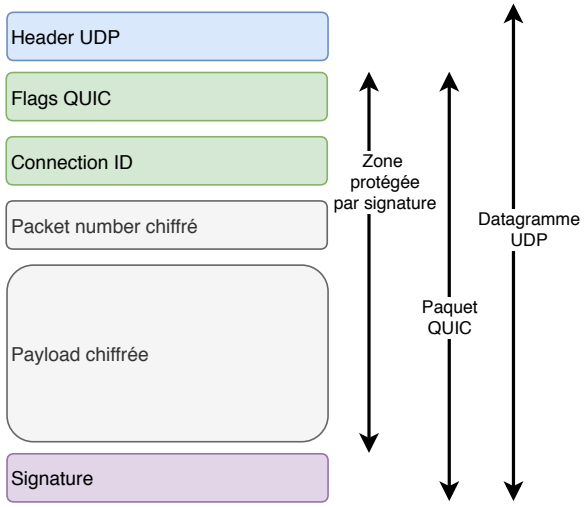


Figure 3.8: Résumé de la constitution d'un paquet QUIC avec le numéro de paquet chiffré.

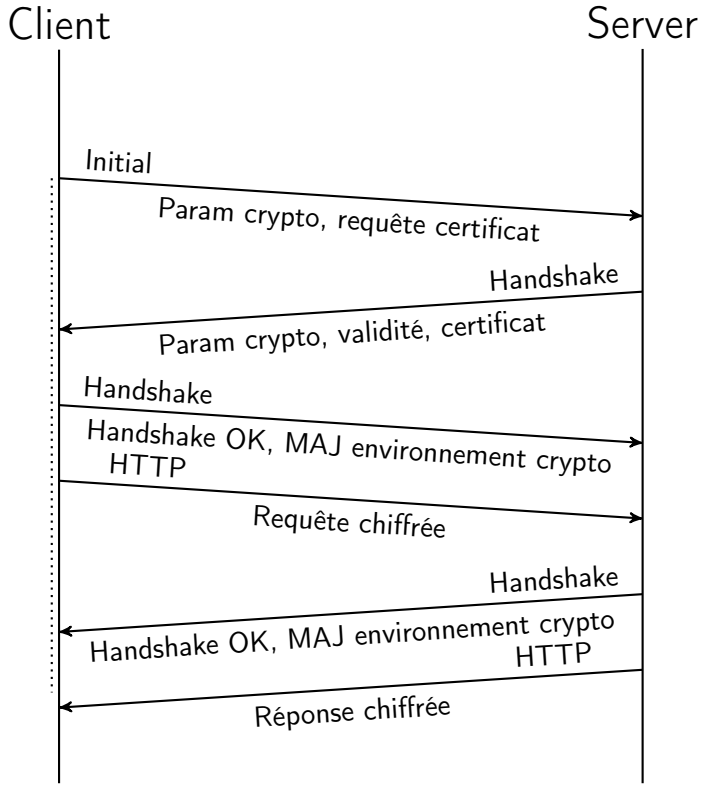


Figure 3.9: Poignée de main QUIC 1RTT : La première fois que le client se connecte au serveur, l'établissement de la connexion nécessite 1RTT, la réponse à la requête est donc reçue en 2RTT. Le client en profite pour enregistrer les paramètres cryptographiques du serveur.

Enfin, le numéro de paquet est lui-même chiffré [56] puis l'en-tête UDP est ajouté.

On obtient finalement le paquet présenté Figure 3.8 dont toute la *payload* est chiffrée et dont une majeure partie est signée.

Si un paquet est déclaré perdu, le contrôleur QUIC liste les frames qu'il contennait et elles sont alors déclarées perdues. Chaque frame perdue est étudiée par le contrôleur QUIC selon des règles dépendant de son type, il peut alors :

- Remettre la frame dans le *pool* des frames prêtes à l'envoi. Elle sera alors renvoyée avec d'autres frames dans un nouveau paquet ayant un nouveau numéro.
- Mettre dans la pool une frame modifiée. Cela peut arriver s'il s'agit d'une frame spéciale et que le contexte a changé depuis son envoi par exemple.
- Il peut enfin simplement ignorer la frame si les données qu'elle contenait sont considérées obsolètes.

Poignée de main

L'établissement de la connexion a subi de nombreuses modifications au cours de la conception de QUIC, nous proposons une version simplifiée permettant de mettre en relief la notion de connexion 0RTT.

Poignée de main en 1RTT: La Figure 3.9 présente le principe de la poignée de main lorsque le client n'a jamais contacté le serveur. Le client envoie un paquet

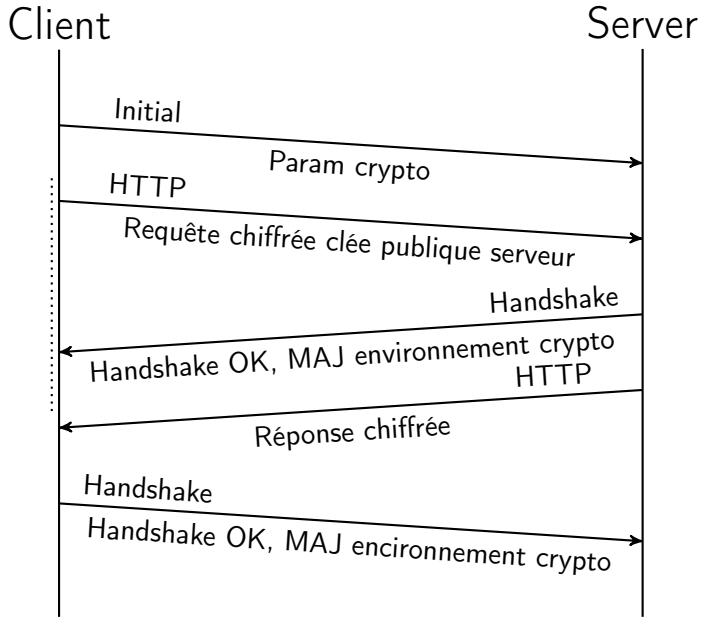


Figure 3.10: Poignée de main QUIC 0RTT : Le client ré-utilise les données cryptographiques du serveur pour envoyer directement sa requête HTTP. On peut alors dire que la connexion est réalisée "en 0RTT" et la réponse HTTP est reçue en 1RTT.

d'initiation de connexion qui contient les paramètres cryptographiques qu'il aura choisi[8], les paramètres de transport (taille initiale de fenêtre) et la requête du certificat. Le serveur répond par ses propres paramètres cryptographiques en précisant leur date limite de validité puis il fournit son certificat TLS.

Une fois que le client a validé le certificat, il stocke les paramètres cryptographiques du serveur et calcule le secret cryptographique de l'échange. Il informe le serveur de la validation de la poignée de main et de la mise à jour de l'environnement des clés des chiffrement. Il envoie en parallèle sa requête HTTP. Le serveur valide de son côté la poignée de main et met à jour son environnement cryptographique. Il en informe le client et répond à sa requête.

Poignée de main en 0RTT: Une fois que le client a récupéré les paramètres cryptographiques du serveur, il est capable de réaliser des poignées de main en "0RTT" (Figure 3.10).

Ici le client envoie un paquet d'initiation de connexion en fournissant de même ses paramètres cryptographiques et de transport. Cependant, comme il a déjà pu valider le certificat du serveur et qu'il en connaît la clé publique, il peut directement envoyer une requête HTTP "0RTT". Cette requête n'est pas chiffrée par les clés de chiffrement de l'échange qui seront générées plus tard, elle est donc un peu moins sécurisée que le reste des messages[9]. Le serveur valide la poignée de main, met à

[8]En particulier le *random seed*, la graine aléatoire qui sera à la base de la constitution des clés. Pour plus d'information sur les échanges de clés, le chiffrement, voir [56].

[9]Parmi les propriétés perdues par rapport aux autres messages, il y a :

- La non-répétabilité: Un attaquant qui intercepte ce message 0RTT peut le rejouer en son nom, sans cependant savoir ce qu'il contient.
- La confidentialité persistante (*Forward security*). Un attaquant ayant enregistré la conversation qui parvient à récupérer dans le futur la clé privée du serveur pourra déchiffrer tous les messages 0RTT du passé, alors que ce n'est pas possible pour le reste des messages.

jour son environnement de chiffrement et réponds à la requête avec les nouvelles clés. Enfin, le client valide l'échange de son côté.

Nuançons pour finir les bénéfices de la poignée de main 0RTT puisque des mécanismes, à l'initiative du serveur, de prévention des attaques *Distributed Denial Of Service* (Attaque distribuée par deni de service) (DDOS) peuvent drastiquement limiter le gain en temps de la poignée de main.

Considérations de sécurité

Justement, QUIC a pu bénéficier de l'expérience de nombreux experts en sécurité informatique, discipline plus développée qu'à l'époque de la création de TCP. En parallèle de l'usage des techniques de chiffrement AEAD et de son intégration de TLS1.3, le protocole contient des mesures :

Contre les attaques DDOS : QUIC utilise des mécanismes de preuve de possession de l'adresse IP par le client, semblables aux TCP SYN Cookies [15, 52]. Ici cependant ces mécanismes présentent plusieurs niveaux de confiance qui peuvent s'adapter à la charge du serveur (voir [32, Section 6.9]). QUIC impose aussi que les paquets initiateurs de connexion soient rejetés si leur taille est trop faible, limitant ainsi les attaques par amplification [27, Section 3] et [32, Section 8.3].

Contre la traçabilité des connexions : Les connexions sont rendues difficilement traçables grâce à l'usage des identifiants de connexion qui résistent au changement de tuple adresse ip/port. De plus, une fois l'environnement de chiffrement mis à jour, les *end-points* échangent une liste d'identifiants de connexion alternatifs qu'ils peuvent utiliser sans refaire de poignée de main. Modifier cet identifiant en même temps que changer de réseau rend ainsi complexe la traçabilité de l'échange pour un observateur externe, tout en rendant l'opération transparente pour les *end-points* [32, Section 6.11.5].

Conséquences pour les SATCOM

Quelles sont donc les conséquences pour les *Performance-Enhancing Proxies* (Mandataires améliorant les performances) (PEPs) de cette association, dans un même protocole, sans découpage vertical, des fonctions de transport fiable, sécurisé, multiplexant des flux ? Quelles sont les conséquences pour les opérateurs de cet usage étendu de l'AEAD ?

Nous indiquions en Section 2.3.3 que l'accélération des connexions TCP par des proxys transparents repose sur la capacité, pour ces middleboxes, à générer des acquittements au nom du serveur (en se "faisant passer pour le serveur"). Avec QUIC, les acquittements font partie de la charge utile chiffrée du paquet. Ce chiffrement s'effectue avec les clés de session générées lors de la poignée de main au cours de laquelle une authentification du serveur est réalisée à l'aide des certificats SSL.

Ainsi, le PEP étant géré par l'ISP et non par l'OTT, il ne possède ni la clé privée du serveur, ni de certificat au nom du serveur. Le PEP ne peut ni lire ni générer d'acquittement. **Une connexion QUIC n'est pas découpable**.

Au delà de cette conséquence pour les SATCOMs, l'AEAD complique la gestion du réseau par les ISP, y compris terrestres : ils ne peuvent par exemple pas se baser sur des mesures passives du numéro de paquet (chiffré), ils ne peuvent procéder à des mesures actives sur l'en-tête du paquet (signé), etc. [17].

Table 3.2: Comparaison de Google QUIC (GQUIC) et IETF QUIC.

Version	GQUIC	IETF QUIC
Début conception	2012	2016
Conception par	Google	QUIC WG de IETF
Statut	Protocole Expérimental	Internet Draft
Conception	Sur la fin	En cours
Tests	Grande échelle	Compatibilité de frameworks
Déploiement (2018-10)	Oui, via Chrome	Non
Spécifications accessibles	Partiellement	Oui

3.2 Déploiement et versions de QUIC

En section précédente, nous indiquions présenter uniquement des caractéristiques générales de QUIC. En effet, le protocole est toujours en développement selon un rythme très intense qui amène à de nombreuses modifications/évolutions des spécifications. Cette partie se propose donc de présenter la manière dont cette technologie est créée, testée et déployée.

Quand on parle de QUIC, il faut d'abord savoir qu'il existe deux "grandes versions", deux macro-projets :

- Google QUIC (GQUIC) est la version originale du protocole QUIC, expérimentale, créée par Google en 2012.
- IETF QUIC est un effort de l'IETF pour standardiser le travail de Google.

Les parties suivantes présentent un aperçu de ces deux versions et la Table 3.2 compare quelques points importants.

3.2.1 GQUIC

QUIC est un protocole initialement inventé par les équipes de Google et décrit dans [50, 41]. Une partie de ses spécifications est accessible sur le site des `Chromium Projects` [23, 40] et dans le code source du navigateur libre `Chromium` [19].

GQUIC a ainsi été développé comme un protocole expérimental incrémental, basé sur des retours d'expérience de tests à grande échelle. Comment Google a-t-il pu opérer ces tests à grande échelle ?

La Figure 3.11 montre que la force de l'entreprise réside dans son contrôle des éléments du SoS Internet nécessaires à l'opération du protocole[10]. D'un côté Google contrôle le navigateur Chrome, largement déployé. Et de l'autre côté, Google contrôle les serveurs hébergeant ses services. Ajoutons que la partie Client de GQUIC est intégrée à Chrome et opère dans le *user space*, elle peut donc facilement se mettre à jour. De plus, GQUIC fonctionne au dessus de UDP, son taux de blocage par les cœurs de réseau est donc limité. Tout ces aspects sont autant d'ingrédients d'un déploiement rapide de GQUIC puisque tout utilisateur de Chrome se connectant à des services Google est susceptible d'utiliser cette technologie.

Une étude exhaustive de ce déploiement est fournie par [35]. La Figure 3.12 présente ainsi l'évolution du nombre de *hosts* dans l'espace IPv4 supportant GQUIC. L'explosion de ce nombre en avril 2018 montre la capacité de Google et Akamai à mettre à jour rapidement leur infrastructure.

[10] Voir section 1.1.1

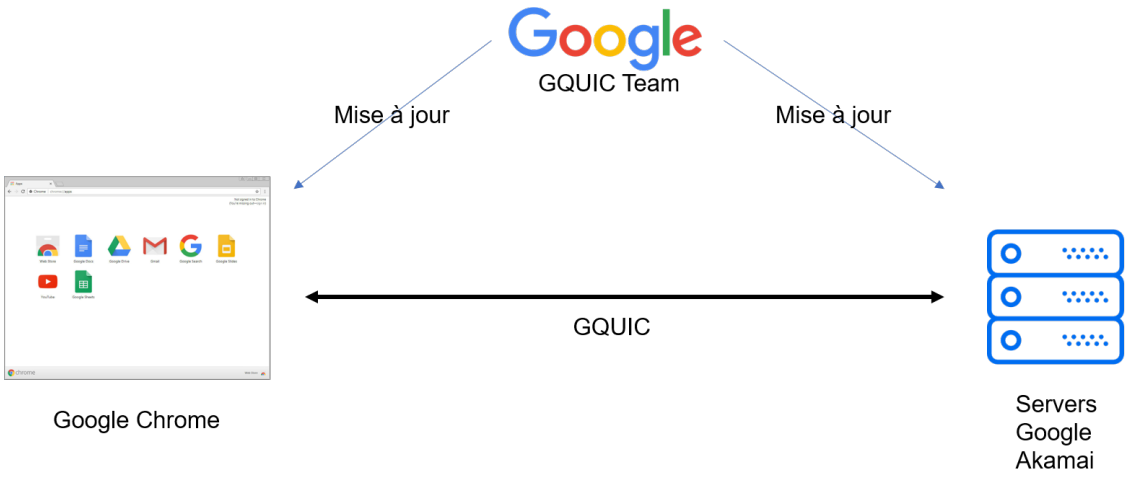


Figure 3.11: Principe du déploiement de Google QUIC (GQUIC) par Google : Une mise à jour du navigateur Chrome et des serveurs Google permet l'usage de GQUIC par tout utilisateur de Chrome se connectant aux services Google.

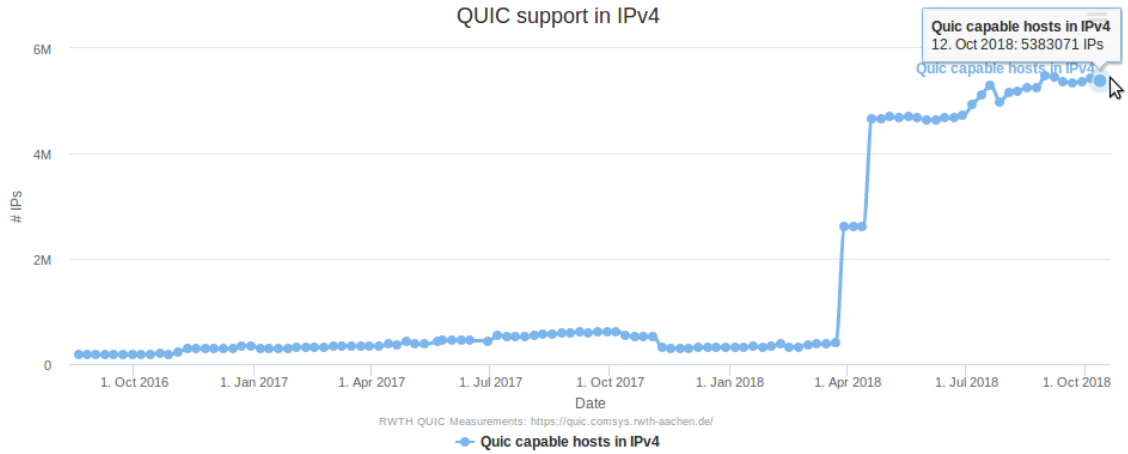


Figure 3.12: Évolution du nombre d'hosts compatibles GQUIC dans l'espace IPv4 : On note aujourd'hui 5.3 millions de hosts et une explosion en avril 2018. A titre de comparaison, mon projet de fin d'étude a commencé le 3 mai 2018. Source : `quic.comsys.rwth-aachen.de`, par les auteurs de [35]. Accédée le 17/10/218.

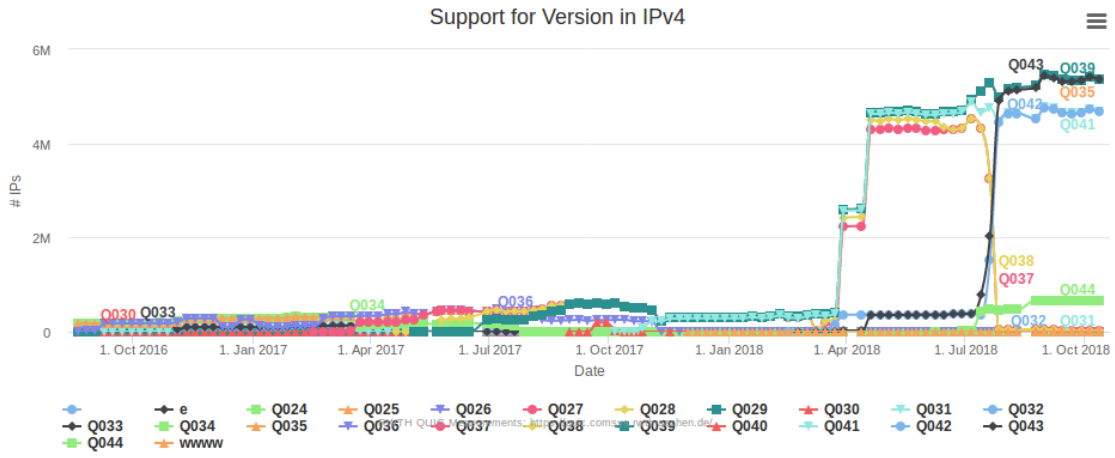


Figure 3.13: Évolution du nombre d'hosts supportant les différentes versions de GQUIC au cours du temps : On met en relation l'explosion d'avril 2018 avec celle observée en Figure 3.12 mais on s'aperçoit ici que Google et Akamai sont capables de rapidement mettre à jour les versions de leurs serveurs, avec le déploiement très rapide de Q043 et la suppression de Q037 et Q038 en juillet 2018. Source : quic.comsys.rwth-aachen.de, par les auteurs de [35]. Accédée le 17/10/218.

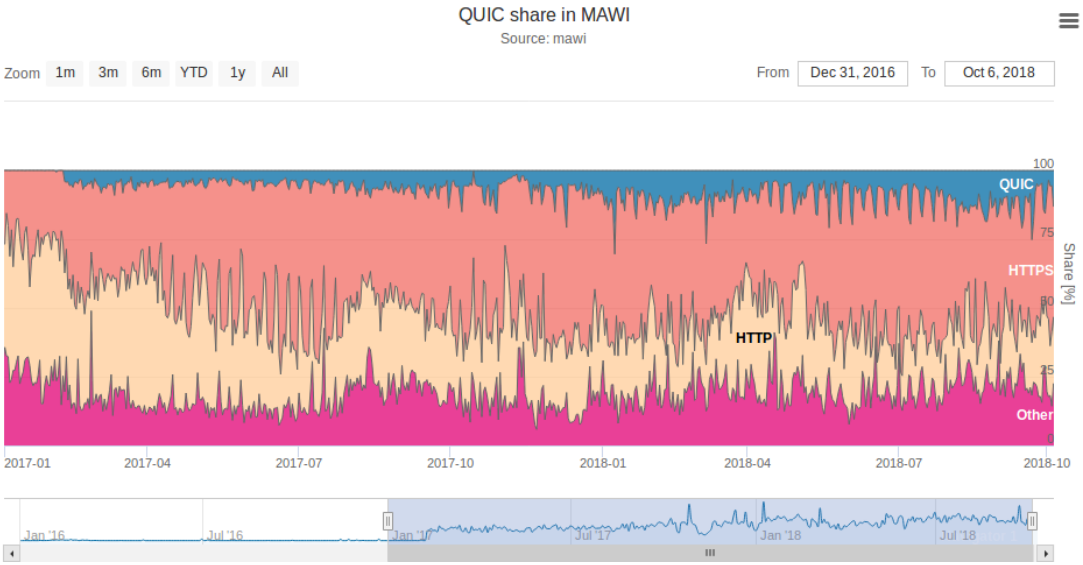


Figure 3.14: Évolution de la répartition du trafic en fonction du protocole dans un ISP japonnais : La part de QUIC augmente progressivement avec des pics à 30% du trafic total. Source : quic.comsys.rwth-aachen.de, par les auteurs de [35]. Accédée le 17/10/218.

