おさえておきたい基本や、
最新動向を解説するコーナーです。

No.88号 10:00 min

HTTP/3について

01 HTTP/3の登場

HTTP/3は2022年6月頃、RFC 9114 HTTP/3として標準化されています。登場の背景について説明するために時間を遡ります。

2013年頃、Webアプリケーションが一般に広まり、どんどんスマートフォンなどの移動端末でHTTP通信の利用が拡大していました。また、HTTPの高速化をめざしたHTTP/2がRFC 7540として標準化され、利用が広がっていた時期でした。そんなおり、Google社が新しいプロトコル QUIC※1の実証実験を行っていることを発表しました。このGoogle版QUICのコンセプトは、HTTPメッセージの送受信をより高速化するために、トランスポートレイヤの機能について改善をしたものでした。

その後、Google版QUICのコンセプトを元に、IETF版QUICおよびHTTP/3が標準化されていきます。

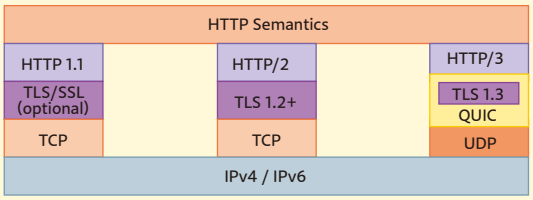
※1　ただし、ここで注意すべきは、このときのQUICと後述するRFC 9000 QUICは別物ということです。Google版QUIC、IETF版QUICと呼び分ける場合もあります。

02 HTTP/3の特徴

HTTP/3はHTTPメッセージをより効率的に転送するためのプロトコルです。HTTPリクエストやHTTPレスポンスといったHTTPメッセージは、HTTP/1.1でもHTTP/2でもHTTP/3でも意味は変わりません。リクエストにはGETといったメソッドがあり、user-agentといったヘッダがあります。このように、各HTTPバージョンはHTTPメッセージの送る通信上のフォーマットが異なりますが、送受信されるそのメッセージの意味が異なっているわけではありません。仕様上もHTTPメッセージのセマンティクスはRFC 9110 HTTP Semanticsという別のRFCとして標準化されています。

HTTP/3の大きな特徴は、後述のRFC 9000 QUICをトランスポートとして利用することです。QUICはUDPを利用するプロトコルですので図1のようなレイヤとなります。

図1　●httpの各バージョンにおけるプロトコルの階層構造●

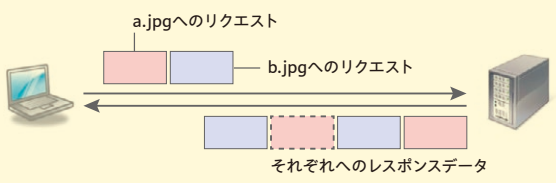


出典 https://en.wikipedia.org/wiki/File:HTTP-1.1_vs._HTTP-2_vs._HTTP-3_Protocol_Stack.svg

HTTP/3の長所を説明するために、まずHTTP/2の課題を説明します。HTTP/2では一つのTCPコネクション上で、複数のHTTPリクエスト・HTTPレスポンスを並列的に送受信することができます。例えば、https://example.com/にa.jpg、b.jpgのリソースがあった場合に、それぞれに対するHTTPリクエストやレスポンスが一つのコネクション上で行われるということです。これにより、つどTCPコネクションを確立する手間もなくなったほか、輻輳制御上も帯域を有効活用できます。しかし、HTTP/2には下位層にTCPを用いるがために課題がありました。パケットロスが起こった際に、ロスしていない後続のパケットを受信できていたとしても、OSレイヤでそのロスしたパケットが回復されるまでユーザーランドのアプリケーションは後続のパケットを読み込めない点です。先程の例で言えば、a.jpg、b.jpgのデータ自体に依存関係はないはずですが、a.jpgのデータでパケットロスが起こるとb.jpgのダウンロードも遅れてしまいます。これは、並列的に処理するといっても大きなデータは分割して、トランスポートの上で直列的に送信されるためです（図2）。

これをHead of Line Blocking（HoLB）と呼びます。この課題を解決するのがHTTP/3です。

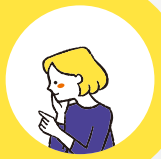
図2　●細かく分割されるデータ●



［ テーマ ］

HTTP/3とは ～QUICとともに～

今回は、広く使われ始めているHTTP/3とQUICについて、その登場背景から特徴までご紹介していきます。



HTTP/3でも単一のコネクション上で複数のHTTPリクエスト･HTTPレスポンスを並列的に送受信できることは同じです。ただし、下位層にQUICを用いることでパケットロスが起こっていた場合でも、受信できている後続のパケットのデータを処理可能です。先の例のように、a.jpgとb.jpgのデータを受信している際にa.jpgデータでパケットロスが起こったとしても、b.jpgのデータを運ぶ後続のデータは問題なく処理を進めることができます。このように、受信しているパケットが処理可能なら処理を進められるのがHTTP/3です。

