https://github.com/flano-yuki/http3-note
ver 0.0.1 (2020/06/01)

2.1 QUICのはじめに

1章で説明したとおり、HTTP/3はトランスポートプロトコルとしてQUICを利用しています。この章では簡単にQUICの提供する機能や、用語を説明していきます。QUICは大変カバーする領域が広く、細かいところを省略はしていますがそれでもトピックは多岐にわたります。

特に重要なのがストリームの概念です。HTTP/3はQUICの提供するストリームという仕組みを活用しており、HTTP/3を理解するにはQUICのストリームを理解することが重要となります。3章と併せて適宜往復しながらお読みいただくのが良いと思います。

QUICの説明に移る前に、QUICプロトコルのバージョンについて補足しておきます。現在は標準化されているのはQUIC v1 と呼ばれるバージョンです。QUICでは標準化を現実てきな期間で終えるため、機能を絞っています。例えばForward Error Correction(FEC)や、Multipathといった機能はQUIC v1のスコープには含まれていません。こういった機能はv2以降、または拡張仕様として議論することになっています。もちろんv2がどのようなものになるかはまだ何も決まってはいません。

この文書ではQUICといった場合はIETFで策定が進められている QUIC v1 のことを指しています。

2.2 QUICの概要

QUICはUDP上で動作するトランスポートプロトコルです。TCPのようなデータの整合性(パケットロスしたデータの回復、データ順番)を担保し、TLSのようにアプリケーションプロトコルレイヤの暗号化を行います。

通信の信頼性を担保するために、輻輳制御、ウィンドウ制御、パケットロスの検出及びデータの再送要求をします。なお、輻輳制御は新しい輻輳制御アルゴリズムを定義するのではなくRenoやCubicといった既存の輻輳制御アルゴリズムを使用します。

アプリケーションデータは、QUICがコネクションの中にもつ仮想的な通信単位であるストリーム上で送信されます。整合性はこのストリーム毎に担保されます、例えばストリーム0用のデータのパケットロスは、ストリーム4には影響を与えず、ストリーム4では処理を継続することができます。この部分については大事ですので、改めて説明します。

また大きな利点の一つとして、QUICではコネクションの確立もより早く行われるようになっており、今までHTTP/2ではTCPハンドシェイクとTLSハンドシェイクを行っていたところを、QUICでは1-RTTでハンドシェイクを行えます。QUICのハンドシェイクは、TLS1.3相当のハンドシェイクを行っており、そこからクライアントとサーバ側ともに鍵の導出を行い以後の通信を暗号化します。QUICでは、QUICレイヤで利用するパケット番号のほか、Ackやコネクションのクローズといった制御用メッセージも暗号化されており、適切な鍵を持つクライアントもしくはサーバのみが通信を制御できるようになっています。

また、このハンドシェイク中にはトランスポート用のパラメータが交換されるほか、通信相手が確かにIPアドレスの所持者であることが確認されます（IPアドレスの詐称は検出されます。確証がない段階では大量のデータ送信は制限されており、アンプ攻撃はできないようになっています。）

QUICはUDP上でどうさするため、QUICレイヤでコネクションを管理しており、それぞれのコネクションはコネクションIDという識別子で管理されます。QUICのコネクションは送信者および受信者のIPアドレスやポート番号には依存しないため、それらが変わってもコネクションは維持され、ハンドシェイクからやり直す必要はありません。クライアント(スマートフォンなど)がWi-Fiからキャリア回線へ意図的に移行する場合や、クライアントが意図せず発生するNATリバインディグなどによるポート番号の変化にたいして耐性があります。QUICの仕様ではコネクションマイグレーションとして、手順が定義されています (このときコネクションIDを変えることで、通信を観測している第三者にはエンドポイントの移行がわからないようになっています。)。

パケットロスによるアプリケーションデータの再送要求も工夫されており、ロスしたパケットをそのまま送り直すのではなく、状況に合わせて必要なデータが送り直されます。欠損したアプリケーションデータは必ず相手に届けられますが、アプリケーションデータも追加のデータがあれば合わせて送ることもできるでしょう。それ以外のメッセージに関しては必ずしもすべてのデータが再送される必要はないため、ものによっては送り直されることはありません。