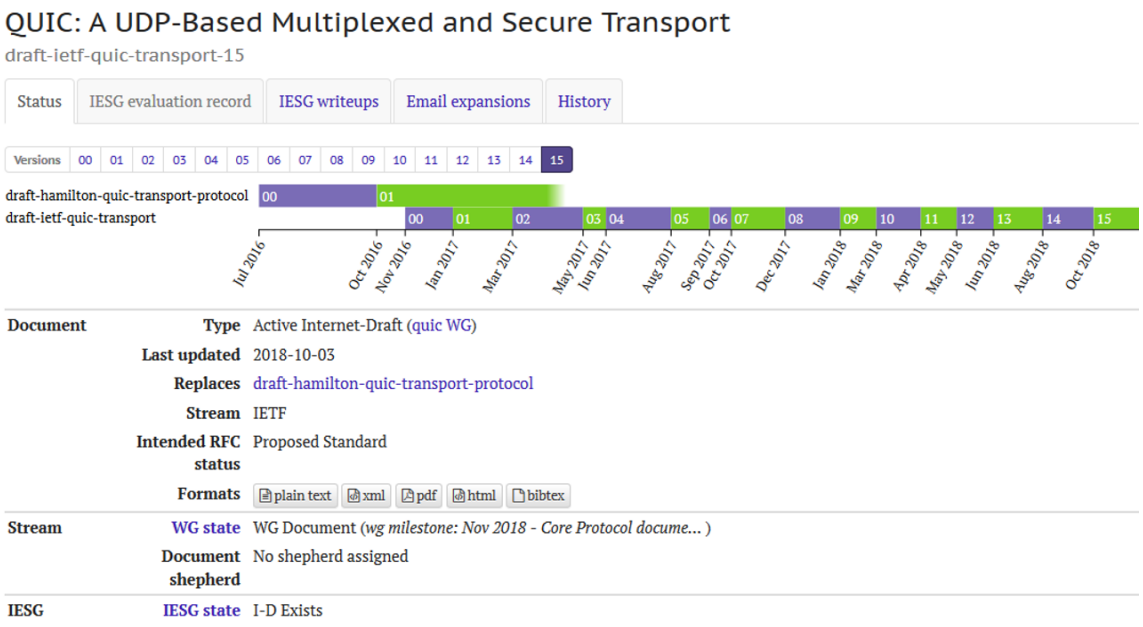


Figure 3.15: Frise chronologique des différents *drafts* de l'IETF QUIC: Le WG publie des brouillons du standard de manière de plus en plus régulière. `Source : datatracker.ietf.org. Accédée le 04/10/218.`

GQUIC est un protocole expérimental, les équipes de conception ont donc pu développer le protocole de manière incrémentale en déployant, testant sur une grande échelle et agrégeant des résultats de performances. Ainsi les versions de GQUIC sont notées QXXX. La figure 3.13 présente l'évolution du support des versions. On peut ainsi apprécier le grand nombre de versions différentes. De plus, de manière complémentaire à la Figure 3.12, on note le déploiement rapide de Q043 en juillet 2018.

Pour finir notre prise de conscience de l'intensité du déploiement, la Figure 3.14 présente la répartition de la part du trafic en fonction du protocole dans un lien reliant un ISP de tiers 1 (ie un ISP global) et un ISP régional japonnais[11]. On note que sur les deux dernier mois, la part de GQUIC oscillait entre 5% et 28%.

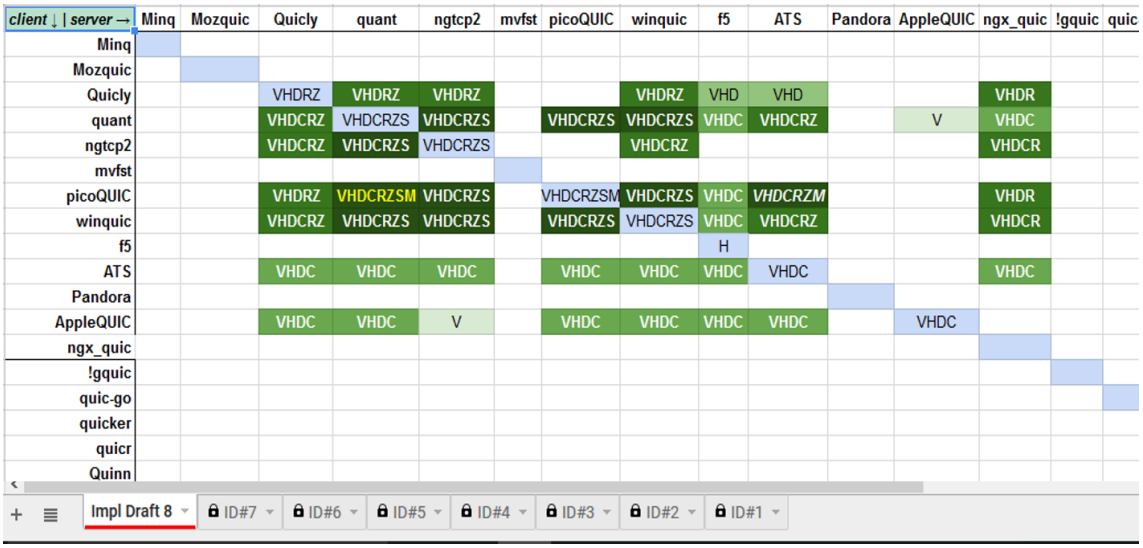
3.2.2 IETF QUIC

Avec IETF QUIC (dans la suite simplement appelé "QUIC"), les objectifs sont tout autres. Pas de déploiement ni de tests grandeur nature mais au contraire un travail de rédaction d'un standard internet en se basant sur les expériences menées par GQUIC.

Comme abordé en Section 1.1.1, un *Working Group* (Groupe de travail) est donc chargé de rédiger ce standard. L'objectif est ici de soumettre la première version (V1) de IETF QUIC à l'IESG[12] en Novembre 2018 [30].

Pour cela, le WG traite de manière incrémentale un certain nombre de sujets et crée des versions intermédiaires, des brouillons (*drafts*). La Figure 3.15 présente l'évolution de ces différents *drafts*. On observe une fréquence assez soutenue malgré un temps de développement long. A titre d'exemple, dans le cadre du Projet de Fin d'Etudes (PFE), le travail d'analyse documentaire du protocole a principalement

[11] Pour plus d'informations, voir la section "samplepoint F" de [44].
[12] Voir le processus de production d'un standard Internet, Section 1.1.1



client ↓ | server →	Minq	Mozquic	Quicly	quant	ngtcp2	mvfst	picoQUIC	winquic	f5	ATS	Pandora	AppleQUIC	ngx_quic	!gquic	quic
Minq															
Mozquic															
Quicly			VHDRZ	VHDRZ	VHDRZ			VHDRZ	VHD	VHD			VHDR		
quant			VHDCRZ	VHDCRZS	VHDCRZS		VHDCRZS	VHDCRZS	VHDC	VHDCRZ		V	VHDC		
ngtcp2			VHDCRZ	VHDCRZS	VHDCRZS			VHDCRZ					VHDCR		
mvfst															
picoQUIC			VHDRZ	VHDCRZSM	VHDCRZS		VHDCRZSM	VHDCRZS	VHDC	*VHDCRZM*			VHDR		
winquic			VHDCRZ	VHDCRZS	VHDCRZS		VHDCRZS	VHDCRZS	VHDC	VHDCRZ			VHDCR		
f5									H						
ATS			VHDC	VHDC	VHDC		VHDC	VHDC	VHDC	VHDC			VHDC		
Pandora															
AppleQUIC			VHDC	VHDC	V		VHDC	VHDC	VHDC	VHDC		VHDC			
ngx_quic															
!gquic															
quic-go															
quicker															
quicr															
Quinn															

Impl Draft 8 | ID#7 | ID#6 | ID#5 | ID#4 | ID#3 | ID#2 | ID#1

Figure 3.16: Matrice d'interopérabilité des implémentations des *drafts* de l'IETF QUIC : Est présentée ici la huitième campagne de tests. Les lettres dans chaque case définissent la validation d'un certain nombre de tests. On observe que parmi le nombre élevé d'implémentations différentes, environ 5 sont régulièrement mises à jour vers les nouveaux *drafts* et passent ainsi les tests. `Source : github.com. Accédée le 03/10/218.`

été réalisé sur les versions *draft*-09 et *draft*-11, puis une veille[13] a simplement été maintenue pour capter les évolutions majeures susceptibles d'influencer l'étude[14].

Enfin, au fur et à mesure de la rédaction du standard, plusieurs participants au WG implémentent des *frameworks* pour tester la technologie, identifier les améliorations et les situations indéfinies. Ce fonctionnement est assez puissant puisque l'interopérabilité des *frameworks* est régulièrement testée ce qui permet de vérifier la justesse et la complétude du standard. La Figure 3.16 montre les résultats de ces tests d'interopérabilité pour la dernière (à la date indiquée) série d'intégration. Dans mon PFE, en anticipation de travaux sur des *frameworks* IETF QUIC finalement non menés, j'ai réalisé un *trade-off* de ces différentes solutions pour en choisir une qui présente la plus forte probabilité de devenir une référence à l'avenir.

[13] Principalement en suivant la *mailing list* du WG, les *Pull Requests* (Requêtes d'intégration) (PRs) et les *issues* postées sur le `git` [30].

[14] Par exemple, le chiffrement du numéro de paquet a été acté au cours de mon PFE et il influe beaucoup sur les capacités de l'opérateur à opérer son réseau.

Chapter 4

Mise en place des expériences

Dans les parties précédentes, nous avons vu comment s'organisait Internet en général et comment le SATCOM s'y intégrait. Nous avons ensuite vu que QUIC, tout en poursuivant des objectifs d'amélioration d'Internet en général, venait chambouler les principes d'opération des ISP SATCOM en raison de son usage étendu du chiffrement. Enfin, nous avons constaté que QUIC est encore en développement auprès de l'IETF, dans un processus assez long. Néanmoins, son déploiement est déjà effectif dans sa version GQUIC, avec une part du trafic Internet global en augmentation et flirtant avec les 20%. De plus, les rapides évolutions des standards et versions ne facilitent pas le suivi de la technologie par les opérateurs.

C'est typiquement dans ce genre de situation que le CNES peut apporter son expertise et ses moyens au service du déploiement des SATCOMs.

Dans ce chapitre, nous aborderons donc les principaux axes de l'étude que nous avons décidée de mener. Nous en justifierons l'existence et les principes puis nous détaillerons la réalisation du banc expérimental et les métriques relevées.

4.1 L'étude

Suivant les objectifs du CNES d'assistance aux opérateurs SATCOM, nous décidons donc de nous poser la question des conséquences de QUIC du point de vue SATCOM.

4.1.1 Pourquoi faire une étude sur QUIC ?

Tout laisse à penser d'après le Chapitre 3 que QUIC a vocation à remplacer la pile TCP/TLS.

Or en se plaçant du point de vue de l'ISP, on peut réaliser une analyse *Strengths, Weaknesses, Opportunities, Threats* (Forces, Faiblesses, Opportunités, Menaces) (SWOT) portée sur le changement de TCP/TLS vers QUIC.

Une telle migration représente donc pour l'ISP :

Une force : En effet QUIC permet dans la majorité des cas une poignée de main en 0RTT, contre 3 avec TLS/TCP. Le coût d'un RTT étant très élevé dans un contexte SATCOM, QUIC présente donc la force de permettre à l'utilisateur de se connecter plus rapidement. Par ailleurs, des études comparatives [36] menées sur des liens typiques terrestres[1] montrent de meilleures performances pour QUIC, même lorsque le Contrôleur de Congestion (CC) est identique sur les deux protocoles, c'est donc une autre force de QUIC.

[1] Y compris de grand RTT.

Une faiblesse : QUIC ne peut en effet pas profiter des *Performance-Enhancing Proxies* (Mandataires améliorant les performances) (PEPs) du fait de ses mécanismes de chiffrement. Cela en fait une faiblesse car le PEP améliore les performances (Section 2.3.3). Par ailleurs, il a été identifié que les performances de QUIC dépendent fortement du contexte et du type de ressource chargée [36]. En particulier, elles sont impactées lorsque le client est un mobile du fait de la charge processeur nécessaire au chiffrement/déchiffrement des paquets.

Une opportunité : Néanmoins s'il s'avérait que les performances de QUIC restent bonnes sans l'usage de PEP, son déploiement représente une opportunité car l'opérateur pourrait alors se passer des PEP, de la complexité de leur infrastructure et de leur opération. Enfin, la puissance de déploiement de QUIC permettrait un remplacement efficace de vieilles piles TCP pour une amélioration globale des performances des utilisateurs.

Une menace : Le déploiement de QUIC représente une menace pour les capacités de contrôle, mesure et surveillance de l'ISP dans son réseau [17]. En effet, une grande partie des informations est inaccessible aux yeux de l'opérateur. Il perd en capacité pour localiser les pertes, mesurer les délais et adapter en conséquence son réseau. Enfin, pour l'opérateur SATCOM, si QUIC s'avérait présenter de mauvaises performances du fait du non-bénéfice de la technologie PEP, son déploiement en remplacement de TCP[2] menace de sévèrement dégrader la qualité d'expérience ressentie par l'utilisateur, à son insu.

Pour peser le poids relatif de chacun de ces éléments SWOT, on voit bien que nous avons besoin d'une étude sur les performances comparées de QUIC et TCP sur un lien satellite utilisant des optimisations pour TCP comme des PEPs.

Notre objectif va donc être de trouver un moyen de mener cette étude dans un contexte représentatif pour un utilisateur accédant à Internet via un SATCOM GEO.

4.1.2 Les risques associés à l'étude

Mener une telle étude maintenant présente cependant un certain nombre de risques:

Volatilité des spécifications

Tout d'abord comme nous l'avons mentionné en Section 3.2.2, les spécifications souffrent d'une certaine **volatilité**. Le risque est donc de faire un étude de performances dont les résultats seraient caduques suite à une prochaine mise à jour du *draft*. Dans le même temps, attendre une stabilisation des spécifications et la sortie officielle de QUIC V1 n'est pas souhaitable vis à vis du timing du PFE.

Influences complexes sur les résultats

Ensuite, l'environnement de QUIC est complexe. Nous avons déjà mentionné la différence entre GQUIC et IETF QUIC. Il faut ajouter à cela que les performances globales ressenties par l'utilisateur dépendent aussi de la manière dont l'application met en œuvre QUIC.

[2]Ce qui ne nécessite qu'une mise à jour en espace applicatif.

Ces interactions inter-couches (*cross-layer*) sont décrites pour IETF QUIC mais les *frameworks* applicatifs mettant en œuvre IETF QUIC sont tout récents, en constante évolution, les performances mesurées peuvent donc dépendre plus de l'implémentation que de QUIC.

De son côté, la manière dont Chrome met en œuvre GQUIC est assez vague.

Le risque de la rétro-ingénierie

Enfin, et cela est lié aux risques précédents, nous devons éviter de tomber dans du *retro-engineering*. Comprendre les mécanismes complexes de Chrome, de GQUIC ou même des *frameworks* implémentant IETF QUIC[3] peut consommer beaucoup de ressources inutilement puisque la compréhension obtenue aura toute les chances d'être dépassée lors d'une prochaine mise à jour des systèmes.

4.1.3 Les principes adoptés pour limiter les risques

Nous avons donc besoin de fixer un certain nombre de principes qui nous permettront de mieux traiter les éventuels problèmes que nous rencontrerons.

Étude comparative

Pour corriger l'influence d'un certain nombre de facteurs, nous orientons résolument notre étude sur un principe de comparaison des performances QUIC vis à vis d'une pile traditionnelle TCP/TLS accélérée par des PEPs.

Définition d'invariants QUIC

Pour maîtriser la question de l'évolution des spécifications, nous définissons des invariants de QUIC. Ce sont des propriétés qui sont communes à IETF QUIC et à GQUIC et qui ont de bonnes chances de rester les mêmes dans les prochaines évolutions des standards. Nous en avons défini deux :

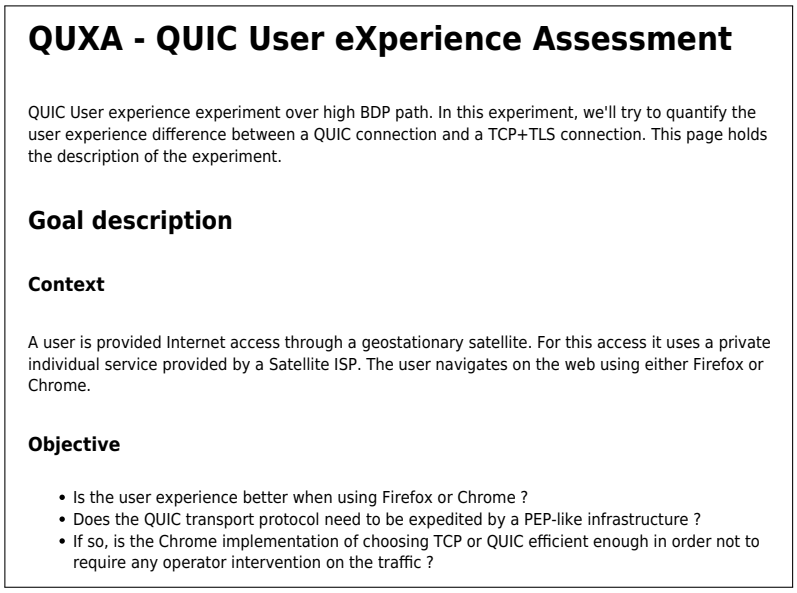
- Les poignées de main de QUIC permettent l'établissement de la connexion (à la fois sur les fonctions de fiabilité et de sécurité) en 1 ou 0 RTT, comme décrit en Section 3.1.2.
- L'usage du chiffrement pour QUIC empêche l'usage de PEP (Section 3.1.2). La connexion est assurée de bout en bout du client au serveur disposant d'un certificat valide. Les intermédiaires de réseau ne peuvent intervenir dans la connexion.

Ainsi notre approche devrait se baser sur ces invariants et les résultats que nous obtiendrons devraient être robustes (rester qualitativement valables) aux changements des autres propriétés.

Orientation usager

Nous orientons notre étude sur des mesures impactant l'expérience d'un utilisateur réel sur un accès SATCOM GEO public. Ce principe nous permet d'une part de contrôler la complexité de QUIC en limitant la finesse nécessaire à l'analyse et d'autre part, nos résultats gagnent en applicabilité puisque nous obtiendrons des mesures valables pour la majorité des clients satellites publics.

[3] En guise d'exemple, une à deux semaines furent dépensées dans la compréhension du *framework* `picoQUIC` [25], finalement abandonné en raison d'une trop grande instabilité du code.



QUXA - QUIC User eXperience Assessment

QUIC User experience experiment over high BDP path. In this experiment, we'll try to quantify the user experience difference between a QUIC connection and a TCP+TLS connection. This page holds the description of the experiment.

Goal description

Context

A user is provided Internet access through a geostationary satellite. For this access it uses a private individual service provided by a Satellite ISP. The user navigates on the web using either Firefox or Chrome.

Objective

- Is the user experience better when using Firefox or Chrome ?
- Does the QUIC transport protocol need to be expedited by a PEP-like infrastructure ?
- If so, is the Chrome implementation of choosing TCP or QUIC efficient enough in order not to require any operator intervention on the traffic ?

Figure 4.1: Capture d'écran de la page de suivi QUXA : On y voit le contexte, la notion de comparaison et les objectifs visés (réponses recherchées). Le choix de comparer `Firefox` et `Chrome` a par la suite été remis en question. `Source : wiki.star. Accédée le 18/10/218. Accessible uniquement depuis l'infrastructure STAR du CNES.`

4.2 Description de l'étude QUXA

Nous avons ainsi défini, mené et exploité l'étude *QUIC User eXperience Assessment* (Evaluation de l'expérience de l'utilisateur QUIC) (QUXA). Cette section présente la composition de l'étude.

Un point important concerne la traçabilité de QUXA. Le service DSO/NT/ST du CNES possède un système de documentation intégré à son infrastructure. QUXA y a donc droit à sa propre page. Elle décrit :

- Les objectifs et le contexte visés.
- L'ensemble des exigences[4] pour l'étude, accompagnées de leur justification. Ces exigences portent sur le banc de test, les versions des logiciels, etc.
- La description chronologique des tests menés, l'analyse des résultats, l'identification des problèmes, les études complémentaires menées pour comprendre les problèmes et les actions réalisées pour les traiter.

La Figure 4.1 présente ainsi le tout début de cette page avec les objectifs initialement identifiés.

4.2.1 Un banc de test

Le banc de test principal utilisé est constitué d'un ordinateur portable connecté à Internet via une offre d'accès à Internet par satellite (Figure 4.2). Il charge des données stockées sur un serveur Google quelque part dans le réseau Internet.

Nous utilisons cependant aussi un autre banc de test en guise de comparaison. Ce dernier utilise un accès Internet 4G (Figure 4.3). Pour des raisons techniques de

[4] 48 au total.

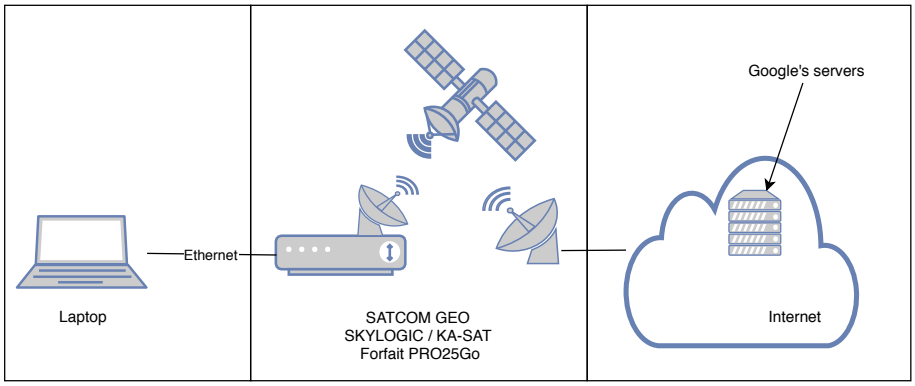


Figure 4.2: Schéma du banc de test SATCOM utilisé pour QUXA : Un ordinateur portable charge des données d'un serveur Google via une offre SATCOM GEO publique.

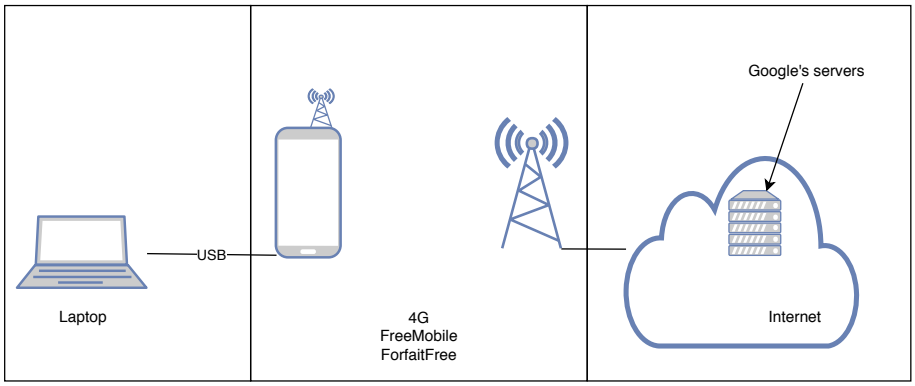


Figure 4.3: Schéma du banc de comparaison 4G utilisé pour QUXA : Un ordinateur portable charge des données d'un serveur Google via un accès 4G public.

Table 4.1: Caractéristiques des offres d'accès à Internet par satellite ou par 4G utilisées. `Sources : konnect-europe.com et mobile.free.fr.`

Accès par…		Satellite	4G
Nom de l'offre		Konnect Pro25Go	FreeMobile
Capacités maximales	Montante	6Mbps	50Mbps
	Descendante	30Mbps	150Mbps
Quota		25Go	100Go

driver USB, l'ordinateur portable utilisé est différent mais le reste du banc de test est identique.

Une infrastructure réelle

Nous décidons de réaliser notre étude *via* des mesures sur une infrastructure réelle.

Justifications : QUIC est en effet un protocole en cours de développement, les implémentations de l'IETF QUIC ou de GQUIC sont donc trop récentes et instables pour qu'on puisse les considérer pour une émulation, bien que le CNES possède toute une infrastructure d'émulation avec `OpenBach` [10] et `OpenSand` [11].

Par ailleurs la simulation est in-envisageable puisque les modules QUIC pour les simulateurs de réseau n'existent pas encore.

