3つのプロトコルから見た QUIC

2024.6.6

技術研究所
山本和彦

おしながき

HTTPから見たQUIC

TLSから見たQUIC

TCPから見たQUIC

QUICとは

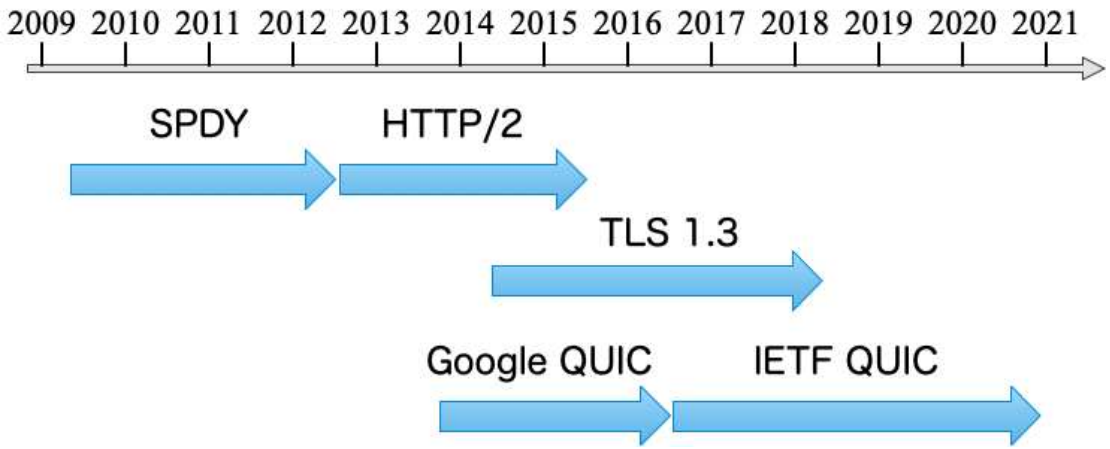
UDPの上に作られた
トランスポートプロトコル

ペイロードはHTTPに限定されず
何でも運べる

デフォルトで暗号化されている

QUICの歴史

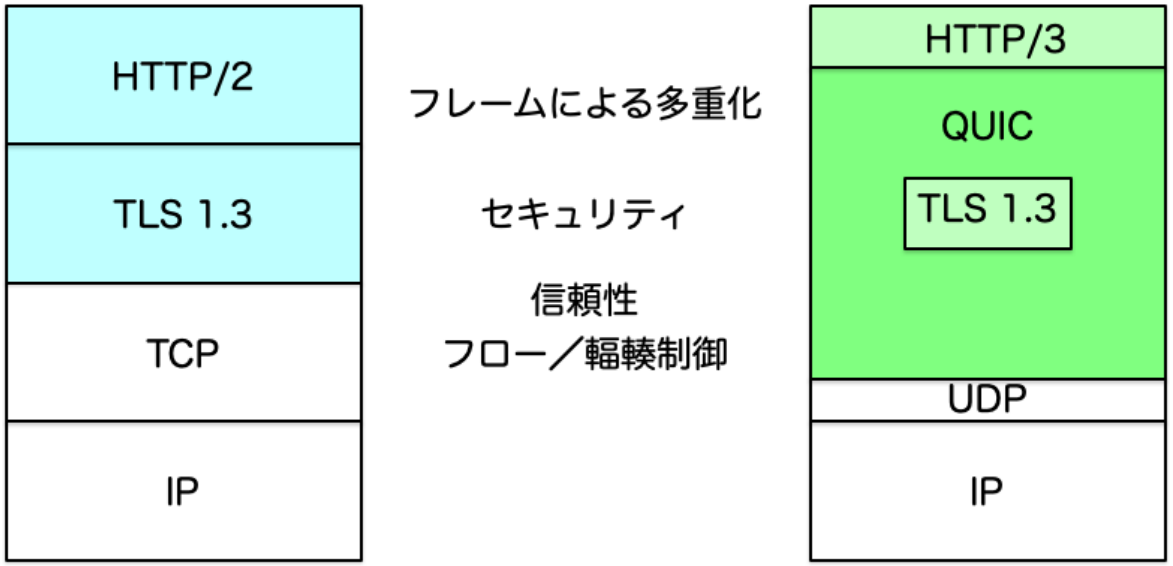
- 2016年6月からIETF で標準化が開始
- Google QUIC (gQUIC) がベース
- 5年間の議論



QUICのRFC

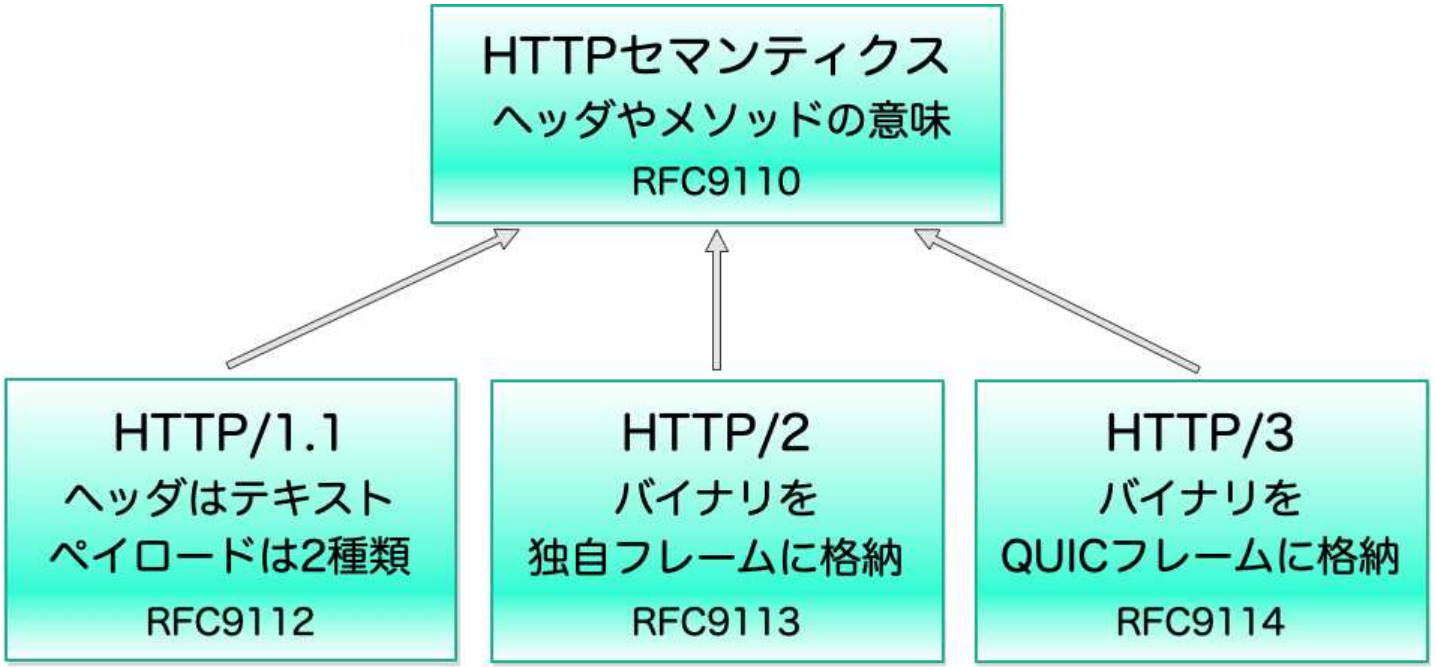
- QUIC Version 1 (2021年5月)
  - RFC 8999: Version-Independent Properties of QUIC
  - RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport
  - RFC 9001: Using TLS to Secure QUIC
  - RFC 9002: QUIC Loss Detection and Congestion Control
- HTTP/3 (2022年6月)
  - RFC 9114: HTTP/3
  - RFC 9204: QPACK: Field Compression for HTTP/3
- QUIC Version 2 (2023年5月)
  - RFC 9368: Compatible Version Negotiation for QUIC
  - RFC 9369: QUIC Version 2

