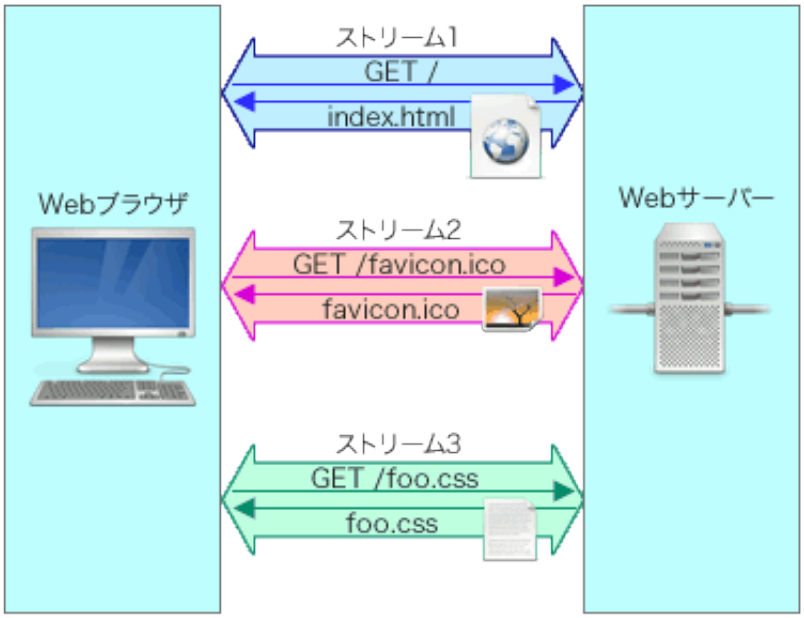
普及が進む「HTTP/2」の仕組みとメリットとは

エンジニア向け

2017.03.24



HTTPの新規格「HTTP/2」が2015年2月に正式に承認されてから約2年が経った現在、WebサーバーやWebブラウザでのサポートも進んでおり、現実的に導入が可能な状況となってきた。そこで今回はこのHTTP/2についての基礎知識と、その活用方法について紹介する。

HTTP/2とは

クライアント（Webブラウザ）とWebサーバーとの間で、どのようにデータをやり取りするかを定めた仕様がHTTP（Hyper Text Transfer Protocol）だ。HTTPの初期バージョン（HTTP/0.9）は1990年に開発されたもので、クライアントはリクエストしたいコンテンツのパスをWebサーバーに送信し、Webサーバーはそれに対応するコンテンツを返す、といった単純なものであった。その後、WebサーバーやWebブラウザの機能強化に応じてPOSTリクエストなどに対応した「HTTP/1.0」や、バーチャルホストなどに対応した「HTTP/1.1」が策定され、現在では多くのWebサーバーやクライアントがHTTP/1.1を利用してやり取りを行っている。

このようにたびたびHTTPはアップデートされていたものの、「1つのリクエストに対して1つのレスポンスを返す」という基本構造はHTTP/0.9から変わっていない。HTTP/1.1ではクライアントが複数のリクエストを同時に送れるようになったが、その場合も必ずリクエストされた順でレスポンスを返すようになっている。そのため、たとえばレスポンスとして容量の大きいファイルを返す場合や、レスポンスを返すための処理に時間がかかるような場合でも、後続のレスポンスは先のレスポンスの送信が完了するまで送信できない（**図1、2**）。そのため、多くのWebブラウザではWebサーバーに対して同時に複数のコネクションを確立することとで、並行してコンテンツをダウンロードできるよう実装されている。