Un accès à Internet

L'accès à Internet par satellite est fourni par le **distributeur** `Skylogic` [54] *via* le satellite géostationnaire `KA-SAT` [16] de `Eutelsat` (**opérateur**, voir Section 2.3.2). Les caractéristiques du forfait utilisé sont disponibles en Table 4.1. Comme pour tout forfait d'accès à Internet par satellite, un quota de données est présent : dans le cas de notre offre, au-delà de 25Go mensuels consommés (somme du montant et du descendant), les capacités sont drastiquement réduites (il devient alors impossible de mener à bien nos tests). Ce quota limite le nombre de tests que nous avons pu mener.

Les caractéristiques théoriques de l'accès 4G sont aussi fournies en Table 4.1.

Justifications: Le choix d'utiliser un accès SATCOM GEO est lié à la prépondérance de ce trafic dans les accès Internet par satellite (voir Section 2.2.2). Nous choisissons un accès public et non un montage spécifique en accord avec notre objectif de rester proche de l'expérience d'un utilisateur lambda. En particulier, nous cherchons à déterminer si les optimisations spécifiques opérées par l'ISP SATCOM n'amèneraient pas des comportements singuliers.

Pour le lien de comparaison, nous choisissons un lien 4G pour sa facilité de mise en œuvre. En effet, un accès 4G public à destination des particuliers implémente des politiques de filtrages (pare-feu, DPI, …) assez limitées et bloque donc très rarement l'UDP, protocole vital puisque son port 443 véhicule QUIC. Or en utilisant une infrastructure terrestre liée au CNES, le trafic aurait dû passer par les moyens de contrôle du CNES qui bloquent l'UDP 443.

Un ordinateur portable

L'ordinateur portable utilisé pour le banc de test SATCOM opère sous Ubuntu Mate 16.04.

L'ordinateur portable utilisé pour le banc 4G opère sous Windows 10 émulant Xubuntu 16.04.

Justifications: D'une manière générale, les architectures sous Ubuntu sont privilégiés pour la facilité à y déployer les scripts de tests et dans un objectif d'unification. `Xubuntu` et `Ubuntu Mate` sont simplement des distributions d'Ubuntu variant dans leurs interfaces graphiques. Nous sommes conscients que les différences entre les deux ordinateurs et entre les OS sont susceptibles d'empêcher la comparaison des performances brutes entre les deux bancs. Cela n'est pas un problème puisque nous ne nous comparerons que les performances comparées de TCP et QUIC, le banc 4G servant surtout de référence aidant à comprendre les mécanismes du SATCOM.

Des serveurs Google

Le contenu chargé par le client est stocké sur un serveur `Google` remplissant toutes les conditions pour permettre l'usage de GQUIC[5].

Justifications : Certes nous ne contrôlons pas le serveur, mais le choix d'un serveur `Google` simplifie énormément la mise en œuvre des tests. Par ailleurs nous ne sommes pas très intéressés par la conception même de QUIC. Ainsi, dans [36] les auteurs ont menés des tests exhaustifs sur la conception de QUIC. Leur étude a nécessité une complexité particulière de mise en œuvre. De plus il faut ajouter qu'une fois l'ISP choisi, celui-ci fait inévitablement sortir les paquets sur le cœur de réseau Internet. Pour revenir vers une infrastructure CNES, il aurait encore fallu repasser par les systèmes de protection du CNES avec toutes les problématiques de blocage des flux, etc.

4.2.2 Un navigateur internet

Pour comparer les performances de GQUIC et TCP/TLS, il nous faut un navigateur récupérant les ressources via GQUIC et un récupérant les ressources via TCP.

Pour mesurer les temps de chargement avec GQUIC, nous utilisons le navigateur `Google Chrome` version `67.0.3396.99`.

Pour mesurer les temps de chargement avec TCP/TLS, nous utilisions initialement `Mozilla Firefox` mais des comportements inattendus de `Firefox` nous ont amenés à l'abandonner et à le remplacer par `Chrome`, configuré avec QUIC désactivé.

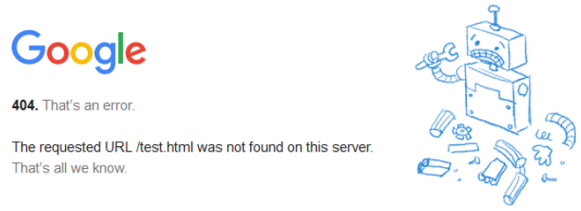
Justifications : Chrome a été choisi pour son support de GQUIC (à l'époque en version `Q039`). L'abandon de `Firefox` est expliqué en Section 4.3.2.

4.2.3 Des pages à charger

Pour mesurer des temps de chargement, il nous faut un contenu et un protocole applicatif pour les charger. Nous choisissons le protocole HTTP2.

[5]Support de la version opérée par le navigateur mais aussi promotion, voir Section 4.3.2.

(a) Cible A : Taille 5.3Mo, 1 objet. `Source : intelligence-airbusds.com`



(b) Cible B : Taille 11Ko, 2 objets. `Source : google.fr`

Figure 4.4: Cibles utilisées, pour lesquelles on mesure le temps de chargement.

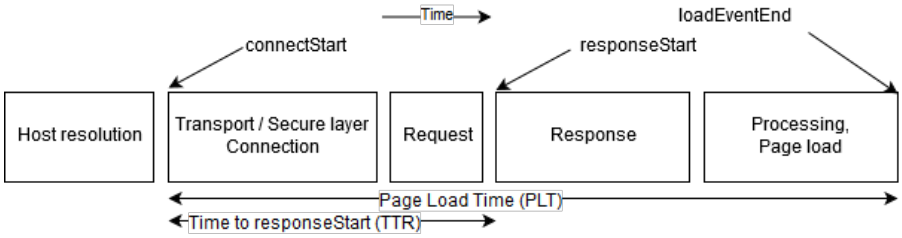


Figure 4.5: Étapes du chargement d'une page et métriques choisies: *Page Load Time* (Temps de chargement de la page) mesure tout le temps de chargement, *Time To* `reponseStart` (Temps écoulé jusqu'à `responseStart`) se concentre sur la connexion/l'envoi de requête. `Reproduite d'après [57].`

Justification : C'est en effet, pour le moment, le seul pour lequel GQUIC possède une interface. De plus il correspond à l'usage le plus fréquent et enfin il est à la racine de la conception de QUIC (Section 3.1.1).

Des cibles sur des serveurs publics

Nous avons réalisé les tests en chargeant deux cibles différentes :

- La cible A est une photo d'une taille totale de 5.3Mo.
- La cible B est une page HTML (une page 404 `Google`) contenant 2 objets pour une taille totale de 11Ko.

Ces deux cibles sont présentées en Figure 4.4.

Justification : L'idée était de fournir deux points de comparaison :

- Une situation favorisant QUIC par rapport à TCP : La cible B est d'une taille très faible mais comporte plusieurs objets. La latence des connexions et la capacité à multiplexer plusieurs chargements devraient donc bénéficier à QUIC.
- Une situation favorisant l'accélération des connexions TCP : La cible A ne comporte qu'un objet, lourd, qui nécessite que le CC soit rapidement capable d'utiliser la ressource à son maximum.

Des mesures de temps de chargement

La figure 4.5 présente les différentes étapes du chargement d'une page web en HTTP ainsi qu'elles sont définies par le *World Wide Web Consortium* (Consortium du WWW) (W3C) [57].

Pour chaque chargement d'une cible avec un navigateur, nous mesurons le *Page Load Time* (Temps de chargement de la page) (PLT), temps écoulé entre le début de la connexion (`connectStart`) et la fin de l'affichage du contenu (`loadEventEnd`). Nous mesurons aussi le *Time To* `reponseStart` (Temps écoulé jusqu'à `responseStart`) (TTR), temps écoulé entre le début de la connexion (`connectStart`) et le début de la réponse du serveur (`responseStart`).

Justifications : Toujours dans l'idée de comparer les situations qui seraient favorables à GQUIC de celles qui lui seraient défavorables, nous souhaitons étudier la contribution dans le PLT du temps nécessaire à établir la connexion avec le serveur d'une part et le temps pour télécharger toutes les données d'autre part, d'où les deux métriques. Aucune des deux n'intègre le temps nécessaire à la résolution de nom car cette partie constitue un facteur d'influence en dehors de notre étude.

4.2.4 Des unités de test

Les tests sont opérés par des scripts `Python3` sur l'ordinateur client et regroupés dans des **unités de test**. Chaque unité procède chronologiquement :

- Configuration du système :
  - La résolution *Domain Name System* (Système de noms de domaine) (DNS) est forcée en imposant l'adresse ip du serveur via le fichier `/etc/hosts`.
  - Le trafic GQUIC est bloqué en sortie à l'exception de celui en destination de l'adresse IP du serveur. Ceci est réalisé grâce à `iptables`.
  - Une surveillance du niveau d'utilisation du processeur est lancée, pour toute la durée de l'unité de test.
  - Une capture de tous les paquets entrant/sortant de l'interface Ethernet de l'ordinateur client est lancée *via* `tcpdump`, pour toute la durée de l'unité de test.
- Chaque demi-heure, l'unité de test effectue un test du lien satellite, constitué de :
  - Quatre tests de capacité de 15 secondes : en destination/provenance d'un serveur public de mesure et en UDP et TCP.
  - Une mesure de RTT vers le serveur public de mesure et vers le serveur contenant la cible.

  En dehors de ces demi-heures, cette partie est sautée par l'unité de test.
- Un tirage au sort détermine si c'est le navigateur avec GQUIC[6] ou celui sans GQUIC[7] qui commencera le premier les tests.
- Puis, pour chaque navigateur :

[6] `Chrome` avec GQUIC activé.

[7] `Firefox` ou `Chrome` avec GQUIC désactivé, selon le test.

  - Le navigateur est configuré :
    * Le stockage (*caching*) du contenu des pages est désactivé.
    * Si nécessaire, l'enregistrement interne des événements est activé[8].
    * Selon le cas, activation ou désactivation de l'option pour utiliser GQUIC. Si GQUIC est activé, nous ajoutons aussi une option pour demander au serveur d'utiliser le CC BBR.
    * *TCP Fast Open* (Ouverture rapide de TCP) (TFO) [12] est activé.
  - La page est chargée 3 fois :
    * Entre chaque chargement, le navigateur est fermé et rouvert.
    * Entre chaque chargement, un temps de repos uniformément réparti entre 5 et 15 secondes est opéré.
    * Pour chaque chargement, dès que le navigateur démarre, la cible est demandée. Une fois chargée, le navigateur récupère les métriques puis se ferme.

- Enfin, si l'unité de test avait réalisé la mesure de la qualité du lien, elle récupère le *METeorological Airport Report* (Rapport météorologique d'aéroport) (METAR) de l'aéroport de Toulouse Blagnac puis le joint aux résultats[9]. Cela est fait *via* l'API de l'Organisation de l'Aviation Civile Internationale (OACI) [29].

Un ordonnanceur lance les unités de tests, avec un rythme de 10 par heure. Les tests sont menés entre 13 et 17h (heure locale). On a donc un potentiel de 40 mesures par jour.

Justifications : La résolution DNS forcée, le blocage du trafic QUIC, le tirage au sort, la désactivation du stockage et la fermeture/ré-ouverture des navigateurs entre chaque test sont toutes des méthodes de contournement de certains problèmes rencontrés. Pour comprendre comment sont définis les problèmes et comment ils sont traités, voir la Section 4.3.

La surveillance du niveau d'utilisation du processeur, la mesure de la qualité du lien et le relevé de la météo sont des mesures préventives permettant l'explication d'éventuelles performances inhabituelles[10].

La capture du trafic est une donnée principale en parallèle des métriques de chargement, elle sera exploitée en Section 5.1.2.

L'activation du système de log interne n'est utilisé que pour améliorer la compréhension des mécanismes, il est désactivé pour les résultats présentés.

BBR pour GQUIC est demandé car cet algorithme est supposé être déjà déployé pour les implémentions TCP dans les serveurs Google [7].

Le choix d'effectuer 10 tests par heure réside dans un compromis entre l'espacement entre les unités de test, le temps nécessaire pour effectuer une unité de test et la volonté de maximiser le nombre de tests réalisés.

Le choix de la période dans la journée pour effectuer les tests réside dans une étude interne du CNES qui montre que le lien est moins congestionné à ces heures.

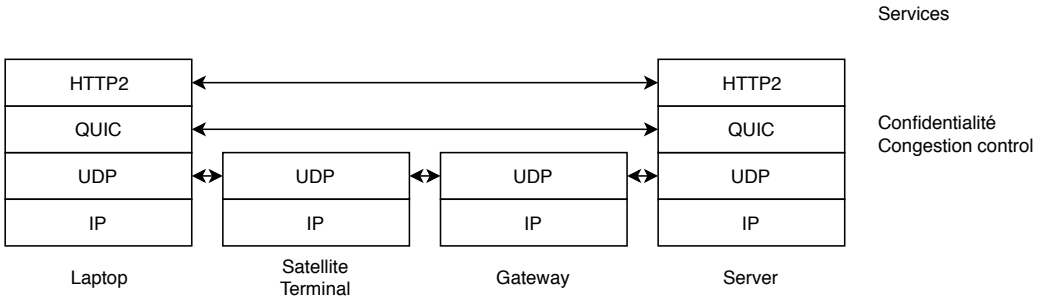


Figure 4.6: Architecture protocolaire du banc de test avec GQUIC. GQUIC ne peut être découpé, ainsi le contrôle de congestion est opéré de bout en bout.

4.2.5 Architecture protocolaire du banc de test

Dans notre infrastructure de test, nos mesures montrent que des PEP sont déployés pour TCP. Du point de vue architecture protocolaire, lorsque le navigateur récupèrera la ressource en TCP, nous aurons ainsi la situation décrite en Figure 2.7. La fonction de contrôle de congestion est répartie en trois segments.

En revanche, une connexion GQUIC ne pouvant être découpée, lorsque le navigateur utilisera ce protocole pour charger la ressource, nous serons dans la situation telle que décrite en Figure 4.6 : le contrôle de congestion est réalisé de bout en bout.

4.3 Les problèmes

Inévitablement dans ce projet orienté R&T, tous les phénomènes ne sont pas connus à l'avance. Nous avons rencontrés des comportements complexes, d'origine pas toujours claire, influençant nos résultats. Pour ne pas tomber dans l'un des pièges mentionnés en Section 4.1.2, nous avons adopté une méthode de gestion des problèmes basée sur les principes mentionnés en Section 4.1.3.

4.3.1 Méthode de gestion des problèmes

Le mot "problème" désignera dans la suite tout phénomène complexe susceptible de perturber nos résultats. Les problèmes ont pu être anticipés ou découverts lors des tests. Pour faciliter leur gestion, nous les classons selon les critères suivants :

- **Le problème affecte-t-il la comparaison entre les deux protocoles ?** En effet, nous avons acté le principe de l'étude comparative. Ainsi, si un problème affecte négativement les deux protocoles simultanément, alors la comparaison ne sera pas affectée.
- **Le problème est-il lié aux invariants ?** En effet nous avons défini le principe des invariants. Ils sont aux nombre de deux : l'impossibilité pour QUIC de bénéficier des PEP et les poignées de main QUIC en 1 ou 0RTT. Si un problème a son origine dans ces invariants ou si ses conséquences impactent les performances *via* les invariants, on dira alors qu'il y est lié.

[8] Pour `Chrome`, cela est réalisé grâce aux pages `chrome://net-export` et `chrome://net-internals`. Pour `Firefox`, la complexité du système de log interne étant assez élevée, cela n'a pas été très utilisé.

[9] Un rapport par demi-heure est publié.

[10] Par exemple, pour un accès SATCOM, la pluie a un impact prépondérant sur les performances.

Cette classification des problèmes nous permet de prendre plus facilement une décision sur les mesures à adopter. Un problème peut ainsi être :

- **Contourné :** Si un problème affecte la comparaison mais qu'il n'est pas lié aux invariants, alors on est dans une situation présentant un risque de *retro-engineering* : un phénomène que l'on ne comprend pas affecte directement notre comparaison mais sa compréhension risque de consommer beaucoup de ressources et d'être caduque lors d'une prochaine mise à jour. L'objectif va donc être de **contourner** le problème en le faisant disparaître/en bloquant ses effets. Cela se fait typiquement par isolation dichotomique.
- **Expliqué :** Si un problème affecte la comparaison et est lié aux invariants, alors il fait partie de notre étude, nous devons le comprendre et **expliquer** la manière dont il interagit avec nos invariants pour affecter notre comparaison.
- **Non expliqué/Non contourné :** Si un problème n'affecte pas la comparaison, alors nous pouvons simplement l'**ignorer**. Cette catégorie peut aussi rassembler tous les problèmes qui affectent la comparaison[11] et n'ont pas été entièrement expliqués/contournés, soit parce-que l'explication, bien qu'incomplète, nous a semblé suffisante pour notre étude, soit parce-que l'influence nous a semblé limitée. La frontière entre la nécessité de traiter un problème et l'acceptation de sa non-maîtrise étant toujours subjective…

4.3.2 Quelques exemples de problèmes

Pour donner une idée de l'utilité de la méthode décrite ci-dessus, nous prenons plusieurs exemples de problèmes rencontrés :

Influence de la météo

L'influence de la météo sur les performances des protocoles (donc sur les métriques) est un problème qui a été anticipé *via* des discussions avec l'équipe. Cependant, comme les protocoles sont testés au sein d'une même unité de test l'un à la suite de l'autre, avec peu de temps de pause entre les deux, nous estimons très faible la probabilité qu'une dégradation météorologique impacte subitement l'une des mesures pour un protocole et pas l'autre.

On classe ainsi ce problème en "n'affectant pas la comparaison", nous choisissons de l'ignorer et nous nous contentons de noter la météo au cas où un résultat serait particulièrement singulier et pourrait être lié à la météo.

Le comportement de Firefox

Nous avions anticipé que `Firefox` ne présenterait pas les mêmes performances intrinsèques (chargement graphique de la page, efficacité du programme…) que `Chrome`, cela affectant la comparaison puisque la mesure de performance TCP est effectuée sous `Firefox` tandis que la mesure de performance QUIC est effectuée sous `Chrome`.

Ainsi la Figure 4.7 présente le temps de chargement de la cible A pour 8 tests lorsque Firefox et Chrome utilisent le même protocole (TCP) pour récupérer la page. On observe bien la différence de temps de chargement et cela n'est certainement pas lié aux protocoles eux-mêmes, donc ce n'est pas lié aux invariants : nous cherchons à contourner le problème.

[11] Et qui donc auraient normalement dû être contournés/expliqués.

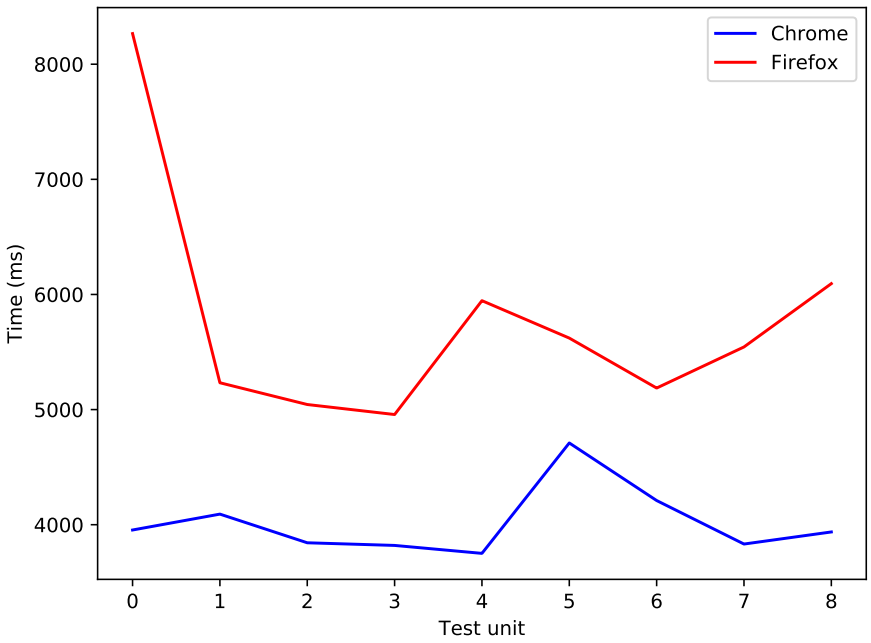


Figure 4.7: Temps de chargement cible A, lorsque Firefox et Chrome utilisent le même protocole : Firefox est intrinsèquement plus lent que Chrome.

Une première idée consistait à réaliser, pour chaque unité de test et pour chaque navigateur, un premier chargement de la page en forçant l'usage de TCP. Cette mesure servirait alors de référence et les chargements suivants (en changeant, pour `Chrome`, de protocole) seraient exprimés comparativement à cette valeur.

Sauf que la Figure 4.8 donne justement la répartition, calculée sur 8 tests, du temps de chargement PLT pour les deux chargements ("Load 2" et "Load 3") qui suivent le chargement de calibration ("Load 1"). On s'aperçoit[12] que Firefox prend plus de temps à charger la page pour les loads 2 et 3 (valeur supérieure à 1 pour 75% des tests). Ce n'est pas normal puisque Firefox utilise toujours TCP pour charger la page. Cela affecte la comparaison et n'est pas lié aux invariants, on cherche ici encore à le contourner.

Finalement, nous choisissons de nous passer de Firefox et nous allons comparer deux instances de Chrome. Pour l'une d'entre-elles, l'usage de GQUIC sera interdit. Pour l'autre, il sera autorisé.

On remarque ainsi qu'avec cette méthode, on n'a pas consommé trop de temps à chercher à expliquer le comportement observé en Figure 4.8.

La découverte de GQUIC par Chrome

L'étude des documents [23, 41] nous a permis d'en apprendre plus sur la manière dont Chrome apprend que GQUIC est supporté par le serveur :

- La première fois que `Chrome` charge la page[13], il se connecte au serveur avec la pile TCP/TLS/HTTP.

[12] Et cela a pu être vérifié dans d'autres situations et sur d'autres métriques, en particulier TTR.
[13] Ce qu'on appelle donc Load 1 et qui servait à la calibration mentionnée en section précédente.

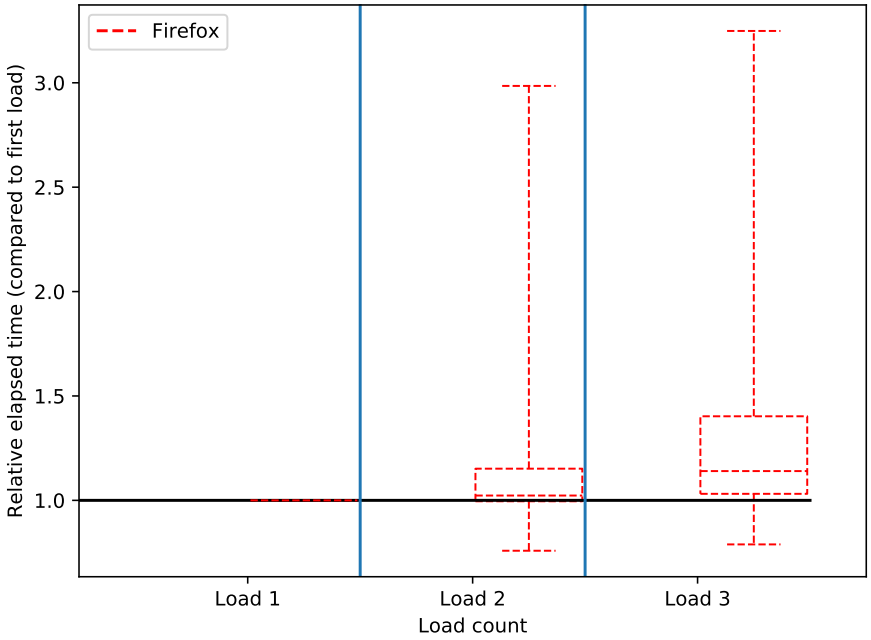


Figure 4.8: Répartition du PLT relatif au chargement de calibrage ("Load 1") en fonction de l'index du chargement effectué dans la même unité de test, pour 8 mesures. Les boites représentent les quartiles 25% et 75%. Le trait central à une boîte donne la médiane. Les moustaches donnent les valeurs extrêmes. Les chargements successifs à celui de calibration durent plus longtemps, ce qui n'est pas normal puisque Firefox utilise toujours le même protocole (TCP).

