2022-01-27 JANOG49

QUICについて話そう

Yuya Kawakami (@yuyarin), SoftBank
Kazuho Oku (@kazuho), Fastly

SoftBank

このプログラムの経緯

JANOG48でライトニングトークをしたところ高評価を頂きましたが、LTで議論ができなかったのでプログラム応募してみました

登壇者

川上 雄也 (@yuyarin)

ソフトバンク株式会社

Senior Network Architect

奥 一穂 (@kazuho)

Fastly K.K.

JANOG48 LTのきっかけ

うえええ、ルーターのNATテーブルを圧迫してるの、
LAN内部に設置したDNSサーバーだけかと思ったら、
QUICか！どうりで、googleとかQUICに対応してるサイト使うとトラブル起きる訳だ。。

1:22 PM · Jun 7, 2021 · twitcle plus

450 Retweets **25** Quote Tweets **836** Likes

このプログラムの目的

QUICは新しいパラダイムのプロトコルなので
既存のネットワークインフラとギャップがある

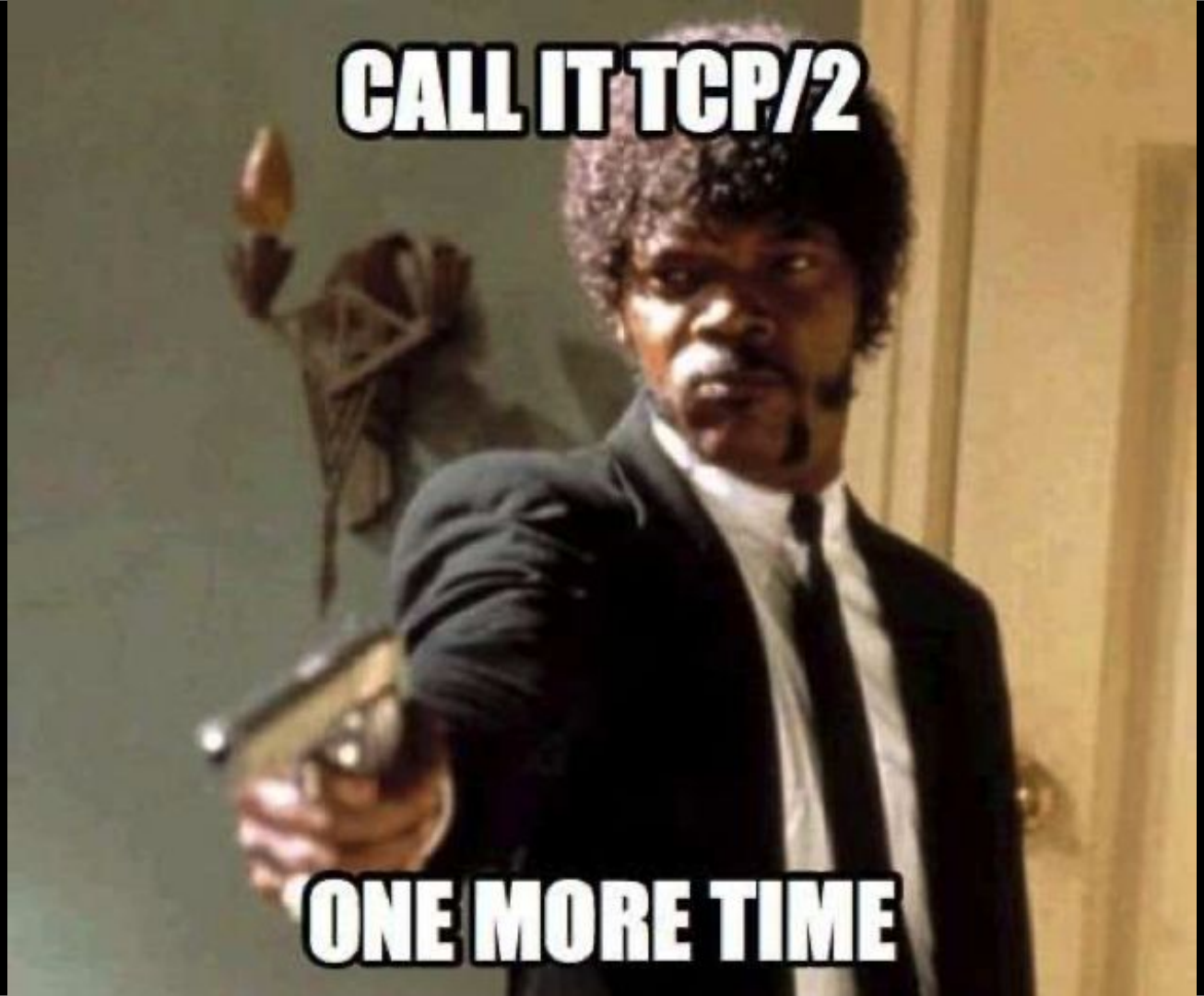
QUICの理念や特性をネットワークオペレータが理解して、
QUICを活かすネットワーク運用ができるようになりたい

プログラムの流れ

- イントロ
- QUICの紹介 by @kazuhoさん
- QUICがネットワークに与える主なインパクトの整理
- FAQ
- 議論

QUICの紹介

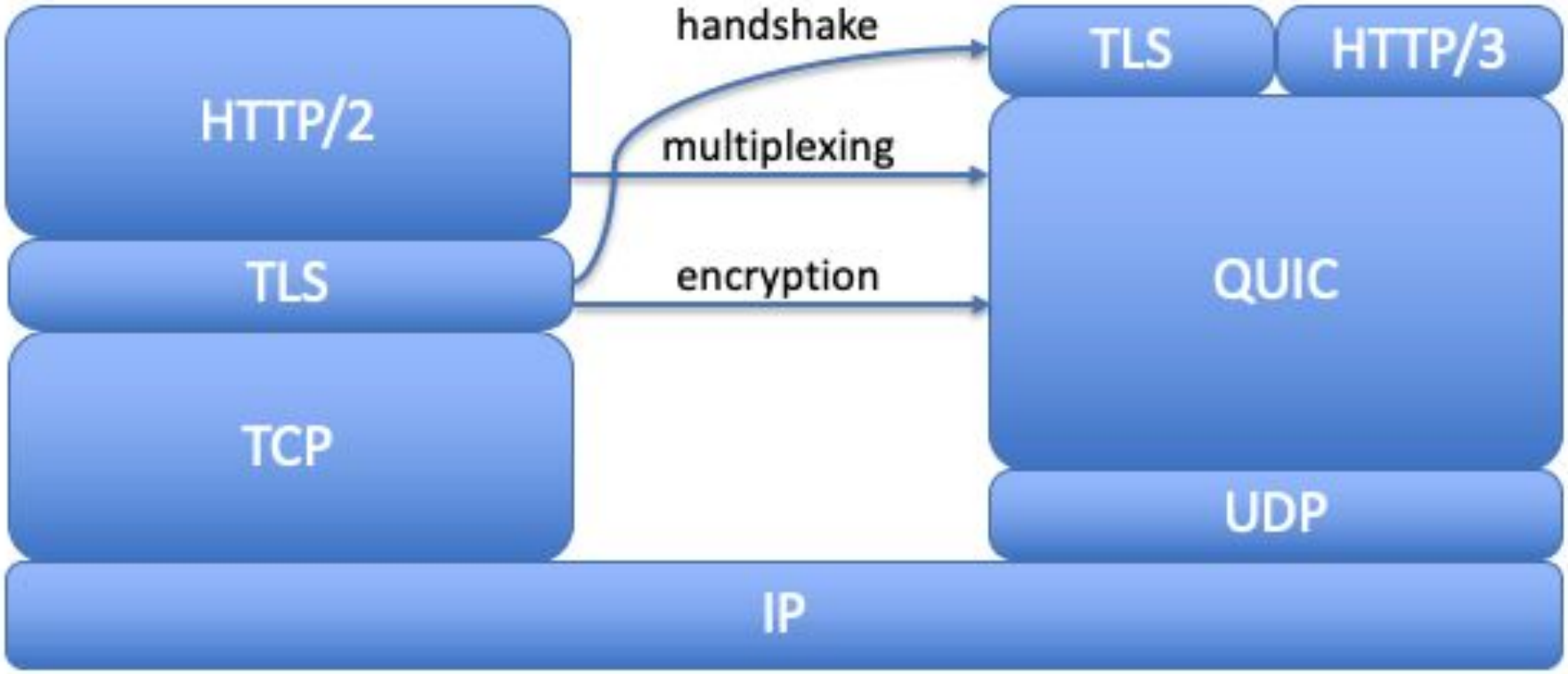
QUICとは?


CALL IT TCP/2
ONE MORE TIME

QUICとは?