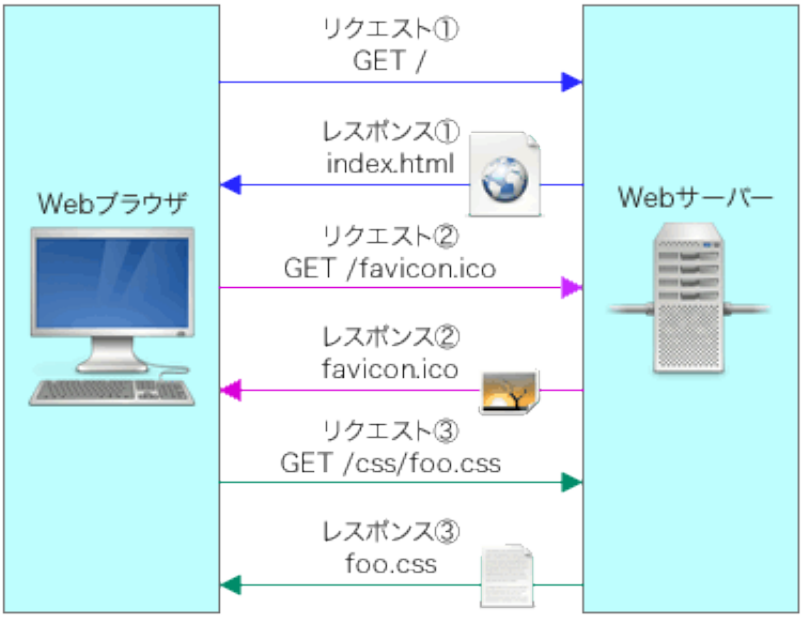


図1 HTTPでは、1つのリクエストに対し1つのレスポンスを返す、というのが基本

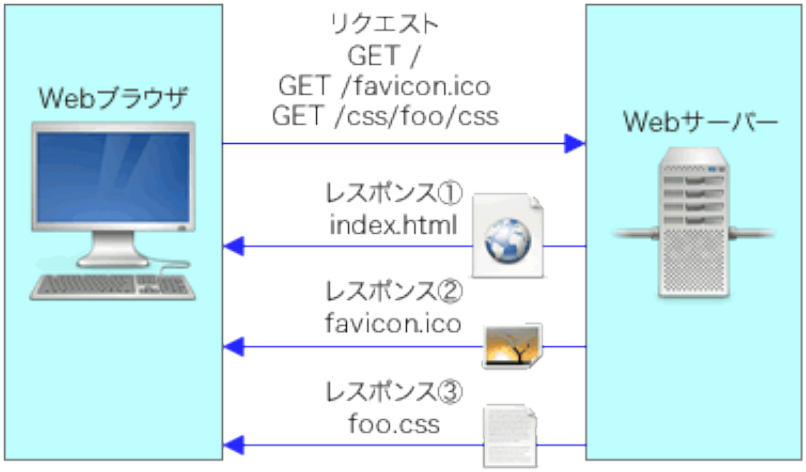


図2 HTTP/1.1ではクライアントが同時に複数のリクエストを送ることが可能になったが、その場合必ずリクエストされた順にレスポンスが返る

また、HTTPではサーバーとクライアント間のやり取りにHTTPヘッダ（HTTPリクエストヘッダおよびHTTPレスポンスヘッダ）を使用するが、近年ではサーバーとクライアント間でやり取りするヘッダの量が増大する傾向にある。たとえば**図3**はWikipediaのトップページ（https://ja.wikipedia.org/wiki/メインページ）を開いた際に送受信されているリクエストヘッダおよびレスポンスヘッダだが、非常に多くのヘッダが送受信されていることが分かる。