- Le serveur répond aux requêtes HTTP en indiquant dans le header HTTP, dans un champ nommé `Alt-Svc` (*Alternatives Services*), que GQUIC est disponible.
- La prochaine fois que `Chrome` se connecte (Load 2) il utilise la pile QUIC/HTTP en effectuant une poignée de main 1RTT. Il récupère les paramètres transports et cryptographiques du serveur.
- Les fois suivantes, `Chrome` utilise QUIC/HTTP et est capable de procéder à des poignées de main 0RTT grâce aux données récupérées par le load 2.

Or, lors de nos différents tests, nous avons constaté un comportement bien plus complexe, dont voici les principaux symptômes :

- Sur le lien terrestre (4G), lors du load 2, le message initial de poignée de main de la part de Chrome contient des informations qu'il n'aurait pas pu posséder sans s'être connecté préalablement en GQUIC au serveur. Ce qui nous parait bizarre puisque Chrome est supposé ne pas avoir utilisé GQUIC au premier chargement.
- Sur le lien SATCOM au contraire, la poignée de main est réalisée au second chargement comme si le navigateur et le serveur se connectaient en utilisant GQUIC pour la première fois.
- On n'observe pour autant aucune poignée de main 0RTT.

On est ici bien face à un problème (phénomène complexe affectant les mesures) non anticipé, affectant la comparaison puisqu'il ne concerne que les mesures réalisées sur GQUIC et relatif aux invariants puisqu'il conditionne en particulier la manière dont les poignées de main 1RTT/0RTT sont mises en œuvre.

Nous avons donc dépensé un certain temps à expliquer le phénomène *via* l'usage de `tcpdump`, de l'outil de log interne à Chrome, etc.

Les conclusions de nos recherches mettent en valeur des procédures complexes impliquant la favicône. Nous ne reprenons pas tous les points mais les deux suivants sont importants:

- La première fois que `ChromeQuic` se connecte au serveur ("Load 1"), il utilise TCP/TLS.
- Toutes les fois suivantes, il utilise GQUIC avec une poignée de main 1RTT.

Chapter 5

Résultats

Nous avons ainsi traité une majorité des problèmes rencontrés et nous pensons être arrivés à une maîtrise du système de test suffisante pour être confiants dans les données, au moins en comparaison. Cette section se propose ainsi de présenter les résultats principaux.

5.1 Résultats pour la cible A sur le SATCOM

Pour commencer, plaçons-nous dans la situation où l'on charge la cible A (image de poids élevé, Section 4.2.3) depuis l'accès Internet par SATCOM. Nous comparons deux instances de Chrome : une avec GQUIC activé (`ChromeQuic`), l'autre avec GQUIC désactivé (`ChromeNoQuic`).

5.1.1 Première métrique : le PLT

Pour rappel, la métrique principale que nous tirons est le *Page Load Time* (Temps de chargement de la page) (PLT). Pour chaque unité de test et pour chaque navigateur, nous obtenons 3 mesures : une pour chaque chargement effectué. On réalise en effet 3 chargements successifs permettant au navigateur de découvrir GQUIC (si activé pour le navigateur) puis de vérifier l'absence d'évolution entre les loads 2 et 3.

La question réside donc dans la manière de présenter les mesures et de définir la projection de la propriété P1 sur nos données.

Définition ordre P1 :
"$A > B$"
$\Longleftrightarrow$
"Le navigateur A a un meilleur *Page Load Time* (Temps de chargement de la page) (PLT) que le navigateur B".

Présentation en boîtes à moustaches et hypothèse d'indépendance

Une première idée est de présenter nos points de mesure sous la forme de boîtes à moustaches comme en Figure 5.1. On y représente la répartition des résultats sur l'échantillon de 40 unités de tests (40 mesures) *via* la position de certains quantiles et de la médiane de l'échantillon.

Les répartitions sont présentées en fonction de l'index du chargement depuis la création du profil du navigateur.

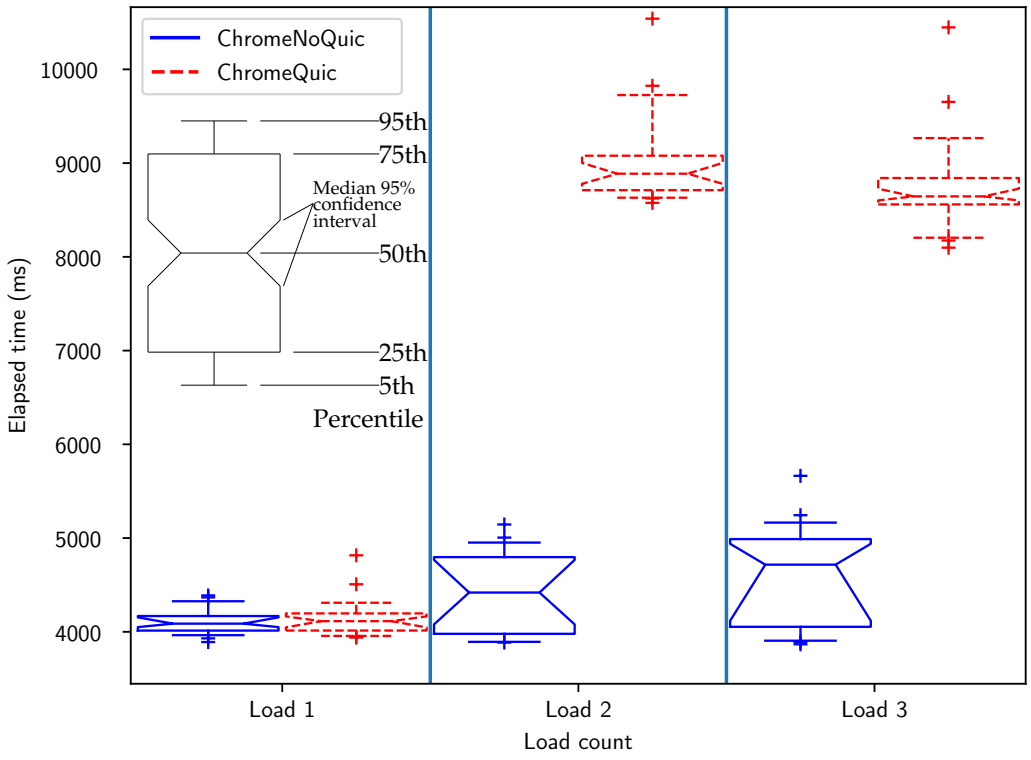


Figure 5.1: Répartition du *Page Load Time* (Temps de chargement de la page) pour la cible A sur l'accès SATCOM en fonction de l'index de chargement.

L'intervalle de confiance de la médiane : On souhaite également représenter l'intervalle de confiance de la médiane à 95% représentant la plage à laquelle la médiane réelle a plus de 95% de chance d'appartenir.

Le calcul se fait grâce au Théorème 2.1 de [5]. De manière assez intéressante, seule l'hypothèse d'indépendance et de même distribution est nécessaire. Ainsi, prenons $x_i^{b,l}$ la mesure du PLT avec b le navigateur (`ChromeQuic` ou `ChromeNoQuic`), l le numéro du chargement ($1 \leq l \leq 3$) et i le numéro de l'échantillon ($1 \leq i \leq 40$). Pour tout i, $x_i^{b,l}$ est une réalisation de la variable aléatoire $X_i^{b,l}$.

On suppose, que, pour tout couple (b, l), toutes les variables aléatoires de l'ensemble des $\{X_i^{b,l}; 1 \leq i \leq 40\}$ sont indépendantes et de même distribution $F^{b,l}$.

Par application du théorème 2.1 de [5], l'intervalle de confiance à 95% de la médiane m de $F^{b,l}$ est calculé et affiché en Figure 5.1 sous la forme d'une zone pincée.

Analyse : Ces calculs et cette présentation permettent une première analyse : pour le chargement 1, on s'aperçoit que les deux navigateurs prennent sensiblement le même temps de chargement, ce qui est consistant avec le fait que les deux utilisent la pile TCP/TLS. En revanche, pour les chargements 2 et 3, `ChromeQuic` choisit d'utiliser GQUIC grâce à l'apprentissage de sa disponibilité à la première itération. Ses performances sont alors dégradées d'un facteur deux et l'intervalle de confiance de `ChromeQuic` est distinct de celui de `ChromeNoQuic`. On décide ainsi de définir l'ordre P1 par :

Re-définition ordre P1 :
Pour le chargement l,
$$"A > B" \iff \mathcal{C}_{0.95}^{A,l} \prec \mathcal{C}_{0.95}^{B,l}$$

Avec $\mathcal{C}_{0.95}^{b,l}$ l'intervalle de confiance à 95% pour le chargement l et le navigateur b, calculé avec nos échantillons. $\prec$ représente ici l'ordre partiel des intervalles sur l'ensemble $\mathbb{R}+$ des durées.

Pour les chargements 2 et 3, les intervalles de confiances étant disctincts, `ChromeQuic` et `ChromeNoQuic` sont comparables et :

$$\texttt{ChromeNoQuic} > \texttt{ChromeQuic}$$

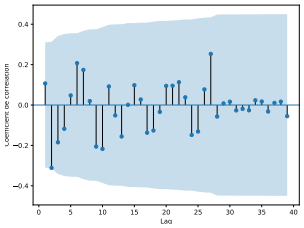
Vérification de l'hypothèse d'indépendance

Pour la partie précédente, nous avons utilisé l'hypothèse que nos variables aléatoires $X_i^{b,l}$ sont indépendantes et suivent la même loi de probabilité.

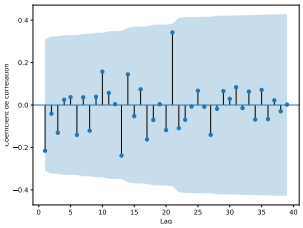
Pour justifier l'hypothèse de même distribution, nous pouvons dire que nous mesurons à chaque fois le même système et que nous pouvons négliger l'évolution du modèle, puisque les conditions météorologiques ont été constantes la plupart du temps, que la charge du lien satellite est restée faible, etc.

Pour justifier l'hypothèse d'indépendance, nous pouvons dire que les scripts repartent de zéro à chaque mesure, en utilisant de nouveaux profils de navigateur. Par ailleurs, la période de mesure est d'environ 5 minutes. On peut donc supposer que les éléments du réseau (NAT, pare-feu, etc.) ont perdu leur état entre deux unités de test.

Enfin, on peut vérifier l'hypothèse d'indépendance en se basant sur la Section 2.3.2 de [5] grâce aux courbes d'auto-corrélation. Concentrons-nous par exemple sur



(a) Coefficients d'autocorrélation pour `ChromeQuic`.



(b) Coefficients d'autocorrélation pour `ChromeNoQuic`.

Figure 5.2: Coefficients d'autocorrélation pour les échantillons du PLT de la cible A sur l'accès SATCOM. Chaque point présente une mesure de la corrélation en fonction du décalage (*lag*) dans l'échantillon. La zone bleue correspond à la zone de confiance à 95% : d'après [5], **si** les variables sont indépendantes, **alors** les coefficients ont une probabilité supérieure à 95% d'appartenir à cette zone bleue.

le second chargement de la cible A en conditions SATCOM. La Figure 5.2 présente le calcul des coefficients de corrélation sur les échantillons en fonction du décalage opéré (*lag*). Si les variables aléatoires sont indépendantes de même distribution, alors la probabilité que les valeurs calculées tombent dans la zone de confiance (zone bleue) est supérieure à 95% [5]. C'est bien ce qu'on observe ici. Bien que ce ne soit qu'une condition nécessaire, on est donc confiants pour la véracité de l'hypothèse.

CDF et dominance stochastique

Une autre manière de présenter les résultats est par l'intermédiaire d'une *Cumulative Distribution Function* (Fonction de répartition cumulative) (CDF). La figure 5.3 présente ainsi la CDF pour la métrique du PLT, pour le second chargement, pour les deux navigateurs et pour la cible A sur un contexte SATCOM.

On observe bien que le PLT pour `ChromeNoQuic` est inférieur aux valeurs pour `ChromeQuic`. La courbe de l'effectif cumulé de `ChromeNoQuic` minore stochastiquement celle de `ChromeQuic`. Re-définissons P1 :

Re-définition ordre P1 :
Pour le chargement l,
$$"A > B"$$
$$\Longleftrightarrow$$
$$\text{CDF}^{A,l} \text{ minore stochastiquement } \text{CDF}^{B,l}$$
$$\Longleftrightarrow$$
$$\forall t \in \mathbb{R}+,\ \text{EC}^{A,l}(t) > \text{EC}^{B,l}(t)$$

Avec $\text{EC}^{b,l}(t)$ l'effectif cumulé de la durée t pour le navigateur b lors du chargement l. L'ordre est toujours partiel puisque la dominance stochastique l'est.

En revanche, pour le PLT du second chargement de A en contexte SATCOM, `ChromeNoQuic` et `ChromeQuic` sont toujours comparables selon cette seconde définition et :

$$\texttt{ChromeNoQuic} > \texttt{ChromeQuic}$$

5.1.2 Seconde métrique : le numéro de séquence

La section 5.1.1 présentait les différentes manières de traiter les mesures pour la cible A en contexte SATCOM de la valeur du PLT. Quelle que soit la définition utilisée

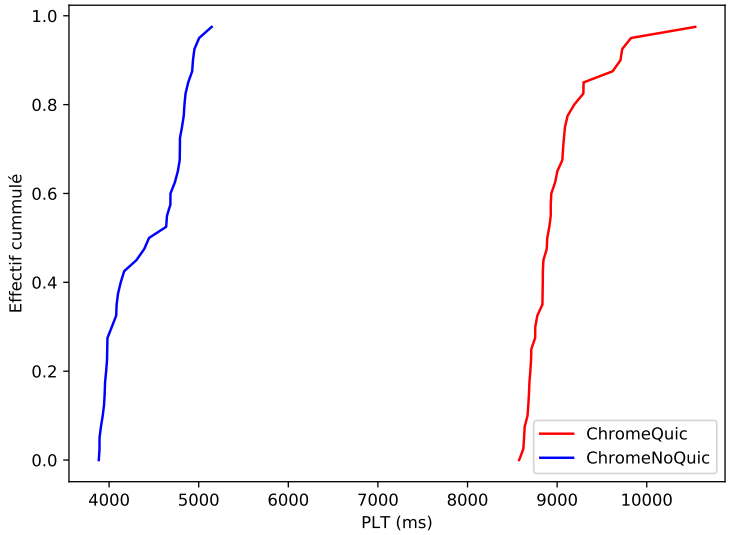


Figure 5.3: *Cumulative Distribution Function* (Fonction de répartition cumulative) (CDF) des *Page Load Time* (Temps de chargement de la page) pour la cible A en contexte SATCOM. Les deux échantillons ont des supports distincts.

de l'ordre P1, nous en avons conclu que, lorsque `ChromeQuic` utilise la technologie GQUIC, `ChromeNoQuic` > `ChromeQuic`.

Pour la cible A, dans un contexte SATCOM, TCP/TLS a ainsi de meilleures performances que GQUIC. Comment expliquer cette différence de performances ?

Tout d'abord, rappelons-nous que le chargement de la page en HTTP est constitué de deux temps :

- Le temps nécessaire à la connexion, écoulé entre l'envoi du premier paquet par le client et la réception du premier bit de réponse HTTP.
- Le temps nécessaire à la réception du contenu, écoulé entre le premier bit et le dernier bit de réponse HTTP reçu.

Dans cette section, nous étudions la seconde partie et pour cela nous définissons une seconde métrique : le numéro de séquence.

Définition du numéro de séquence

Tout fragment d'un flux applicatif continu est associé à un offset qui représente la position du fragment dans le flux. Ainsi, la cible A peut être vue comme un flux de données d'une longueur de 5.3Mo. Lorsque le fragment d'offset 5.3Mo sera reçu, l'image pourra être affichée à l'écran, le chargement sera ainsi achevé.

Pour TCP avec HTTP2, puisque la cible A ne contient qu'un objet, il y a bijection entre le flux TCP et le flux applicatif. Comme le numéro de séquence d'un segment TCP n'est pas chiffré, nous pouvons simplement le lire via un outil de capture de trafic.

Pour GQUIC, on perd cette bijection : chaque paquet GQUIC peut contenir plusieurs flux[1] et des frames spéciales. Par ailleurs les numéros de séquence sont chiffrés. Pour gérer ce problème, nous avons considéré que nous ne sommes intéressés que par le comportement global. Nous définissons donc un numéro de

[1] En particulier un flux de contrôle cryptographique.

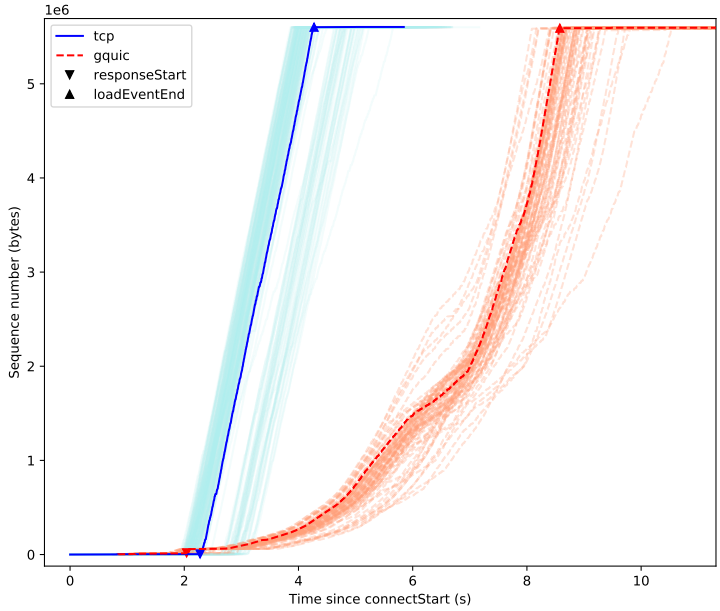


Figure 5.4: Évolution du numéro de séquence pour TCP et GQUIC en fonction du temps écoulé depuis `connectStart` (premier paquet envoyé par le client). Les triangles donnent la position temporelle de certains évènements (voir Figure 4.5). Une fois la poignée de main réalisée, le débit est quasi constant avec TCP tandis qu'il prend du temps à s'établir avec GQUIC.

séquence équivalent, que nous pensons représentatif du numéro de séquence applicatif. Sa définition exacte est disponible en Annexe B.

Évolution du numéro de séquence : le *slow start*

La Figure 5.4 présente l'évolution des numéros de séquence pour les flux TCP et GQUIC chargeant la ressource. Parmi l'ensemble des chargements effectués, on en met un de chaque protocole en valeur et on y ajoute la position des relevés des événements `responseStart` (premier bit de réponse) et `loadEventEnd` (page entièrement chargée).

On observe ainsi que la pile HTTP2/GQUIC/UDP reporte l'événement "premier bit reçu" avant la pile HTTP2/TLS/TCP mais elle termine le chargement bien après. En effet, TCP réalise une montée quasi immédiate de son débit tandis que GQUIC prend plus de temps pour gagner en débit.

Si l'on compare le comportement global du débit GQUIC sur la Figure 5.4 avec le comportement théorique décrit dans [8], on note une phase de croissance exponentielle du débit (entre 2 et 6 secondes environ) suivie de trois phases de débit constant (trois segments entre 6 et 9 secondes).

Cette première phase de croissance exponentielle est caractéristique du *slow start* présenté en Section 2.3.3. Or en Section 4.2.5 nous avions indiqué que, dans le cas GQUIC, le contrôle de congestion est opéré de bout en bout. Ainsi on peut appliquer la Formule 2.1 sur la situation présentée en Figure 4.6, avec $\mathcal{R} = 750$ms (de bout en bout), $\mathcal{I} = 32$ fois la taille d'un segment TCP[2] et $\mathcal{B}_f = 25$ Mbps, capacité maximale

[2]Comme en Section 2.3.3.

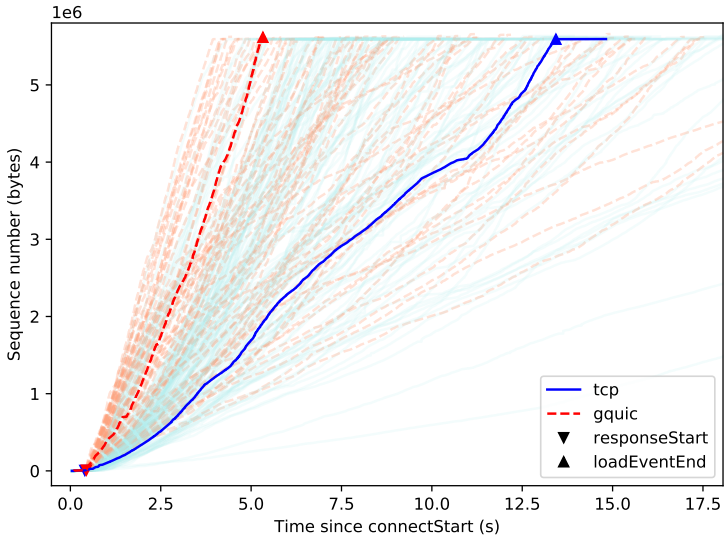


Figure 5.5: Évolution des numéros de séquence pour la cible A en 4G pour TCP et GQUIC en fonction du temps écoulé depuis `connectStart` et position de certains évènements.

du lien mesurée par la dérivée des courbes en Figure 5.4. On calcule alors une durée théorique du *slow start* avec GQUIC de 4.5 secondes, ce qui de l'ordre des durées relevées sur la Figure 5.4.

Pour TCP cette fois, la fonction de contrôle de congestion est découpée en trois segments comme en Figure 2.7. En Section 2.3.3, nous avons vu que la durée totale du *slow start* de bout en bout peut être approximée par la durée maximale du *slow start* pour chacun des trois segments. Ici en plus on rajoute l'hypothèse que le segment central (*i.e.*, le segment du lien satellite, voir Figure 2.7) ne nécessite pas de *slow start*. En effet, les *end-points* TCP de ce segment sont contrôlés par l'ISP qui connait la capacité du lien et peut ainsi effectuer du paramétrage inter-couches (*cross-layer*) pour régler directement le débit à utiliser dans les implémentations. Finalement, si l'on applique la Formule 2.1 aux segments périphériques de la Figure 2.7, on obtient que le *slow start* est fini en moins de 30ms. Or les courbes TCP de la Figure 5.4 atteignent leur débit final en moins de 50ms, ce qui est encore proche de la valeur théorique.

5.1.3 Comparaison avec les résultats 4G

Pour bien comprendre l'importance de cette durée de *slow start* dans le temps de chargement, nous réalisons les mêmes tests sur l'accès 4G (voir Figure 4.3).