QUICのプロトコル階層



HTTPから見たQUIC

意味と配送方法の分離


HTTPセマンティクス
ヘッダやメソッドの意味
RFC9110
HTTP/1.1
ヘッダはテキスト
ペイロードは2種類
RFC9112
HTTP/2
バイナリを
独自フレームに格納
RFC9113
HTTP/3
バイナリを
QUICフレームに格納
RFC9114

HTTP/1.1

- ヘッダはテキスト

```
% gtelnet www.iij.ad.jp 80
GET /dev/ HTTP/1.1↓
Host: www.iij.ad.jp↓
Connection: close↓
↓
```

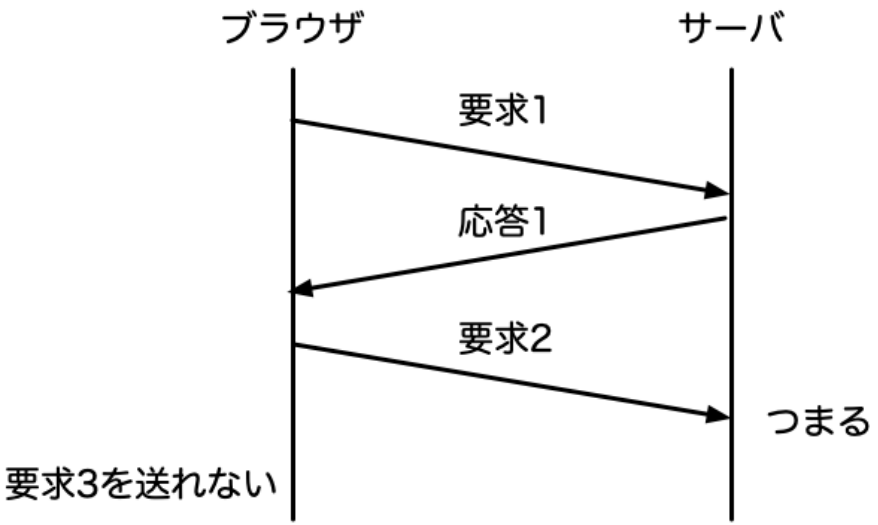
- ペイロード
  - 長さを指定したバイナリ
    - Content-Length
  - ストリーム用のチャンク
    - Transfer-Encoding: chunked

HTTP/1.1 の問題点

同期性	Head-of-line ブロッキング
低い並列性	ドメインシャーディング
非効率なヘッダ	帯域の浪費

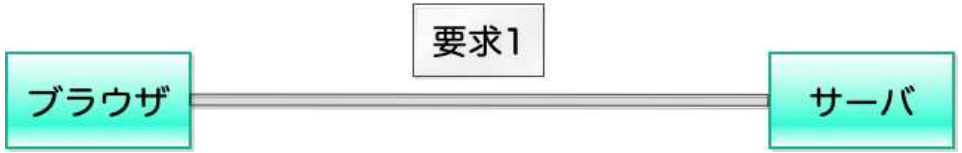
同期性

- HTTP/1.1 は同期的なプロトコル
  - サーバ：　　ある要求を処理した後、応答を返す
  - クライアント：応答が返ってきたら、次の要求を出す
- Head-of-line (HoL) ブロッキング
  - ある要求への応答に時間がかかると、すべての処理が止まる

