解説

HTTP/2の最新動向

Recent Progress of HTTP/2 Standardization

大津繁樹

Abstract

ブラウザとWebサーバの間をつなぐ次世代のWebプロトコル（HTTP/2）は，IETFにおいて2014年の標準化完了を目指し，現在急ピッチで仕様策定作業が進められている．HTTP/2は，Googleが開発したSPDYプロトコルをベースとして，その無駄な部分を見直し，より効率的で汎用性が高い通信を実現している．2013年8月からプロトタイプを使った相互接続試験が開始され，2014年夏に最終仕様案の候補が提出される予定である．本稿では，HTTP/2のこれまでの歩み，仕様の概要，相互接続試験で見えた現状の課題と今後の展望について述べる．

キーワード：Web，HTTP，TLS，TCP，高速化

1. HTTP/2仕様化の歩み

World Wide Webが，CERN（欧州原子核研究機構）で誕生してから今年で25周年になる．当初加速器のデータや膨大な実験結果を共有する目的でWebは始まったが，今では日常的に使われる重要な情報通信技術となっている．Webコンテンツも初期のテキストを中心としたページから，次第に画像や動画像などを含むものに変わってきた．最近のソーシャルネットワークサービスの普及とブラウザの高機能化に伴い，高いリアルタイム性と多様なコンテンツを持つWebサービスが求められるようになった．

そのWebの技術の根幹は，HTML（Hypertext Markup Language）とHTTP（Hypertext Transport Protocol）の二つの技術仕様に基づく．HTMLは，2007年からHTML5の仕様策定が進み，従来のブラウザの枠組みを超えるような機能追加が進んでいる．その結果，ブラウザは高度な機能を持つアプリケーションプラットホーム環境に変わりつつある．

他方HTTPは，これまで幾つかの拡張仕様などが追加されているものの，その基本的な仕組みは1990年代に仕様策定されたバージョン1.1から大きく変わっていない．httparchive.orgの統計によると，2011年2月から3年間でHTTPによる平均転送サイズは約2.4倍，リクエスト数は1.2倍に増加している[(1)]．利用者の環境もスマートフォンの急速な普及によって，モバイル環境の通信が主流になり，通信遅延や回線帯域の制限などの影響を大きく受けるようになった．このような環境の変化の中で，従来のHTTPを使ってこれ以上の高速化を図るには限界があった．

Webの高速化を図るGoogleは，Webが表示されるまでの時間が，どの要因に依存しているのか調査した．その結果，回線帯域の増加よりRTT（Round Trip Time）の短縮が有効であることが判明した[(2)]．一般的にRTTを短縮するには物理的な回線の制約があるため，RTTの影響を極力受けない新しい仕組みを作る必要があった．

この結果を受けGoogleは，2009年にSPDY（スピーディ）という新しいWeb向けのプロトコル仕様を発表した[(3)]．SPDYは従来のHTTPで課題となっていた様々なボトルネックを解決し，Webの表示の高速化を図るものである．2010年からChromeブラウザにSPDYが標準実装され，2011年にはGoogleのほぼ全部のサービスで利用されるようになった．2012年以降は他のブラウザに実装が進み，サーバのオープンソース実装が数多く公開された．現在ではTwitterやFacebookなどGoogle以外の大手サービスでSPDYが利用されて

大津繁樹　(株)インターネットイニシアティブプロダクト本部
E-mail　ohtsu@iij.ad.jp
Shigeki OHTSU, Nonmember (Product Division, Internet Initiative Japan Inc., Tokyo, 101-0051 Japan).
電子情報通信学会誌　Vol.97 No.8 pp.727-733 2014年8月

いる．

インターネットの標準仕様を策定するIETF（Internet Engineering Task Force）のhttpbis WG（ワーキンググループ）は，2006年からHTTP/1.1仕様の見直し作業を進めていた．曖昧な記述の修正と関連する拡張仕様の整理を続け，その作業の完了にある程度めどがついたので，2012年に新たにHTTP/2の仕様化を開始することを決定した．