また、QUICにおけるパケット番号は純粋に送信側パケットを送った順番に採番されていき、再送するさいはTCPと異なり新しいパケット番号がつけられます。これにより、受信側はそれが再送されたものと識別できるようになり、パケロスが発生したことが識別できるようになり、その情報を輻輳制御などにフィードバックできるようになります。パケット番号は純粋に送信者が送信した順番を示すので、そのパケット(実際はSTREAMフレーム)が運ぶアプリケーションデータは、全体のうちどの部分なのか(オフセット)については別途指定することになっています。

なおQUICでは通信を暗号化しない通信方法は存在しません。

この文書では以下のトピックおよび、セキュリティ考慮事項は省略されます。

- Core Transport Protocol
    - Explicit Congestion Notification (ECN)
    - Address Validation
    - コネクションマイグレーションの手順
    - フロー制御
    - Ack関連
    - Spin Bit
    - PMTU Discovery関連
- Loss Detection & Congestion Control
    - パケットロスの検出方法
    - 輻輳制御
- Using TLS to Secure QUIC
    - ハンドシェイク及び、鍵の導出
    - Packet Protection
    - 鍵更新

2.3 QUICコネクションとQUICパケットの基礎

QUICではUDP上でメッセージをやり取りします。ここでは、QUICを理解する上で一番基本的な、QUICコネクションとQUICパケットについて説明します。

- QUICコネクション: QUICのコネクションの単位。コネクションIDという識別子で管理される。

- QUICパケット: UDPパケットに格納される、QUICのパケット。QUICパケットごとに送信される。

QUICでは、ロングヘッダパケットとショートヘッダパケットという2種類のパケットが定義されており。これらのQUICパケットと呼ばれるメッセージがUDPパケットに含まれて送信されます。

ロングヘッダパケットはコネクションの確立時に使用され、その後はショートヘッダパケットが使用されます。これらのQUICパケットはPacket Protectionにより殆どの部分が暗号化されます。

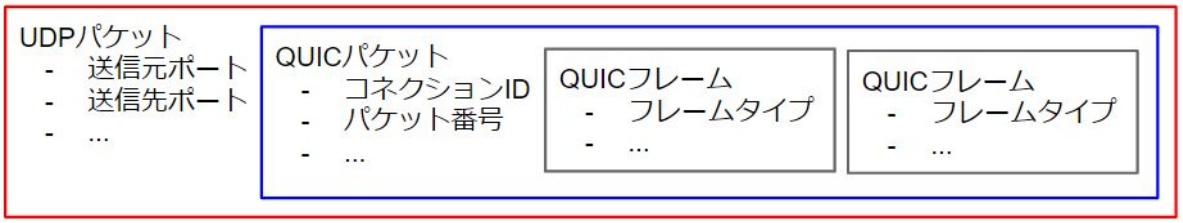
QUICはUDP上で動作しますが、コネクションを一意に識別するためにコネクションIDを持ちます。送信元コネクションIDと送信先コネクションIDとがあり、ロングパヘッダは両方が付加されますが、ショートヘッダでは送信先コネクションIDのみが格納され送信元コネクションIDは省略されます。

また、QUICパケットはパケット番号を持ち。これは、送信者が送った通りに採番されていきます。このパケット番号を見ることで受信者は、パケットロスなどの判断を行っていきます。なお、これらのパケット番号も暗号化されるため、経路上の第三者はパケット番号を見ることも出来ず、パケットロスが発生しているかどうかも測定することは出来ません。

ここまで説明した、QUICパケットはQUICでのメッセージを送るための入れ物です。ハンドシェイクやアプリケーションデータ、コネクションの制御のメッセージなどはフレームという形式でQUICパケットに格納されます。

2.4 フレームについて

QUICパケットには、複数のフレームというメッセージが格納されます。



用途別に20のフレームタイプが存在しており、フレームタイプによってどのようなデータを持つかは異なります。それぞれType値を持っておりQUICパケット内ではこの値を持って、なんのフレームかが識別されます。