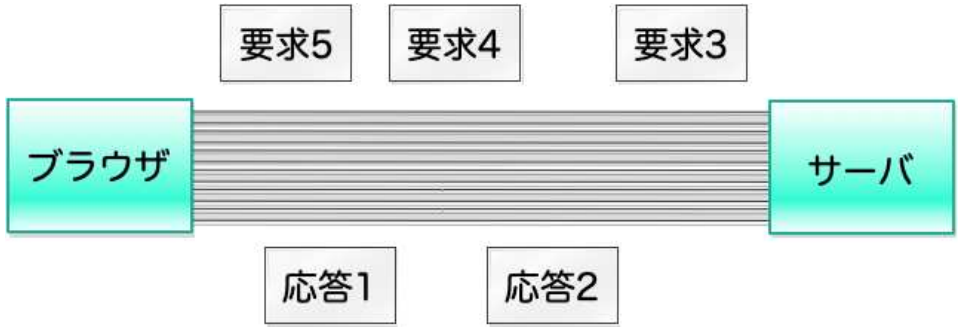


低い並列性

- 1つのコネクション上では高々1つの仕事
  - 要求が1つ、応答が1つ、なにもない
  - 要求応答パイプラインイングは事実上使われてない

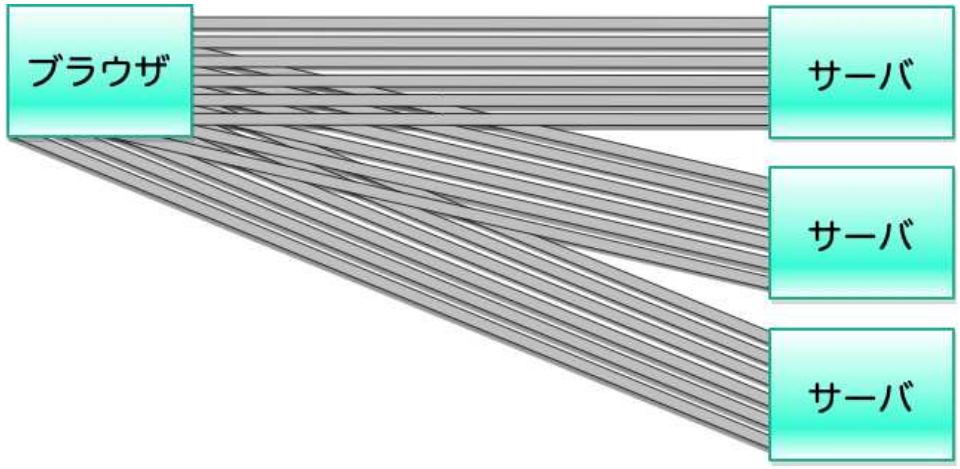


- 同時コネクションの数はドメインごとに6つ



ドメインシャーディング

- ドメインを増やして並列性を上げる
  - コンテンツが分散し、管理が困難になる



非効率なヘッダ

- 帯域の浪費
  - 要求ヘッダの長さの平均は800バイト
  - 似通った要求ヘッダが毎回送られる

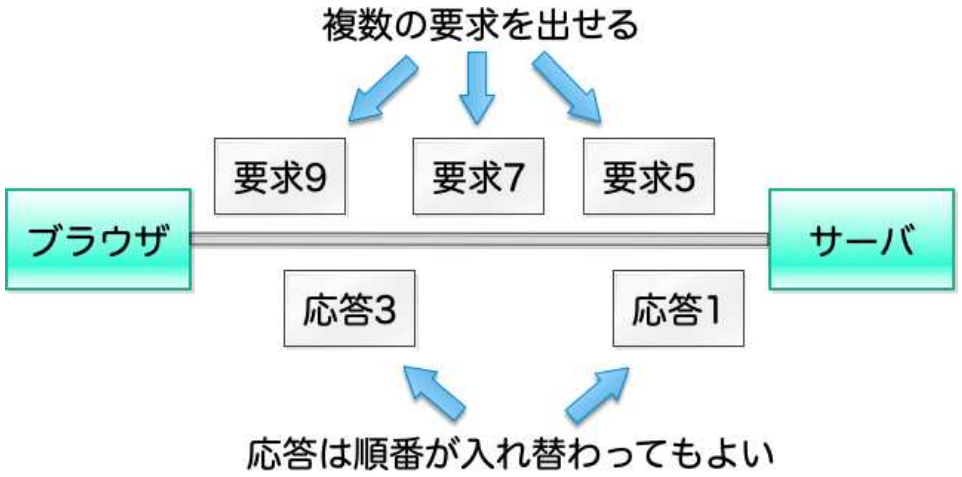
```
GET /roversync/ HTTP/1.1
Host: rover.ebay.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8;
 rv:16.0) Gecko/20100101 Firefox/16.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.ebay.com/
Cookie: ebay=%5Esbf%3D%23%5E; dp1=bpbf/%2380000000000055
 276504d^u1p/QEBfX0BAX19AQA**5276504d^; cssg=c67883f113a
 0a56964e646c6ffaa1abe; s=CgAD4ACBQlm5NYzY3ODgzZjExM2EwY
 TU2OTY0ZTY0NmM2ZmZhYTFhYmUBSgAYUJZuTTUwOTUxY2NkLjAuMS4z
 LjE1MS4zLjAuMeN+7JE*; nonsession=CgAFMABhSdlBNNTA5NTFjY
 2QuMC4xLjEuMTQ5LjMuMC4xAMoAIFn7Hk1jNjc4ODNmMTEzYTBhNTY5
 NjRlNjQ2YzZmZmFhMWFjMQDLAAFQlSPVMX8u5Z8*
```

HTTP/2による解決

HTTP/1.1	HTTP/2
同期性	フレームによる非同期性
低い並列性	フレームによる 高い並列性
非効率なヘッダ	ヘッダ圧縮

フレームの非同期性と高い並列性

- 高い並列性
  - 利用されるコネクションは1つ
  - 複数のフレームが多重化される
- 非同期性
  - 準備ができた応答から返してよい



ヘッダ圧縮と疑似ヘッダ

- HPACK の疑似ヘッダ

```
GET /dev/ HTTP/1.1      :method: GET
                    =>  :path: /dev/
Host: www.iij.ad.jp     :authority: www.iij.ad.jp
```

静的テーブルと動的テーブル

番号	ヘッダ名	ヘッダ値
0	:authority	
1	:method	GET
2	:method	POST
3	:path	/
...		
61	www-authenticate	
62	0	www.iij.ad.jp

```
:method: GET                    {1}
:path: /dev/                    {3, "/dev/"}
                                (登録しない)
:authority: www.iij.ad.jp       {0, "www.iij.ad.jp"}
                                (登録、次から62)
```