- 新しい暗号化されたトランスポート層プロトコル
  - UDPベース
  - Googleの実験をもとにIETFで標準化
- TCP + TLSの性能を改善
  - 接続確立にかかる時間を最小化
  - 再送制御の高度化
  - ヘッドオブラインブロッキングの発生を防止
- 新しい機能
  - ミグレーション

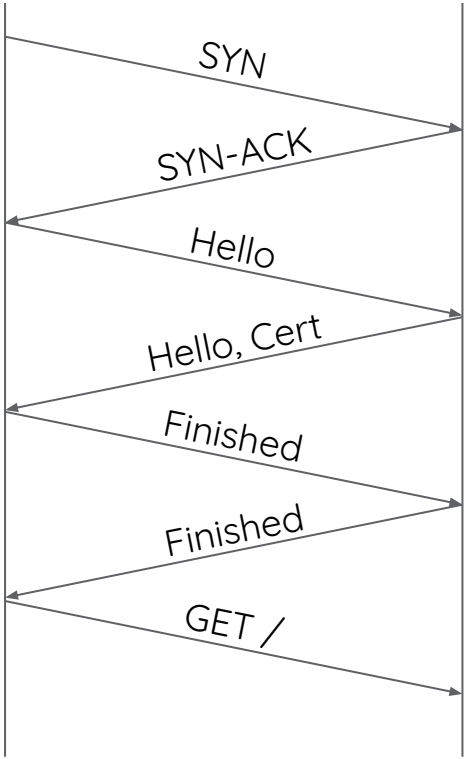
QUICのレイヤ構成


HTTP/2
TLS
TCP
handshake
multiplexing
encryption
TLS
HTTP/3
QUIC
UDP
IP

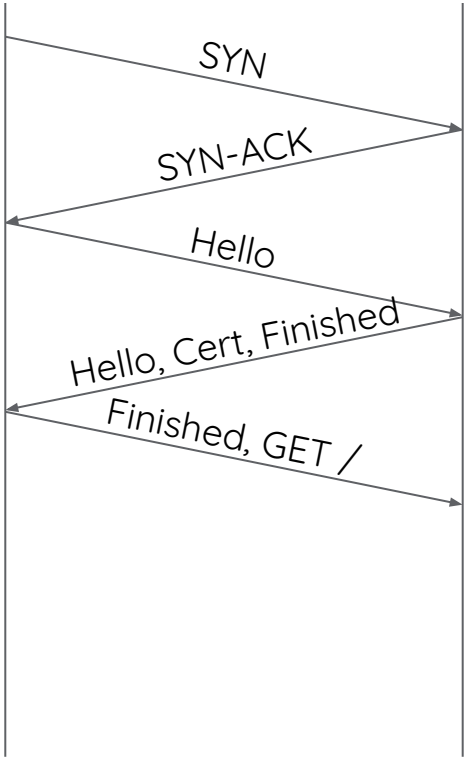
QUICの標準化

- QUIC: 2021年5月完了
  - RFC 8999 - Invariants
  - RFC 9000 - Transport
  - RFC 9001 - Using TLS to Secure QUIC
  - RFC 9002 - Recovery
- HTTP/3:
  - HTTP/3 - RFC Ed Queue
  - QPACK - RFC Ed Queue
  - Extensible Priorities - RFC Ed Queue

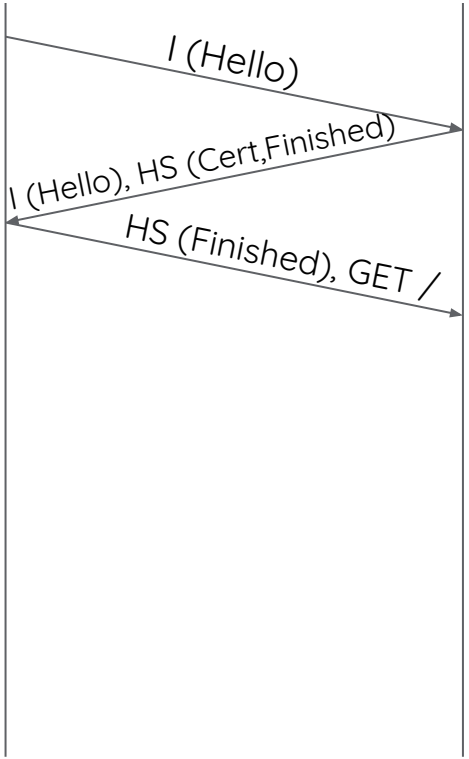
接続確立にかかる時間を最小化



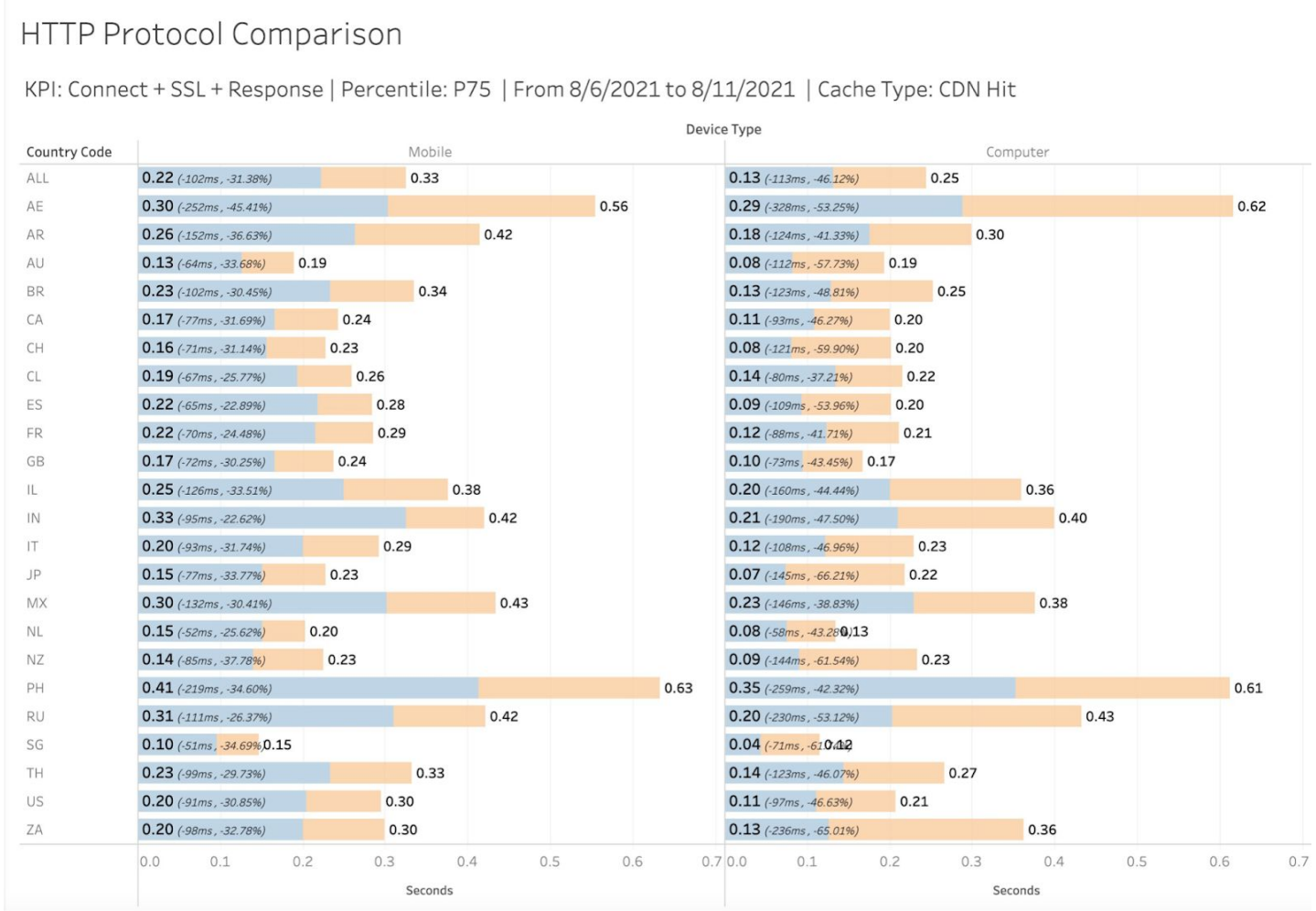
TCP + TLS 1.2



TCP + TLS 1.3



QUIC



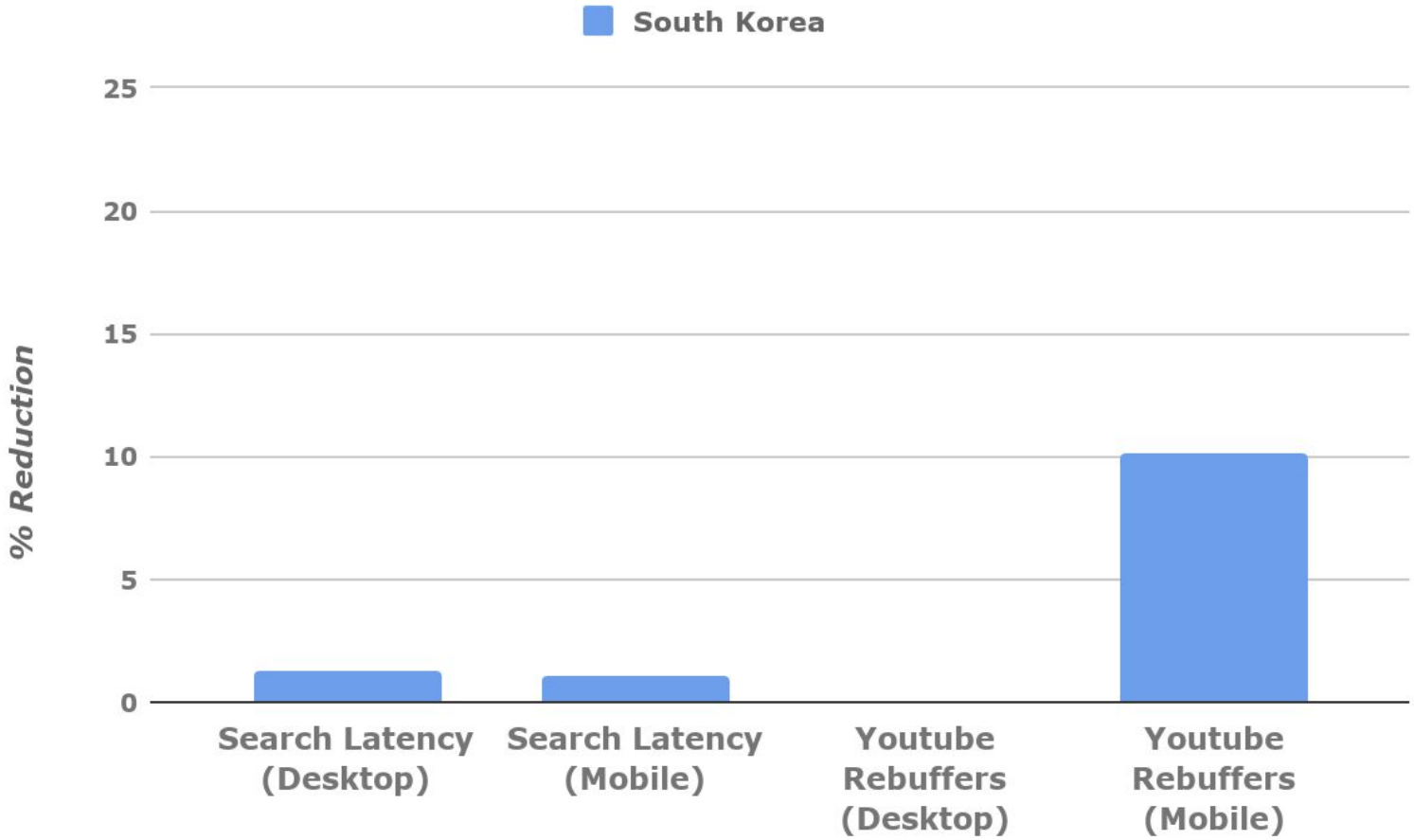
Source: https://twitter.com/alonkochba/status/1425851135265492993