Ici, le *Bandwidth-Delay Product* (Produit capacité-délai) (BDP) est beaucoup plus faible[3], TCP n'est pas coupé et les *slow start* de chaque protocole sont égaux, d'une durée théorique inférieure à 80ms.

L'évolution du numéro de séquence

Les numéros de séquence relevés sont présentés en Figure 5.5.

[3]Capacité moyenne mesurée $\mathcal{B}_f$ de 5Mbits et RTT de 80ms.

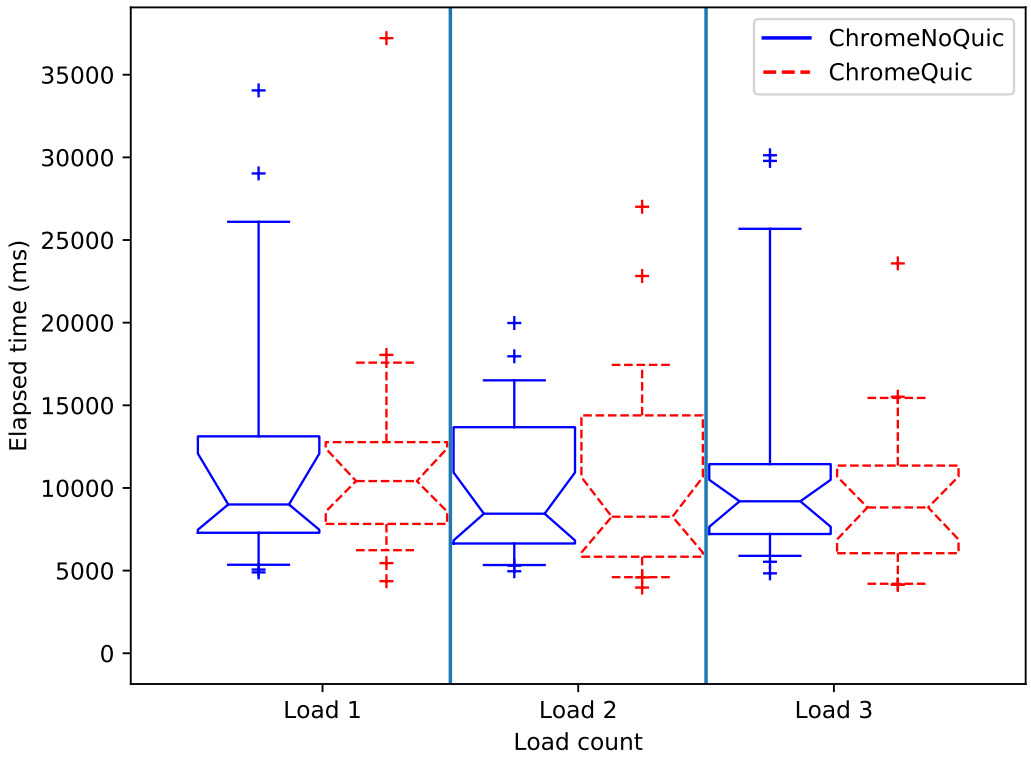


Figure 5.6: Répartition du *Page Load Time* (Temps de chargement de la page) pour la cible A sur l'accès 4G en fonction de l'index de chargement.

Ainsi sur la Figure 5.5, les deux protocoles passent la grande majorité du chargement dans leur état constant[4]. On retrouve alors que GQUIC est meilleur que TCP dans ces conditions [36]. En particulier on note une augmentation progressive du débit TCP entre 1 et 4 secondes. Cela n'est pas lié au *slow start* mais à la difficulté qu'a TCP à prendre sa part de capacité équitable $\mathcal{B}_f$ comparativement à GQUIC qui est plus agressif en simulant plusieurs connexions (voir Section 3.1.1 et [9]).

Conséquence pour le PLT

Comme conséquence, on s'attend ainsi à ce que le temps de chargement PLT soit un peu meilleur avec GQUIC qu'avec TCP, ie un peu meilleur pour les chargements 2 et 3 avec `ChromeQuic` qu'avec `ChromeNoQuic`.

Les résultats sont présentés sous la forme de boîtes à moustache en Figure 5.6 et sous la forme de *Cumulative Distribution Function* (Fonction de répartition cumulative) (CDF) pour le troisième chargement en Figure 5.7.

On observe en effet que pour le troisième chargement, par exemple, `ChromeQuic` est un peu meilleur que `ChromeNoQuic`. En revanche, pour aucune des deux définition de l'ordre P1 de la Section 5.1.1 les navigateurs ne sont comparables puisqu'en Figure 5.6, les intervalles de confiance à 95% s'intersectent et en Figure 5.7, la dominance stochastique n'est pas atteinte.

[4] *Steady state*, parfois aussi appelé état de *Congestion Avoidance*.

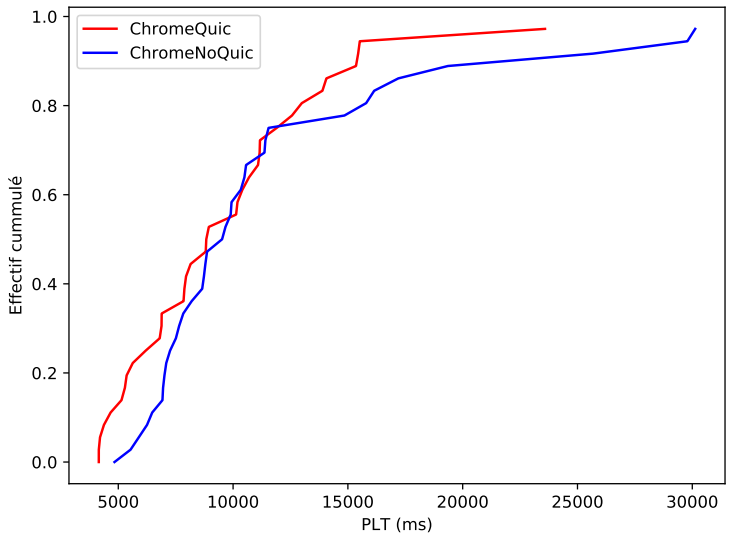


Figure 5.7: CDF des PLT pour la cible A en contexte 4G.

5.1.4 La poignée de main

Dans les deux Sections précédentes (5.1.2 et 5.1.3), nous avons mis en lumière l'influence de la durée du *slow start* sur le délai entre les évènements `responseStart` et `loadEventEnd` et le gain important apporté par la découpe de la connexion TCP en trois segments.

Dans cette section, nous allons désormais nous focaliser sur la première partie du chargement d'une page web : entre `connectStart` et `responseStart`. Cette durée est principalement liée au temps nécessaire à la poignée de main. Or GQUIC a justement été conçu avec pour objectif de réduire la latence des connexions (Section 3.1.1). On s'attend donc à obtenir de meilleurs *Time To* `reponseStart` (Temps écoulés jusqu'à `responseStart`) (TTRs) avec GQUIC en condition SATCOM qu'avec TCP/TLS.

Un gain limité

La Figure 5.8 donne la CDF pour la mesure du TTR de la cible A en condition SATCOM lors du second chargement.

On observe l'absence de dominance stochastique : le gain en établissement de connexion en condition SATCOM avec GQUIC n'est pas notable.

On note en réalité deux phénomènes :

- Tout d'abord, une forte dispersion de l'échantillon de mesure avec TCP, à environ la moitié de l'effectif. Tout se passe comme si la loi de distribution avait une composante en loi de Bernoulli (binaire). Notre manque de maîtrise du banc expérimental ne nous a pas permis d'identifier la source de cette dispersion mais certaines références suggèrent un comportement imputable aux serveurs Google [36].

- Enfin, on note que les deux poignées de main avec TCP et GQUIC durent environ la même durée et en tout cas le gain médian avec GQUIC est bien inférieur à 1RTT (750ms).

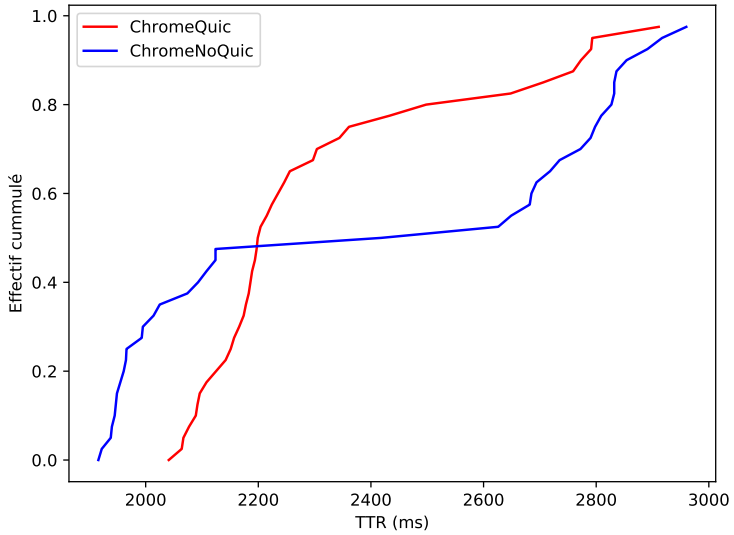


Figure 5.8: CDF des *Time To* `reponseStart` (Temps écoulés jusqu'à `responseStart`) (TTRs) pour la cible A en contexte SATCOM.

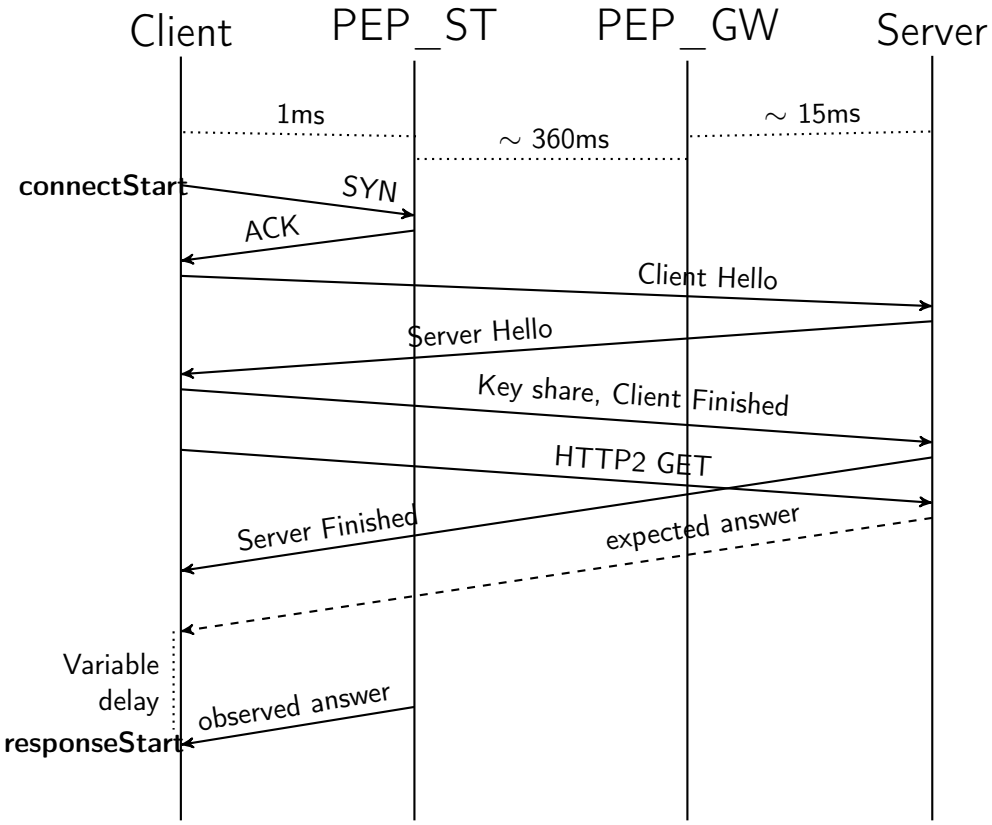


Figure 5.9: Détails de la poignée de main TCP/TLS réellement observée sur le lien SATCOM. La présence d'un mandataire TCP à proximité immédiate du client rend négligeable la durée de la poignée de main TCP. Par ailleurs l'usage de TLS1.2 False Start réduit aussi la durée de la poignée de main TLS. Enfin, la réponse HTTP subit une forte dispersion dont on observe les conséquences sur les valeurs du TTR (Figure 5.8).

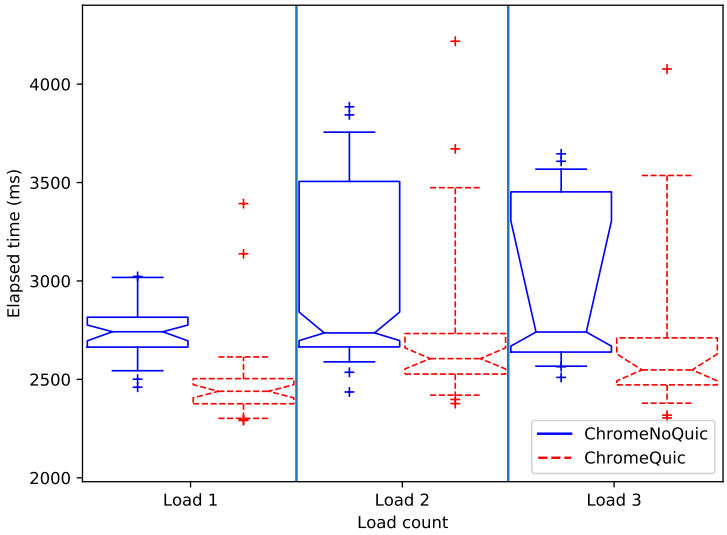


Figure 5.10: Répartition du PLT pour la cible B en contexte SATCOM. Le gain visible sur le chargement 1 peut être ignoré car d'origine externe à notre étude. Sur les chargements 2 et 3 on observe cependant un petit gain de `ChromeQuic` sur `ChromeNoQuic`.

Une combinaison complexe

Comment expliquer ce gain si faible ?

En Section 3.1.1 nous indiquions que le temps théorique écoulé entre `connectStart` et `responseStart` est de :

- 4RTT avec TCP/TLS : 1 pour la poignée de main TCP, 2 pour TLS et 1 pour l'envoi de la requête HTTP et la réception de la ressource.
- 1RTT avec GQUIC : 0 pour l'établissement de la connexion GQUIC ("0RTT", voir Section 3.1.1) et 1 pour l'envoi de la requête HTTP et la réception de la ressource.

En réalité, notre étude avancée des captures de trafic et des systèmes de log internes à `Chrome` a mis en lumière une combinaison de phénomènes qui annule totalement ce gain théorique de 3RTT.

Tout d'abord, comme mentionné en Section 4.3.2, `Chrome` n'utilise pas les connexions GQUIC en 0RTT dans notre procédure de test pour des questions de sécurité. La durée du TTR en GQUIC est ramenée à 2RTT comme sur la Figure 3.9.

Ensuite, en ce qui concerne TCP, la présence d'un PEP à proximité immédiate du client rend négligeable le temps nécessaire à la poignée de main TCP (Figure 5.9). Par ailleurs, `Chrome` utilise TLS1.2 False Start [39] qui lui permet d'envoyer la requête HTTP avant la fin de la poignée de main TLS. L'ensemble réduit ainsi la durée théorique du TTR à 2RTT soit exactement la même durée qu'avec GQUIC !

Par ailleurs, l'origine de la dispersion observée sur la distribution cumulée en Figure 5.8 est isolée au dernier aller-retour, lors de la réception de la ressource en HTTP (Figure 5.9).

5.2 Résultats pour la cible B

Dans la section 5.1, nous avons ainsi vu que, sur un SATCOM :

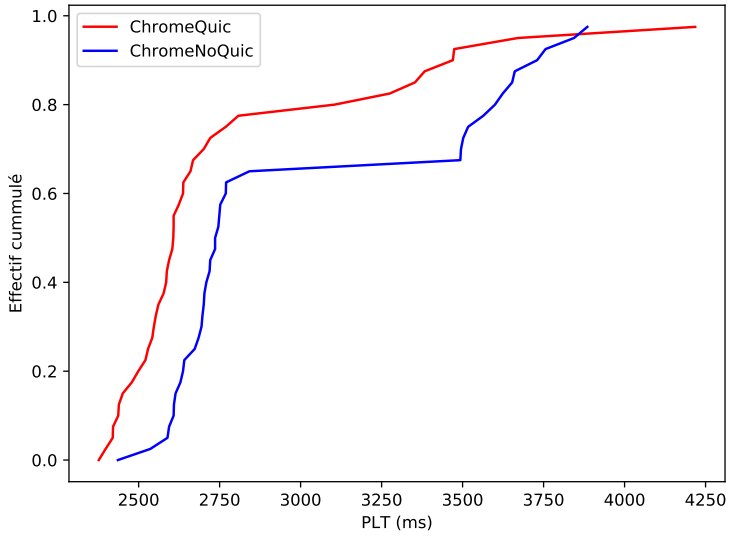


Figure 5.11: CDF des PLTs pour la cible B en contexte SATCOM. On a presque dominance stochastique de `ChromeQuic` sur `ChromeNoQuic`.

- Le gain de GQUIC sur TTR est faible et n'est pas directement lié à la technologie QUIC.
- L'impossibilité pour GQUIC à bénéficier de l'accélération par les PEP mène à un long *slow start* qui aggrave significativement le temps de chargement de A.

La cible B est différente de la cible A : elle est beaucoup plus petite et peut même être envoyée dans une fenêtre de congestion initiale. Ainsi, ni TCP ni GQUIC n'auront à effectuer de *slow start*.

Pour la cible B, on s'attend à ce que la composante principale du PLT soit le TTR. GQUIC présentant de légères meilleures performances sur le TTR (Section 5.1.4), le PLT de la cible B en contexte SATCOM devrait être plus faible avec GQUIC.

Sur la Figure 5.10 est présenté le PLT de la cible B en SATCOM sous forme de boîtes à moustaches. La différence dans le premier chargement entre les deux navigateurs est un problème que nous n'avons pas pu contourner mais que nous avons expliqué et qui ne concerne pas notre étude. Nous nous concentrons donc sur les chargements 2 et 3 et l'on observe effectivement que `ChromeQuic` a de meilleures performances que `ChromeNoQuic` dans ce contexte.

Ce léger gain est confirmé par la CDF du second chargement en Figure 5.11. On note toutefois que le gain relatif de GQUIC sur TCP/TLS n'est pas du même ordre de grandeur que la différence relative observée pour la cible A.

5.3 Discussions

Ainsi, notre étude montre que, dans un contexte SATCOM :

- Pour une page web de taille élevée (cible A), le chargement de la page en utilisant GQUIC est environ deux fois plus long que le même chargement avec TCP/TLS.
- Cette différence s'explique principalement dans les faibles performances du CC non délégué lors du *slow start*. Lorsque cette fonction est déléguée à des

sous-segments comme dans le cas TCP/TLS, le *slow start* est beaucoup plus rapide.

- L'établissement des connexions est légèrement plus rapide avec GQUIC mais cela ne compense pas le problème ci-dessus.

De manière très concrète pour l'ISP, en plus de perdre ses capacités de contrôle et mesure, ses clients voient leur expérience dégradée à leur insu par le seul choix de `Chrome` d'utiliser GQUIC.

QUIC a été conçu avec un point de vue *Fiber To The Home* (Fibre optique jusqu'à la maison). Or il ne peut y avoir de protocole de transport adapté à toutes les applications pour tous les contextes [53]. L'ISP SATCOM se retrouve ainsi à devoir prendre une décision pour conserver une bonne qualité d'expérience pour ses clients. Ces actions dépendent grandement de la capacité du WG QUIC (principalement motivé par des entreprises OTT) à inclure les considérations des ISP. Les récentes discussions entre `Orange` et le WG pour permettre des mesures passives de RTT montrent qu'à une date aussi proche de la sortie officielle du standard, rien n'est gagné.

D'après la Formule 2.1, pour réduire la durée du *slow start* sans diminuer la capacité, il faudrait soit :

- diminuer r, en coupant les connexions QUIC.
  - On peut envisager la mise en place de mandataires transparents QUIC à la manière des PEP TCP. Cela nécessite cependant que les proxys détiennent un certificat valide pour les sites Internet concernés, ou une délégation de sécurité équivalente, c'est donc une hypothèse très peu probable pour un WG guidé par "l'esprit Edward Snowden".
  - On peut envisager que le standard inclut des mécanismes de délégation sécurisée de certaines fonctions (en particulier de contrôle de congestion). Cela nécessiterait cependant d'importantes modifications dans le standard.
  - L'ISP peut aussi proposer à ses clients d'installer son navigateur optimisé [6]. Celui-ci pourrait alors contenir les certificats nécessaires à la délégation de sécurité permettant au client d'utiliser les mandataires GQUIC de l'ISP. Cette solution n'est pas viable dans les cas où le client du service n'est que temporaire (passager dans un avion par exemple).
- augmenter $\mathcal{I}$. Si l'ISP arrive à négocier la mise en place d'un centre *Content Delivery Network* (Réseau de distribution du contenu) (CDN) à proximité de son téléport, l'administrateur du CDN aura les certificats nécessaires pour opérer QUIC au nom du site web. L'ISP pourra alors négocier avec lui des valeurs adaptées de fenêtre de congestion initiales.

Il existe cependant d'autres solutions alternatives :

- On peut envisager que les CC utilisés dans les implémentations QUIC acceptent les données inter-couches telles que celles fournit par des mécanismes comme IP-*Explicit Rate Notification* (Notification explicite de débit) (ERN) [47]. Cependant ces technologies n'ont de réel intérêt que pour les liens à gros BDP. Les implémentations, concentrées sur des usages terrestres, risquent de ne pas en tenir compte.

- Enfin, à défaut, l'ISP peut aussi choisir de mettre en place des mécanismes pour empêcher l'usage de QUIC[5]. En effet si GQUIC est bloqué, `Chrome` se rabat sur l'usage de TCP/TLS qui, lui, peut être accéléré par l'infrastructure de l'ISP. De plus, lors de nos tests, nous avons pu constater que cette alternative s'enclenche sans délai supplémentaire donc sans impact pour l'expérience de l'utilisateur. Toutefois bloquer QUIC devient de plus en plus difficile politiquement : les capacités de Google en déploiement rapide du protocole (Section 3.2.1) placent les ISP devant le fait accompli car plus de 20% du trafic mondial est déjà véhiculé par GQUIC.

5.4 La question de la publication

Quoi qu'il en soit, vis à vis de ces résultats, le CNES peut avoir un objectif double :

- D'abord, informer les opérateurs SATCOM sur les réelles performances susceptibles d'être perçues par les clients lorsqu'ils utilisent `Chrome`. En effet, de précédentes études donnaient des résultats tout autres mais n'utilisaient pas de réel accès SATCOM [61]. Publier les résultats de la présente étude permettrait aux opérateurs d'obtenir une vision plus exacte de la situation.

- Ensuite, le CNES peut aussi tenter de porter à la connaissance du WG QUIC de tels problèmes sur les réseaux utilisant des PEPs afin que soit envisagées les collaborations nécessaires à la mises en place des solutions détaillées en Section 5.3. En effet, bien que le trafic Internet par satellite présente un part limitée, la technologie des PEP est pressentie aussi dans le cadre de la 5G [34, 37]. Le WG peut ainsi trouver intérêt à intégrer les problématiques relevées dans ce rapport.