アプリケーションデータはSTREAMフレームで送受信されます。それ以外のフレームはハンドシェイク時に使われたり、エラー時を通知するのに送信されたりします。

各フレームは以下のとおりです

Type名	Type値	説明
PADDING	0x00	意味を持たず、QUICパケットのサイズを増やすために使用される。
PING	0x01	通信相手生存していること、相手にパケットが正しく届くことを確認するのに使用される。
ACK	0x02 - 0x03	届いたパケットを相手に伝える、確認応答をするのに使用される。Type値が 0x03のものはECN情報がつく。
RESET_STREAM	0x04	ストリームを終了するのに使用される。
STOP_SENDING	0x05	QUICを使用するアプリケーションの都合により、送られたデータは破棄されるため、相手にストリーム上のデータ送信を止めるよう通知するのに使用される。
CRYPTO	0x06	暗号ハンドシェイク用のデータを送るのに使用される。
NEW_TOKEN	0x07	のちに、再接続する際にハンドシェイク中に使用されるトークン(Address Validation用)を送るのにサーバが使用される。
STREAM	0x08 - 0x0f	ストリームを作成し、ストリームデータ(アプリケーションデータ)を送信するのに使用されます。Type値によってOffsetフィールドを持つか、Lengthフィールドを持つか、ストリー

		ムの最後のデータかの違いがあります。
MAX_DATA	0x10	フロー制御において、相手が送信可能なデータ量を通知するのに使用される。
MAX_STREAM_DATA	0x11	フロー制御において、相手がストリーム上で送信可能なデータ量を通知するのに使用される。
MAX_STREAMS	0x12 - 0x13	相手がオープンすることができるストリームの累積上限を通知するのに使用される。単方向ストリームと双方向ストリームで使用するType値が異なる。
DATA_BLOCKED	0x14	コネクションレベルのフロー制御によりデータを送信できないことを相手に通知するのに使用される。
STREAM_DATA_BLOCKED	0x15	ストリームレベルのフロー制御によりデータを送信できないことを相手に通知するのに使用される。
STREAMS_BLOCKED	0x16 - 0x17	ストリームをオープンしたいが上限に達していることを相手に通知するのに使用される。単方向ストリームと双方向ストリームで使用するType値が異なる。
NEW_CONNECTION_ID	0x18	後に、使用できるコネクションIDを相手に通知するのに使用される。
RETIRE_CONNECTION_ID	0x19	NEW_CONNECTION_IDで発行されたコネクションIDを使用しないことを相手に通知するのに使用される。
PATH_CHALLENGE	0x1a	コネクションマイグレーション時に、新しい通信経路が有効か確認するのに使用される。
PATH_RESPONSE	0x1b	PATH_CHALLENGEへの応答として使用される。
CONNECTION_CLOSE	0x1c - 0x1d	コネクションをクローズする際に使用される。QUICレイヤに起因するものと、アプリケーションプロコルレイヤに起因するものでType値が異なる。
HANDSHAKE_DONE	0x1e	ハンドシェイクが完了したことを通知するのに、サーバから送信される。

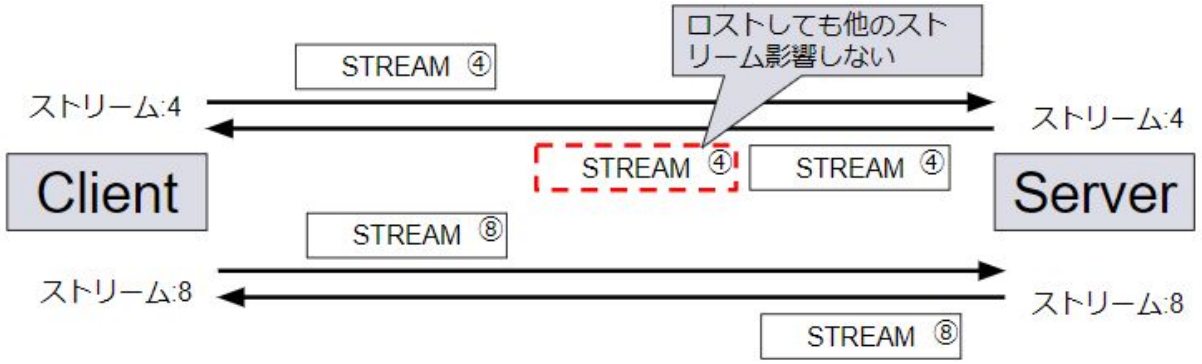
将来、QUICの拡張仕様のなかで、新しいフレーム (拡張フレーム)を定義し、送信するということも出てくるでしょう。しかし、QUICでは相手が理解できないフレーム

を送信してはいけないことになっています。そのため、そのような拡張フレームを送信する場合は、別途新しく定義したトランスポートパラメータで特定の拡張仕様をサポートしていることを事前に通信相手に通知し合います。こうすることで、拡張フレームを送信できるようになります。