その他HTTP/3とHTTP/2のその他の違いを紹介します。やはり、ベースにあるのは下位層にQUICを使い、HoLBを起こすような機能で改善が入れられています。

- HTTP/2で導入された優先度制御機能があります。これは並列化されたHTTPリクエストのうち、サーバはどれを優先的に処理するか決める仕組みでした。HTTP/2で導入された優先度制御は複雑な上、処理の依存関係を表現するものでした。その依存関係を示す仕組みも、パケットの順番が入れ替わると制御が難しくなるものでした。HTTP/3ではかわりに、HTTPヘッダで優先度を伝えるRFC 9218 Extensible Prioritization Scheme for HTTPとしてHTTPバージョンによらない優先度制御の仕組みで置き換えられました。

- HTTPヘッダ（フィールド）を圧縮する仕組みもHTTP/2で導入されました。ハフマン符号化や、よく使われるヘッダを辞書に登録することでHTTPリクエスト･レスポンスのヘッダを圧縮する仕組みです。ただし、HTTP/2のヘッダ圧縮は処理に依存関係ができたため、パケットロスがあると処理を進めることができません。そこで、パケットの順番が入れ替わったり、パケットロスがあっても処理可能であれば処理を進めるQPACKというヘッダ圧縮方式が導入されています。QPACKもハフマン符号化と辞書データを用いてヘッダ圧縮するのは同様です。

- HTTPメッセージを並列化するためにHTTP/2で導入されたストリーム機能は、QUICレイヤで提供されるものを使用します。

03 HTTP/3の利用

現在のHTTP/3利用状況について紹介します。現在、Google Chrome、Firefox、Safariといった主要なブラウザではすべてHTTP/3がサポートされています。サーバ側もNginxをはじめとし、いくつかの実装が公開されています。もちろん、クラウドサービスであるGoogle Cloudのマネージドロードバランサや、各社CDNもHTTP/3をサポートしており、広く使用できる状況となっています。

Cloudflareの利用統計※2では、25%程度がHTTP/3を用いた通信になっていることが報告されています。

※2 https://blog.cloudflare.com/ja-jp/http3-usage-one-year-on/

04 HTTP/3の拡張

例えば、Webアプリケーションでリアルタイムの双方向メッセージを実現するためにRFC 6455 The WebSocket Protocolといった技術があります。WebSocketは、プロトコルの観点で見ると、HTTPコネクションを確立したあとに、そのコネクション上でHTTPメッセージ以外のデータを送受信するというものです。QUICを下位層に使うHTTP/3ではよりユースケースが広がり、さまざまな仕組みが議論されています。

○ WebSocket over HTTP/3

WebSocketの仕組みをHTTP/3でも使えるようにする仕様。ブラウザのJavaScriptも既存のAPI仕様が使えるため、アプリケーションとしては利用がしやすい。

○ WebTransport

WebSocketのようにHTTPコネクションを確立したあとに、双方

向メッセージを可能にする仕組み。HTTP/3の利用を前提としており、それにより可能になった送受信の仕方をサポートしています。具体的には、パケットロスが発生しても再送を必要としないDATAGRAM送信をサポートしているほか、HTTP/3の特徴にあったように送信された順序によらず届いたパケットからデータを処理することが可能です。現在標準化が進められており、W3Cでも合わせてブラウザ側のAPI仕様について策定中です。

○ Media over QUIC

WebTransportの上で、メディアデータの配信を行うプロトコルです。既存の仕組みとは異なり、配信者がアップロードする部分・CDNを介してリレーする部分・実際にエンドユーザー（視聴者）に配信するまでをスコープに含めたプロトコルになっています。

○ MASQUE

RFC 9298 Proxying UDP in HTTPのように、HTTPコネクション確立後に、そのコネクション上でUDPパケットを中継する仕組みを定義しています。VPNのような通信をトンネリングさせることが考えられます。UDPパケットのほか、IPパケットを中継させるRFC 9484 - Proxying IP in HTTPも標準化されています。

QUICについて

01 QUICの登場

先述の通りQUICは、Google版QUICのコンセプトを元にIETFで標準化が進められました。その後、2021年5月に下記の通り標準化されました。

○ 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

IETF版QUICは、トランスポートプロトコルとして設計されています。HTTPでの利用を最初のターゲットとしておりましたが、現在はさまざまなアプリケーションプロトコルでの利用が想定されています。

02 QUICの特徴

QUICはUDP上で動作するプロトコルです。UDP上でTCPのような信頼性(再送制御・輻輳制御)と、TLSのような通信の暗号化を提供します。

それ以外にも下記の特徴があります。

○ ラウンドトリップの少ないコネクションの確立

QUICのコネクション確立では1-RTTのハンドシェイクでアプリケーションデータの送受信が可能になります。このハンドシェイクでは、TLS1.3相当のハンドシェイクを行い、通信の暗号に必要な鍵を共有します。また、このやりとりの中で、相手が本当にそのIPアドレスを所持していることの確認も同時に行われます。