WGはまずHTTP/2に必要な技術要件を定義し，その仕様を公募した．GoogleのSPDYを含む3案の仕様が候補として提案され，議論の末にSPDYがHTTP/2のベース仕様として採用された．仕様化の進め方も通常のIETF総会とは別に中間会議を頻繁に開催し，2014年春にHTTP/2の仕様化完了を目指すこととなった．

httpbis WGは，新たな仕様策定作業の試みとして，検討中のHTTP/2仕様のドラフトを，ソフトウェア開発プロジェクトサービスのGitHubで管理することにした[(4)]．GitHubでは，編集途中のリリース前ドラフトが公開され，誰もがレビューして文面の修正依頼を送付することができる．ただし機能設計に関連する仕様変更は，WGのメーリングリストで議論が必要だ．2014年2月時点でHTTP/2仕様ドラフトはバージョン10（draft-10）まで更新されている．

従来HTTP/2は，これまでHTTP/2.0のようにマイナーバージョンを指す少数値を付けて表記されていた．しかし2014年初めに，今後仕様間での互換性を保持しない方針が決まり，マイナーバージョンの表記が廃止された．今後はHTTP/2.1といったHTTP/2.0と通信互換を持つ仕様の策定を行わず，新しくHTTP/3の開発が行われる予定である．本稿では統一してHTTP/2の表記を利用する．

2. HTTP/2仕様の概要[(注1)]

SPDYをベースとしてHTTP/2の仕様策定が始まったが，基本的にSPDYのプロトコルアーキテクチャを踏襲しただけで，全く同一のものではない．HTTP/2は，より一層の効率化と拡張性を確保するため，SPDYの無駄な部分を廃止し，実運用で判明したSPDYの技術的な課題の解決を目指している．

ほかにもHTTP/2の利点は，従来のWebサーバでHTTP/1.1向けに使用しているコンテンツをそのまま利用できることにある．ただし，ドメイン分割などHTTP/1.1向けに固有なパフォーマンス改善手法を使っている場合，かえって性能が低下する可能性もある．またページ内で必要とされるリソースの量が少ないサイトの場合もHTTP/2による性能向上効果が余り望めないであろう．HTTP/2の性能を最大限に活用するには，その特性を意識したコンテンツ構成にすることが必要である．

HTTP/2を導入する際には，全てのサーバやクライアントを一気に入れ替える必要はない．後述する初期接続機能によって，サーバとクライアントが自動的にネゴシエーションを行い，双方が利用可能である場合に限りHTTP/2の通信が行われる．どちらかがHTTP/2に対応していない場合は，従来のHTTP/1.1で通信を続けることもできる．HTTP/2が普及した後もHTTP/1.1は継続して利用されるため，両者を併用して使い続けることも可能だ．多くの場合，HTTP/2への移行は利用者には全く気付かないものになるであろう．

次にHTTP/2の技術で重要な項目について述べる．

2.1 初期接続

SPDYは，初期接続の制限によって暗号化されたTLS（Transport Layer Security）上の利用に限定されている．HTTP/2では，TLSだけでなく平文通信で接続する方法も提供されている．HTTP/2の初期接続手法は，従来のHTTP/1.1と通信の共存を実現する重要な仕組みである．一方，Webサーバのコンテンツの読込みを高速化するには，できるだけ初期接続のオーバヘッドを少なくする必要がある．また，通信経路途中にあるファイヤウォールやプロキシなどの影響も考慮しなければならない．

このような背景の下，HTTP/2では初期接続で3種類の接続方式が定義されている．TLSのALPN（Alternate Protocol Negotiaion）拡張を使った方法，HTTP Upgradeヘッダを使った方法，事前知識による方法の三つである．それぞれについて，次に簡単に解説する．

2.1.1 TLSのALPN拡張を利用する方法

TLSは，https://でHTTP/2接続を行う場合に利用される暗号化通信である．SPDYでは，TLSの拡張フィールドを利用したNPN（Next Protocol Negotiation）仕様が用いられていたが，Googleが独自で拡張した仕組みであったため，標準化にあたり見直しが行われた．その結果，新たにALPN拡張が仕様化された．

NPNとALPNは，どちらもTLSの拡張フィールドを利用してプロトコルを決定する方式である．TLSのハンドシェイク時に付随してデータ交換が行われるため，初期接続のオーバヘッドが少ないといったメリットがある．NPNは，クライアントがプロトコルを決定する手順に対して，ALPNはサーバで決定を行う点が異なる．これは一般的にセキュリティ関連技術が，サーバに決定権を持たせる手順にすべきであるという方針による．TLS接続は通信経路途中でフィルタされる可能性