2.5 ストリームについて

QUICは一つのコネクション上に、仮想的な通信単位であるストリームを複数持ちます。各ストリームはストリームIDというIDを持ちます。送受信されるSTREAMフレームは、どのストリームようのデータなのかを示すためのストリームIDを持ちます。

ストリーム上でやり取りされるデータは、ストリーム内では順番が担保されています。例えば、ストリームID 4を持つSTREAMフレームが、パケットロスやパケットの順番が入れ替わった場合、それらを回復してからアプリケーション側に渡されます。異なるストリームIDを持つSTREAMフレームはそれらに関わらず処理をすすめることができます。



もちろんこれはイメージですので、実際にはSTREAMフレームはQUICパケットに格納され1つ1つ順番に送受信されていきます。

ストリームには2つの種類があります。単方向ストリームと双方向ストリームです。その名の通り、単方向ストリームはストリームをオープンした側からしかSTREAMフレームを送信できません。一方、双方向ストリームはストリームをオープンした後双方向からSTREAMフレームを送信できます。

先述の通り各ストリームはストリームIDを持ちますが、下位2bitによって使い分けられます。

下位2bit	種類
00	クライアントが開始する双方向ストリーム
01	サーバが開始する双方向ストリーム

10	クライアントが開始する単方向ストリーム
11	サーバが開始する単方向ストリーム

各種類ごとに値が小さいストリームIDを順番に利用していきます。飛ばすことはできません。”クライアントが開始する双方向ストリーム” では、0, 4, 8, 16 と順番に使用して行きます。

ストリームはSTREAMフレームを送ることでオープンし、単方向ストリームではすべてのデータが送り終わるか(FINフラグのたったSTREAMフレームを送り、ACKされる)、RESET_STREAMにより終了されることでクローズされます。

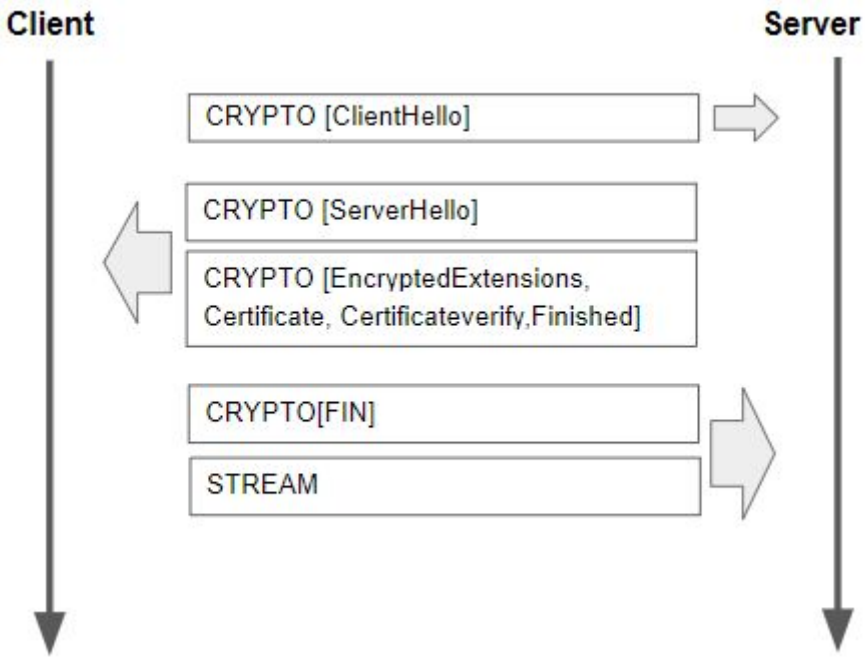
なお、双方向ストリームでは、どちらかがデータ送信を終了した状態があり、ハーフクローズと呼ばれます。双方向で送信を終了するとストリームがクローズ状態になります。

一度使い終わったストリームIDは再利用することはできません。

2.6 コネクションの確立

先述の通り、QUICはTCP+TLSを使う場合より、早くコネクション確立を送り、アプリケーションデータを送信するまでの準備は短くなっています。通常は1-RTTハンドシェイクが使用されますが、一度通信とは0-RTTハンドシェイクも行えます。