再送制御の高度化

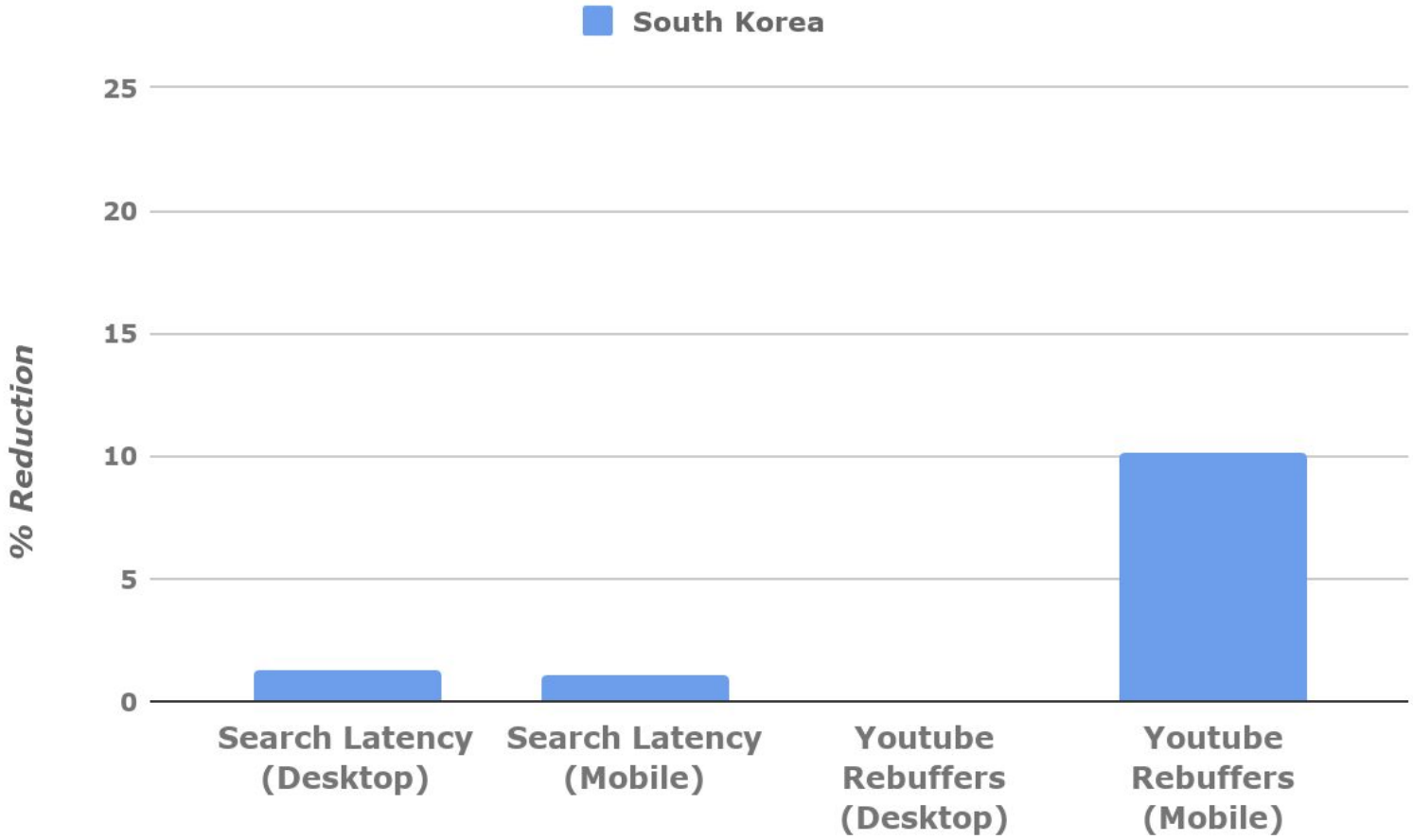
- TCP ACK
  - 先頭から何バイト目まで受信したか（シーケンス番号）
    → ロスしたか、配送遅れか区別がつかない
  - 欠落通知は最大４ヶ所（SACK）
    → 4ヶ所では足りない
  - ACKの圧縮（とりまとめ）による遅延と経路遅延の区別がつかない
    → 再送タイミングのチューニングが難しい

再送制御の高度化

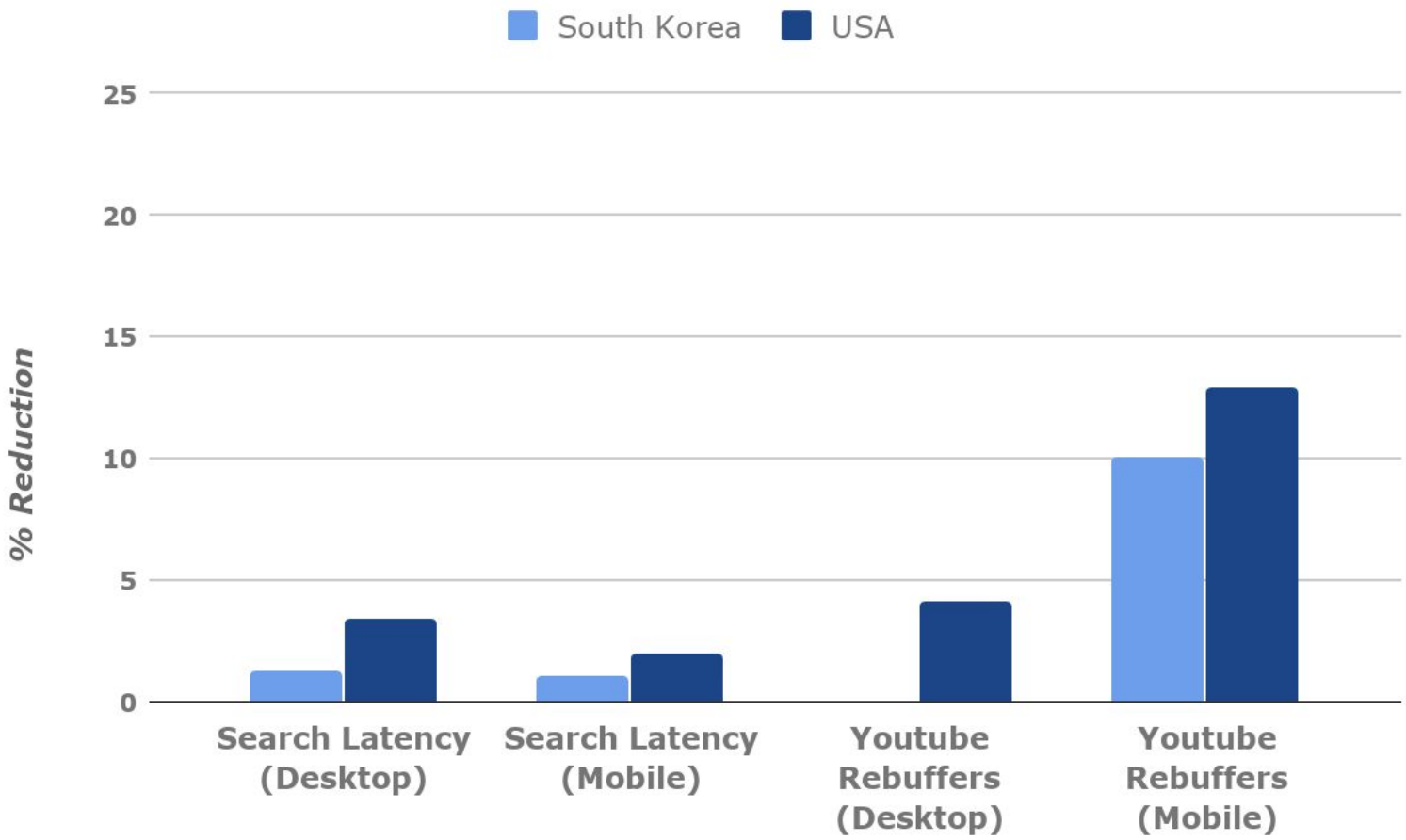
- QUIC ACK
  - 再送の際も変化する「パケット番号」を利用
    → ロスと配送遅れの区別がつく
  - 欠落通知は100ヶ所以上
  - ACKに、エンドポイント内遅延を表すack delayフィールドを追加
    → RTT測定の精度が向上