HTTP/2 の問題点

TCP

Head-of-line ブロッキング

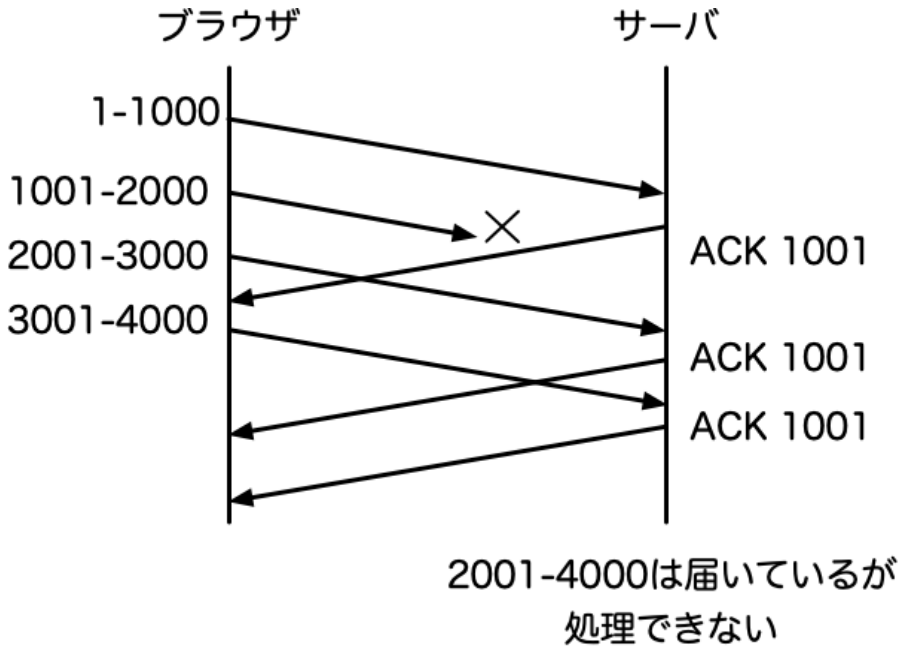
TCPセグメントが1つ落ちると
処理が進まない

HPACK

Head-of-line ブロッキング

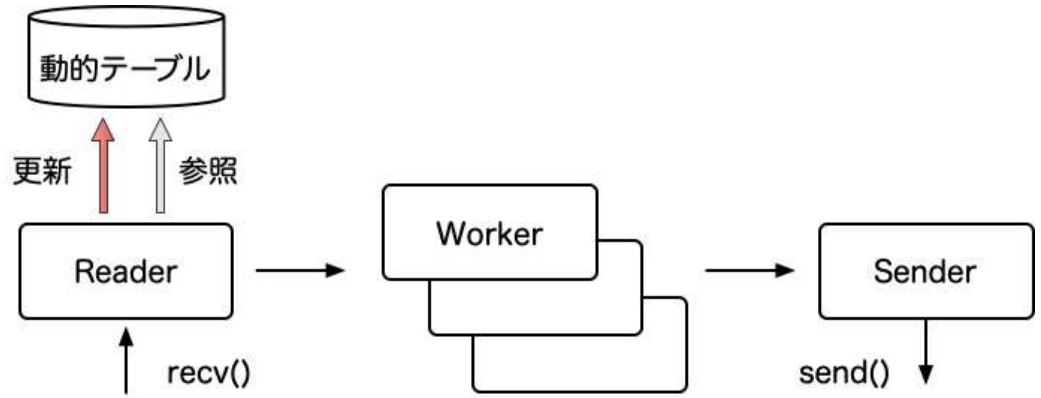
複数のヘッダ間には順序が仮定されてる

TCP の HoL ブロッキング

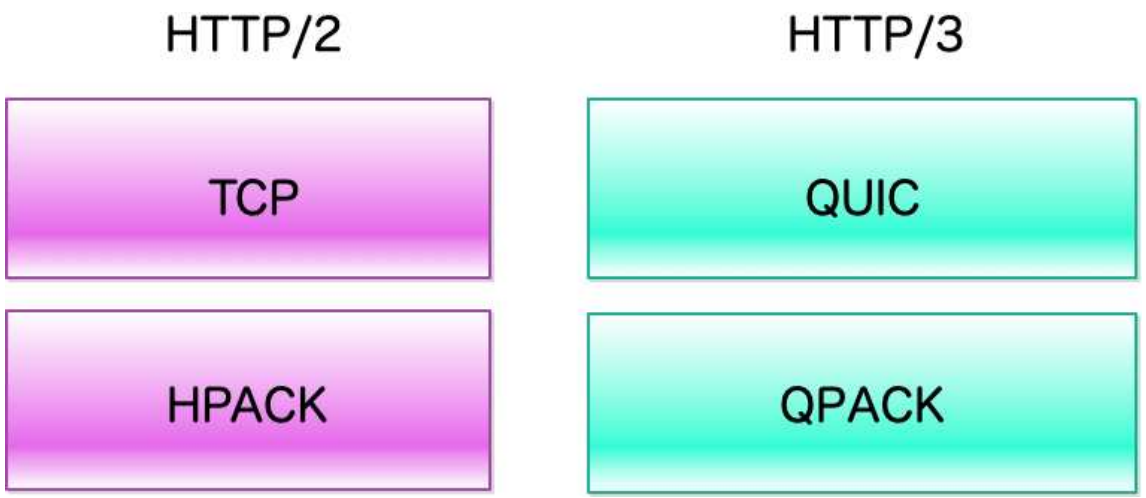


HPACK の HoL ブロッキング

- HPACKには順番がある
  - 圧縮表現と動的テーブルの更新命令が一体化している

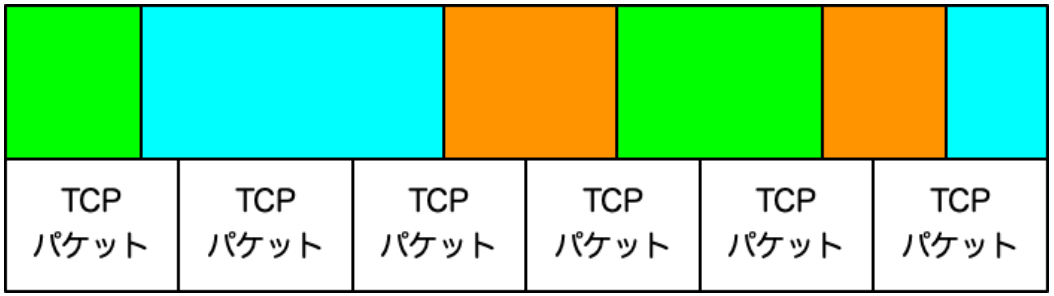


HTTP/3 による解決

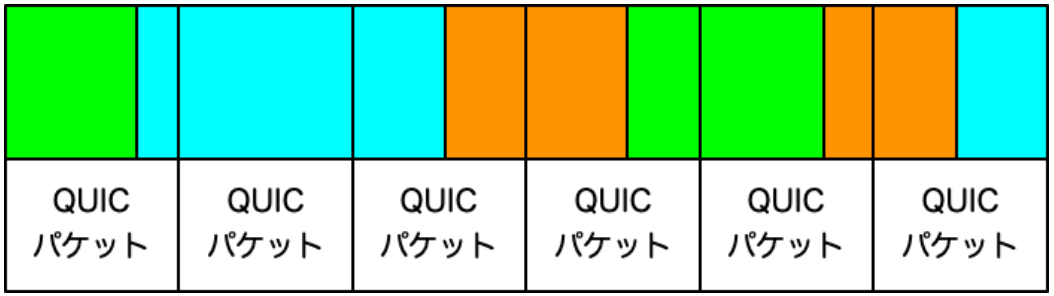


QUICのストリーム

- TCPパケットとHTTP/2フレーム
  - HTTP/2フレームは複数のTCPパケットに跨ることがある

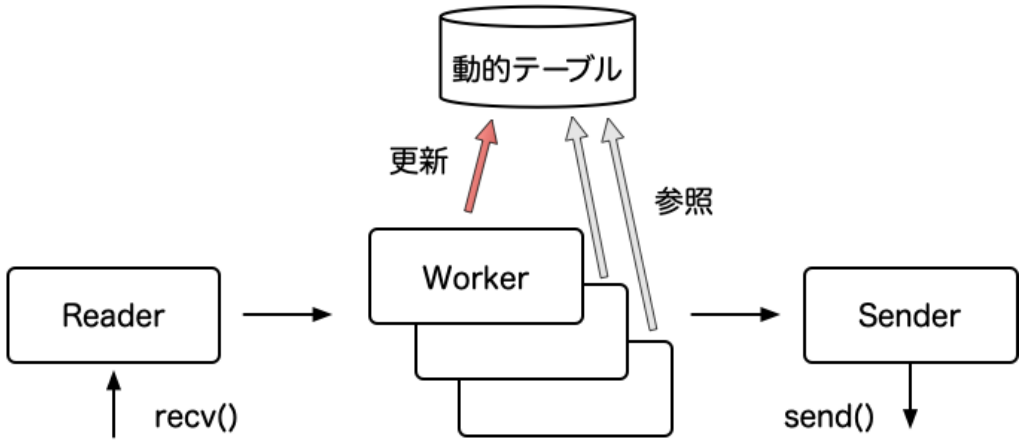


- QUICパケットとHTTP/3フレーム
  - HTTP/3フレームは必ず1つのQUICパケットに収まる



QPACK

- QPACKによるHoLの緩和
  - 圧縮表現と動的テーブルの更新命令を分離
  - 動的テーブルの更新命令には専用の(一方向)ストリームを使う
  - あるストリームのパケットの欠落が他のストリームに影響を与えない