QUICではコネクションを確立する際、TLS1.3のハンドシェイクメッセージ（ClientHello、ServerHello、Fin）相当のやりとりをします。相当というのは、TLS1.3のメッセージに保護を加え、CRYPTフレームをQUICパケットで送るためフォーマットが異なるためです。メッセージとしての意味は変わりません。なお、TLS1.3で行えた、0-RTTハンドシェイクも同様に使うことができます。ここでは、1-RTTハンドシェイクについて説明を指定いきます。



やりとりとしてはTLSのハンドシェイクで行われていることとおなじですので、このハンドシェイクのなかでクライアントはサーバ証明書を取得しサーバの真正性を確認し、後続のパケットを暗号化するための鍵をクライアントとサーバで共有します。

クライアントからClientHelloを、サーバからServerHelloなどを受け取ると、クライアントは暗号ハンドシェイクとしてFinを送り続けてアプリケーションデータをSTREAMフレームに詰めて送信できます。このやりとりは、アプリケーションデータを送るまでに1往復のパケット交換(1-RTT)ですみます。

このハンドシェイクの中でAddress Validationも行われています。サーバは、このやり取りの中でクライアントが確かに受け取ったパケットの送信元IPアドレスをクライアントが所持していることを確認します。このAddress Validationが行われることで、悪意ある攻撃者が送信元IPアドレスを詐称しサーバにパケットを送ることで、サーバが攻撃対象者に大量のトラフィックを送信するように仕向けるアンプ攻撃を防止します(最初のClientHelloを送るパケットはパディングを行って、パケットサイズを一定上にすることになっています。これにより増幅率が高くなりません)。

これらのハンドシェイクメッセージでは、TLS1.3でも使われていたSNIなどのTLS拡張も同様に使用されます。ただQUICでは、コネクションに関わるパラメータ（トランスポートパラメータ）を交換するのにquic_transport_parametersというTLS拡張を付加します。

トランスポートパラメータには例えば以下のものがあります

- max_idle_timeout: 最大アイドル時間

- stateless_reset_token: ステートレストークン
- max_ack_delay: ackを送るまでの最大遅延時間
- initial_max_streams_bidi: オープンできる双方向ストリーム数の初期値
- initial_max_streams_uni: オープンできる単方向ストリーム数の初期値

通常のTLSと同様、このハンドシェイクの中で使用するアプリケーションプロトコルについてもネゴシエーションされます。例えばHTTPSでは443番ポートを使用しますが、HTTP/1.1を使うかHTTP/2を使うかのネゴシエーションは後述のALPNという仕組みが使われています。（もちろん、TCPとTLSを使えるアプリケーションプロトコルと、QUICを使えるアプリケーションプロトコルは異なるため適切なアプリケーションプロトコルを選択する必要があります。）

RFC7301 ALPN(Application-Layer Protocol Negotiation)で定義されている仕組みにのっとり、TLS拡張を用いてクライアント側から使用したいアプリケーションプロトコルのリストを提示し、サーバがそのなかから選択します。HTTP/3の場合はh3という識別子をリストに含めて送信します。このALPNはクロスプロトコル攻撃を防ぐのにも役に立ちます。

2.7 コネクションのクローズ

QUICのコネクションのクローズには3種類あります。

- アイドルタイムアウト
- 即時クローズ
- ステートレスリセット

アイドルタイムアウトはその名の通り、事前に決めたアイドルタイムアウト値を超えた場合にクローズされます。

即時クローズは、CONNECTION_CLOSEフレームを送信することでコネクションを即時にクローズすることができます。

ステートレスリセットは、コネクションをクローズするための最後の手段です。例えばサーバ側の再起動などにより鍵情報が失われた場合。クライアントから送られるQUICパケットを読むこともできず破棄するしかありません。また、鍵がないためサーバからクライアントにCONNECTION_CLOSEフレームを送信することもできません。こうなってしまうと、クライアントはタイムアウトまで待つことしかできなくなってしまいます。

この問題を防ぐために、コネクションが確立したときに事前にトークンを発行しておきます。何らかの事情で鍵情報を失った際に、解読できないQUICパケットを受信すると、ステートレスリセットパケットというパケットでトークンを送ります。こ

のパケットを受け取ったエンドポイントは、そのトークンと紐づくコネクションをクローズします。

一度使ったトークンは二度と使用できません。

2.8 負荷分散・トラフィックのオペレーション

QUICはUDPを利用しますが、一連のやりとりは同一のクライアント・サーバで行われます。負荷分散時の振り分け先サーバをどのように決めるか。または、QUICパケットの多くのフィールドが暗号化される点について、いくつかのオペレーション上の話について触れます。