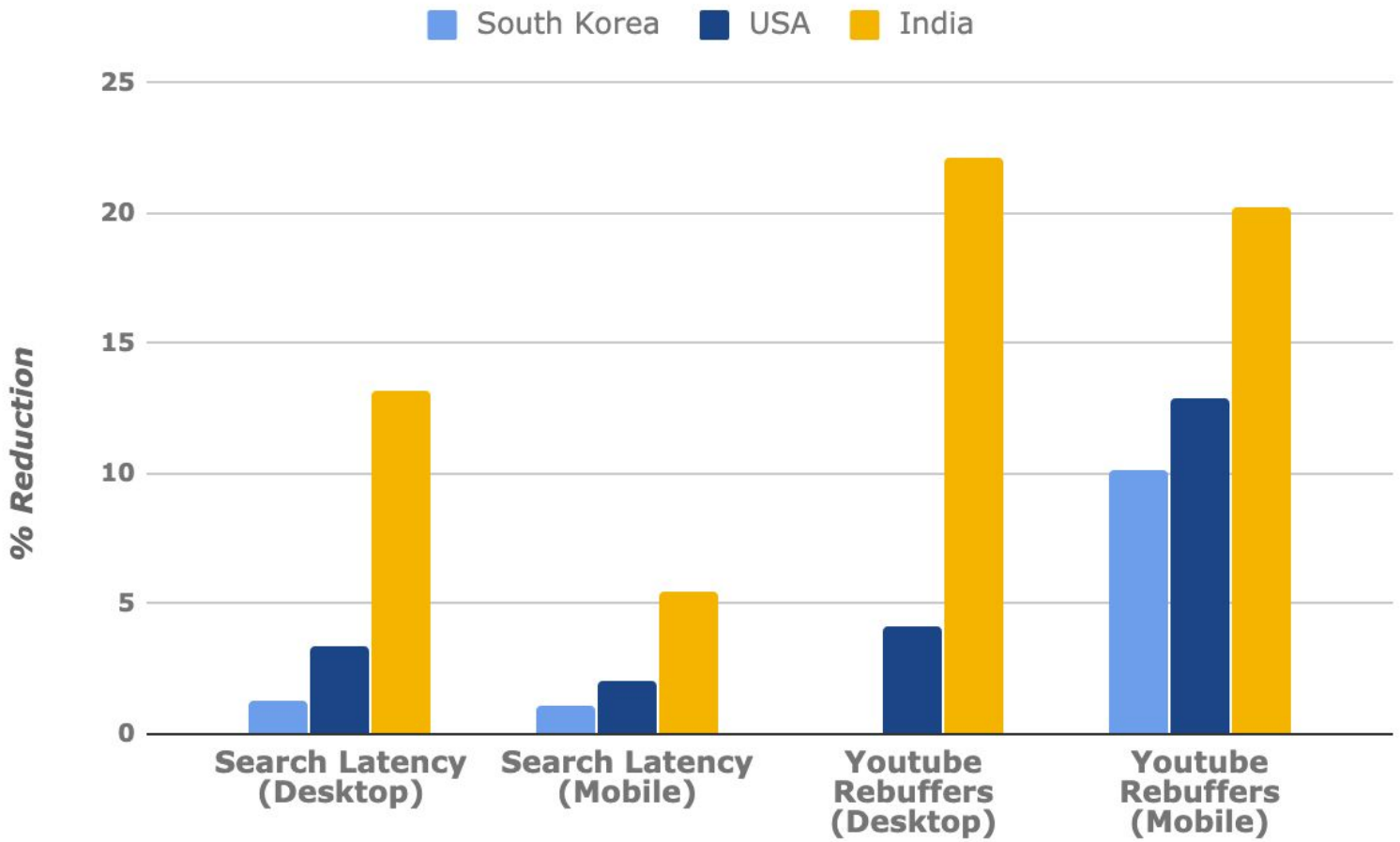
Source: https://research.google.com/pubs/archive/46403.pdf



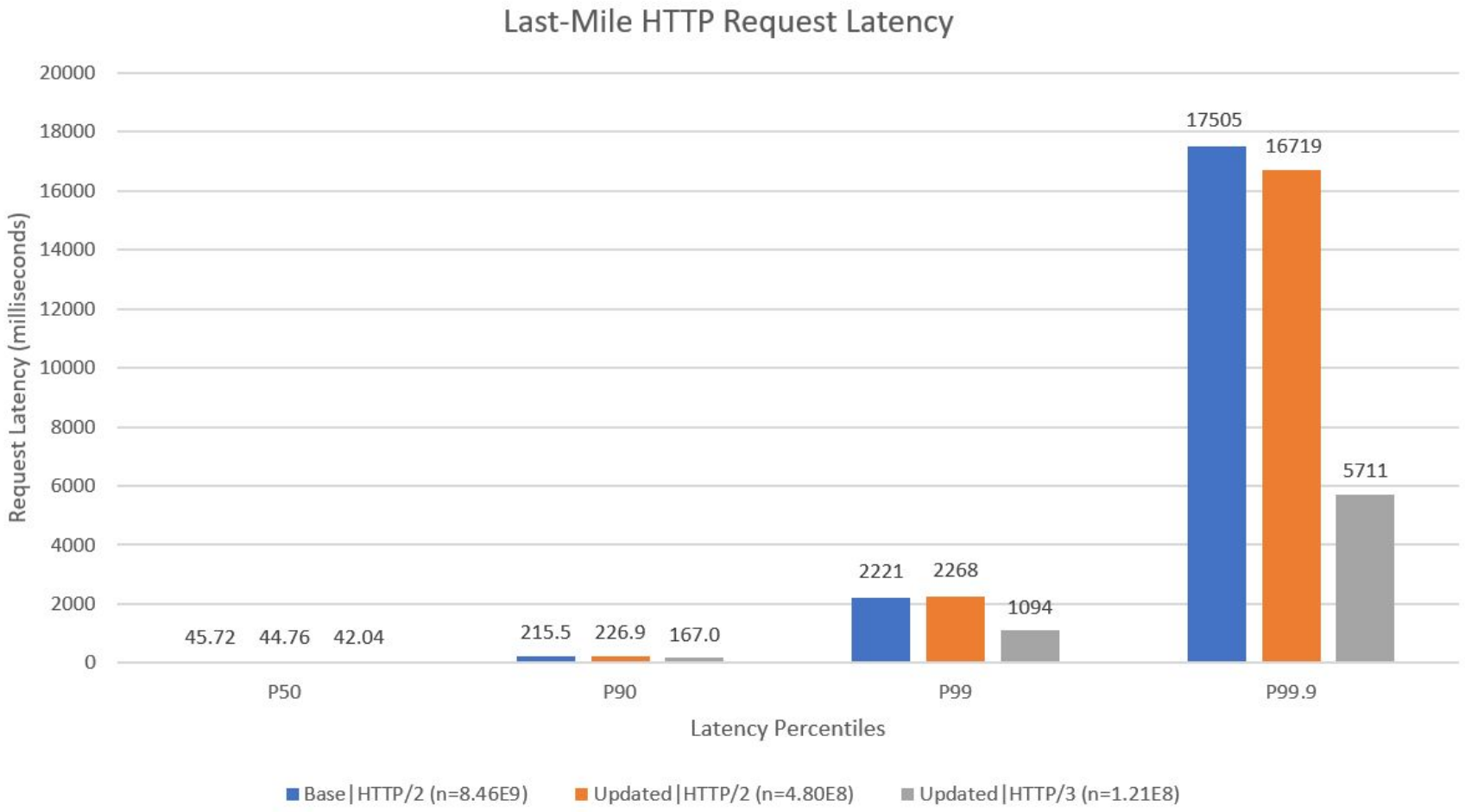
Source: https://research.google.com/pubs/archive/46403.pdf