(注1)：2014年3月時点のドラフトに基づいた解説を行うため，最終版で変更されている可能性があることを留意されたい．

が最も低く，クライアントとサーバの間で透過的に通信が行われることが多い．そのため ALPN の利用は，現状最も効率が良く HTTP/2 の接続が確立しやすい通信方法であると言える．

2.1.2 HTTP Upgrade を利用する方法

http://で指定されているページに接続する場合，ブラウザは平文通信を利用する．HTTP Upgrade は，この平文通信上で HTTP/2 の利用を行う初期接続手法である．まず最初に，Upgrade ヘッダを持つ HTTP/1.1 のリクエストを送信し，サーバ側が HTTP/2 を受入可能であれば HTTP/2 の通信に切り換える．これは，HTML5 で規定されている WebSocket 通信の開始と同様の仕組みである．現在使われている HTTP/1.1 の仕様を切換方式に利用しているので既存環境との親和性が期待されるが，現実にはファイヤウォールやプロキシなどで Upgrade ヘッダの利用を禁止している場合も多く，実際のネットワーク環境で利用できない場合もある．

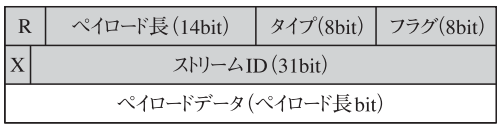
2.1.3 事前知識を利用する方法

接続するサーバが，あらかじめ何らかの方法で HTTP/2 に対応していることが判明している場合，サーバとクライアントの間でプロトコルを決定するステップを省略することができる．この場合，事前にプロトコル情報を入手する方法として，DNS レコードの情報や HTTP ヘッダに指定された情報を参照する方法などが現在検討されている．これらの方法の詳細は，HTTP/2 と別な仕様として今後提案される予定である．

各接続方式の長所と短所をまとめた比較を表 1 で示す．

表 1　HTTP/2 初期接続方法の比較

初期接続方法	長所	短所
TLS ALPN 拡張	最良の接続効率	暗号化必須
HTTP Upgrade	平文で利用可能	禁止環境の存在
事前知識	別仕様で検討予定	



R：予約フィールド（2bit），X：予約フィールド（1bit）
灰色の部分は，フレームヘッダを表す．

図 1　HTTP/2 のフレーム書式　HTTP/2 のフレームは，8 Byte の固定ヘッダとペイロードデータから構成される．最大 16 kByte のペイロードデータを指定でき，10 種類のフレームのタイプと各タイプに応じたフラグが規定されている．フレームにストリームを区別する ID を付与することにより多重化通信を実現している．

2.2　フレーム・ストリームによる多重化通信

HTTP/2 は，バイナリデータで構成されたフレームという単位でデータのやり取りを行う．この点は，テキストをベースとした HTTP/1.1 と大きく書式が異なる．HTTP/2 のフレームフォーマットを図 1 に示す．フレームは 8 Byte の固定のヘッダを持ち，最大 16 kByte までペイロードのデータを運ぶことができる．draft-10 時点では，10 種類のフレームタイプが用意され，各タイプに応じたフラグを指定できる．フレームには後述するストリームの ID が付与されている．サーバとクライアントの間でやり取りする HTTP ヘッダや Web コンテンツのデータは，全てフレームのペイロード領域に格納されるため，ネットワークスタックの視点で見ると，HTTP/2 仕様の位置付けは図 2 に示すとおりフレーミング層の役割であると定義できる．

HTTP/2 は，単一の TCP 接続の性能を最大限に利用することによって性能を向上させている．このことによって，RTT の影響を受けにくいプロトコルを実現する．一つの TCP 上で複数の HTTP リクエストとレスポンスを同時に処理させるために，サーバとクライアントの間で特定のフレームを交換して仮想的にストリームを生成する．各ストリームはフレームで指定された ID で区別され，一つの TCP 接続内で多重化通信を実現している（図 3）．

既存の Web 環境との親和性も重要である．HTTP/2 では，従来の HTTP/1.1 で使われている GET や POST といった HTTP メソッドや，クッキー等の HTTP ヘッダといった HTTP/1.1 のセマンティクスを極力変更しない仕様になっている．一部に制限や変更があるが，HTTP/1.1 でやり取りされているデータの大半は，そのまま HTTP/2 に変換することが可能である．仕様では，HTTP/1.1 を HTTP/2 へ，若しくはその逆の変換を行うプロキシ中継機能が広く利用されることを想定して，