C'est suivant le second objectif que nous avions décidé de soumettre nos résultats à une conférence de l'*Association for Computing Machinery* (Association pour les systèmes d'informations) (ACM) portant sur QUIC (voir annexe C). Suite au refus de notre papier, parmi les retours que nous avons eut, certains pointent légitimement notre manque de contrôle du banc de test. D'autres en revanche (voir annexe D) mettent en relief le manque de considération du WG QUIC pour ces problématiques.

A défaut, nous nous sommes rabattus sur le premier objectif et le papier est en cours de soumission à l'*International Journal of Satellite Communications and Networking* (Journal international sur les communications et réseaux satellites) (IJSCN), un journal par et pour les opérateurs SATCOM.

[5] Tels que le blocage du trafic QUIC, l'injection de paquets de terminaison de connexion, etc.

Conclusion

Ce projet de fin d'études a permis de réaliser la première évaluation des performances de GQUIC sur un accès SATCOM réel. Il met en valeur la nécessité des tests sur des architectures réelles car certains mécanismes d'optimisations peuvent être mal représentés par des simulations ou émulations.

Au cours de ce projet, j'ai dû m'adapter à un environnement protocolaire complexe, en constante mutation. J'ai dû définir la viabilité d'une nouvelle étude sur QUIC vis à vis de la documentation existante. J'ai dû définir, mener et exploiter des essais complexes tout en traitant de manière efficace les phénomènes inexpliqués. J'ai également dû valoriser les résultats et rédiger un papier pour la soumission. Je pense donc que ce PFE constitue une bonne expérience de recherche qui me sera utile dans le cadre de ma thèse.

Appendix A

Description du phénomène HOL

Cette annexe décrit avec un exemple simple le phénomène de *Head-Of-Line blocking* (Blocage par tête de flux) (HOL) qui affecte les flux lorsqu'ils sont multiplexés au dessus d'un service de remise ordonné.

Pour cela nous utilisons un exemple issu d'une présentation réalisée le 29 octobre 2018 devant les services DSO/NT/ST et DNO/DA/AP du CNES.

A.1 Situation initiale : des flux multiplexés

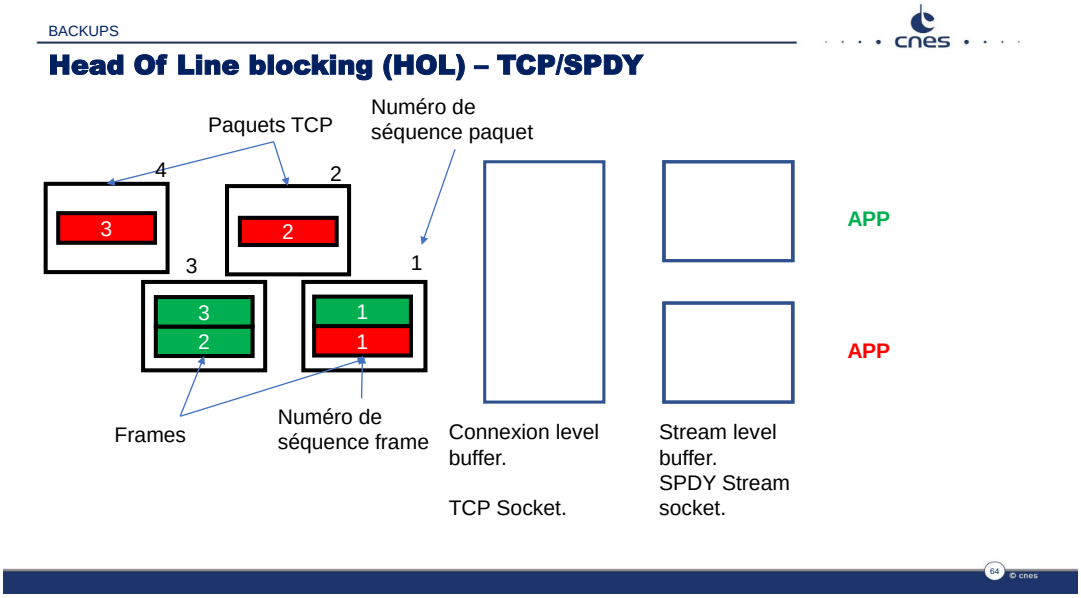


Figure A.1: *Head-Of-Line blocking* (Blocage par tête de flux) (HOL) : Situation initiale. Deux flux applicatifs partagent une même connexion TCP. Leurs **frames** sont multiplexées dans des segments TCP.

La figure A.1 présente la situation initiale :

- Deux flux, vert et rouge, sont multiplexés dans une même connexion TCP.
- Chaque flux est destiné à une application (verte ou rouge, à droite de l'image).
- Chaque application a un buffer peuplé par le protocole SPDY (ou HTTP2) qui s'occupe de démultiplexer les données depuis le buffer de la *socket* TCP.
- Les unités de données de chaque flux (souvent appelées **frames**) sont représentés par des rectangle de couleur. Leur numéro correspond à leur ordre de lecture.

- Les frames sont regroupées (multiplexées) dans des segments TCP (qu'on peut ici appeler paquets puisqu'ils perdent ici la notion de segments d'un flux unique). Les numéros sur les paquets représentent leurs numéros de séquence.

A.2 Perte d'un paquet

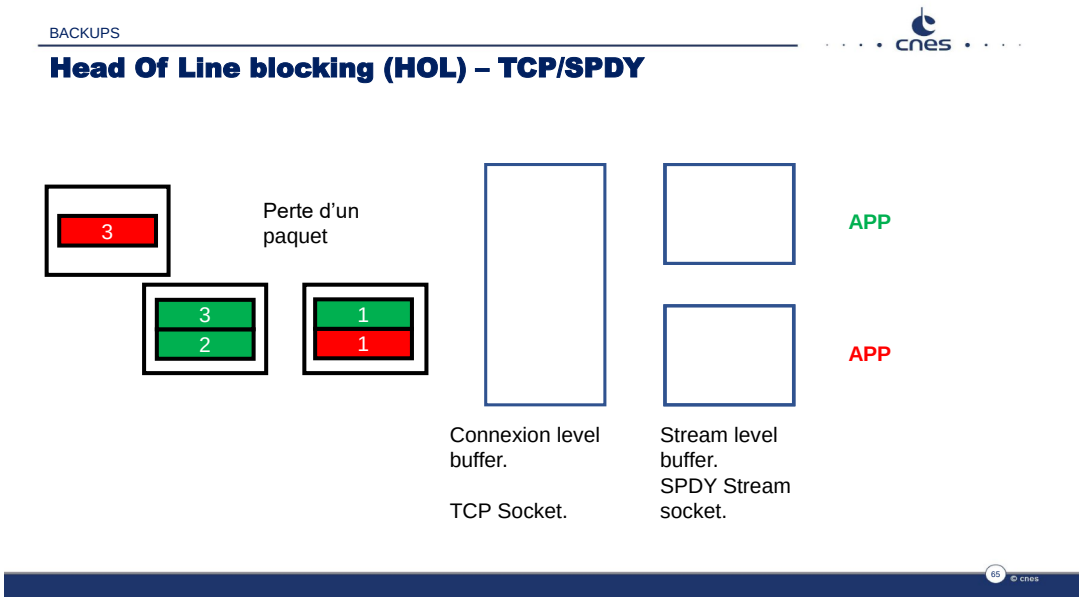


Figure A.2: HOL : Perte d'un paquet. Le segment TCP 2 est perdu (retardé).

Supposons que le segment TCP numéro 2 est perdu sur le chemin (Figure A.2) : il devra donc être réémis et subit ainsi un délai.

A.3 Remise ordonnée, niveau TCP

Le premier service à être rendu est celui de la remise ordonnée au niveau TCP (voir Figure A.3). Ainsi, puisque le segment TCP de numéro de séquence 2 n'a toujours pas été reçu (données représentées striées sur la Figure A.3), les données contenues dans les segments 3 et 4[1] sont conservées mais ne sont pas rendues disponibles au protocole suivant (ici SPDY).

A.4 Démultiplexage, niveau SPDY

Ainsi, SPDY, qui s'occupe du démultiplexage des frames, n'a pas accès à celles qui sont détenues par le protocole TCP en attendant le segment 2. En particulier, il ne peut pas fournir à l'application verte les frames 2 et 3. Pourtant le paquet TCP perdu (numéro 2) ne contenait pas de données pour l'application verte : le *Head-Of-Line blocking* (Blocage par tête de flux) (HOL) est caractérisé lorsque le ralentissement d'un flux (ici pour cause de perte) affecte les autres flux même si ces derniers ne sont pas liés à la perte.

[1] C'est à dire les frames SPDY 2 vert, 3 vert et 3 rouge.

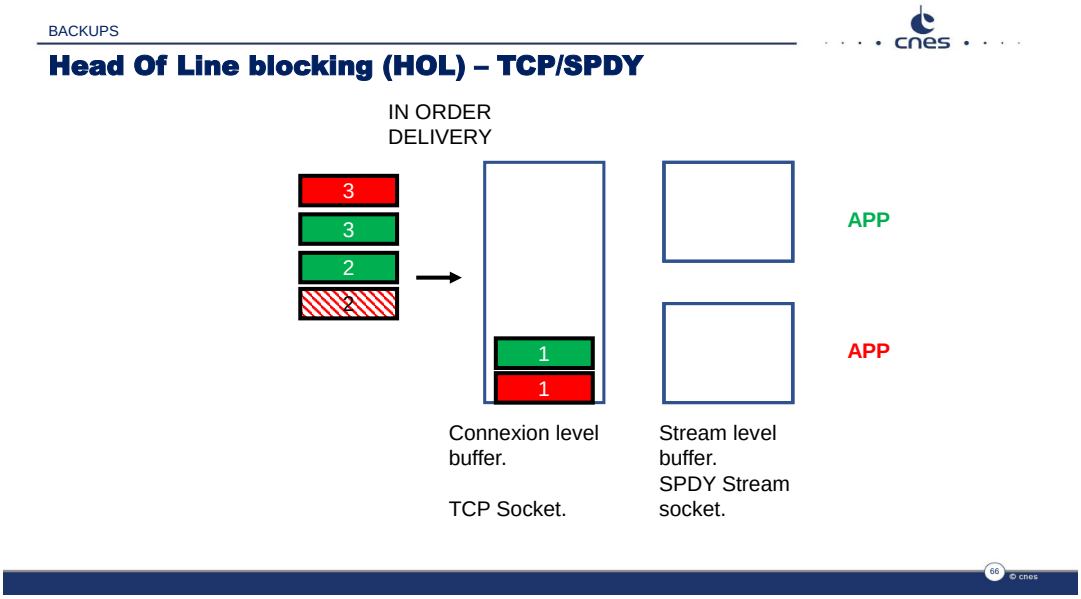


Figure A.3: HOL : Remise ordonnée par TCP. Les unité de données 2 et 3 verts et 3 rouges ne peuvent pas être rendues disponibles dans le buffer TCP car elles appartiennent à des segments TCP de numéro de séquence supérieur au segment 2, pas encore reçu.

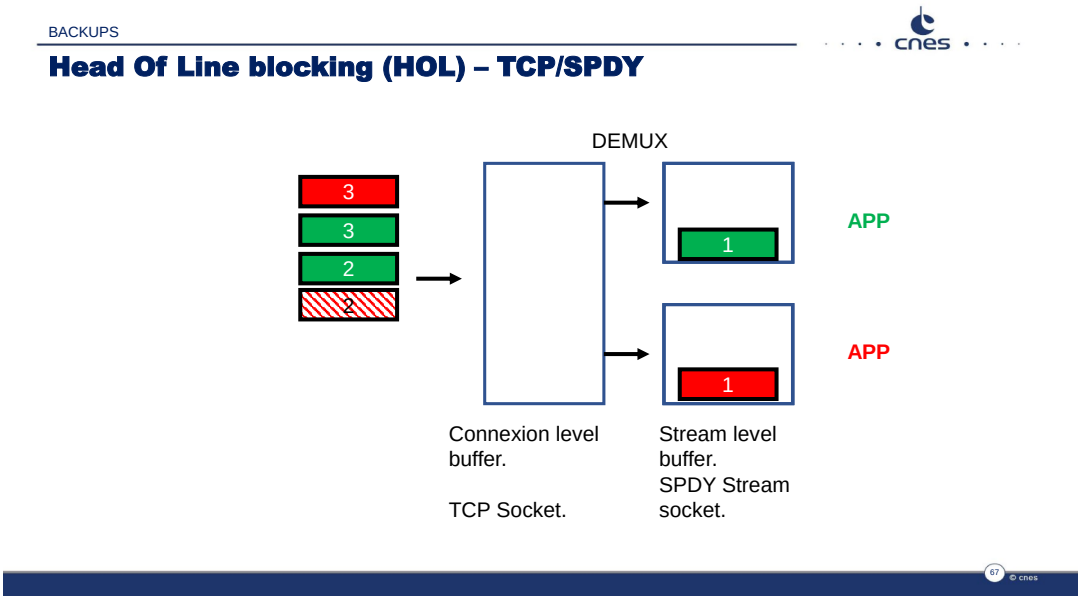


Figure A.4: HOL : Démultiplexage par *Speedy* ("Rapide") (SPDY). Ce dernier n'a pas accès aux frames 2 et 3 vertes, ni 3 rouge, stoppées par TCP en attendant l'arrivée du segment 2.

A.5 Solution : inverser l'ordre des services

Avec les figures ci-dessus, on conçoit bien que si les services étaient inversés et que la remise ordonnée s'appliquait au niveau des frames et non des segments TCP, un tel problème de HOL serait évité. Notons toutefois que TCP n'a pas été conçu comme support pour multiplexer plusieurs flux mais bien comme transporteur de flux unique, en témoignent le numéro de séquence en octets, la notion de "segment"

et le principe d'unique remise ordonné, ici problématique.

Appendix B

Calcul du numéro de séquence équivalent GQUIC

Cette annexe décrit (en anglais) le calcul d'un numéro de séquence équivalent pour GQUIC.

To better understand how the second phase of a page load influence the whole PLT, we need to compare the sequence number evolution within a TCP connection with an equivalent sequence number for a GQUIC connection.

Such number shall be comparable with the TCP sequence number. Especially, we expect that it counts the bytes received and the final value shall be the same as the final value with TCP (for the same HTTP object). To meet those expectations:

- for the test unit i in the set of test units $\mathcal{I}$,
- for the GQUIC connection k in the set of connections $K(i)$ used during test unit i,

we compute the following function of time :

$$S^{i,k}_{\texttt{GQUIC}}(t) = \alpha_i \sum_{p}^{0 \leq p \leq P^{k,i}_{\max}(t)} L_{\texttt{IP}}(p,k,i)$$

Where:

- $P^{k,i}_{\max}(t)$ is the number of GQUIC packets received in GQUIC connection k during test unit i at time t,
- $L_{\texttt{IP}}(p,k,i)$ is the length of the IP datagram containing the GQUIC packet p of the GQUIC connection k during test unit i, in bytes[1] and
- $\forall i \in \mathcal{I}$, α_i is a scaling coefficient computed so that :

$$\max\left\{S^{i,k}_{\texttt{GQUIC}}(t \to \infty); k \in \mathcal{K}(i)\right\} = \max\left\{S^{i,r}_{\texttt{TCP}}(t \to \infty); r \in \mathcal{R}(i)\right\}$$

  with $\mathcal{R}(i)$ the set of TCP connections used during test unit i.

In other words, te sequence number equivalent for a GQUIC connection is the cumulative sum of the IP-level bytes received on this connection, to which we apply a

[1] Under the hypothesis that exactly one GQUIC packet is contained in an IP datagram.

scaling coefficient so that, for each test unit, the last sequence number of the biggest GQUIC connection equals the last sequence number of the biggest TCP connection.

We reckon this removes any overhead difference between the two protocols.

Appendix C

Papier rédigé sur la base des résultats

Les pages suivantes contiennent une copie du papier soumis et refusé au *Workshop on the Evolution, Performance and Interoperability of QUIC* de la conférence CoNEXT 2018 de l'ACM. Pour plus d'informations sur le *Workshop*, voir :
https://conferences2.sigcomm.org/co-next/2018/#!/workshop-epiq.

Le papier est désormais disponible sous `Arxiv.org` à l'adresse suivante :
https://arxiv.org/abs/1810.04970.
Les résultats ont été présentés à `Orange` et lors de la réunion du WG *Measurement and Analysis for Protocols Research Group* (Groupe de recherche pour les mesures et analyses des protocoles) (MAPRG) de l'IETF103 à Bangkok.
Ils seront également soumis pour un article dans l'IJSCN.

QUIC and SATCOM

7 pages

Ludovic Thomas
ISAE-SUPAERO
Toulouse, France
ludovic.thomas@student.isae-supaero.fr

Emmanuel Dubois
CNES
Toulouse, France
emmanuel.dubois@cnes.fr

Nicolas Kuhn
CNES
Toulouse, France
nicolas.kuhn@cnes.fr

Emmanuel Lochin
ISAE-SUPAERO
Toulouse, France
emmanuel.lochin@isae-supaero.fr

ABSTRACT

We analyze QUIC transport protocol behavior over a satellite communication system. Such systems usually split end-to-end protocols to improve end users' quality of experience while fully encrypted QUIC might jeopardize this solution. Using a real satellite public access, we observe that heavy page load time is approximately twice longer with QUIC than with TLS over TCP. Although faster, QUIC connection establishment does not compensate an inappropriate congestion control.

1 INTRODUCTION

Quick UDP Internet Connections (QUIC) is a transport-layer protocol running on top of User Datagram Protocol (UDP) [21] developed by Google since 2012 and currently under discussion at Internet Engineering Task Force (IETF) [11]. QUIC benefits from years of development, rapid deployment and large scale testing. Despite evolving versions, several properties are expected to remain invariant such as encryption of both application data and transport parameters.

Fully-encrypted QUIC might lead to discrepancies between Over-The-Top (OTT) protocol design choices and Internet Service Provider (ISP) policing mechanisms. While OTT and ISP may not always be seen as competitors [4], the deployment of QUIC could lead to some ISP issues: (1) to select the appropriate Quality of Service (QoS) policy for the applications carried out; (2) to enable the right shaping policy according to both end user's contracts and access network characteristics; or (3) to optimize the use of the constrained resources such as on cellular networks.

As a matter of fact, ISP operating networks do not evolve at the same pace than End-to-End (E2E) protocols. Furthermore, they should not only be influenced by new emerging protocols, but also with existing and potentially old fashioned protocol stacks. Indeed, both the low Transmission Control Protocol (TCP) Initial congestion Window (IW) values measured in [15] and the analysis of TCP variants in the wild [30] highlight that some web services are still using outdated transport protocol flavours. QUIC could balance the part of old stacks currently used which is a great illustration of the impact OTT have over Internet traffic. Note that QUIC takes a traffic part ranging from 2.6 % to 9.1 % of the Internet with rapidly changing versions [14].

SATellite COMmunication (SATCOM) networks typically break the E2E paradigm to adapt the transport protocol to long delay links. This allows to cope with quick protocol changes. Although recent E2E protocols may exhibit fair performance over high Bandwidth-Delay Product (BDP) paths, splitting TCP allows for adaptation of both TCP slow-start and loss-recovery mechanisms. This results in lower page load times [23]. Moreover, older stacks would anyway need specific acceleration. It is worth pointing out that cellular networks may also introduce the same kind of Performance Enhancing Proxy (PEP) to adapt *e.g.* TCP for the upcoming Fifth-Generation Mobile Communications System (5G): this is not only seen through research papers [13, 17] but also in 3rd Generation Partnership Project (3GPP) study items [1]. In this context, the trend towards the deployment of protocols like QUIC questions the actual E2E protocols' adaptations. This motivates this study where we seek to assess whether a geostationary public SATCOM access influences the performance obtained by QUIC.

To the best of our knowledge, this paper reports the first evaluations of Google QUIC (GQUIC) using a real public SATCOM access by assessing the web browsing Quality of Experience (QoE). Our main findings are:

- for a heavy web page, the page load time is approximately twice longer with GQUIC compared to Transport Layer Security (TLS)/TCP (section 3.1);

- this difference in larger page load time resides in the poor performance of the non-delegated Congestion Controller (CC) in GQUIC (section 3.2);
- although faster, GQUIC connection establishment does not compensate the above issue (section 3.3).

2 EXPERIMENT SETUP

We exploit a public SATCOM Internet access and repeatedly download two pages with different profiles. We report raw QoE that represents today's end user experience. Our approach provides a fair comparison between GQUIC and an SATCOM-optimized TCP. Controlled experiments could hardly be envisioned since the operator's ground segment implements specific optimizations and QUIC available frameworks may not be relevant [16].

2.1 SATCOM and 4G Internet accesses

To better explain the behavior of GQUIC over a SATCOM access, we also perform some tests with a 4G access as a reference. This section focuses on the description of the SATCOM Internet access.

The public SATCOM operates in Europe with geostationary satellites[1]. To roughly estimate the likely performance of this access and provide an initial sanity check, we have measured that the network is less congested between 2 pm and 4 pm and have decided to run our evaluations at that moment of the day. The data plan of our contract limits the variety of performed experiments, but our tests let us assess the estimated SATCOM end-users' QoE. Furthermore, we reckon this also illustrates the impact of OTT protocol design decisions over a SATCOM provider.

A description of a generic SATCOM architecture can be found in [3, 19]. One important aspect to consider in order to interpret the results of this evaluation is the fact that TCP PEP are deployed (*i.e.*, TCP connections are split).

2.2 Web pages and QoE

To assess the QoE of a web browsing user, we measure the Page Load Time (PLT), defined as the elapsed time between the `connectStart` and `loadEventEnd` events [28]. As shown in Figure 1, the PLT can be decomposed into (1) the time needed to complete the handshake and send the request and (2) the time required to download the content and to process it. In addition to PLT, we measure Time To `responseStart` (TTR), the moment at which the user receives the first byte of the Hypertext Transfer Protocol (HTTP) response from the server. This helps us assess the contribution of the connection establishment and the request transmission in PLT. Host resolution is not considered. Metrics are measured when accessing two different targets :

- *Target A :* one picture with a 5.3 MB total size;
- *Target B :* one Google's 404 page with two objects and 11 kB of total size.

Both page are hosted on Google's GQUIC capable servers. This is consistent with our objectives as we don't focus on GQUIC design but on user QoE over SATCOM.

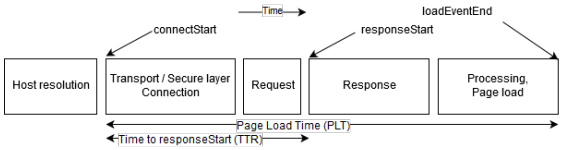


Figure 1: Steps of a page load [28]

Table 1: SATCOM and 4G access networks

Metric	SATCOM	4G
Downlink capacity	25 Mbps	5 Mbps
Uplink capacity	5 Mbps	2.5 Mbps
RTT	750 ms	80 ms

2.3 Scenario configuration

Involved protocols collect network and server information using parameters caching, TCP Fast Open (TFO) [9], QUIC discovery [12] and QUIC connection resumption[2] mechanisms. These data are then used to improve the following loads. To analyze their impact on PLT and TTR, each test unit is composed of three web pages downloads before purging the browser profile. For each load, the client fetches one of the web pages and then closes the browser when the page is retrieved. Elapsed time between two loads is uniformly distributed between 5 and 15 seconds.

Automated weather and link quality reports are linked to the test units. Measures were performed during good weather conditions, with no rain and few clouds. Average link characteristics are presented in Table 1. We also provide worth-noting comparison of the size of the targets with the BDP and the IW in Table 2. We assume servers' default IW is set to 32 TCP Maximum Segment Size (MSS) as observed in GQUIC source code [25]. We use the Selenium automation tools to control the browser and retrieve PLT and TTR. Tests are operated on a laptop with 4.15.0-29-generic Linux.