＜TODO＞

2.9 その他 (FEC, Multipath, LB)

IETF QUIC WGでは、Forward Error Correction(FEC)とMultipath機能の拡張性について言及されています。QUIC v1のスコープからは外されており、将来議論されるものになります。仕様の詳細については踏み込みませんが、どういったものなのか簡単に紹介します。

また、ここではQUIC-LBという拡張仕様にも簡単に紹介します。なお、DATAGARAM拡張については4章で紹介します。

2.9.1 Forward Error Correction(FEC)

2013年頃、Google QUICでは、Forward Error Correction(FEC)という機能が検討されていました。これは、QUICのパケットに冗長性データを含め、パケットロスが発生した場合でもその他の受信できているパケットからロスしたデータを復元できるようにする機能です。冗長データを余分に送ることになりますが、ロスしたデータを再送なしに復元できるようになります。

当時検討されたFECの方式としてはXORベースの方式でした。いくつかのQUICパケットをFECグループとして扱い、同じFECグループに属するQUICパケットのXORを取ったデータを追加パケットで送信します。こうすることで、1つのFECグループ内で1つのパケットまではロスしても復元できるようになりました。

このXOR方式は、実際にGoogleのサーバ及びブラウザで実験実装され、Youtubeや検索画面のパフォーマンスの測定が行われました。結果としては、Youtubeでの再生に対して良くない結果となりました。詳細なレポートは下記を参照。

- https://groups.google.com/a/chromium.org/forum/?nomobile=true#!topic/proto-quic/Z5qKkk2XZe0

この調査結果を受けて、Google QUICではFEC機能のサポートを見送りました。IETF QUICでも議論の中でトピックとしてFECについて言及が出てきてはいますが、具体的な方向性はまだ未定となっております。

2.9.2 MP-QUIC

Multipath TCP(RFC 6824)という複数ネットワークインターフェースを利用して通信を行う仕様があります。

例えば、最近のスマートフォンではWi-FiとLTE(4G, 5Gなど)の2つの通信方法が行えるものが出てきています。このような端末において、Wi-Fiおよびキャリア側の通信方法を同時に活用することが出来ます。それぞれの経路で個別にTCPコネクションを確立するのにくらべ、より柔軟に通信をコントロールできるようになります。経路1, 2 があったときに、輻輳は向きがありますので送信は経路1・受信は経路2といったようにすることでメリットを享受するケースもあります。iOSなどではすでにこのMultiPath TCPをサポートしています。

このような複数ネットワークインターフェースを利用した通信方法をQUICでも可能にする拡張仕様がMP-QUICです。2020年5月時点では、まだ正式にQUIC WGで扱われいてる仕様では有りませんが、下記の通り提案仕様が出ております。

Multipath Extensions for QUIC (MP-QUIC)
- https://tools.ietf.org/html/draft-deconinck-quic-multipath-04

2.9.3 QUIC-LB

QUIC-LBの拡張仕様について紹介する

QUIC-LB: Generating Routable QUIC Connection IDs
- https://tools.ietf.org/html/draft-ietf-quic-load-balancers-02

<TODO>

2.10 応用例

QUICはトランスポートプロトコルですので、様々なアプリケーションプロトコルで使うことが可能でしょう。IETF QUIC WGでは、まずアプリケーションプロトコルとしてHTTP/3を検討を行っています。その他のアプリケーションプロトコルについ

ては、まだ本格的に標準化は進められていませんが、いくつかトピックとして紹介しておきます。

- DNS over QUIC:
  QUIC上でDNS通信を行うプロトコル。IETF DNS Privacy WGで「Specification of DNS over Dedicated QUIC Connections」として 標準化が進められています。
- QBone:
  Googleがデータセンタ内をつなぐために利用しているQUIC as a VPNプロトコル。仕様は公開されていませんが、下記資料で簡単に触れられています。
  https://datatracker.ietf.org/meeting/107/materials/slides-107-masque-masque-master-slide-deck-02

もちろん、議論のなかでその他にもQUICの利用が検討されていますが、確度が低いためここでは紹介しません。

TODO

- コネクションID
- Address Validation
- パケット保護
- 再送関連