TLS, HTTP/2, QUIC

IIJ 大津 繁樹

リクルートテクノロジーズ社内勉強会

2016年8月19日

自己紹介

- 2013〜2015年にHTTP/2の標準化作業に参画
- 2015年よりIPA主催セキュリティ・キャンプにて TLS技術の講師を担当している。
- オープンソースプロジェクト Node.js の Core Technical Committee メンバー、TLS/crypto 関連機能の技術担当。

本日の内容

1. Webプロトコルの進化

    - 過去、現在。未来でWebプロトコルがどう変わっていくのか？(HTTP/2,TLS,QUICの位置付け)

2. 常時TLSへ至る背景

    - どういう経緯で常時TLSが必要となったのか？

3. 常時TLSとHTTP/2

    - HTTP/2のメリット・デモ、常時TLSとの関係

4. これからどう変わっていくのか？

    - 次期バージョンTLS1.3、新プロトコルQUICについて（デモ）

何よりもまず
常時TLS化から始めましょう

- Googleは、なぜ常時TLS(HTTPS Everywhere)を推奨しているのでしょうか？

- HTTP/2はTLS(暗号化)/平文の両方で使用できるよう規定されているのに、実質TLS必須です。なぜ？

- そもそも常時TLSは本当に必要なんでしょうか？

全ては2013年6月が転機

1994 SSL2.0

2008 TLS1.2(RFC5246)

2012 HTTP/2仕様化開始

2013 **6月 エドワード・スノーデン事件(*)**

11月 TLS1.3仕様化開始 **常時SSLへの動きが本格化**

2014 Googleブログ「HTTPS as a ranking signal」

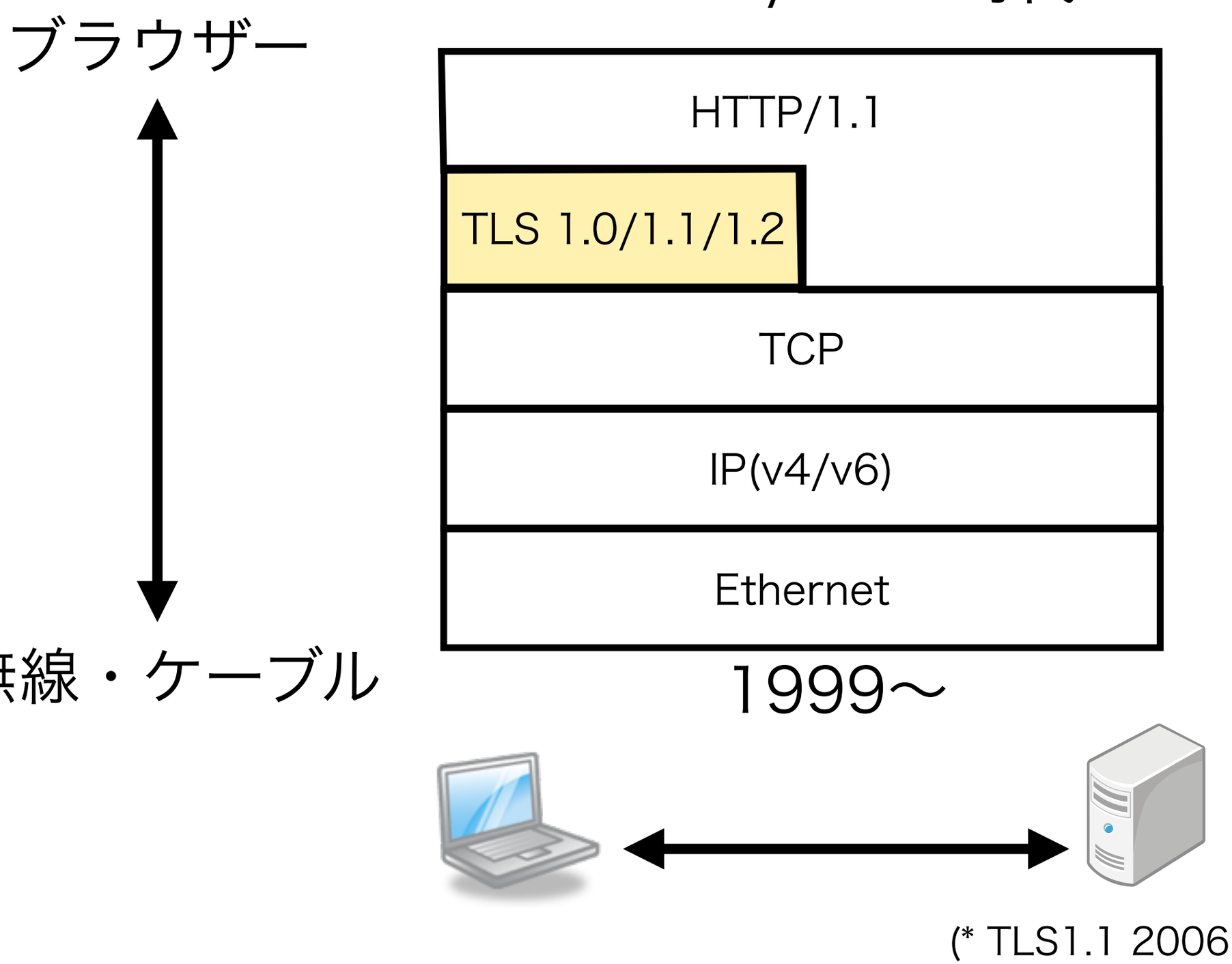
2015 HTTP/2仕様化完了(RFC7540)