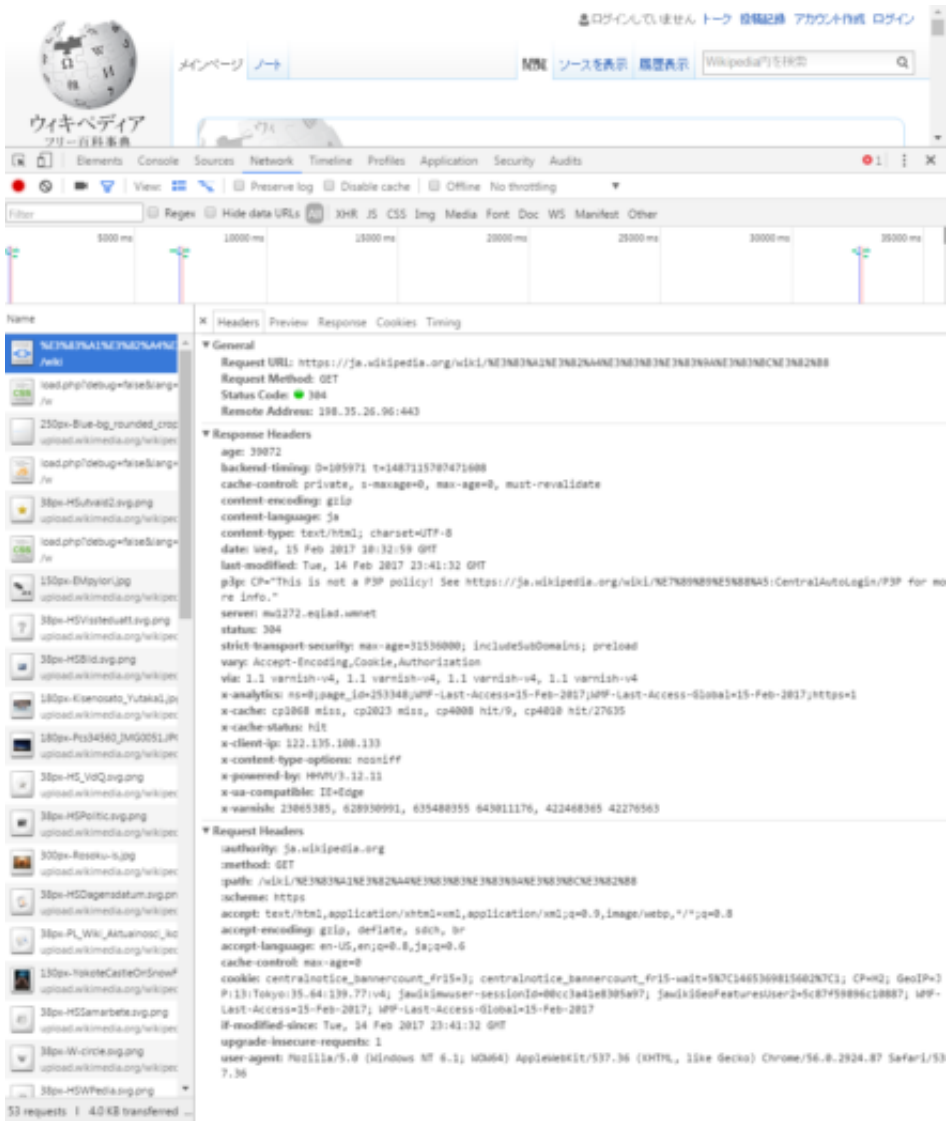


図3 Wikipediaトップページを開いた際に送受信されるHTTPヘッダ

これらの問題はHTTPの根本的な仕様が原因であるため、従来の延長線的な仕様変更では対応が難しかった。そのため新たに開発されたのがHTTP/2である。

HTTP/2の特徴

HTTP/2は従来のHTTPを踏襲しつつも、新たな転送手段を提供することで既存の問題点を解決し、より少ない通信量でより迅速にやり取りを行えるようになっている。

HTTP/2の最大の特徴は、「ストリーム」という概念を導入したことだ。これによって1つのコネクション内で同時に並行して複数のリクエスト/レスポンスを処理できるようになった（**図4**）。

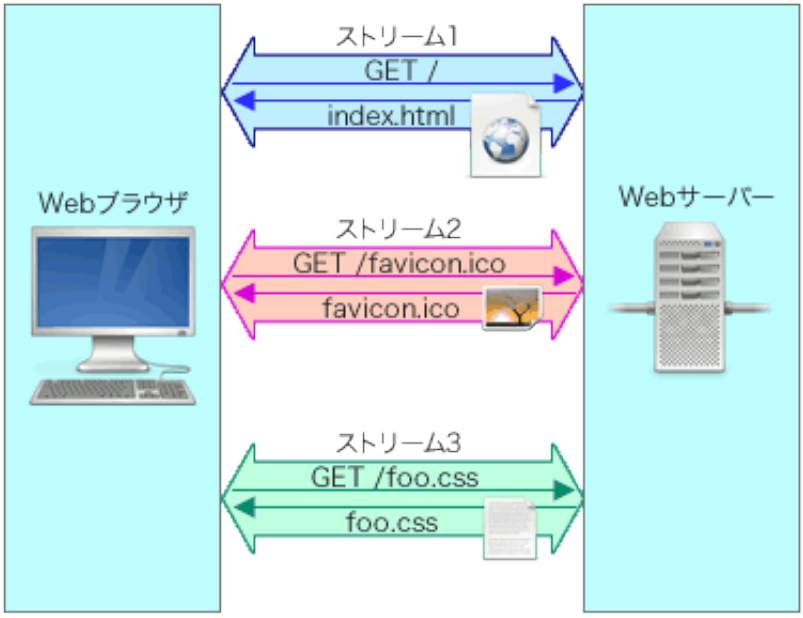


図4 HTTP/2では「ストリーム」という概念を導入し、1つのコネクション内で複数のリクエスト/レスポンスを並行処理できるようになった

ストリームはクライアント–サーバー間を結ぶ仮想的なコネクションのようなもので、1つのHTTP/2コネクション内で同時に複数のストリームを確立できる。HTTP/2ではこのストリーム内でリクエストやレスポンスをやり取りする仕組みになっている。