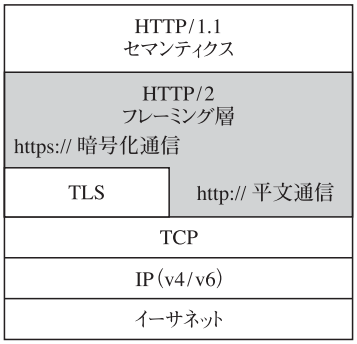


図 2　HTTP/2 のネットワークスタック上での位置付け
HTTP/2 は暗号化通信（TLS）・平文通信（TCP）上の通信をサポートし，HTTP のデータは HTTP/2 のフレーム内に全て格納される．HTTP/2 は，従来の HTTP/1.1 セマンティクスを基本的に維持する．

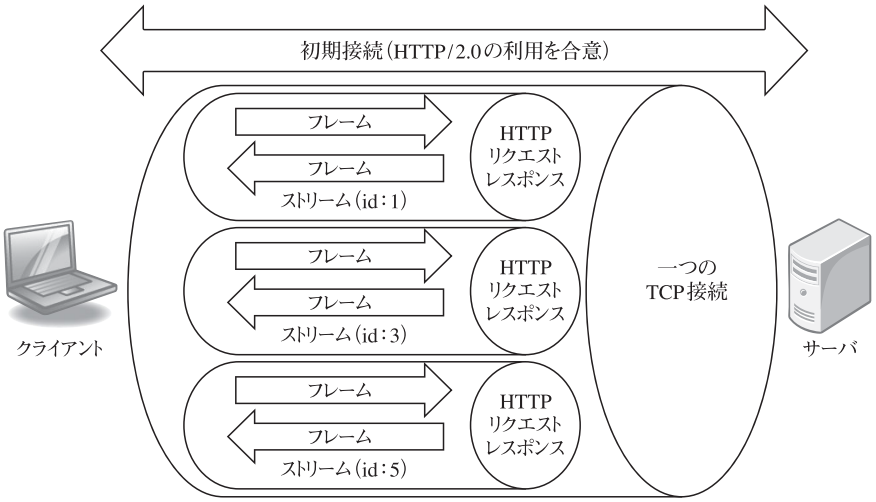


図3　HTTP/2のフレームとストリームによる多重化通信の実現　一つのTCP接続内で仮想的にストリームによって多重化を行い，複数のHTTPリクエスト・レスポンスを同時に処理する．各ストリームはidによって区別される．

HTTP/1.1のセマンティクスの扱いを細かく規定している．

2.3　フロー制御・優先度指定

HTTP/2は，一つのTCP接続中で複数のストリームが同時にデータの送受信を行う．そのためストリーム間で通信帯域を食い合う競合が発生する．特にプロキシ経由で複数の接続先に分散されている状況では，それぞれの通信速度の違いから，ストリーム間で通信量の偏りが発生しやすい．このような問題を解決するために，HTTP/2ではフロー制御機能が用意されている．

HTTP/2の各ストリームは，初期値に64 kByteのウィンドウサイズを持ち，データを受け取るとそのサイズだけウィンドウサイズを消費する．ウィンドウサイズが0になるとデータの送信を停止する．対向からウィンドウサイズを更新するフレームを受け取ると，指定された容量分ウィンドウサイズが増加する．これによってデータの受信側が，送信側のデータを送信するタイミングを制御することが可能となる．フロー制御は，ストリーム単位とTCP接続全体の2種類で用意されている．

ほかにもHTTP/2では，クライアントでWebの画面生成を最適化するために，ストリームの優先機能が用意されている．クライアントは，ファイルの種類によってレスポンスを受信する優先度を指定できる．例えば，画面生成するHTMLファイルやJavaScriptのコードは，画像や動画像よりも高い優先度を指定する．これによってユーザから見た画面表示の体感の向上が図れる．今後優先度の依存性を新たに導入し，複数のリソースの優先度をまとめて変更するといった高度な仕組みも導入することが検討されている．