[1] Due to double-blind policy, we could not provide more details on the public Internet access at the time of submission.

[2] Also called zero-RTT and denoted 0RTT in the following.

Table 2: Ratios between target sizes, BDP and IW - *e.g.* Target A is 113 times larger than the IW

Target	Size / BDP		Size / IW
	SATCOM	4G	
A (5.3 MB)	4.8	212	113
B (11 kB)	0.01	0.44	0.2

2.4 Few words on the browsers

An analysis of `Chrome`'s behavior combined with [12, 21] has provided us with the following expectations:

- When `Chrome` starts, it opens connections with several servers. To limit the potential influence of those connections on the page load, any GQUIC traffic not intended to the server holding the web-page or its objects is blocked.
- Before using GQUIC, `Chrome` needs to learn about its availability. GQUIC discovery procedure is described in more details in [12, 21]. Important to note here is that `Chrome` always use TCP for the first time it contacts a server.
- `Chrome` will not attempt to use 0RTT connections since it is restarted between each load. We expect only 1RTT connections.

For each test unit, we use two instances of `Google Chrome 67.0.3396.99`:

- `ChromeQuic`: GQUIC is enabled with Bottleneck Bandwidth and Round-trip propagation time (BBR) congestion control instead of CUBIC[3];
- `ChromeNoQuic`: GQUIC is disabled.

`ChromeNoQuic` will always use a HTTP2-TLS1.2-(split)TCP (HTT) stack, whereas `ChromeQuic` starts with HTT and switches to HTTP2-GQUICv39-UDP (HQU) whenever possible. For both instances, TFO is enabled and content caching is disabled. Finally, Selenium tools add their own flags that enable browser control. Changes described in this section are the only ones performed on publicly available `Google Chrome`.

3 RESULTS

3.1 PLT of a heavy page

Figure 2 presents the PLT for target A as a function of the load index since last profile purge. Metrics are computed according to [5] over 40 test units. At the bottom of the figure, we also provide the *Welch's t-test p* parameter [29] of the two distributions under the *null* hypothesis. For the first load, we expect similar performance for both browser

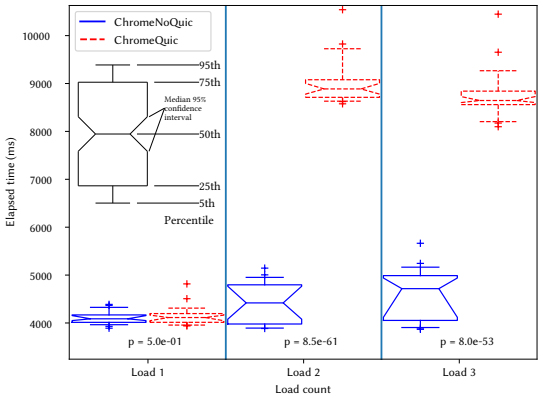


Figure 2: Page Load Time for target A (heavy picture) on a SATellite COMmunication access

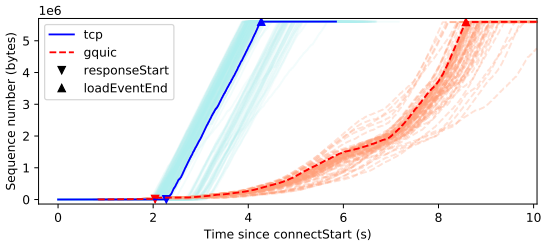


Figure 3: Evolution of the sequence number for the flows downloading the target A on a SATCOM access

versions as they use the same protocol stack (HTT). The two distributions are indeed located at the same time values and they both show low dispersion. However, with `ChromeQuic` learning about GQUIC availability, HQU is then permanently used for that browser starting from the second load.

We observe the worst performance with HQU where PLT is up to twice longer than `ChromeNoQuic` which uses HTT. The extremely low p values confirm the statistical relevancy of that observation. Each of `ChromeNoQuic` and `ChromeQuic` exhibits a higher PLT dispersion and an intersection of the confidence intervals for loads 2 and 3. It substantiates that above mentioned optimization mechanisms are all performed within one load and we do not expect any further evolution of the PLT with additional loads.

3.2 Packet sequence numbers rate

To understand the performance gap between HQU and HTT, we first focus on the second load of target A and more particularly on the second phase of a PLT, *i.e.*, the time elapsed

[3]This is performed following the method described in [22]. The objective is to get a consistent comparison as BBR is expected to be deployed for TCP on Google's infrastructures [7].

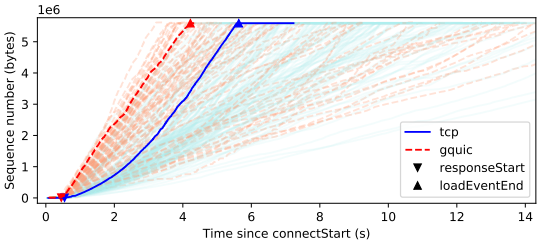


Figure 4: Evolution of the sequence number for the flows downloading the target A on a 4G access network

between the reception of the first response byte and the reception of its last.

To mitigate the issue of encryption keys, we define a sequence number equivalent for GQUIC connections. It is defined as the cumulative sum of the bytes received over the connection, scaled in order to reach the same last value as with TCP. We recognize that the value may present local differences compared to stream offsets embedded in GQUIC packets. Nonetheless, we do believe that global behaviors can be compared.

Figure 3 presents those computations on a subset of HTT and HQU connections. Downward [resp. upward] triangles report the location of TTR [resp. PLT] measurements. We observe that the HQU stack fires the `responseStart` event before HTT. However, the download is completed way after. This can be explained by HTT getting "up-to-speed" and showing a stable and high goodput, while HQU ramps up its transmission rate slowly.

We first focus on HQU. To discriminate the origin of this slow increase between (a) a UDP throughput control from the ISP or (b) the BBR CC itself, we can compute the duration of the BBR *Startup* phase : $t_s = \ln_2(\mathcal{B}/\mathcal{W}_i)\mathcal{R}$, with $\mathcal{B}$ the BDP of the link, $\mathcal{R}$ its Round Trip Time (RTT) and $\mathcal{W}_i$ the IW. For the given parameters of the whole E2E link[4] we obtain $t_s \approx 3.5$s. Compared to expected behavior of BBR we can note on Figure 3 a *Startup* phase of approximatively 4s (between 2s and 6s), followed by three constant-rate segments. The observation is in line with the computed value. With target A only four times bigger than the BDP (Table 2), we can note that the CC above GQUIC spends more than two third of the download in its *Startup* phase. It means that we can expect a similar performance for any other CC using a binary search including TCP New Reno and CUBIC, as long as they are implemented above GQUIC.

In comparison, HTT achieves a near-constant downloading rate, as noted by the quasi-linear increase of sequence numbers. Thanks to a deeper analysis of the traffic, it appears that the TCP path is split into three connections. One for the satellite link and one at each of its edges : connecting the server to the Gateway (GW) and the Satellite Terminal (ST) to the client (see Figure 5). Let's suppose the segment $\text{PEP}_{\text{GW}} \rightarrow \text{PEP}_{\text{ST}}$ uses proprietary protocols and does not require any startup probing phase. The TCP slow start can be neglected for the segment $\text{PEP}_{\text{ST}} \rightarrow$ Client since its RTT is around 1ms. Finally, the binary search is also expected to last less than 1RTT for the segment Server $\rightarrow \text{PEP}_{\text{GW}}$ before the later toggle down the emission with its flow control[5]. Computations are again in line with the observed values as we note that the final constant TCP throughput is reached in less than 50ms on the row data of Figure 3. In our scenario, splitting TCP and using proprietary protocols in the central segment allows each outer segment to present low BDP and thus permit a fast binary search. On the contrary, GQUIC cannot be split because of transport-level Authenticated Encryption with Associated Data (AEAD).

To better explain the poor performances of GQUIC, we run the same computations on a comparative 4G access link. Results are shown in Figure 4. First, we note the HQU stack fires the `responseStart` event before HTT and the median gap is around 90ms. Section 3.3 provides insights for that difference. Second, we can note that the BDP of the path is here lower than the IW. Thus, the *Startup* phase is completed in less than 1RTT. And last, the CCs spend the most part of the load in their *Steady* state because target A is significantly bigger than the BDP (Table 2). On that state, GQUIC shows better performances which is consistent with studies performed on non-split paths [16]. On the geostationary link, the gap in Time To `responseStart` was weak and GQUIC was penalized in its slow start compared to split TCP.

3.3 Focus on the handshake

We just saw that GQUIC encryption prevents any proxy to split the connection, which results in long CC *Startup* phase. But GQUIC was also designed to reduce the handshake duration. In this section, we focus on that phase of the download: from the first packet sent by the client to the first HTTP response bytes received. Results are shown in Figure 6 that presents the notch boxes for the TTR metric.

Loads 2 and 3 show a high dispersion for `ChromeNoQuic` and incompatible confidence intervals for `ChromeQuic`. It questions the hypothesis that learning mechanisms are performed within one load. On a 4G access network (Figure 7), this behavior cannot be seen : we note a gain between load 1 and the followings but loads 2 and 3 show low dispersion and compatible confidence intervals. We assume that

[4]Measured throughput : 25Mbps. Measured RTT : 750ms. Default IW : 32.

[5]Throughput before reaching flow control limitation : 25Mbps. Computed RTT based on the difference of the RTT with the Server and with the GW : 30ms. Default IW : 32.

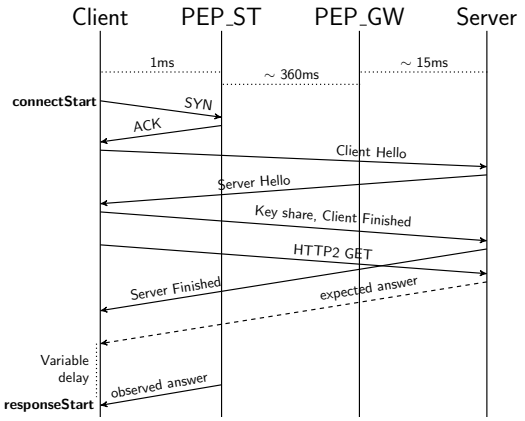


Figure 5: HTTP2-TLS1.2-(split)TCP handshake sequence. TCP handshake duration can be neglected thanks to Performance Enhancing Proxy (PEP)

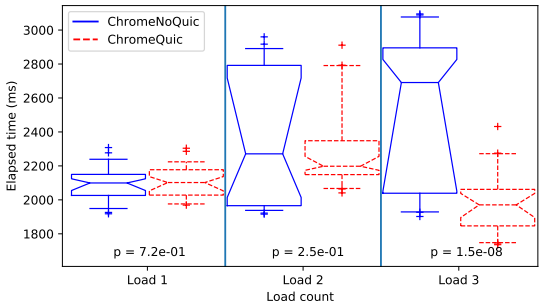


Figure 6: Time to `responseStart` (first HTTP byte) for target A on a SATCOM access

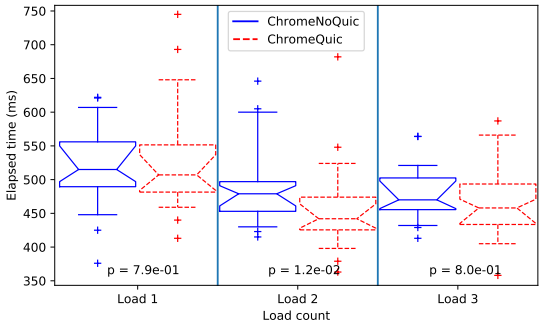


Figure 7: Time to `responseStart` (first HTTP byte) for target A on a 4G access

the SATCOM ISP policy might disturb the results based on recent traffic history.

One could expect that HQU would gain at least one RTT during the handshake, our results do not reflect that expectation. To understand why, we first need to note that neither GQUIC nor TLS1.2 use connection resumption since the browser is restarted each time. However, analysis of traffic indicates that `Chrome` is using TLS1.2 False Start [20]. The observed HTT handshake is presented in Figure 5. The TCP handshake is performed with the immediate PEP on the path. Its duration can be neglected compared to the rest of the sequence. The TLS1.2 Client Hello packet does not suffer any extra delay. So, it appears that the middle segment use a proprietary protocol or already existing connections and no handshake is required between the two PEPs. As a consequence we expect to receive the first byte with the HTT stack (*i.e.* TTR_{HTT}) within twice the E2E RTT. In reality, we observe a high-dispersion extra-delay before receiving the HTTP answer (see Figure 5). We put those results in relation with the same dispersive delay observed in [16, 31] and we could not identify with certainty its origin.

For the HQU stack, since connection resumption is not used, GQUIC will perform 1RTT handshakes (Figure 4 of [21]). Here again we can expect : $\text{TTR}_{\text{HQU}} \approx 2\text{RTT}_{\text{E2E}}$. In conclusion, thanks to the distributed PEPs, to TCP False Start and despite the 1RTT handshake in GQUIC, both the HTT and the HQU stacks present the same theoretical TTR. In practice, HQU is here slightly faster.

To further justify our analysis, we compare the TTR on a 4G access network (Figure 7). In this network, the TCP handshake cannot be neglected anymore because PEP are not deployed. Thus, $\text{TTR}_{\text{HTT}} = 3\text{RTT}_{\text{E2E}}$, *i.e.*, HQU is faster by at least one E2E RTT which is consistent with Figure 4 and Table 1 as the measured TTR gap is around 90 ms. It explains why, when compared to HTT, HQU might present better performances on a mobile network than on a SATCOM access.

3.4 PLT on a small page

In this section, we aim at assessing the impact of the size on the above mentioned conclusions and further analyze the impact of the ratio between the BDP, the IW. Figure 8 presents those results.

Target B can be sent within an initial congestion window (See Table 2): `ChromeQuic` exhibits a better PLT for loads 2 and 3 of target B than `ChromeNoQuic`. Indeed, the CC does not need to probe the link and the main contributor of the PLT should be the TTR. Moreover, as we saw in section 3.3, HQU generally fires the `responseStart` event slightly before HTT.

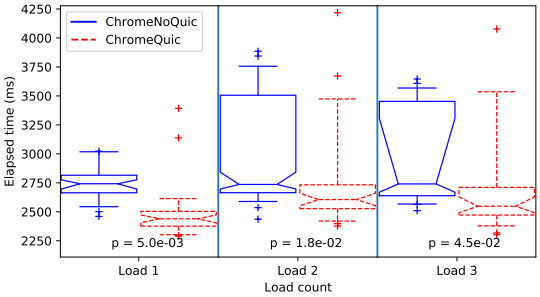


Figure 8: Page Load Time (PLT) for target B

That being said, as opposed to target A, objects of target B are located on `www.google.com`[6]. These objects are downloaded by `ChromeQuic` using an already existing GQUIC connection. This is due to the initial connections to `www.google.com` that `Chrome` opens when starting. They enable `ChromeQuic` to discover GQUIC support and even to reuse the previously opened GQUIC connection to fetch the two objects of page B. *I.e.*, by reducing the duration of several handshakes to several servers, the gains are summed up and the PLT is reduced. However, we note that the gain is limited compared to the PLT difference with target A.

4 RELATED WORK

Performance comparisons of GQUIC and TCP have already been conducted under various network conditions and for various applications [8, 16]. For instance, the authors in [31] have performed an evaluation of GQUIC on an emulated platform with scenarios involving SATCOM in LEO and GEO contexts. They concluded that GQUIC outperforms TCP but their testbed did not include any PEP. On the contrary, our study demonstrates that transparent proxying is the cornerstone of better user QoE with TCP when compared to GQUIC. The impossibility for GQUIC to benefit from the PEP technology has already been identified in [16]. Here again, the authors concluded that GQUIC continues to outperform TCP even when the later is split by a proxy. Our tests on a real access highlight the influence of complex PEP deployment schemes on the comparative performance of both protocols.

5 DISCUSSION

The important variability in applications' requirements makes it hard to define a transport protocol that suits them all. The relevance of application and transport layers protocols depends on both (a) how application data packets are generated and carried out and (b) the network underneath. Taking as example a SATCOM Internet access, the performance of different web applications highly depends on the web pages characteristics [26]. Moreover, for a given application service, QUIC showed a contradictory interest if the network is stable or not [10]. There is no "transport layer silver bullet", *i.e.* a transport protocol that would suit to any application and any network conditions [27].

5.1 ISP point-of-view

For a SATCOM ISP, the deployment of QUIC could be seen as challenging. The opportunity resides in the fact that (1) we can expect interest gains for short files transmission and (2) without the need for PEP, ground segment infrastructures may be cheaper and easier to operate. That being said, the performance gains for large files transmissions, the inadequacy of the end-to-end congestion control to SATCOM links and the rapid evolutivity of the protocol may be seen as a threat to good end-user quality of experience.

5.2 Towards a middlebox-friendly QUIC

To date, while QUIC seems not to be blocked by ISP companies [14], it may not be the case when this UDP traffic becomes a greater part of the Internet traffic. Indeed, for the reasons mentioned in 5.1, quickly evolving protocols may not be easy to deal with, from an ISP point-of-view. Interactions between the end-to-end protocol and the operator middleboxes could be enabled, such as discussed in [18]. Moreover, bits could be made available for the ISP operations, such as load-balancing and statistics on the current flows - "a few bits are enough" [2]. To adapt the congestion control and data rate transmission to a specific SATCOM scenario, more important modifications may be required. They may involve changing the advertised receive window, such in IP-Explicit Rate Notification (IP-ERN) architectures [24], delegate the security to the network operator for better quality of experiences or update the browser [6].

5.3 An Internet for all

SATCOM accesses may still account for a small part of the Internet, they are essential to provide connectivity when other types of accesses can not be made available. This happens when flying over an ocean, living in a rural area or providing Internet access after a natural disaster. This article focuses on geostationary satellite Internet accesses, because geostationary satellite Internet accesses still accounts for most of the satellite-based Internet accesses. Large constellations of satellites projects are not here actually deployed and their economical viability is yet to be proven.

SATCOM systems might encounter performance issues with a protocol that has been designed with a "fiber-to-the-home access" in mind. Moreover, this specific type of access

[6]Given by analysis of the internal log of `Chrome`

may not be the only one with such specificities. Leaving an opened door for in-network operations would allow adaptations for each context; this can be essential for good end-users quality of experience.

REFERENCES

[1] 2018. *New SID on Study on NR Enhancements for TCP.* 3GPP Study Item.

[2] A.Ferrieux, I.Hamchaoui. 2018. *The case for passive measurement in QUIC.* IETF 101 HotRFC presentation.

[3] Toufik Ahmed, Emmanuel Dubois, Jean-Baptiste Dupé, Ramon Ferrùs, Patrick Gélard, and Nicolas Kuhn. [n. d.]. Software-defined satellite cloud RAN. *International Journal of Satellite Communications and Networking* 36, 1 ([n. d.]), 108–133. https://doi.org/10.1002/sat.1206

[4] Angelos Antonopoulos, Chiara Perillo, and Christos Verikoukis. 2017. Internet Service Providers vs. Over-the-Top Companies: Friends or Foes? - Short Talk. *SIGMETRICS Perform. Eval. Rev.* 44, 3 (Jan. 2017), 37–37. https://doi.org/10.1145/3040230.3040242

[5] Jean-Yves Le Boudec. 2011. *Performance Evaluation of Computer and Communication Systems.* EFPL Press.

[6] Viasat Browser. last accessed: 08/08/2018. https://www.exede.com/viasat-browser/

[7] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-Based Congestion Control. *Queue* 14, 5, Article 50 (Oct. 2016), 34 pages. https://doi.org/10.1145/3012426.3022184

[8] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2015. HTTP over UDP. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15.* ACM Press. https://doi.org/10.1145/2695664.2695706

[9] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. 2014. *TCP Fast Open.* RFC 7413. IETF. https://www.rfc-editor.org/rfc/rfc7413.txt

[10] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui. 2017. QUIC: Better for what and for whom?. In *2017 IEEE International Conference on Communications (ICC).* 1–6. https://doi.org/10.1109/ICC.2017.7997281

[11] IETF QUIC Working Group. 2018. QUIC Working Group Website. (sept 2018). https://quicwg.org

[12] Ryan Hamilton. 2014. *QUIC Discovery.* Design Document. The Chromium Projects. https://www.chromium.org/quic

[13] Xiaoxiao Jiang Jae Won Chung, Feng Li. 2017. Driving Linux TCP Congestion Control algorithms around the LTE network Highway. In *NETDEV 2.1.*

[14] Christoph Dietzel Jan Rüth, Ingmar Poese and Oliver Hohlfeld. 2018. A First Look at QUIC in the Wild. In *Proceedings of the 19th Passive and Active Measurement Conference.* 1–6. https://doi.org/10.1007/978-3-319-76481-8_19

[15] Jan Rüth, Christian Bormann, Oliver Hohlfeld. 2018. *On the use of TCP's Initial Congestion Window in IPv4 and by Content Delivery Networks.* IETF 101 presentation.

[16] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a long look at QUIC. In *Proceedings of the 2017 Internet Measurement Conference on - IMC '17.* ACM Press. https://doi.org/10.1145/3131365.3131368

[17] B. H. Kim, D. Calin, and I. Lee. 2017. Enhanced Split TCP with End-to-End Protocol Semantics over Wireless Networks. In *2017 IEEE Wireless Communications and Networking Conference (WCNC).* 1–6. https://doi.org/10.1109/WCNC.2017.7925842

[18] Mirja Kuehlewind and Brian Trammell. 2017. *Manageability of the QUIC Transport Protocol.* Internet-Draft draft-ietf-quic-manageability-01. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-quic-manageability-01.txt

[19] Nicolas Kuhn and Emmanuel Lochin. 2018. *Network coding and satellites.* Internet-Draft draft-kuhn-nwcrg-network-coding-satellites-05. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-kuhn-nwcrg-network-coding-satellites-05.txt

[20] A. Langley, N. Modadugu, and B. Moeller. 2016. *Transport Layer Security (TLS) False Start.* RFC 7918. IETF. https://www.rfc-editor.org/rfc/rfc7918.txt

[21] Adam Langley, Al Riddoch, Alyssa Wilk, Antonio Vicente, Charles 'Buck' Krasic, Cherie Shi, Dan Zhang, Fan Yang, Feodor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Christopher Dorfman, Jim Roskind, Joanna Kulik, Patrik Göran Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, and Wan-Teh Chang. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment.

[22] Dong Mo, Ian Swett, and Nimrod Aviram. 2015. Best practice to test congestion control part of QUIC. (jan 2015). http://bit.ly/2MqnVpy

[23] Nicolas Kuhn. 2017. *MPTCP and BBR performance over Internet satellite paths.* IETF 100 ICCRG presentation.

[24] Dino Martin Lopez Pacheco, Tuan Tran Thai, Emmanuel Lochin, and Fabrice Arnal. 2012. An IP-ERN architecture to enable hybrid E2E/ERN protocol and application to satellite networking. *Computer Networks* 56, 11 (2012), 2700 – 2713. https://doi.org/10.1016/j.comnet.2012.04.011

[25] Google Chromium Projects. 2018. Chromium source. (sept 2018). https://chromium.googlesource.com/chromium/src.git

[26] R. Secchi, A. C. Mohideen, and G. Fairhurst. 2016. Performance analysis of next generation web access via satellite. *International Journal of Satellite Communications and Networking* 36, 1 (dec 2016), 29–43. https://doi.org/10.1002/sat.1201

[27] Anirudh Sivaraman, Keith Winstein, Suvinay Subramanian, and Hari Balakrishnan. 2013. No Silver Bullet: Extending SDN to the Data Plane. In *Twelfth ACM Workshop on Hot Topics in Networks (HotNets-XII).* College Park, MD.