ストリームはそれぞれ独立しており、あるストリームでやり取りされている内容がほかのストリームに影響を及ぼすことはない。そのため複数のリクエスト/レスポンスを同時並行的に処理でき、レスポンスを返すための処理に時間がかかるようなケースでも、その間に別のリクエスト/レスポンスを処理することが可能となった。たとえば**図4**の例では3つのストリームが確立されており、それぞれのストリーム内でのリクエストやレスポンスはほかのストリームの状態によらず送受信が可能になっている。

ストリームごとに「優先度」を設定することも可能で、たとえばコンテンツの表示に必須のデータを優先して送信する、といった処理も可能だ。また、ストリーム毎にフロー制御を行う機能も用意されている。

そのほか、従来のHTTPはクライアント/サーバー間はテキストベースで行われていたが、HTTP/2ではクライアント/サーバー間の通信がバイナリベースで行われるようになった。さらにクライアント/サーバー独自に機能を拡張するための仕様も策定されている。

このように、HTTP/2ではクライアント/サーバー間でやり取りを行う方式が大きく変わっているが、送受信されるデータについては従来のHTTPからは変わっていない。リクエストやレスポンスに「キー:値」形式のHTTPヘッダが付与される点も同じだ。そのため、たとえばCookieやHTTPヘッダを使った認証などはまったく同じように動作する。さらにHTTPヘッダの送受信時には圧縮/展開が行われるようになり、効率的なやり取りが可能となった。

また、HTTP/2のもう1つの大きな新機能として、リクエストされていないコンテンツをサーバーがクライアントに送信する「サーバープッシュ」機能がある。たとえばクライアントがHTMLファイルをリクエストした場合、それに続いてそのHTMLファイル内で指定されているCSSファイルやスクリプトファイルなどがリクエストされる可能性が高い。サーバープッシュ機能を利用することでそういった関連ファイルを自動的にサーバーからクライアントに送信することが可能になり、クライアントがリクエストを行うオーバーヘッドを削減できる。

HTTP/2で利用されるポートとスキーマ

HTTP/2はHTTP/1.1と完全な互換性が保たれており、使用するデフォルトのポート番号もhttpの場合80番、httpsの場合443番で変わらない。

また、HTTP/2ではポート番号とは別に、どのような方式で接続を行うかを決めるための「識別子」が定義されている。定義されている識別子は「h2」（HTTP/2 over TLS）と「h2c」（HTTP/2 over TCP）の2つで、h2はTLSで保護されたコネクション上（いわゆるhttps）でHTTP/2を利用することを示し、h2cは暗号化されていないTCPコネクション上（いわゆるhttp）でHTTP/2を利用して通信を行うことを示すものだ。

HTTP/2では規格上はTLSによる暗号化があってもなくても利用できるのだが、TLSを利用するh2にしか対応していないクライアントも多い。特にGoogle ChromeやFirefox、Safari、Internet Explorerといった一般的なWebブラウザはh2cに未対応となっているため、HTTP/2を利用するためには事実上TLSが必須という状況になっている。これは、プライバシ保護などのためにWWWで暗号化通信（HTTPS）を必須とするような動きが進んでいることが背景にある。また、HTTP/2でTLSを使用する際はTLS 1.2以降が必須とされている。

HTTP/2での通信の流れ

HTTP/2は従来のHTTPと完全な互換性を保つようになっているため、そのURLやポート番号といった情報だけでは、接続しようとしているサーバーがHTTP/2に対応しているかどうかは判断できない。そのため、HTTP/2に対応するクライアントはまずはHTTP/1.1以前と同じ方法でWebサーバーに接続を行い、サーバーが対応していればHTTP/2に移行する、という手順で通信を行う。

具体的には、まずWebサーバーに接続したクライアントはリクエストと同時に「Connection:」ヘッダや「Upgrade:」ヘッダ、「HTTP2-Settings:」ヘッダをリクエストヘッダとして送信することで、クライアント側がHTTP/2に対応していることを示す。

たとえば次の例は、「http://server.example.com/」というURLへのリクエストを行う場合に送信されるリクエストだ。

```
GET / HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
```

```
HTTP2-Settings: ...
```

ここで「HTTP2-Settings」ヘッダには通信方法に関する設定をBase64形式でエンコードした情報が含まれている。これに対しHTTP/2に対応するWebサーバーはステータスコード101のレスポンスを返し、HTTP/2コネクションに移行する。