Source: https://research.google.com/pubs/archive/46403.pdf



Source: https://research.google.com/pubs/archive/46403.pdf

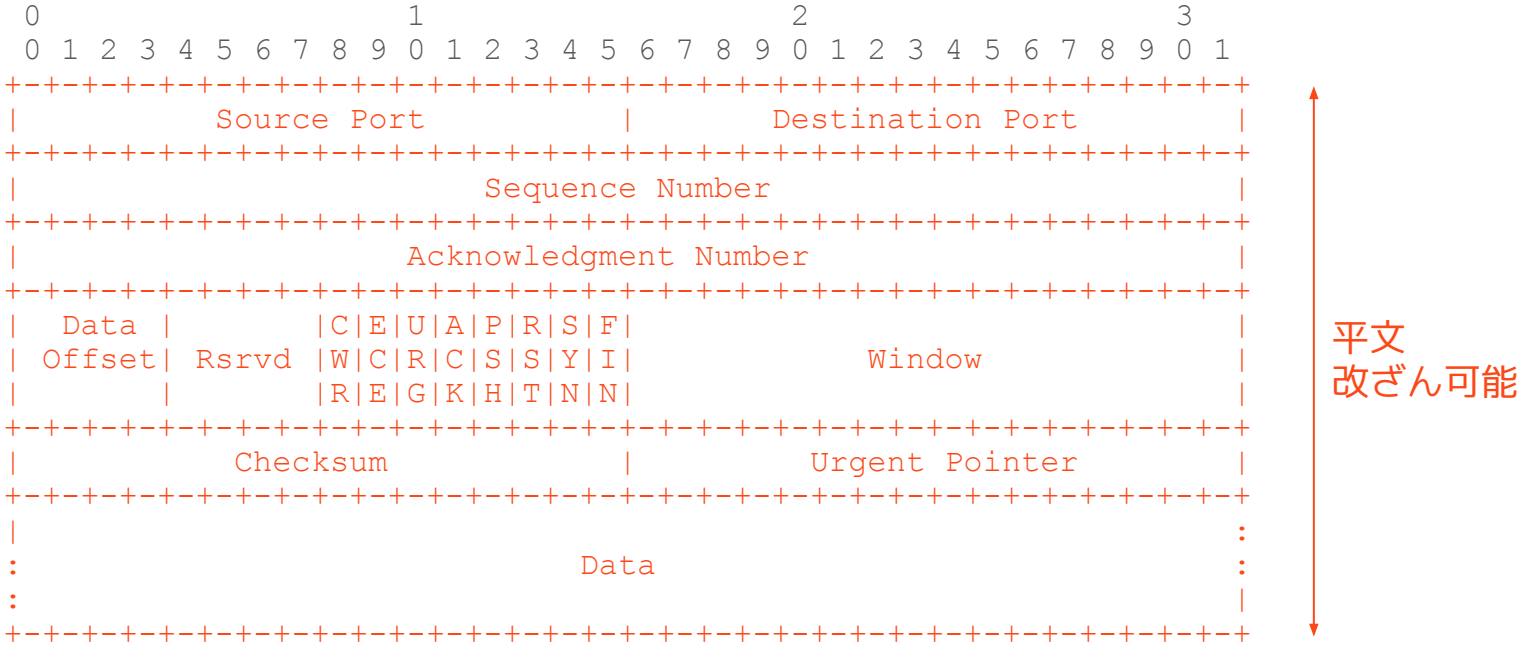


Source: https://techcommunity.microsoft.com/t5/networking-blog/deploying-http-3-on-windows- server-at-scale/ba-p/2839394

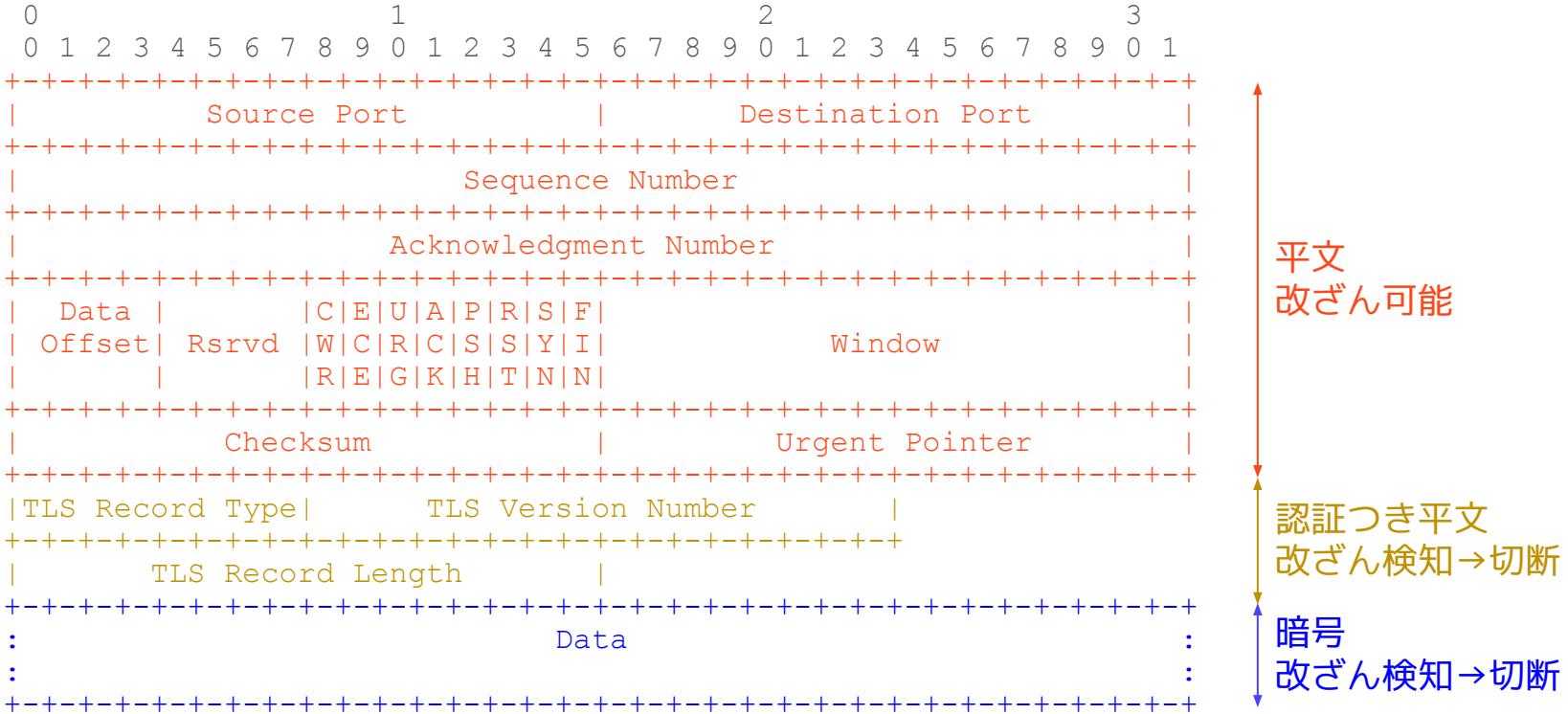
QUICとは?

- 新しい暗号化されたトランスポート層プロトコル
  - UDPベース
  - Googleの実験をもとにIETFで標準化
- TCP + TLSの性能を改善
  - 接続確立にかかる時間を最小化
  - 再送制御の高度化
  - ヘッドオブラインブロッキングの発生を防止
- 新しい機能
  - ミグレーション

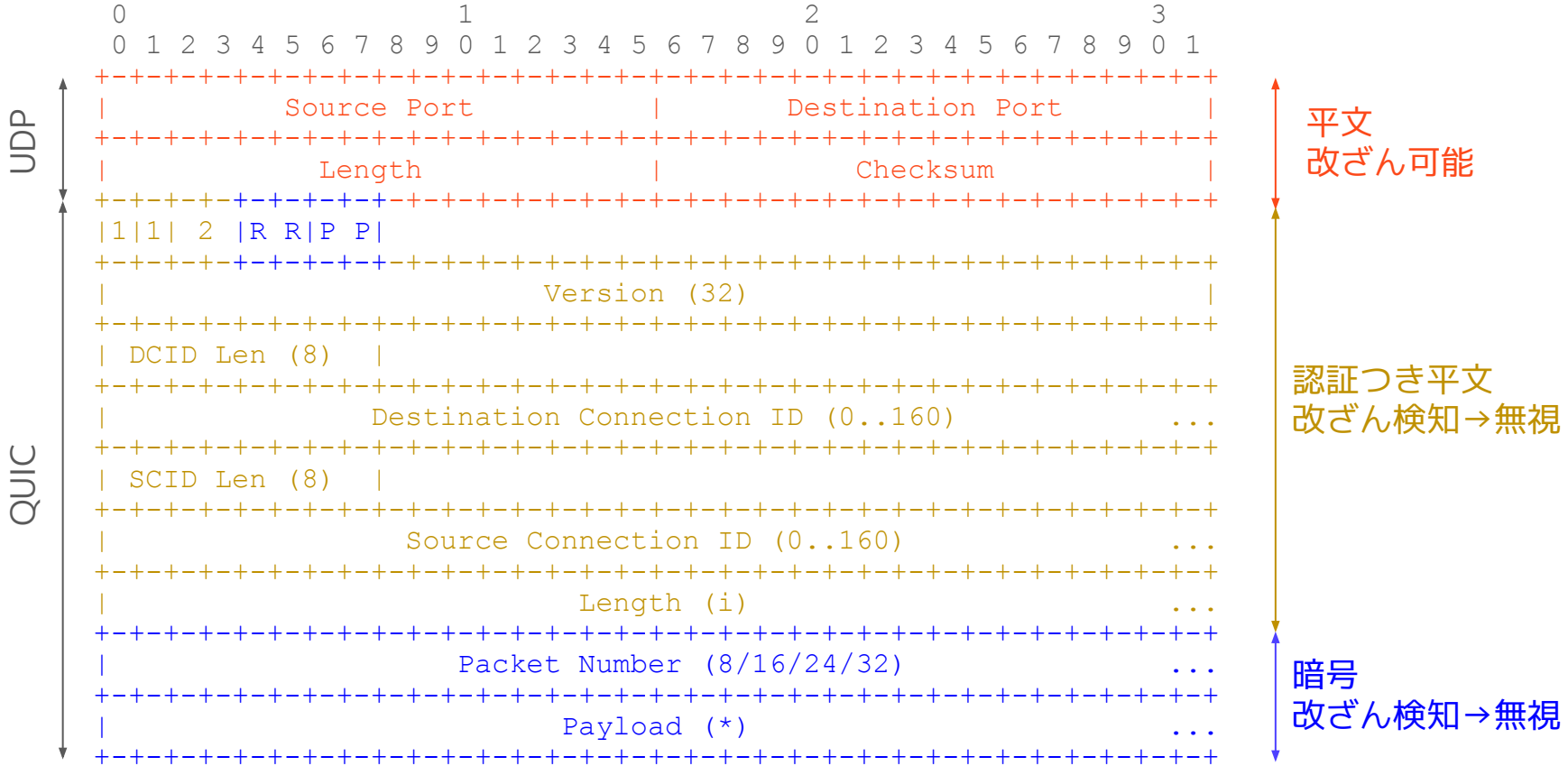
TCPパケット


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Rsrvd
C|E|U|A|P|R|S|F
W|C|R|C|S|S|Y|I
R|E|G|K|H|T|N|N
Window
Checksum
Urgent Pointer
Data
平文
改ざん可能