TLSから見たQUIC

TLSの状況

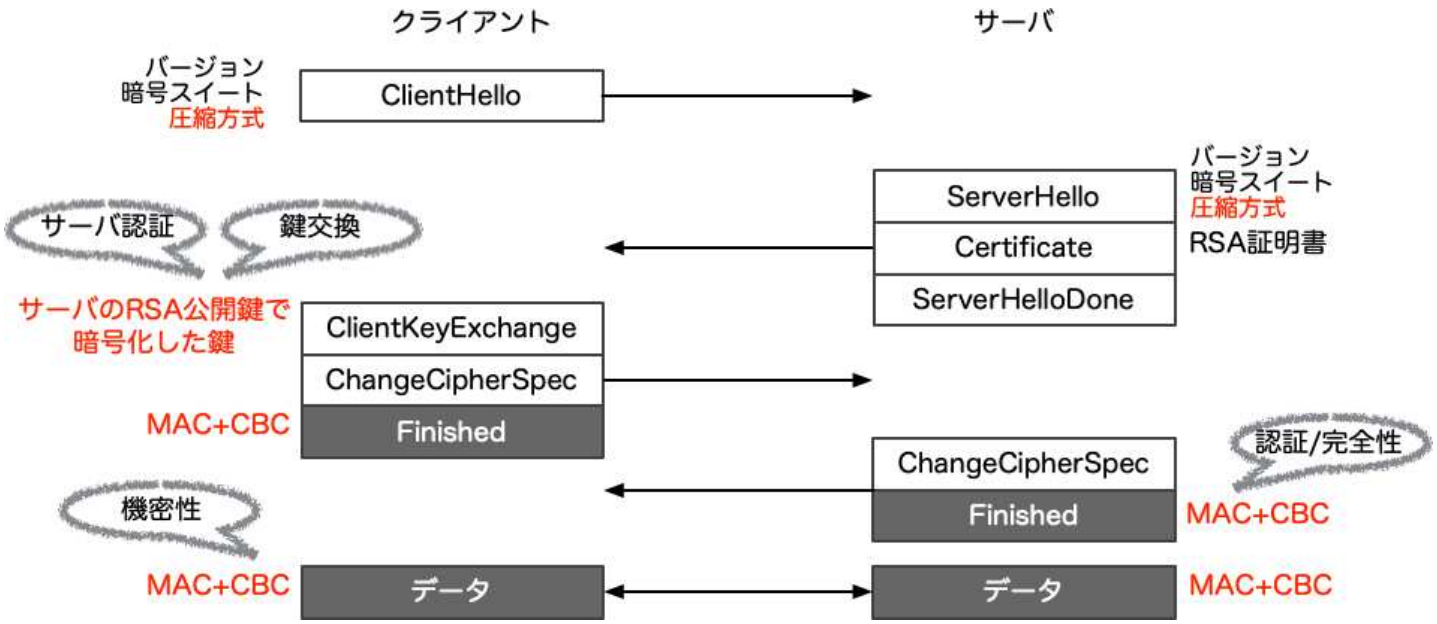
	策定年	攻撃	状況
SSL 2.0	1995	DROWN	RFC 6176により禁止
SSL 3.0	1996	POODLE	RFC 7568により禁止
TLS 1.0	1999	BEAST	RFC 8996により禁止
TLS 1.1	2006	脆弱なCBC	RFC 8996により禁止
TLS 1.2	2008		認証付き暗号(AEAD)がある
TLS 1.3	2018		プロトコルの大幅な変更

TLS の常識

- TLS 1.1 より前は使わない
- TLS 1.2 は適切なパラメータと使う
  - 圧縮は使わない
  - 共通暗号には認証付き暗号(AEAD)を使う
    - ブロック暗号のCBCモードやストリーム暗号は使わない
  - 鍵交換には前方秘匿性を持つ Diffie Hellman 系を使う
    - DHE (Diffie Hellman Ephemeral)
    - ECDHE (Elliptic Curve Diffie Hellman Ephemeral)
  - 再ネゴシエーションは使わない
  - EMS (Extended Master Secret)は必須
- TLS 1.3 がオススメ
  - 設計により安全

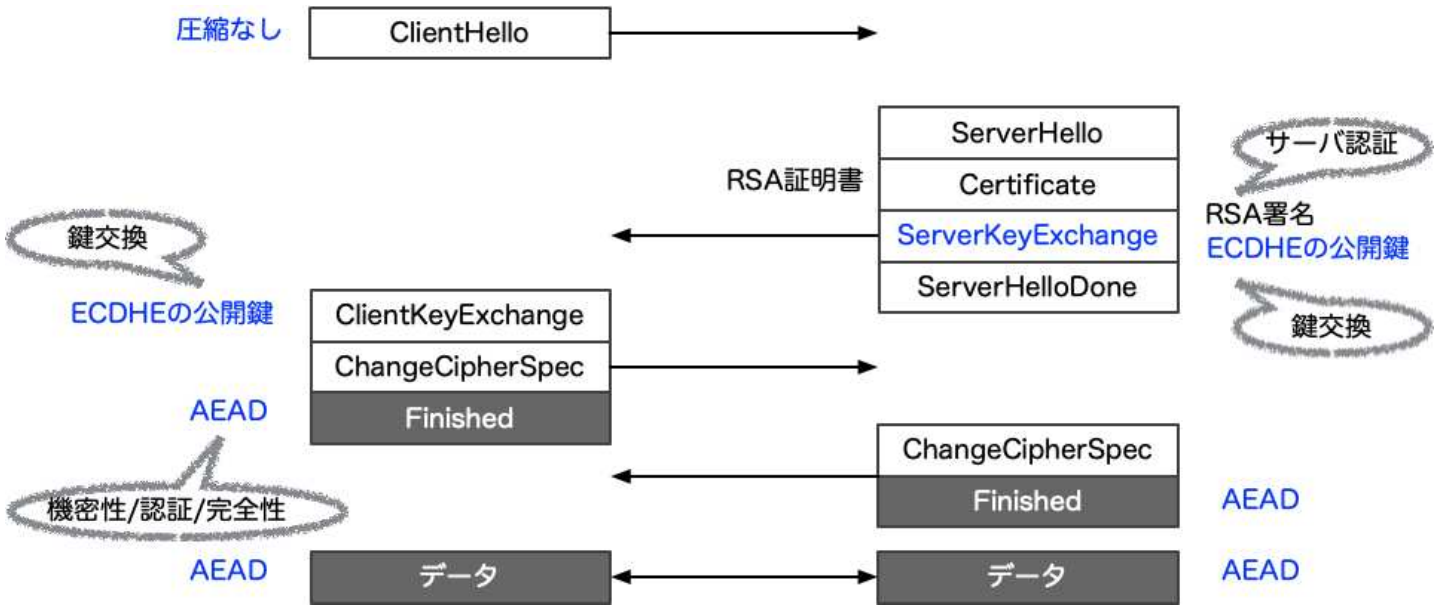
TLS 1.2 のフルハンドシェイク(NG)