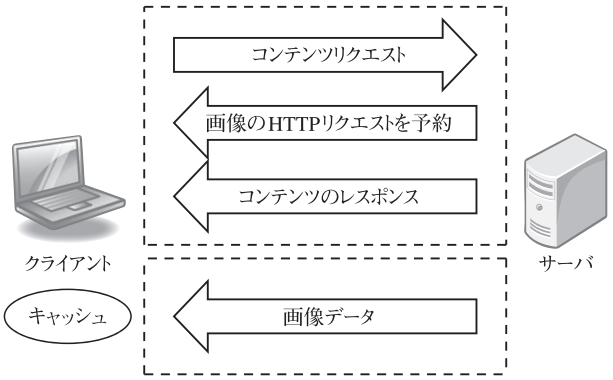


図4　サーバプッシュ機能　サーバはコンテンツの中身を判断し，あらかじめコンテンツに含まれている画像のリクエストを予約する．予約された画像リクエストはクライアントからサーバに送らずに，クライアントはサーバ側からの画像データの送付を待つ．サーバから送信された画像データは，クライアントのキャッシュに保存される．

2.4　サーバプッシュ機能

サーバは，クライアントからコンテンツのリクエストを受信した際に，コンテンツに含まれているリンク情報から，次にクライアントがアクセスするリクエストを予測することができる．HTTP/2では，サーバがクライアントにHTTPレスポンスを返す前に，将来クライアントが送信するHTTPリクエストを事前に予約してサーバからコンテンツを送り込むことが可能である．これをサーバプッシュ機能と呼ぶ（図4）．クライアントは，予約されたストリームのリクエストを送信せず，サーバからコンテンツが送られるのを待つ．サーバがコンテンツを送信すると，クライアントはそのデータをキャッシュする．この機能によって，クライアントが新

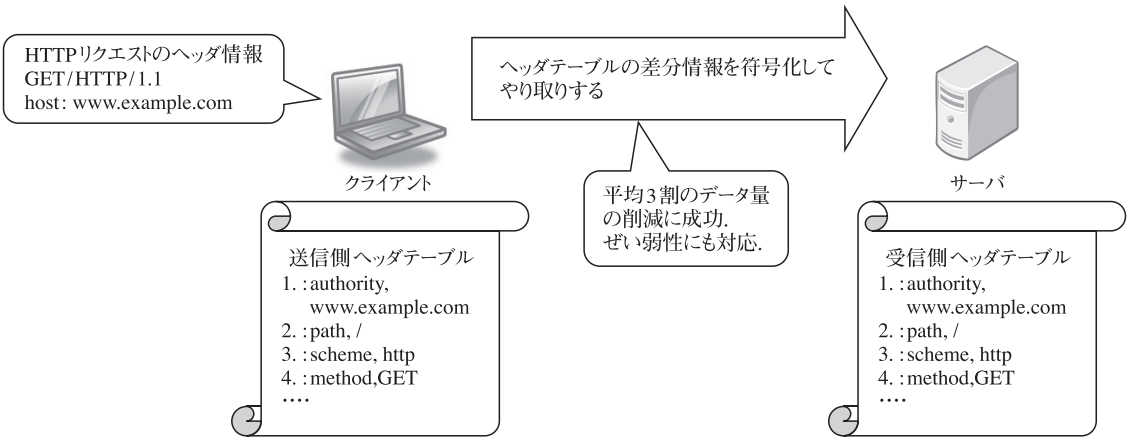


図5　新しいHTTPヘッダ圧縮仕様HPACK　クライアントとサーバ間でヘッダテーブル情報を共有し，その差分情報を符号化したデータを相互にやり取りする．冗長で繰返しの多いヘッダ情報に対して極めて高いデータ圧縮率を達成する．

しくリクエストを送信するオーバヘッドを解消することができる．

2.5　新しいHTTPヘッダ圧縮技術HPACK

HTTPのリクエストとレスポンスには，必ずヘッダ情報が付随する．近年，様々な付加情報がヘッダに追加されるようになったため，その容量は年々増加している．しかもHTTPヘッダは，クッキーのように毎回同じデータを繰り返し送信しているものも多く，冗長性が高い．特にモバイル環境のように回線帯域が限られているネットワークでは，HTTPヘッダの通信量増加による影響は大きく，これをどう削減・効率化を行うかが課題となっている．SPDYでは，一般的な圧縮手法を使ってヘッダデータ量を削減していたが，暗号化された通信上でも圧縮されたヘッダ情報が漏えいするぜい弱性が公表され，現在クライアントでのヘッダ圧縮機能は停止されている．HTTP/2ではそのぜい弱性対策を行い，HTTPヘッダに特化して効率的にデータ量の圧縮を行うHPACKという手法が開発された（図5）．