TCPパケット + TLS


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Rsrvd
C|E|U|A|P|R|S|F
W|C|R|C|S|S|Y|I
R|E|G|K|H|T|N|N
Window
Checksum
Urgent Pointer
TLS Record Type
TLS Version Number
TLS Record Length
Data
平文
改ざん可能
認証つき平文
改ざん検知→切断
暗号
改ざん検知→切断

QUIC Handshake Packet



なぜ「暗号化トランスポート」なのか

- 攻撃耐性
  - TCPは、man-on-the-side攻撃[注]で切断可能
    - TCPの上で動くTLSでは、防ぎようがない
  - QUICは、暗号を使ってパケットの改ざんを検知・破棄
    - Man-on-the-side攻撃が困難
      - ハンドシェイク時のみタイミング依存な攻撃可能
    - パケットを全部落とすオンライン攻撃は可能

注: 傍受とパケット送信はできるがパケット改変・ドロップできないアクターによる攻撃

なぜ「暗号化トランスポート」なのか

- プライバシー保護
  - ネットワークミグレーションの際に、Connection IDやパケット番号を利用したユーザトラッキングを防止
    - Connection ID: あらかじめ予備を（暗号化されたパケット上で）交換し、ミグレーション時に使用
    - パケット番号: 暗号化

なぜ「暗号化トランスポート」なのか

- 硬直化 (ossification) 対策
  - QUICのような性能改善はTCP拡張では難しい
    - TCPを解釈し、介入するミドルボックスの存在
    - 例:
      - TCP Fast Openで誤動作するファイアウォール
      - 再送制御に介入するプロキシ
        - a.k.a. performance-enhancing proxy
  - QUICでは今後、継続的な改善ができるようにしたい
    → 暗号化

QUICの使われ方

- ウェブブラウザの対応状況
- サーバの対応状況
- フォールバック

HTTP/3 protocol - OTHER

Usage % of all users ?

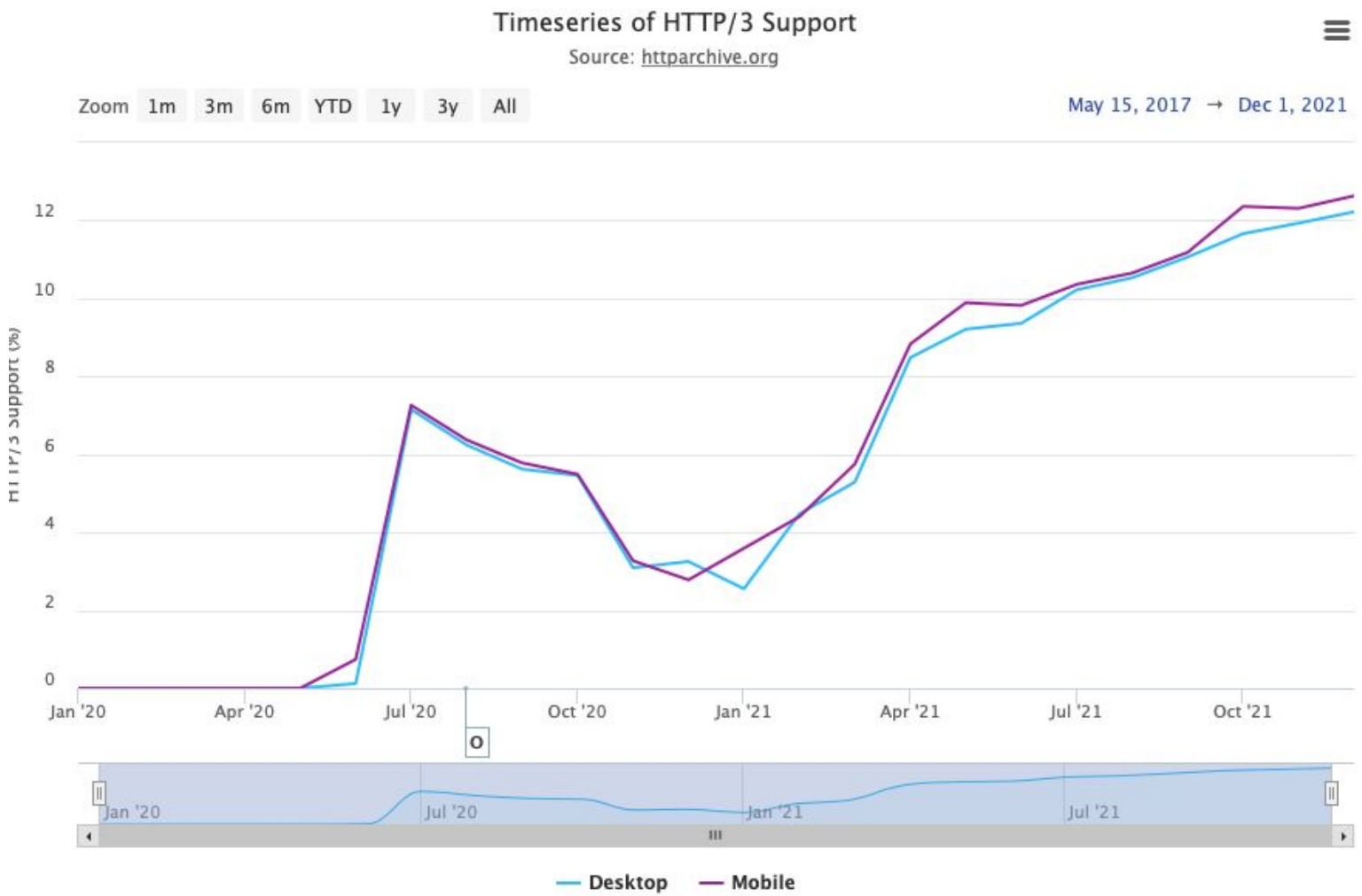
Global 71.28%

Upcoming version of the HTTP networking protocol, which is currently a draft. Previously known as HTTP-over-QUIC. Uses QUIC as its transport layer protocol.

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android
	12-18		4-78									
	2 79-84	2-71	2 79-84		10-72							
	3 85-86	1 72-87	3 85-86	3.1-13.1	3 73	3.2-13.7						
6-10	87-96	88-95	87-96	4 14-15.1	74-81	14-15.1		2.1-4.4.4	12-12.1			
11	97	96	97	4 15.2	82	15.2	all	97	64	97	1 95	12.12
		97-98	98-100	4 TP								

Source: https://caniuse.com/?search=http%2F3



Source: https://httparchive.org/reports/state-of-the-web#h3

QUICの使われ方

- ウェブブラウザの対応状況
- サーバの対応状況
- フォールバック
  - サーバ運用者は、QUIC対応をアドバタイズ
    - Alt-Svc HTTPヘッダまたはDNSのHTTPSレコード
  - ウェブブラウザはTCPとQUICでhappy eyeballs

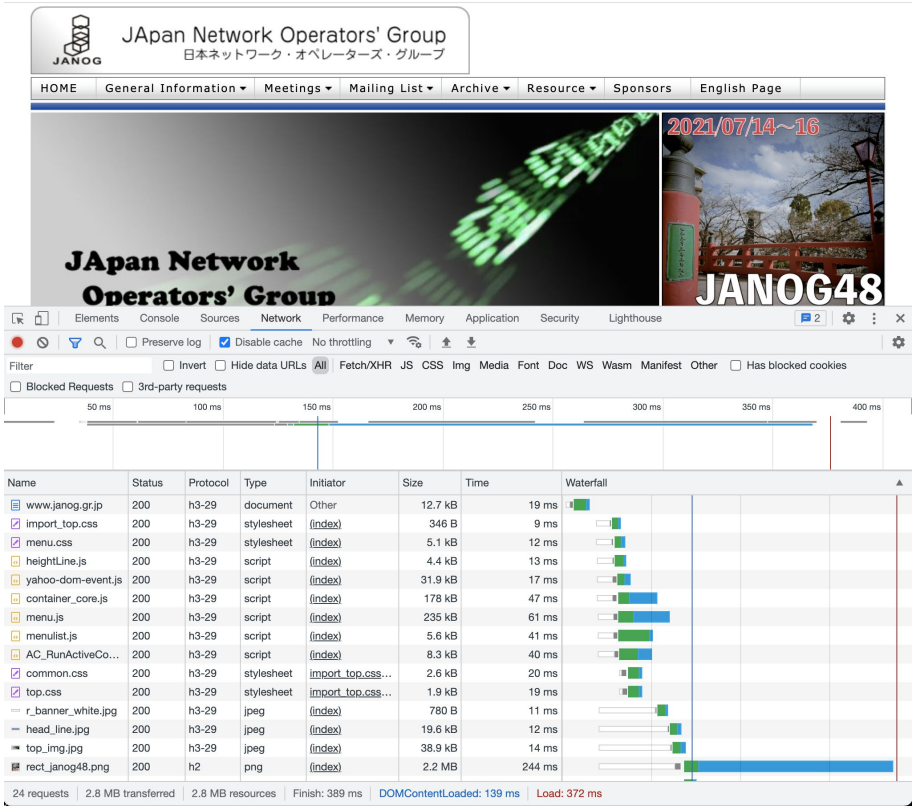
QUICがネットワークに与えるインパクト

余談

JANOGのWebサイトをHTTP/3にしようと秋頃に頑張ったんですが、proxyしているいくつかのバックエンドアプリケーションがうまく動かなかったので戻しました…

https://github.com/yuyarin/nginx-quic-dockerfiles

https://hub.docker.com/repository/docker/yuyarin/nginx-quic-centos7.9



QUICはどういうプロトコルなのか？

ネットワークオペレータの観点では以下を押さえておく

1. HTTP/3のためにUDP443番で使われている
2. UDPを使うトランスポートのプロトコルである
3. 最低限以外のヘッダとペイロードがTLSで暗号化されている
4. 複数のHTTPストリームを重畳できる

QUICを阻害するネットワークトラブル

QUICを使用したときにトラブルを起こす既存のネットワークインフラの例

1. NATやSPIのテーブル溢れ
2. NATの送信元ポート番号枯渇
3. UDP送信元ポート番号フィルタリング

1. NATやSPIのテーブル溢れ

NAT (Network Address Translation)