TLS1.2のデフォルトの暗号スイート
TLS_RSA_WITH_AES_128_CBC_SHA



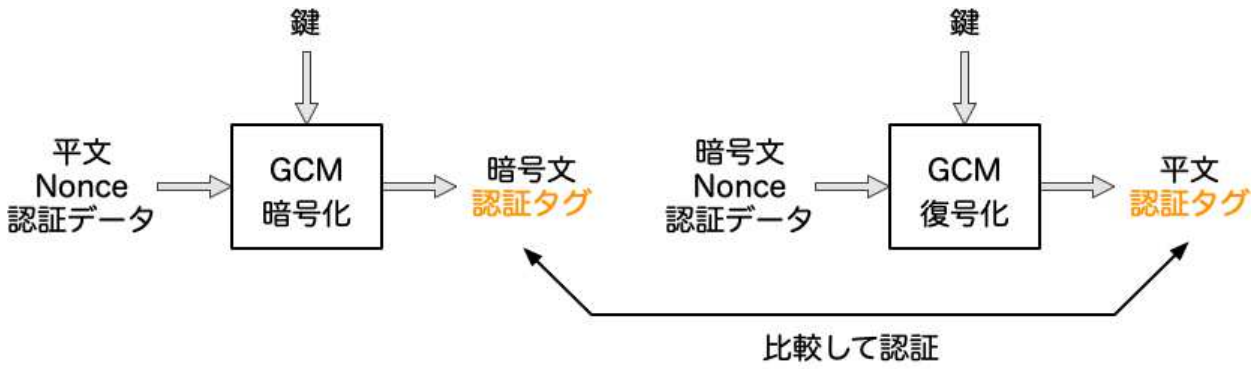
TLS 1.2 のフルハンドシェイク(OK)

HTTP2のデフォルトの暗号スイート
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256



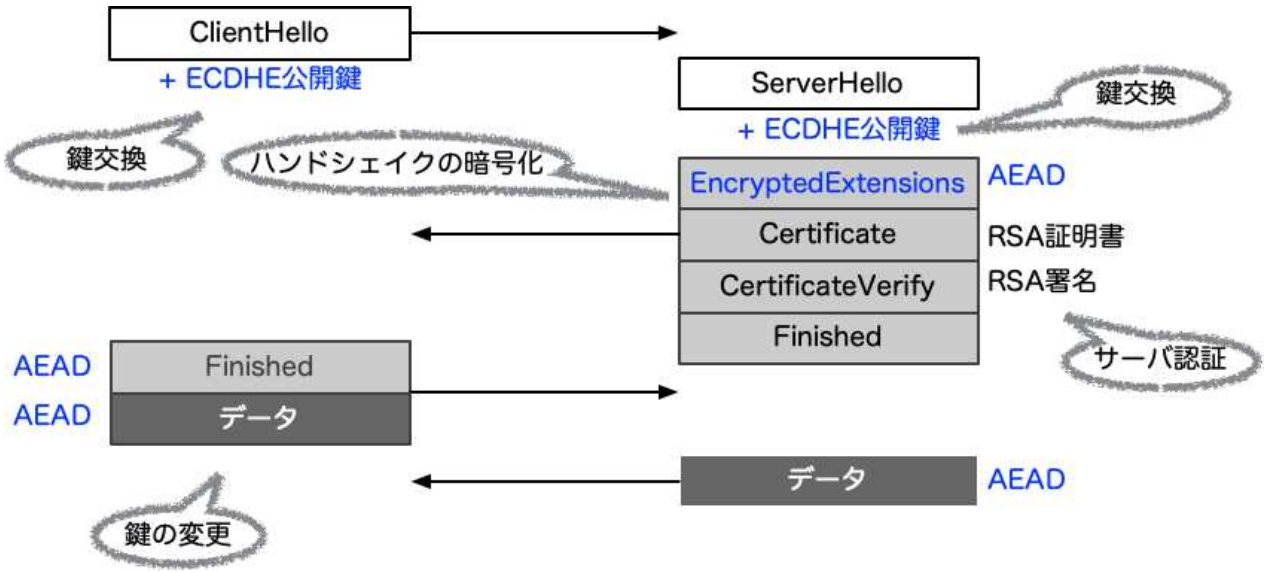
認証付き暗号

- TLS 1.2 で採用された第三の暗号化手法
  - AEAD: Authenticated Encryption with Associated Data
  - 暗号化と同時に認証
  - ハッシュ関数は使わない
- 2つの認証付き暗号
  - GCM (Galois/Counter Mode)
  - CCM (Counter and CBC MAC Mode)