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c

以下、HTTP/2コネクションとなる
:
:
```

HTTPSによる接続の場合も、Upgrade:ヘッダで送信される識別子が「h2」になる以外の流れは同じだ。

また、接続先のサーバーがHTTP/2に対応していることが分かっている場合、クライアントは最初からWebサーバーに対してHTTP/2で接続を行うことも可能だ。この場合、クライアント/サーバーともに最初に「HTTP/2コネクションプリフェイス」と呼ばれる24オクテットのデータを送信するようになっている。

なお、これまでの説明はあくまでHTTP/2の概要であり、より具体的な規格内容などに興味のある方はRFC7540などの文書を確認して欲しい。

HTTP/2に対応するサーバー/クライアント

2015年2月にHTTP/2が正式に承認されてから約2年が経った現在、多くのサーバーやクライアントがHTTP/2に対応している。たとえば、GitHub上にあるHTTP/2仕様書の公式リポジトリ（http2/http2-spec）内のWikiページで公開されているImplementationsページではHTTP/2に対応する50以上のクライアントやサーバー一覧がまとめられている。ここから分かるように、昨今ではRubyやGo、Python、Java、Node.jsなど各種言語向けのWebサーバーライブラリもHTTP/2に対応しつつある。なお、この表ではh2およびh2cのサポートについても記されているが、そこにある「h2-14」というのはHTTP/2のドラフト14を意味している。

Webブラウザについては現時点でInternet Explorer 11およびEdge 14移行、Firefox 50以降、Google Chrome 49（Android版は55）以降、Safari 10（iOS版は9.3）以降、Opera 42以降、Android 4.3以降な

どがHTTP/2に対応している。ただし、前述のとおり基本的にサポートされているのはTLSを使用するh2のみで、多くのWebブラウザでは平文でやり取りを行うh2cについては未サポートとなっている。

HTTP/2を利用するメリット

さて、ここまででではHTTP/2の技術的な話について紹介してきたが、多くの方が興味があるのは、HTTP/2を利用することで実際にどのようなメリットがあるのか、という点だろう。

HTTP/2導入のメリットとしてよく言われているのが、クライアントーサーバー間の通信が効率化されることでコンテンツの伝送が高速化されるというものだ。とはいえ、前述のとおりHTTP/2でやり取りされる情報は従来のHTTPとまったく同じであり、その伝送方法が変更されたに過ぎない。そのため、たとえば既存のWebサイトにおいて、Webサーバーの設定等を変更して単純にHTTP/2に対応させただけでは、大きなパフォーマンス向上は見られない可能性もある。

しかし、それだけでHTTP/2は実際にはあまり効果がない、というのは早計だ。HTTP/2で重要なのは、優先度制御や非同期でのリクエスト/レスポンス処理が可能になった点にある。これらによってWebブラウザにおけるページレンダリング過程などを考慮した効率的なリクエストが可能になり、結果としてWebブラウザにおいて画面表示が完了するまでの時間を短縮できる可能性がある。

具体的にはGoogleが公開している「クリティカル レンダリング パスのパフォーマンスを分析する」というドキュメントが参考になるが、昨今のWebブラウザではそのページのHTMLファイルに加えて、そのHTML内で指定されているCSSファイルやJavaScriptファイルなどのダウンロードが完了しないとページのレンダリングが開始されない。いっぽうでページ内に含まれている画像などのコンテンツについては、ダウンロードの完了を待たずにページレンダリングを開始できるようになっている。

HTTP/2では優先制御やサーバープッシュといった機能が用意されており、これによってWebブラウザ側でレンダリングに必要なコンテンツを優先的に送信するようリクエストしたり、Webサーバー側で送信するコンテンツの内容を確認し、送信する際の優先度を変更することが可能になっている。たとえばこういった機能を使って画像よりもCSSやJavaScriptファイルを優先してWebブラウザ側に送信することで、Webページを表示する見かけ上のスピードを向上させられる可能性がある。

そのためにはWebブラウザ側での適切なリクエスト生成や、Webサーバー側での設定などが必要になってくるのだが、現状ではまだいくつかのWebブラウザでは優先度制御がサポートされていなかったり、またサーバープッシュについてもWebブラウザ側でキャッシュされている情報をサーバーが無駄に送信する可能性がある、という問題がある。こういった問題を解決するための手法も考案されているが、まだ普及までには時間が必要だろう。

HTTP/2を利用するためのサーバー設定

ここまではHTTP/2の技術的な話について述べてきたが、本記事の後編では実際にHTTP/2に対応するWebサ

ーバーでHTTP/2を利用するためのサーバー設定について紹介する。

ネットワーク , サーバー管理 , 運用管理 , ソフトウェア , HTTP/2