プライベートIPアドレスを持つ複数の端末でパブリックIPアドレスを
共有するためにTCP/UDP/ICMPのポート番号等まで含めた
アドレス変換を行う仕組み(NAPT/広義のNAT)

主に使われている場所

- ブロードバンドルータやホームゲートウェイ
- IPoEのIPv4共有技術 (DS-Lite AFTR, MAP-E CE)
- マンションISPのインターネット接続ルータ
- 企業ネットワークやカンファレンスネットワークのインターネット接続ルータ
- ISPやモバイルキャリアのCGN
- クラウドサービスのNATゲートウェイ
- スマホのテザリング

NATのテーブル溢れとは

- NATでは5-tupleで通信を区別して状態管理を行っている
  - 新しい通信が来たらNATテーブルにエントリーを作成して管理を開始する
  - 通信が「終わる」とエントリーを削除する
- NATテーブルのエントリー数には上限がある
- 上限を超える＝溢れると新規通信ができなくなる
  - QUICだけでなく**TCPも含めた他の通信もできなくなる**
    - 多くの実装ではTCP/UDP/ICMPでNATテーブルを共有している

NATテーブル

Local					Global				
Src IP Addr	Dst IP Addr	Proto	Src Port	Dest Port	Src IP Addr	Dst IP Addr	Proto	Src Port	Dest Port
192.168.0.2	198.51.100.8	tcp	55350	443	203.0.113.24	198.51.100.8	tcp	55350	443
192.168.0.3	198.51.100.17	udp	60914	53	203.0.113.24	198.51.100.17	udp	49360	53
192.168.0.4	198.51.100.220	icmp	19717	19717	203.0.113.24	198.51.100.220	icmp	19717	19717

なぜQUICでNATテーブルが溢れやすいのか

QUICの特徴1〜3

1. HTTP/3のためにUDP443番で使われている
2. UDPを使うトランスポートのプロトコルである
3. 最低限以外のQUICヘッダとペイロードがTLSで暗号化されている

たくさん使われていて
UDPで
中身が見えないから

特徴1: HTTP/3のためにUDP443番で使われている

- Googleなどのサービスやブラウザでデプロイされている
  - W3Techsによれば既に7.5%のWebサイトがQUICを使用
- さらに増え続けると予測される
  - HTTPトラフィック自体の増加
  - HTTP/1.1, HTTP/2のWebサイトのHTTP/3化による増加

Usage of QUIC for websites, 24 Jan 2022, W3Techs.com

特徴2: UDPを使うトランスポートのプロトコルである

NATのエントリー削除の方法は

TCPの場合

- FINでコネクションの終了を検知してマッピングを消すことができる

UDPの場合

- 「コネクション」の終了が分からないので能動的にマッピングを消すことができない
- NATルータは**無通信時間(タイマー)を基に「コネクション」終了を判断するしかない**

TCPのHTTP/2では即時で消えていたものがUDPのHTTP/3では
タイマーの時間だけエントリーが残り続けることでテーブルを圧迫する

特徴3: 最低限以外のQUICヘッダとペイロードがTLSで暗号化されている

🤔 「QUICヘッダを見てコネクションの終了を検知すればいいのでは？？？」

1. **コネクションの終了を知らせるフレームも暗号化されている**ため
   途中のルータではコネクションの終了を知ることはできない
2. そもそもプロトコル仕様上クローズせずにコネクションが終了することもある

一方で溢れにくくなる要素もある

4. 複数のHTTPストリームを重畳できる
    - 従来複数のTCPコネクションを使用していたものが1本のUDPにまとめられる
    - 1つのWebサイトを開くときに必要な**NATエントリーの数は減る**はず

対策になりえるのは？

1. IPv6を使う？
2. NATのUDPタイマーを短くする？
3. NATのアルゴリズムを変える？
4. NATテーブルのサイズを増やす？
5. UDP443を塞ぐ（最終手段）？？？

対策になりえるのは？

1. **IPv6を使う？**
2. NATのUDPタイマーを短くする？
3. NATのアルゴリズムを変える？
4. NATテーブルのサイズを増やす？
5. UDP443を塞ぐ（最終手段）？？？

対策案1: IPv6を使う

「多くのHTTP/3対応WebサイトはIPv6に対応しているから、
IPv6を使えばNATの問題は解決できるのでは？？？」

SPI (Stateful Packet Inspection)

TCP/UDP/ICMPなどのポート番号や通信状態を管理して
ファイアウォールなどの機能を実現する仕組み

例

- EstablishになったTCPは許可する
- 内側から通信を開始したUDPの戻りパケットは許可する

主に使われている場所

- ステートフルファイアウォール
- ブロードバンドルータ・ホームゲートウェイ(特にIPv6)
- フローベースでトラフィックを扱うSDNの仮想ルータソフトウェア

NAT同様に5-tupleをキーにしたテーブルで管理する

対策1: IPv6を使う効果はあるのか？

議論ポイント

IPv4のNATの問題を軽減できるという意味では価値があると思う

- Stateful Firewallの導入は避けられない
- IPv4では構成によってはNATだけでなくSPIも気にする必要がある

対策になりえるのは？

1. IPv6を使う？
2. **NATのUDPタイマーを短くする？**
3. NATのアルゴリズムを変える？
4. NATテーブルのサイズを増やす？
5. UDP443を塞ぐ（最終手段）？？？

対策2: NATのUDPタイマーを短くする

「無通信時間(タイマー)を基に「コネクション」終了を判断するのであれば、タイマーの時間を短くすればいいのでは？？？」

ブロードバンドルータのNATの実装

国内でよくNATルータとして使われる機器のUDPマッピングの消去方式はタイマー方式が多く、**デフォルトのタイムアウト値は300秒(5分)が多い**

ルータ	方式	タイマーのデフォルト値	設定可否	NATセッション数上限
A	タイマー	300秒	可	2048
B	不明	不明	不可	2048
C	不明	不明	不可	不明
D	タイマー	180秒	不可	不明
E	タイマー	900秒	可	4096-65534
F	タイマー	300秒	可	数千以上(メモリ次第)
G	タイマー	300秒	可	数千以上(メモリ次第)

NATのUDPマッピングのタイムアウトの要件

RFC4787 "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP" によるとUDPのタイムアウトを**2分未満にしてはならず**、
デフォルト値としては**5分以上が推奨**される

REQ-5: **A NAT UDP mapping timer MUST NOT expire in less than two minutes**, unless REQ-5a applies.

a) For specific destination ports in the well-known port range (ports 0-1023), a NAT MAY have shorter UDP mapping timers that are specific to the IANA-registered application running over that specific destination port.

b) The value of the NAT UDP mapping timer MAY be configurable.

c) A default value of **five minutes or more for the NAT UDP mapping timer** is RECOMMENDED.

NATのUDPマッピングのタイムアウトの例外

とはいえ長くしすぎると短い時間でコネクションを終了するアプリケーションで
マッピングが埋まるので、**特定の宛先ポートのタイムアウトは短くして良い**

➡ **DNSのタイムアウトを短くすることができる実装**は存在する

a) Some UDP protocols using UDP use very short-lived connections. There can be very many such connections; keeping them all in a connections table could cause considerable load on the NAT. **Having shorter timers for these specific applications is, therefore, an optimization technique**. It is important that the shorter timers applied to specific protocols be used sparingly, and only for protocols using well-known destination ports that are known to have a shorter timer, and that are known not to be used by any applications for other purposes.

QUICの実装・デプロイにあたって

Manageability of the QUIC Transport Protocol ではRFC4787の2分の値を参照しつつ、フィールドでは**実際には30秒や60秒でタイムアウトしてしまう**ことに言及している

> [RFC4787] requires a network state timeout that is not less than 2 minutes for most UDP traffic. However, in practice, **a QUIC endpoint can experience lower timeouts, in the range of 30 to 60 seconds**.

ブロードバンドルータのSPIの実装

NATルータのうちStatefull Firewall機能を機器のSPIのUDPエントリーの消去方式はタイマー方式が多く、**30秒でタイムアウトするものも存在**する

➡「フィールドでは実際には30秒や60秒でタイムアウトしてしまう」の原因

ルータ	方式	タイマーのデフォルト値	設定可否
A	タイマー	300秒	可
B	不明	不明	不可
C	タイマー	**30秒**	可
E	タイマー	**30秒**	可
F	タイマー	なし	可

QUICの実装の要件

RFC9000 "QUIC: A UDP-Based Multiplexed and Secure Transport" では
タイムアウトを防ぐために**30秒ごとにパケットを送信することが必要である**と
述べられている

> Though REQ-5 in [RFC4787] recommends a 2-minute timeout interval, experience shows that sending packets **every 30 seconds is necessary** to prevent the majority of middleboxes from losing state for UDP flows [GATEWAY].