[28] Zhiheng Wang. 2012. *Navigation Timing.* W3C Recommendation. W3C. http://www.w3.org/TR/2012/REC-navigation-timing-20121217/

[29] Wikipedia. 2018. Welch's *t*-test. (sept 2018). https://en.wikipedia.org/wiki/Welch's_t-test

[30] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu. 2011. TCP Congestion Avoidance Algorithm Identification. In *2011 31st International Conference on Distributed Computing Systems.* 310–321. https://doi.org/10.1109/ICDCS.2011.27

[31] Han Zhang, Tianqi Wang, Yue Tu, Kanglian Zhao, and Wenfeng Li. 2019. How Quick Is QUIC in Satellite Networks. In *Communications, Signal Processing, and Systems*, Qilian Liang, Jiasong Mu, Min Jia, Wei Wang, Xuhong Feng, and Baoju Zhang (Eds.). Springer Singapore, 387–394.

Appendix D

Commentaires des évaluateurs sur le papier

Les pages suivantes contiennent la copie du mail de refus du papier, avec les commentaires de revue associés.

ludovic.thomas@woolab.fr

De: EPIQ'18 HotCRP <noreply@epiq18.hotcrp.com>
Envoyé: lundi 8 octobre 2018 13:51
À: Dubois Emmanuel; Emmanuel Lochin; Thomas Ludovic; L. Thomas; Kuhn Nicolas
Cc: lars@eggert.org; ott@in.tum.de
Objet: [EPIQ'18] Rejected submission #8 "QUIC and SATCOM" (poster/demo invite)

Dear authors,

The ACM CoNEXT 2018 Workshop on the Evolution, Performance, and Interoperability of QUIC (EPIQ'18) program committee is sorry to inform you that your submission #8 was rejected, and will not appear at the workshop as a paper.

However, the TPC selected your submission as one of the few to which we'd like to extend the option of presenting your work as a poster or demo at the workshop. Please let us know in the next few days whether you are interested in pursuing this option.

Title: QUIC and SATCOM
Authors: L. Thomas (ISAE-SUPAERO)
E. Dubois (CNES)
N. Kuhn (CNES)
E. Lochin (ISAE-SUPAERO)
Site: https://epiq18.hotcrp.com/paper/8?cap=08aasy_V7aowrg

7 papers were accepted out of 19 submissions.

Reviews and comments on your paper are appended to this email. The submissions site also has the paper's reviews and comments, as well as more information about review scores.

Contact Lars Eggert <lars@eggert.org> and Jörg Ott <ott@in.tum.de> with any questions or concerns.

- EPIQ'18 Submissions

Review #8A
==========

Overall merit
---
4. Accept

Reviewer expertise
---
4. Expert

Paper summary

This paper measures performance of gQUIC over a public SATCOM access compared to the use of TCP with deployed PEPs. As a reference, measurements over a 4G network are evaluated as well.

Strengths

This paper provides a measurement study over a real network and good analysis of the presented results.

Weaknesses

The measurement study is quite narrowly focused. Especially the impact of the used congestion control is not clear and may make the results not comparable. The authors claim that the impact is low, as the flow spends most time in slow start, which is probably true for the satellite link but it would still be interesting to also see measurement results with other congestion control schemes in use (e.g QUIC with cubic). Also is it known which congestion control approach is used on the satellite link?

And of course it would also be interesting to see results for an setup that uses the IETF-QUIC version.

Detailed comments for author
---
1) QUIC is a name, not an acronym (at least for the version developed in the IETF).

2) "Selenium tools add their own flags..." Can you say more? Why is that important?

3) In Figure 8: why is there such a big different for the first load? Shouldn't this both be TCP again?

4) As mention above, it would at least be nice to provide some more background on the use of congestion control on the split satellite link. I guess the first hop from the google server to the one end of the satellite link is also using BBR, while I would guess that the 3rd hop from the other end of the satellite link to the client implements cubic (as default in Linux)? For the satellite link itself (which is the important bit) however you only say that a customized approach is used...

* *

Review #8B
==========

Overall merit
---
2. Weak reject

Reviewer expertise
---
4. Expert

Paper summary

This paper compares the performance QUIC and unclear TCP variant over a SATCOM satellite link. The measured performance is expressed by the page load time of a web site measured in a Chrome browser.

Strengths

- Improving transport protocols for satellite communication is an interesting use case (e.g., currently becoming relevant with on-board wifi services in commercial aircrafts)
- Performance evaluation using a real-world satellite link

Weaknesses
---
1. The evaluation is an apples and oranges comparison; rather than testing QUIC vs.\ TCP, the authors _likely_ test TCP CUBIC vs. BBR in QUIC for high latency links. I say likely, because the paper nowhere mentions which TCP congestion control scheme is used in this evaluation (which appears unknown to the authors).
2. The authors evaluate flow-completion time, not QoE. QoE should be entirely removed from the paper.

3. The evaluation settings are not properly described.
4. (arguably weak) uncited related work on QUIC over satellite links exist

Detailed comments for author
---
1. The evaluation is an apples and oranges comparison; rather than testing QUIC vs.\ TCP, the authors _likely_ test TCP CUBIC vs. BBR in QUIC for high latency links. I say likely, because the paper nowhere mentions which TCP congestion control scheme is used in this evaluation.

The authors compare the performance of QUIC BBR CC to an TCP with an unknown CC and find the performance to be different due to CC. This is a very unsurprising finding and rather unfair comparison. It also flaws the contribution QUIC is slower than TCP. Is is very likely that BBR is slower then CUBIC in your setting, which is then a difference due to the used CC algorithms and _NOT_ due to TCP vs. QUIC.

In this regard, the paper should clearly state which CC is being compared to QUIC BBR. This is likely to be hard, since I assume it to be unknown to the authors (proprietary implementation by the satellite operators). However, in this absence of this knowledge, the results are of questionable value and cannot be properly understood.

What is a fairly interesting finding is the effect of Performance Enhancing Proxies on TCP performance by splitting flows. This is a really interesting setting since - as the authors say - the use of such transparent proxies is no longer possible with QUIC. The authors have hinted that the QUIC performance suffers by not being split. This result, the most interesting in the paper, is only very superficially evaluated. A deeper evaluation would be interesting. It is currently unclear what the lesson learned is, other than not using the PEP lets QUIC suffer.

2. The authors evaluate flow-completion time, not QoE. QoE should be entirely removed from the paper

What is described in this paper is the time to download a 5.3MB vs. a 11kB file (guessing from Table 2, since the workload is sadly not described). What the authors therefore measure is the flow-completion time. Not wrong, but it should be labeled as such.

If the workload is a file only, it is unclear why the authors remote controlled a Google Chrome browser to measure the Page Load Time (PLT). Measuring the PLT would be interesting, if the browser needs to fetch further decencies (e.g., images or to load and render javascripts that then also issue further requests). Such a more complex setting that better represents typical web sites would be a more interesting evaluation setting since the added latency by the satcom link can slow down the request behavior of the browser.

Further, the PLT is NOT QoE. It can be a quality indicator, but PLT does not directly express end-user perception. It can be mapped though, e.g., by using old ITU G.1030 model, which has its own set of limitations. Since the authors did not evaluate QoE, these claims must be removed from the paper entirely.

3. The evaluation settings are not properly described.

As described in 2), the evaluation workload described in Table 2 is unclear. Are these simple files or web sites with decencies?

4. (arguably weak) uncited related work on QUIC over satellite links exist

- Mile High WiFi: A First Look at In-Flight Internet Connectivity (WWW 2018)
  http://www.aqualab.cs.northwestern.edu/publications/354-mile-high-wifi

The contribution of this EPIQ paper is to evaluate QUIC on a real satellite link, whereas the above mentioned paper only measures the latency of satellite links and uses it to asses QUIC in a testbed (due to proxies in in-flight connections).

In summary: The comparison of QUIC to an optimized TCP but unknown TCP variant is not insightful and should not be published. The take aways from this work are not clear, other than the fact that proprietary and not understood PEPs are working well in optimizing TCP performance.

What could be an interesting direction for an interesting paper

The fact that the observed PEPs make it impossible to test TCP variants, can be a motivation for using QUIC variants as measurement tool. By design, QUIC circumvents PEPs and other middleboxes that crated trouble in this evaluation. Thus, QUIC can be a nice tool to evaluate various protocol variants on a number of links, e.g., the SATCOM link utilized by the authors. This is because QUIC will surely not be optimized by the observed PEPs and thus would allow the authors to study protocol effects directly. This would make an interesting paper.

* *

Review #8C
==========

Overall merit
---
1. Reject

Reviewer expertise
---
2. Some familiarity

Paper summary

Using very limited testing of page load time and downloading elephant flows over an unnamed commercial SATCOM network, the paper concludes that QUIC in Chrome is inferior to a TCP-based Chrome browser. The paper uses a very limited and artificial test case and is trying to test a moving target, Google's QUIC protocol, which is under active development. Hence, sweeping conclusions seem unwarranted.

Strengths

Studies seem to described well.

Weaknesses

Language is non-standard and confusing ("heavy page load"), last paragraph of page 4 first column "Thanks to a deeper analysis of the traffic" where "deeper analysis" is unexplained, explicitly tried for a lightly loaded network, fails to understand that moving away from PEPs is considered a positive thing in the industry.

Detailed comments for author

QUIC is a protocol under active development and Google is constantly responding to issues and making improvements. Thus, the current release version and the version you tested are not going to be the same. This raises the issue of what is the point of this work? This appears to be a very limited issue which might better have been raised on an IETF or a QUIC developers mailing list.

What do you think might happen under more congested conditions? When you say you did "deeper analysis" how did you do it and what tools were employed?

Figure 3 appears to show QUIC's use of BBR ramping up slowly as it probes for the bottleneck bandwidth vs a faster ramp up. Do you know what dynamics are behind that and whether they are still the way QUIC operates?

SATCOM is moving toward heavy FEC to remove errors rather than PEPs. Does this affect your conclusions?

* *

Review #8D
==========

Overall merit
---
2. Weak reject

Reviewer expertise
---
2. Some familiarity

Paper summary

This paper contains empirical measurements of QUIC's behavior over satellite links. In these measurements, QUIC has comparable performance to H2 on small files, but substantially worse performance on large files, which the authors attribute to the presence of a TCP PEP.

Strengths

The best part about this paper is that it contains real field measurements on a live network. That's useful information.

Weaknesses

The choice to use commodity implementations where the authors don't really have access to the details makes it very hard to interpret these results. Specifically:

- The authors use Chrome directly, which leads to some hard to explain noise.
- The authors don't really explain the nature of the PEP device in the network

Detailed comments for author

This paper would be far stronger if instead of using stock Chrome you used a local gQUIC and HTTPS client. This would let you isolate the influences of the session cache, etc. (so you wouldn't need to look at the first connection) as well as attempts to fetch other resources. It would also let you have the traffic keys so you could directly look at what's going on with QUIC rather than having to infer it from packet traces.

The browser events you are using are not intended for the purpose you are using them for and are gated on all sorts of other things (e.g., JS parsing time, loading subresources, etc.). As above, just instrumenting the network stack would give you much more information.
DNS resolution time is another factor.

Can you provide the specific URLs you are using here?

This paper would be much stronger if you were able to determine from your carrier what the exact PEP topology they have in place here is.

The very high dispersion in loads 2 and 3 for HTT is very concerning.
That makes all the results for that scenario very questionable.
They can't really be trusted without an explanation.

Bibliography

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4):34–41, 2005. ISSN: 0018-9162. DOI: 10.1109/MC.2005.136.

[2] O. Anstett. Internet broadband: a new source of growth. URL: https://www.eutelsat.com/files/contributed/investors/pdf/Capital-Markets-Day-2015/Internetbroadband_a_new_source_of_growth.pdf.

[3] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, May 2015. DOI: 10.17487/RFC7540. URL: https://rfc-editor.org/rfc/rfc7540.txt.

[4] E. Blanton, D. V. Paxson, and M. Allman. TCP Congestion Control. RFC 5681, Sept. 2009. DOI: 10.17487/RFC5681. URL: https://rfc-editor.org/rfc/rfc5681.txt.

[5] J.-Y. L. Boudec. *Performance Evaluation of Computer and Communication Systems*. EFPL Press, 2011. ISBN: 1439849927, 9781439849927.

[6] V. Browser. In last accessed: 08/08/2018. URL: https://www.exede.com/viasat-browser/.

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. Bbr: congestion-based congestion control. *Queue*, 14(5):50:20–50:53, Oct. 2016. ISSN: 1542-7730. DOI: 10.1145/3012426.3022184. URL: http://doi.acm.org/10.1145/3012426.3022184.

[8] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. Bbr: congestion-based congestion control. *ACM Queue*, 14, September-October:20 –53, 2016. URL: http://queue.acm.org/detail.cfm?id=3022184.

[9] G. Carlucci, L. D. Cicco, and S. Mascolo. HTTP over UDP. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*. ACM Press, 2015. DOI: 10.1145/2695664.2695706. URL: https://doi.org/10.1145/2695664.2695706.

[10] Centre National d'Etudes Spatiales. Openbach. NET4SAT, editor. Last accessed : 2018-10-19. URL: http://www.openbach.org.

[11] Centre National d'Etudes Spatiales and Thales Alenia Space. Opensand. NET4SAT, editor. Last accessed : 2018-10-19. URL: http://www.opensand.org.

[12] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP Fast Open. RFC 7413, IETF, 2014. URL: https://www.rfc-editor.org/rfc/rfc7413.txt.

[13] J. Crowcroft, I. Wakeman, Z. Wang, and D. Sirovica. Is layering, harmful? (remote procedure call). *IEEE Network*, 6(1):20–24, 1992. DOI: 10.1109/65.120719. URL: https://doi.org/10.1109/65.120719.

[14] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF, 2008. URL: https://www.rfc-editor.org/rfc/rfc5246.txt.

[15] W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, Aug. 2007. DOI: 10.17487/RFC4987. URL: https://rfc-editor.org/rfc/rfc4987.txt.

[16] Eutelsat. Ka-sat. Last accessed : 2018-10-18. URL: https://www.eutelsat.com/fr/satellites/flotte/EUTELSAT-KA-SAT.html.

[17] A Ferrieux and I Hamchaoui. Spin bit and beyond. QUIC WG Interim presentation, 2018. URL: https://github.com/quicwg/wg-materials/raw/master/interim-18-09/Spin%20bit%20and%20beyond%20%40%20Orange%20v1.pdf.

[18] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, Sept. 2001. DOI: 10.17487/RFC3168. URL: https://rfc-editor.org/rfc/rfc3168.txt.

[19] Google Chromium Projects. Chromium source. Google, editor, Website, 2018. URL: https://chromium.googlesource.com/chromium/src.git.

[20] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC 3135, June 2001. DOI: 10.17487/RFC3135. URL: https://rfc-editor.org/rfc/rfc3135.txt.

[21] S. Ha and I. Rhee. Taming the elephants: new TCP slow start. *Computer Networks*, 55(9):2092–2110, 2011. DOI: 10.1016/j.comnet.2011.01.014. URL: https://doi.org/10.1016/j.comnet.2011.01.014.

[22] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008. ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. URL: http://doi.acm.org/10.1145/1400097.1400105.

[23] R. Hamilton. QUIC Discovery. Design Document, The Chromium Projects, Oct. 2014. URL: https://www.chromium.org/quic.

[24] P. E. Hoffman and S. R. Harris. The tao of ietf - a novice's guide to the internet engineering task force. RFC 4677, 2006. DOI: 10.17487/RFC4677. URL: https://rfc-editor.org/rfc/rfc4677.txt.

[25] C. Huitema and al. Github picoquic. Private Octopus, editor. Last accessed : 2018-10-18. URL: https://github.com/private-octopus/picoquic/.

[26] IAB and L. Daigle. The RFC Series and RFC Editor. RFC 4844, July 2007. DOI: 10.17487/RFC4844. URL: https://rfc-editor.org/rfc/rfc4844.txt.

[27] IAB, M. J. Handley, and E. Rescorla. Internet Denial-of-Service Considerations. RFC 4732, Dec. 2006. DOI: 10.17487/RFC4732. URL: https://rfc-editor.org/rfc/rfc4732.txt.

[28] IAB and O. M. Kolkman. RFC Editor Model (Version 1). RFC 5620, Aug. 2009. DOI: 10.17487/RFC5620. URL: https://rfc-editor.org/rfc/rfc5620.txt.

[29] ICAO Safety. Api data service. Last accessed : 2018-10-19. URL: https://www.icao.int/safety/iStars/Pages/API-Data-Service.aspx.

[30] IETF QUIC Working Group. Quic working group website. IETF, editor, Website, 2018. URL: https://quicwg.org.

[31] International council on systems engineering website, last accessed: 29/09/2018. URL: http://www.incose.org/.

[32] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-Draft draft-ietf-quic-transport-15, Internet Engineering Task Force, Oct. 2018. 134 pages. URL: https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-15. Work in Progress.

[33] V. Jacobson. Congestion avoidance and control. In *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, pages 314–329, Stanford, California, USA. ACM, 1988. ISBN: 0-89791-279-9. DOI: 10.1145/52324.52356. URL: http://doi.acm.org/10.1145/52324.52356.

[34] X. J. Jae Won Chung Feng Li. Driving linux tcp congestion control algorithms around the lte network highway. In *NETDEV 2.1*, 2017.

[35] C. D. Jan Rüth Ingmar Poese and O. Hohlfeld. A first look at quic in the wild. In *Proceedings of the 19th Passive and Active Measurement Conference*, pages 1–6, 2018. ISBN: 978-3-319-76481-8. DOI: 10.1007/978-3-319-76481-8_19.

[36] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove. Taking a long look at QUIC. In *Proceedings of the 2017 Internet Measurement Conference on - IMC '17*. ACM Press, 2017. DOI: 10.1145/3131365.3131368. URL: https://doi.org/10.1145/3131365.3131368.

[37] B. H. Kim, D. Calin, and I. Lee. Enhanced split tcp with end-to-end protocol semantics over wireless networks. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2017. DOI: 10.1109/WCNC.2017.7925842.

[38] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach, Global Edition*. Pearson Education Limited, 2016. ISBN: 1292153598.

[39] A. Langley, N. Modadugu, and B. Moeller. Transport Layer Security (TLS) False Start. RFC 7918, IETF, 2016. URL: https://www.rfc-editor.org/rfc/rfc7918.txt.

[40] A. Langley and W.-T. Chan. QUIC Crypto. Design Document, The Chromium Projects, Dec. 2016. URL: https://www.chromium.org/quic.

[41] A. Langley, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, A. Riddoch, W.-T. Chang, Z. Shi, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, and I. Swett. The QUIC transport protocol. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM '17*. ACM Press, 2017. DOI: 10.1145/3098822.3098842. URL: https://doi.org/10.1145/3098822.3098842.

[42] P. Manière. Orange recrute des clients chez sfr, mais pas seulement. *La Tribune*, 2018. URL: https://www.latribune.fr/technos-medias/orange-recrute-des-clients-chez-sfr-mais-pas-seulement-fabienne-dulac-771000.html.

[43] G. Maral and M. Bousquet. *Satellite Communications Systems*. John Wiley & Sons, Ltd, 2009. DOI: 10.1002/9780470834985. URL: https://doi.org/10.1002/9780470834985.

[44] MAWI Working Group. Packet traces from wide backbone. MAWI, editor. Last accessed : 2018-10-17. URL: http://mawi.nezu.wide.ad.jp/mawi/.

[45] D. Meyer and R. Bush. Some Internet Architectural Guidelines and Philosophy. RFC 3439, Dec. 2002. DOI: 10.17487/RFC3439. URL: https://rfc-editor.org/rfc/rfc3439.txt.

[46] Oneweb World. Oneweb website. Oneweb.World, editor, Website, 2018. URL: http://www.oneweb.world.

[47] D. M. L. Pacheco, T. T. Thai, E. Lochin, and F. Arnal. An ip-ern architecture to enable hybrid e2e/ern protocol and application to satellite networking. *Computer Networks*, 56(11):2700 –2713, 2012. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2012.04.011. URL: http://www.sciencedirect.com/science/article/pii/S138912861200151X.

[48] G. Poncet. Cyberattaque, pourquoi les câbles sous-marins sont le maillon faible. *Le Point*, 2017. URL: https://www.lepoint.fr/high-tech-internet/cyberguerre-les-cables-sous-marins-le-maillon-faible-28-01-2017-2100734_47.php.

[49] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, Aug. 2018. DOI: 10.17487/RFC8446. URL: https://rfc-editor.org/rfc/rfc8446.txt.

[50] J. Roskind. Quic: design document and specification rationale. Google, editor, Online document. Last accessed:2018-10-11. URL: https://docs.google.com/document/d/1RNHkx_VvKWyWg6Lr8SZ-saqsQx7rFV-ev2jRFUoVD34/.

[51] L. A. Sanchez, M. Allman, and D. D. Glover. Enhancing TCP Over Satellite Channels using Standard Mechanisms. RFC 2488, Jan. 1999. DOI: 10.17487/RFC2488. URL: https://rfc-editor.org/rfc/rfc2488.txt.

[52] W. A. Simpson. TCP Cookie Transactions (TCPCT). RFC 6013, Jan. 2011. DOI: 10.17487/RFC6013. URL: https://rfc-editor.org/rfc/rfc6013.txt.

[53] A. Sivaraman, K. Winstein, S. Subramanian, and H. Balakrishnan. No silver bullet: extending sdn to the data plane. In *Twelfth ACM Workshop on Hot Topics in Networks (HotNets-XII)*, College Park, MD, 2013.

[54] Skylogic. Skylogic's website. Last accessed : 2018-10-18. URL: http://skylogic.com.

[55] The Chromium Projects. Spdy : project's home page. Google, editor, Online document. Last accessed:2018-10-12. URL: https://dev.chromium.org/spdy.

[56] M. Thomson and S. Turner. Using Transport Layer Security (TLS) to Secure QUIC. Internet-Draft draft-ietf-quic-tls-16, Internet Engineering Task Force, Oct. 2018. 32 pages. URL: https://datatracker.ietf.org/doc/html/draft-ietf-quic-tls-16. Work in Progress.

[57] Z. Wang. Navigation Timing. W3C Recommendation, W3C, Dec. 2012. URL: http://www.w3.org/TR/2012/REC-navigation-timing-20121217/.

[58] Wikipedia. Starlink (satellite constellation) — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Starlink%20(satellite%20constellation)&oldid=861727897, 2018. [Online; accessed 02-October-2018].

[59] T. Y. C. Woo, R. Bindignavle, S. Su, and S. S. Lam. Snp: an interface for secure network programming. In *IN PROCEEDINGS OF USENIX'94 SUMMER TECHNICAL CONFERENCE*, pages 45–58, 1994.

[60] E. Wyatt and N. Cohen. Comcast and netflix reach deal on service. *The New York Times*, 2014. URL: https://www.nytimes.com/2014/02/24/business/media/comcast-and-netflix-reach-a-streaming-agreement.html.

[61] H. Zhang, T. Wang, Y. Tu, K. Zhao, and W. Li. How quick is quic in satellite networks. In Q. Liang, J. Mu, M. Jia, W. Wang, X. Feng, and B. Zhang, editors, *Communications, Signal Processing, and Systems*, pages 387–394. Springer Singapore, 2019. ISBN: 978-981-10-6571-2.