HPACKでは，サーバとクライアントの両者がヘッダテーブルというヘッダ情報を格納する領域を持ち，テーブルに対する操作とデータの差分情報を符号化して相互に送る仕組みになっている．特に同一のヘッダ情報を繰り返し送るような場合では，差分がないため全くデータを送信しないで済む．HPACKは，平均で3割程度のデータ量に圧縮されると試算されている．HPACKによってぜい弱性のリスクはかなり低減されたが，まだ完全にぜい弱性が解決されたわけではなく，ヘッダ情報の機密性に応じて圧縮するかどうかの判断が必要となる．更にHPACKは仕組みが複雑であるため，一般的にどうやって圧縮が最適化できるのかといった課題も存在する．

3.　実装状況・相互接続試験結果と現状の課題

通常のIETF標準化活動と異なり，HTTP/2は中間会議を頻繁に開催し，極めて短期間で仕様策定完了を目指している．2013年の8月の第3回中間会議以降は，実装ドラフトと呼ばれる仕様を事前に公開し，各自が実装したHTTP/2のプロトタイプを持ち寄って，HTTP/2の相互接続試験を行っている．2014年3月まで計3回の相互接続試験が行われ，ChromeやFirefoxといったブラウザのプロトタイプ実装や各種プログラミング言語で実装されたHTTP/2のソフトウェアが持ち込まれた[(5)]．実験的な試験にもかかわらず導入に積極的なTwitterは，通常のサービスを提供している本番環境を使ってHTTP/2の相互接続試験を行っている．筆者もサーバサイドのJavaScript実行環境Node.jsを使って独自に実装を行い，相互接続試験に参加した．相互接続試験では，仕様検討の議論では明らかにならなかった記述が曖昧な点や考慮されず見逃していた部分などが数多く判明した．

多くの実装者は，既にSPDYの導入で実装のノウハウを持っていたため，相互接続試験で多重化通信の基本的な動作に問題が発生することはなかった．しかしHPACKによるヘッダ圧縮手法は全く新しい試みであったため，当初相互接続性の確認に苦労した．HPACKではクライアントとサーバの間でヘッダ情報を格納するヘッダテーブルのデータが常に同一であることが前提である．テーブルデータは常に更新されるため，当初問題がなくても通信を続けるうちにバグによって不整合が発生する場合もある．そうなるとサーバとクライアントで相互のテーブルの状態を比較し，それまでやり取りしたヘッダ更新情報の突き合わせを行う必要が生じる．この作業は一般的に困難であり，送信側・受信側のどちらに

不具合があるのか特定して解決するのは短時間では非常に難しい．将来的には何らかの仕組みによって，効率的にデバッグができる手法が必要であろう．

HTTP/2 仕様の技術項目は，全て実装が必須なものではなく，幾つか選択可能な項目も多い．例えば初期接続方法において，Chrome や Firefox は TLS 接続のみ実装しているが，平文通信には対応していない．そのため HTTP Upgrade の相互接続試験は，TLS 接続ほど行われていないのが実情である．このように各プロトタイプが実装している技術項目の範囲に偏りがあり，将来的に仕様で規定されている HTTP/2 の機能が，どこまで実際に使われるかは未知数だ．これまで相互試験では主に機能的なチェックが中心であり，最近になって HTTP/2 のテストをどのように行うのかといった議論や取組みが始まったばかりである．今後品質を担保するためにどれだけ網羅的なテストシナリオを作れるかが課題であろう．

4. 今後の展望

現在 HTTP/2 は，仕様化完了に向けて大詰めの作業を迎えている．当初から技術的な検討課題に入っていた項目の仕様化は既に終わっている．そのほか新たに発生した課題については，その重要度や優先度を判断し，後回しができるものは HTTP/3 以降の検討項目として先送りをしている．全体的に HTTP/2 仕様完了作業の足止めにならない方向で議論が進んでいる．

今後は，優先度の依存性指定など仕様を盛り込んだ実装ドラフトをリリースし，6 月上旬に中間会議と相互接続試験を行う予定である．当初 2014 年春に仕様化完了を目指していたが，多少遅延し 2014 年夏に httpbis WG 内で HTTP/2 の最終仕様を合意するラストコールを行い，その後 IETF 組織全体のラストコールを経て仕様化が完了する予定である．