QUICの実装例

Google ChromeのQUICの実装 “QUICHE” では

15秒おきにPING Frameを送信してNATのタイムアウトを防いでいる

master/quic/core/quic_constants.h

```
// Default ping timeout.
const int64_t kPingTimeoutSecs = 15;  // 15 secs.
```

master/quic/core/quic_connection.cc

```
void QuicConnection::SetPingAlarm() {
  if (perspective_ == Perspective::IS_SERVER &&
      initial_retransmittable_on_wire_timeout_.IsInfinite()) {
    // The PING alarm exists to support two features:
    // 1) clients send PINGs every 15s to prevent NAT timeouts,
    // 2) both clients and servers can send retransmittable on the wire PINGs
    // (ROWP) while ShouldKeepConnectionAlive is true and there is no packets in
    // flight.
    return;
  }
```

QUICの推奨タイマー値

Manageability of the QUIC Transport Protocol ではQUICは30秒のタイムアウトで動くように作られているが、それでも**最低2分のタイムアウト値**にするように求めている

> Even though QUIC has explicitly been designed to tolerate NAT rebindings, decreasing the NAT timeout is not recommended, as it may negatively impact application performance or incentivize endpoints to send very frequent keep-alive packets.
>
> The recommendation is therefore that, even when lower state timeouts are used for other UDP traffic, **a state timeout of at least two minutes ought to be used for QUIC traffic**.

対策2: タイマー値を減らして効果はあるのか？

議論ポイント

NATのUDPタイマー値を減らすことは一時的な対処としては有効

- 減らすならば2分(120秒)に設定する
- SPIが入っている場合はSPIのタイマーも揃える

対策になりえるのは？

1. IPv6を使う？
2. NATのUDPタイマーを短くする？
3. **NATのアルゴリズムを変える？**
4. NATテーブルのサイズを増やす？
5. UDP443を塞ぐ（最終手段）？？？

対策3: NATのアルゴリズムを変える

「タイマー方式をやめてLRU (Least Recently Used)方式に
すればいいのでは？？？」

※キャッシュメモリなどの記憶領域が溢れたときに
最も古いエントリーを消すアルゴリズム

対策3: LRU方式に変えて効果はあるのか？

議論ポイント

RFC4787に従うとタイムアウト方式にならざるを得ないので
実装しているルータはほとんど存在しないのでは？

- RFC4787を守っていないと**ポートの再利用間隔が2分未満になる**問題がある
  - 特にマルチテナンシー環境では十分に注意する必要がある
- LRUでTCPとUDPでマッピングテーブルを共有している場合、Keepaliveの長いTCPベースのプロトコルが割を食う可能性がある
  - Google Cloud MessagingはKeepaliveが15分
- LRUで消されないように新しいプロトコル同士でKeepaliveを短くする競争が激化する可能性もある

対策になりえるのは？

1. IPv6を使う？
2. NATのUDPタイマーを短くする？
3. NATのアルゴリズムを変える？
4. **NATテーブルのサイズを増やす？**
5. UDP443を塞ぐ（最終手段）？？？

対策4: NATテーブルのサイズを増やす

 「NATテーブルが足りないんだったら増やせばいいのでは？？？」

NATテーブルのサイズを増やすことができるのか？

議論ポイント

ソフトウェアリミットの場合

- 変更可能であれば設定変更＆再起動などで変更できる

ハードウェアリミットの場合

- 増やすことが困難なので機器交換が必要になる
- 新しく導入する場合はNATのエントリー数が多い機器を選ぶのが良い
- ブロードバンドルータなどの場合、コストに直接影響してくる

対策になりえるのは？

1. IPv6を使う？
2. NATのUDPタイマーを短くする？
3. NATのアルゴリズムを変える？
4. NATテーブルのサイズを増やす？
5. **UDP443を塞ぐ（最終手段）？？？**

対策5: UDP443を塞ぐ

 「UDP443を塞いでQUICを使えないようにしてしまえば？？？」

本当にUDP443を塞いでいいのか？

議論ポイント

やったら負け

- どうしても通信品質が保てなくて緊急避難として行う最終手段に留める
- 塞いだ場合QUICがどのような挙動を示しエンドユーザにどういう影響があるのかを理解しておくことは重要（のちほど）

2. NATの送信元ポート番号枯渇

送信元ポート番号枯渇とは

パブリックIPアドレスが1つしかないときNATで使用できるのは64512ポート

- 64512 = 65536-1024
- CGNの場合は1加入者あたり512, 1024, 2048ポートなどでもっと少ない

NATのテーブルエントリーに余裕があっても使用できる
ポート番号が枯渇するとQUIC以外のUDPも含めて通信できなくなる

※Endpoint-Independent Mappingなどで実際にはもう少し多く使える場合があるが
ここでは簡単のため基本的には1UDP接続で1ポート消費と考える

CGNのポートの再利用禁止期間

CGN(Carrier-Grade NAT) の要件を示した RFC6888 では一部の例外を除いて
ポートの再利用禁止期間を2分と定めている

➡ **タイムアウトを含めると無通信開始から最低4分は再利用不可？**

※例外として**トラックされたTCPポートは即時再利用することができる**

> REQ-8: Once an external port is deallocated, **it SHOULD NOT be reallocated to a new mapping until at least 120 seconds have passed**, with the exceptions being:
>
> a. If the CGN tracks TCP sessions (e.g., with a state machine, as in Section 3.5.2.2 of [RFC6146]), **TCP ports MAY be reused immediately**.

※シングルテナンシー用途のブロードバンドルータがこのRFCに従うべきかどうかは別の問題です

送信元ポート番号枯渇対策

1. パブリックIPアドレス数を増やす
    - IPv4アドレス枯渇により難しい
2. CGNでの割当ポート数を増やす
    - サービスイン後の設計変更は難しいことが多い
3. IPv6を使う
    - これしかなさそう

3. UDP送信元ポート番号フィルタリング

UDP送信元ポート番号フィルタリングとは

- UDPのリフレクション攻撃のへの対応としてネットワーク側でフィルタリングを行っていることがある
  - 代表的なもの: NTP(123)、memcached(11211)、SSDP(1900)
  - HTTP/3では well-known ポートを送信元に使うことはないので非well-knownを考慮する

well-knownポート以外のUDP増幅攻撃の対象

ポート番号	プロトコル	増幅率	主な利用者
11211	memcached	10,00051,000	コンテンツプロバイダ
27910	Quake 2	63.9	Eyeball/クラウド
1900	SSDP	30.8	Eyeball
1433	MSQL	25	エンタープライズ/クラウド
4672	Kad	16.3	Eyeball
27000-27030	Steam Protocol	5.5	Eyeball/コンテンツプロバイダ/クラウド
1024-65534	BitTorrent	3.8	Eyeball
5353	mDNS	2-10	Eyeball

UDP送信元ポート番号の選択

- QUICクライアントではこれらのポート番号を送信元ポート番号として使用しないようにする動きがある
  - 通常エフェメラルポートと重なることは多くないのでそこまで心配ではない

各OSのエフェメラルポートレンジ

OS	ポート番号
Windows	49512 - 65535
macOS	49152 - 65535
Linux	32768 - 60999

NATによる影響

しかし…

QUICクライアント側でこれらのポート番号の使用を避けても
NATしたときに送信元ポート番号として割り当てられる可能性がある

- その結果としてQUICの通信がフィルタにかかり通信できなくなる可能性がある
- フィルタリング箇所が適切であれば問題にならないかも？
  - 参考「QUICやHTTP/3で利用を避けるべき送信元ポートの議論についての考察」

UDP送信元ポート番号フィルタリングの対策

議論ポイント

1. まず自身の管理しているネットワークでのフィルタリング状況を知っておく
2. CGNにおいて特定のポート番号だけを避けることは難しいですよね？
   - MAP-E: 該当するポート番号が含まれるPSIDをアサインしない
   - NAT/DS-Lite: 該当するポート番号を送信元ポートとして使用しない
3. IPv6を使う
   - NATしなければよい

各種の問題への対応まとめ

- 使用しているNATルータのスペックを把握しておこう
- NATだけではなくSPIのUDPタイマーも気にしよう
- 自身のネットワークのUDPのフィルタリング箇所について理解しておこう
- HTTP/3でウェブサービスを提供するならIPv6対応しよう！
- HTTP/3が増えてきたのでISPもIPv6対応しよう！

議論

@kazuhoさんに聞きたいこと

- Path MTU DiscoveryやMSS Clampingに相当する処理はどうなっていますか？
  - ➢ PLPMTUD(RFC4821)によるPath MTU Discoveryを実装しているパターンと一定サイズ(Google Chromeの場合1350バイト)のパケットを投げて通るかどうかを見てQUICを使うかどうかを判断している実装がある
- H2OでのNATタイムアウト対策の実装はどうしていますか？
  - ➢ 仕様上は両方がPINGを送る形になっているが実際は片方だけが送れば問題いし、15秒以内にレスポンスが返ってこない状況はほとんどないので現状は実装していないが、将来的に実装する可能性がある
- HTTP/3対応Webサイトにアクセスしようとしたときに途中経路でUDP443が塞がれていたときのクライアントの挙動はどうなりますか？
  - ➢ TCPとHappy Eyeballsを行うので最初から塞がれている場合はTCPにフォールバックすることで通信ができる
  - ➢ セッションが確立してからSPIなどでフィルタされた場合は通信ができなくなる

みなさんと議論したいこと

- いま困っていることや、実際にあったトラブル事例があれば教えて下さい。
- NATテーブル溢れに対する対策は有効でしょうか？
- NATテーブルエントリ数の向上など、QUICに向けて対応を検討しているブロードバンドルータメーカさんはいらっしゃいますか？
- HTTP/3対応を検討しているコンテンツプロバイダさんはいらっしゃいますか？その場合はIPv6対応してますか？

さいごに

- QUICについてしっかり理解して前向きに対応していこう
- QUIC&HTTP/3の世界のためにIPv6対応しよう！