○ 制御用メッセージも含む通信の暗号化

TLSではアプリケーションプロトコルのデータが暗号化されており、下位層のTCPレイヤの情報は暗号化されていませんでした。QUICでは、再送制御に必要なメッセージやコネクションの切断といった制御用メッセージも暗号化されます。

○ QUICレイヤによる接続制御により、IP・ポート番号が変わってもコネクションが維持可能

QUICはUDP上で動作しますが、QUICのレイヤでコネクションを管理します。QUICのパケットにはコネクションIDが含まれており、それによってコネクションを識別します。そのため、送信元IPや送信元ポート番号が変わってもコネクションを維持することができます。これにより、例えばWi-Fiからキャリア回線への変更があった際もコネクションの確立からやり直す必要はありません。

○ 効率的な再送制御

トランスポートプロトコルとして、輻輳制御や再送制御を持つQUICですが、既存のアルゴリズムを基本的には踏襲しています。再送制御の点での改善点では、受信できてないパケットの情報を多く相手に伝えることができる点です。また、QUICではパケットロスしたパケットをそのまま再送するのではなく、再送すべきデータのみを新しいパケットで送り直します。

○ ストリーム構造による、並列化されたアプリケーションデータのHoLB回避

QUICはコネクション内に、仮想的な通信単位であるストリームという概念を持ちます。一つのコネクション内に複数のストリームが存在します。各ストリームは独立しており、ストリームが異なるものは別々に処理できます。ストリーム0のデータがパケットロスしたとしても、ストリーム4のパケットが受診できていればストリーム4のデータは処理を進めることができます。

そのほか、UDPベースのプロトコルですのでアンプ攻撃（トラフィック増幅攻撃）への対策や、ユーザー追跡を行えないようにプライバシー観点での対策も行われています。

またプロトコル上の特徴ではありませんが、QUICはUDPレイヤで動作するためユーザーランドで多くの機能が動くのも特徴です。TCPのようにカーネルレイヤで動作する機能は、開発後に実際にOSで実装され展開されるまで時間を要します。トランスポートレイヤの機能改善が早くエンドユーザーに届けられるのもメリットの一つと言えるでしょう。

03 QUICの中身

より具体的なイメージを持ちやすいように、QUICの中身について紹介します。QUICにはQUICパケットと呼ばれるパケット形式があります。この中に複数のフレームと呼ばれるメッセージが格納されます（図3）。

QUICパケットには、コネクションを識別するためのコネクションIDや、パケット番号が含まれています。パケットペイロードにフレームデータが格納されています。なお、パケットペイロードは暗号化されています。

フレームにはアプリケーションデータを送信するのに使うSTREAMフレームのほか、通信の制御に使うフレームをあわせて合計20種類ほどあります。ここでは簡単にいくつか紹介します。

○ PING

コネクションが使用できることを確認するためのフレーム。PINGフレームを受け取った相手は、同様にPINGフレームで応答する必要があります。PINGフレームは任意のタイミングで送信することができます。

○ ACK

再送制御として使用されるフレーム。受け取ったパケット番号の情報を相手に通知するのに使用されます。具体的にはパケット番号の幅が記載されていて、例えば 1~100、102~105、107~110と言ったように受信できてないパケット番号をギャップとして伝えることができます。これにより、届いてないパケットのデータが再送されます。

○ CRYPTO

QUICのコネクション確立する際に行われる、ハンドシェイクの中で使用されます。具体的には、TLS1.3のハンドシェイクメッセージ相当であるClientHello、ServerHello、Certificateなどの情報が格納されます。

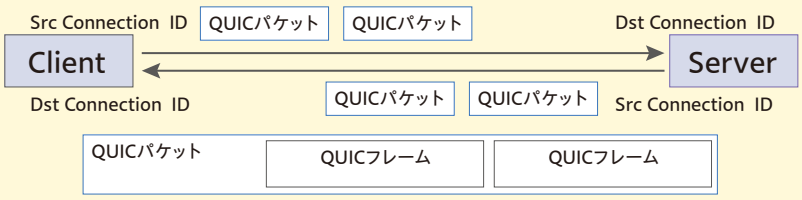
○ STREAM

アプリケーションデータが格納されます。このフレームはストリーム番号を持ち、どのストリームに属するデータなのかが分かるようになっています。また、このSTREAMのデータはパケットロスがあった場合再送されます。

○ CONNECTION_CLOSE

コネクションを切断する際に使用され、エラーコードや、エラーの理由も格納されています。

図3 ● QUICパケットの構造 ●



04 QUICを使うアプリケーションプロトコル

QUICをトランスポートプロトコルとして使うアプリケーションプロトコルは現在も複数議論されています。例えば以下のようなものがあります。

○ DNS over QUIC　○ BGP over QUIC
○ SMB over QUIC　○ RTP over QUIC

トランスポートとしてQUICを使うことで、QUICの特徴として書いたメリットを享受することができます。HTTP同様、そのままアプリケーションプロトコルを乗せるのではなく、ストリームをどのように使うか仕様上で議論されています。

引き続きパフォーマンスが求められるところではQUIC利用が検討されることでしょう。

グリー株式会社　後藤浩行