現在 HTTP/2 仕様と並行して議論されている幾つかの技術課題について主要なものを次に三つ紹介する．

（1） WebSocket 対応

WebSocket は，Web サーバとクライアントの間で双方向のデータが通信できる仕様である．HTTP/2 と異なり，WebSocket は，任意のデータ形式で相互にやり取りし，JavaScript のインタフェースを持っているのが特徴である．HTTP/1.1 では，HTTP Upgrade を用いて切換を行う仕組みであったが，HTTP/2 では TCP 接続を維持した状態で別のプロトコルに Upgrade で切り換えることを明示的に禁止している．そのためインタフェースの互換を維持しつつ，WebSocket のフレーム書式を変更して HTTP/2 上にデータをマップしていく方針で現在検討が進んでいる．

（2） 日和見暗号（opportunic encryption）

NSA（アメリカ国家安全保障局）の元職員エドワード・スノーデンによるインターネットの広範囲な盗聴行為の暴露により，ネットワークセキュリティの脅威はこれまで以上に深刻なものとなった．ほかにもモバイルの公衆無線回線サービスの普及により，一般のユーザが知らないうちに通信を盗聴されるセキュリティリスクも増加している．

こうした状況を受け，インターネットをもっと堅ろうなものに変えていくべきであるという意見が IETF 内で日増しに高まってきた．この動きを受けて httpbis WG 議長は，HTTP/2 を平文の通信で利用することを禁止し，TLS 利用に限定するという提案を行った．議論は賛否が分かれ，結局合意に至らなかった．代わりにこれまで平文通信であった http://へのアクセスを暗号化するといった日和見暗号の採用について議論が進んだ．

日和見暗号は，接続先のサーバ証明書による認証を行わない TLS 接続方式を指す．この場合，経路途中でデータを改ざんする攻撃には対処できないが，通信の暗号化が行われるため単純にデータを盗聴する攻撃に対応できるものである．HTTP/2 の初期接続やフレームのやり取りで日和見暗号接続に切り換えられるようにすれば，既存の平文でやり取りされている状況よりもセキュリティ的により強固な状態になるとの判断である．ただし先述したとおり限定的なセキュリティ対応になるため，導入に否定的なベンダも多い．

（3） 明示的なプロキシ機能（Explicit Proxy）

明示的なプロキシは，クライアントとプロキシサーバの間を TLS を使って HTTP/2 接続する構成を指す．このプロキシサーバは，コンテンツフィルタやウイルスチェック，コンテンツキャッシュ等の機能を提供する．平文の HTTP/2 通信はプロキシで処理されるが，暗号化された TLS 接続は何もせず透過的に通信を転送する．

明示的である理由は，プロキシサーバ用の証明書を新たに定義し，クライアントがプロキシの接続時にユーザを選択できるようなユーザインタフェースを用意することを指していることによる．現在 Google は，スマートフォン向けにコンテンツ圧縮を行うサービスを展開しているが，同様のサービスにプロキシ利用の明示的な仕組みを追加するのが目的である．

以上の課題は，HTTP/3 以降で導入が検討されている項目である．

今やブラウザは数か月単位でバージョンアップを自動的に行うのが普通である．HTTP/2 もある程度仕様が固まれば，標準化の手続きを待たずしてブラウザで一般の利用者が使えるようになるであろう．Google や Twitter などサービス大手は，ドラフト段階で本番サービス

のHTTP/2対応を進めている．新プロトコルの様々な課題が明らかになるとHTTP/3以降の仕様検討が直ちに始まる予定である．HTTP/2は10年以上続いたHTTP/1.1とは異なり，これまでにないような早いサイクルでバージョンアップが行われる予定である．HTTP/2の仕様完了は，HTTPプロトコルの新たな更新サイクルを切り開く第一歩として非常に大きな意義を持っているものと言える．

文　　献

（1） http://httparchive.org/trends.php?s=All&minlabel=Jan＋31＋2011&maxlabel=Feb＋1＋2014#bytesTotal&reqTotal

（2） https://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

（3） http://www.chromium.org/spdy

（4） https://github.com/http2/http2-spec

（5） https://github.com/http2/http2-spec/wiki/Implementations

（平成26年4月1日受付　平成26年4月16日最終受付）

大津（おおつ）　繁樹（しげき）

平3東大・工卒．平7同大学院博士課程中退．博士（工学）．平11㈱インターネットイニシアティブ入社．現在同社プロダクト本部アプリケーション開発部所属．HTTP/2やNode.jsなど先端技術の検証・評価に従事．