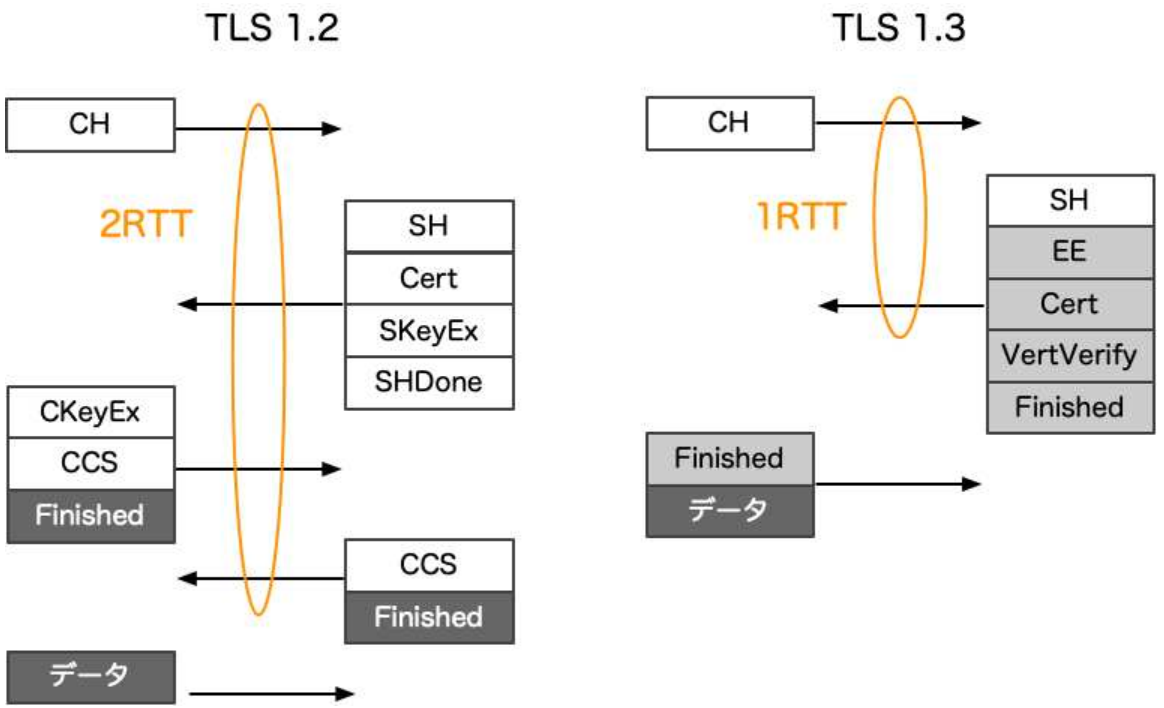


TLS 1.3のフルハンドシェイク

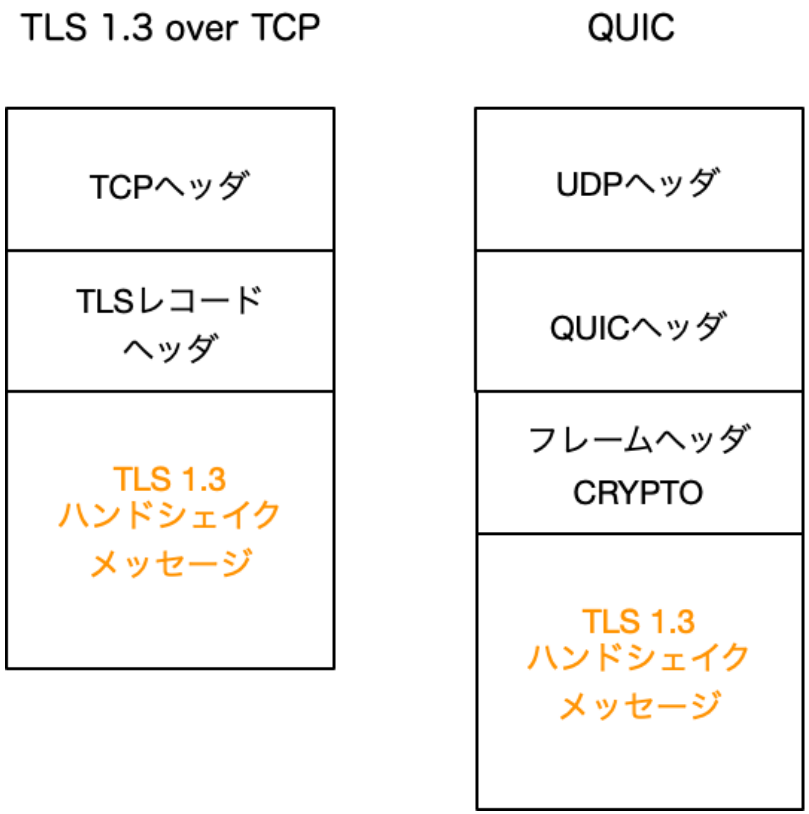
TLS 1.3のデフォルトの暗号スイート
TLS_AES_128_GCM_SHA256



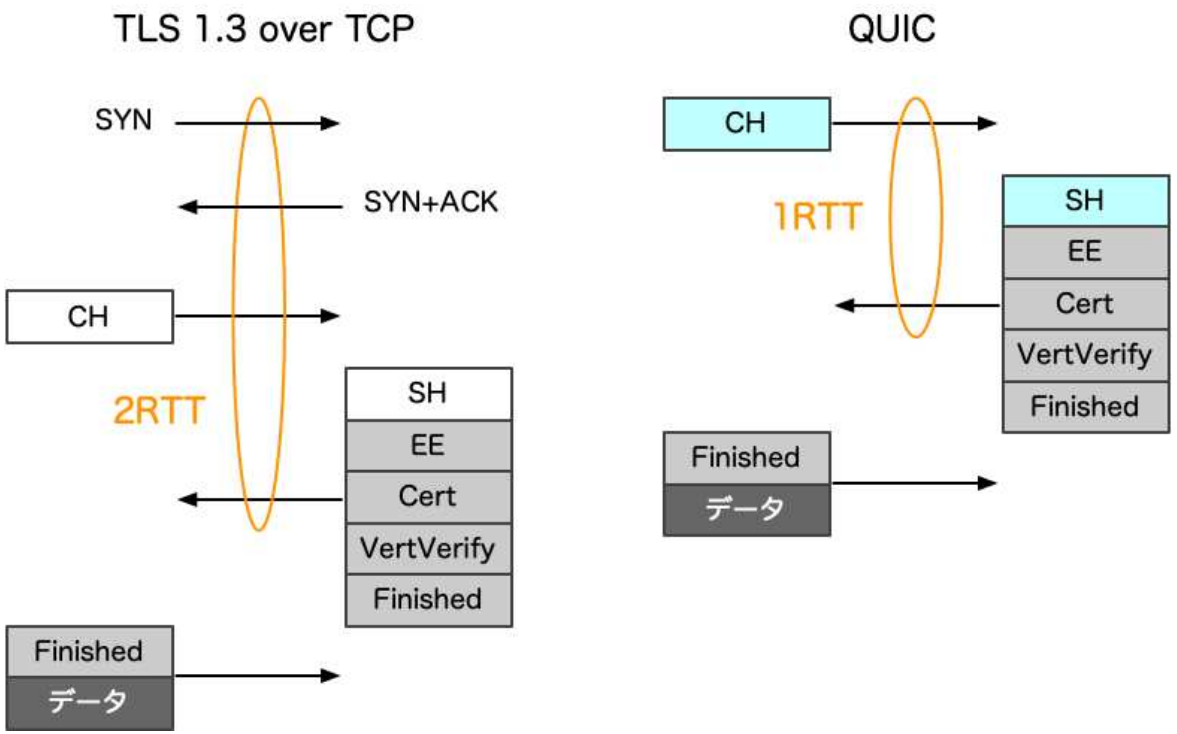
データを送り出せるまでのRTT



TLSレコードとQUICフレーム



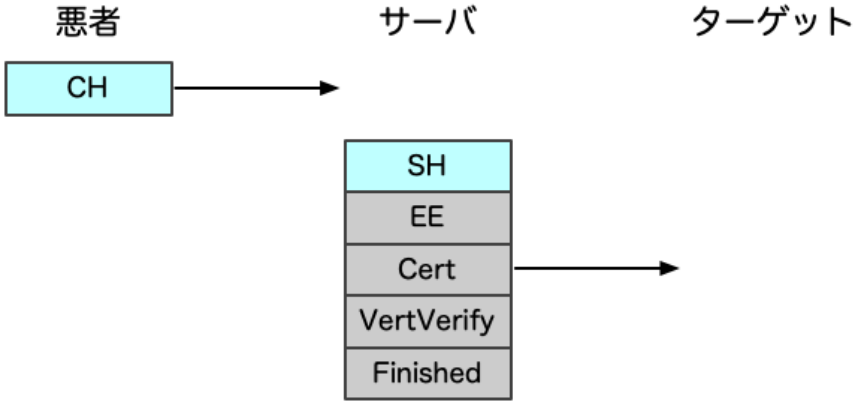
QUICのフルハンドシェイク



TCPから見たQUIC

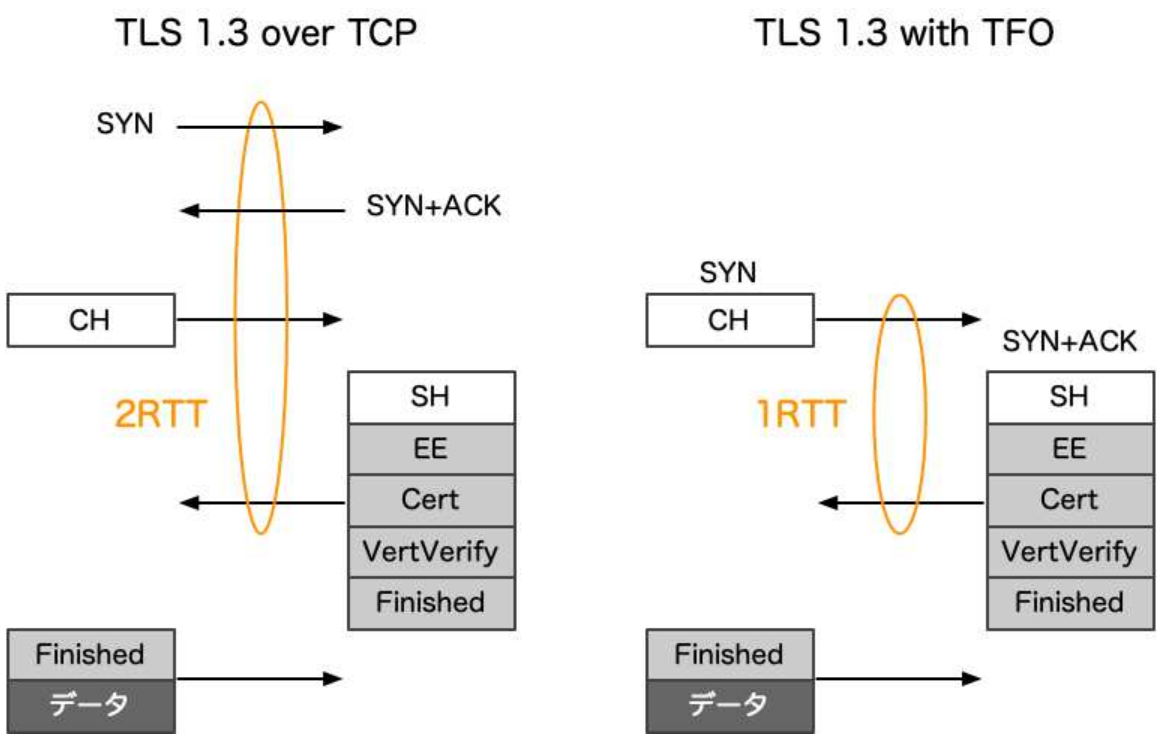
増幅攻撃への対策

- 2番目のパケットが大きく増幅攻撃に使える



- 3倍まで返しルール
  - クライアントが送ってきたデータ量の3倍までしか
    サーバは送り返してはならない

TCP Fast Open



TCPを硬直化させる中間装置

知らないTCPオプションを削除する

SYNパケットにデータが乗っていた
場合はパケットを落とす

TCP Fast Open は実質的に使えない

QUICでの硬直化の回避

- ヘッダの保護
- Chromeマジック
- QUICバージョン2

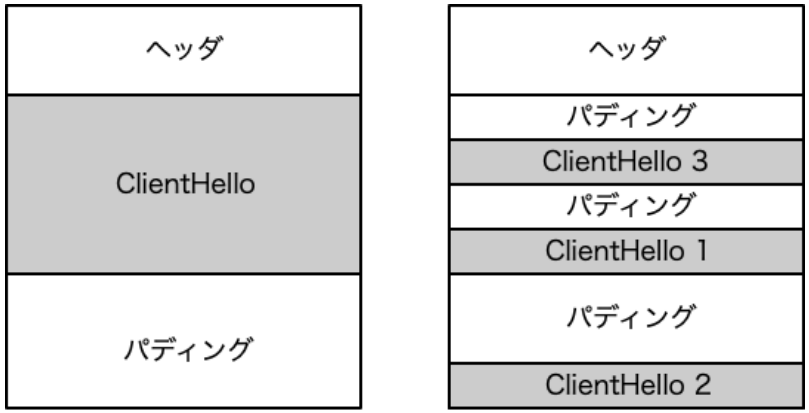
ヘッダの保護

- 最初のパケットのペイロードをキーとしてヘッダを暗号化
- RRビットは予約 (00)

- RRビットは保護されている
- 「00以外ならパケットを落とす」ことはできない

Chromeマジック

- 最初のパケットは中間装置でも復号できる



- 中身のClientHelloが分割されシャッフルされている
- 「ClientHelloからサーバ名オプションを取り出す」ことが難しくなっている

QUIC バージョン2

	バージョン1	バージョン2
バージョン	0x00000001	0x6b3343cf
Initial	0b00	0b01
0-RTT	0b01	0b10
Handshake	0b10	0b11
Retry	0b11	0b00

その他、鍵生成のパラメータも異なる

説明しなかった特徴

- マイグレーション
- 正確なACK
- バージョン・ネゴシエーション

- 詳しくは「QUICをゆっくり解説」を参照

QUIC の今後

- QUIC 自体の拡張
  - パルチパス
  - ロードバランサ
- QUIC の応用
  - DNS over QUIC (DoQ), DNS over HTTP/3 (DoH)
  - Masque
    - HTTP/3で接続した後、さまざまなプロトコルを通す
    - 応用はVPNなど
  - Media over QUIC (MoQ)
    - post WebRTC