2016 TLS1.3仕様化完了見込み

(* NSA:アメリカ国家安全保障局による広範囲な盗聴行為の暴露)

Webプロトコルの進化

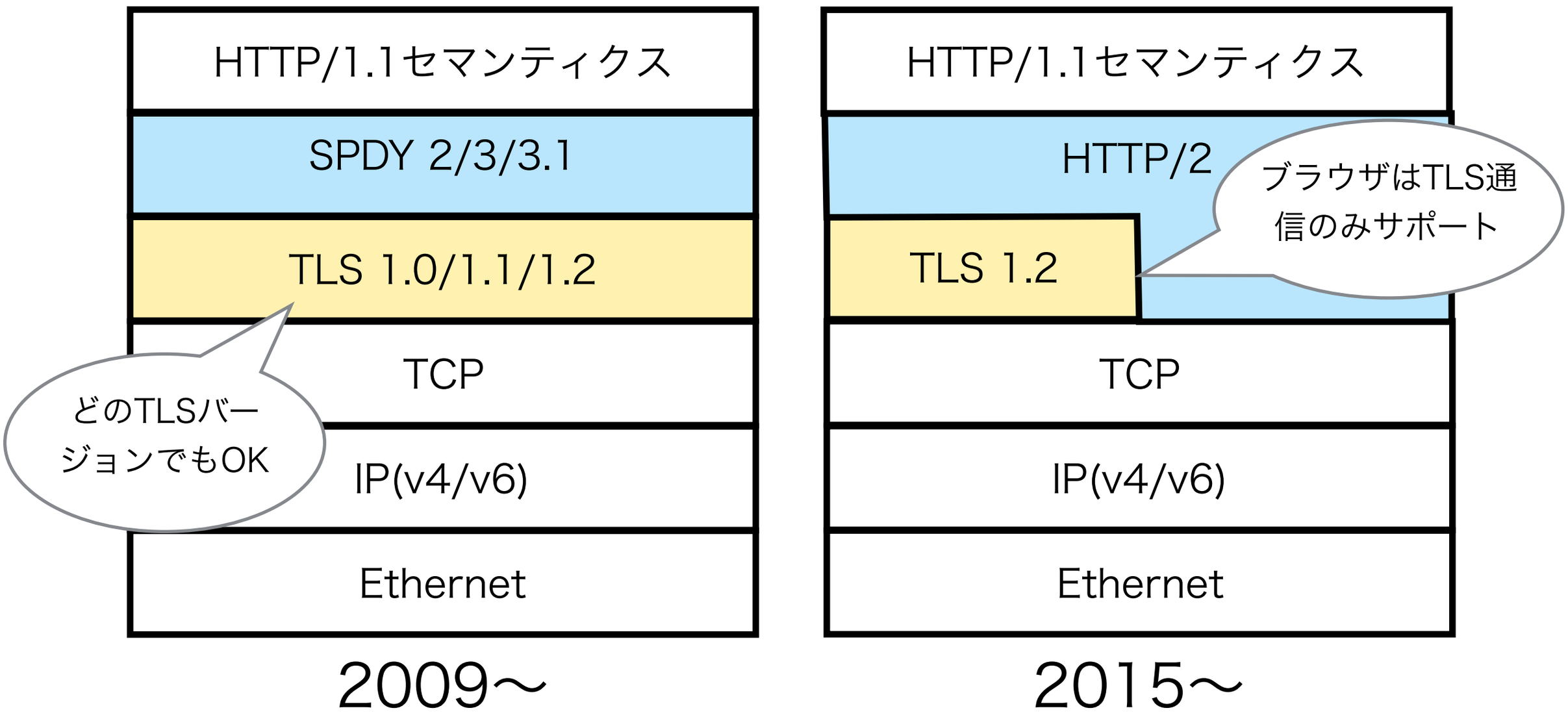
HTTP/1.1の時代



(* TLS1.1 2006〜) (* TLS1.2 2008〜)

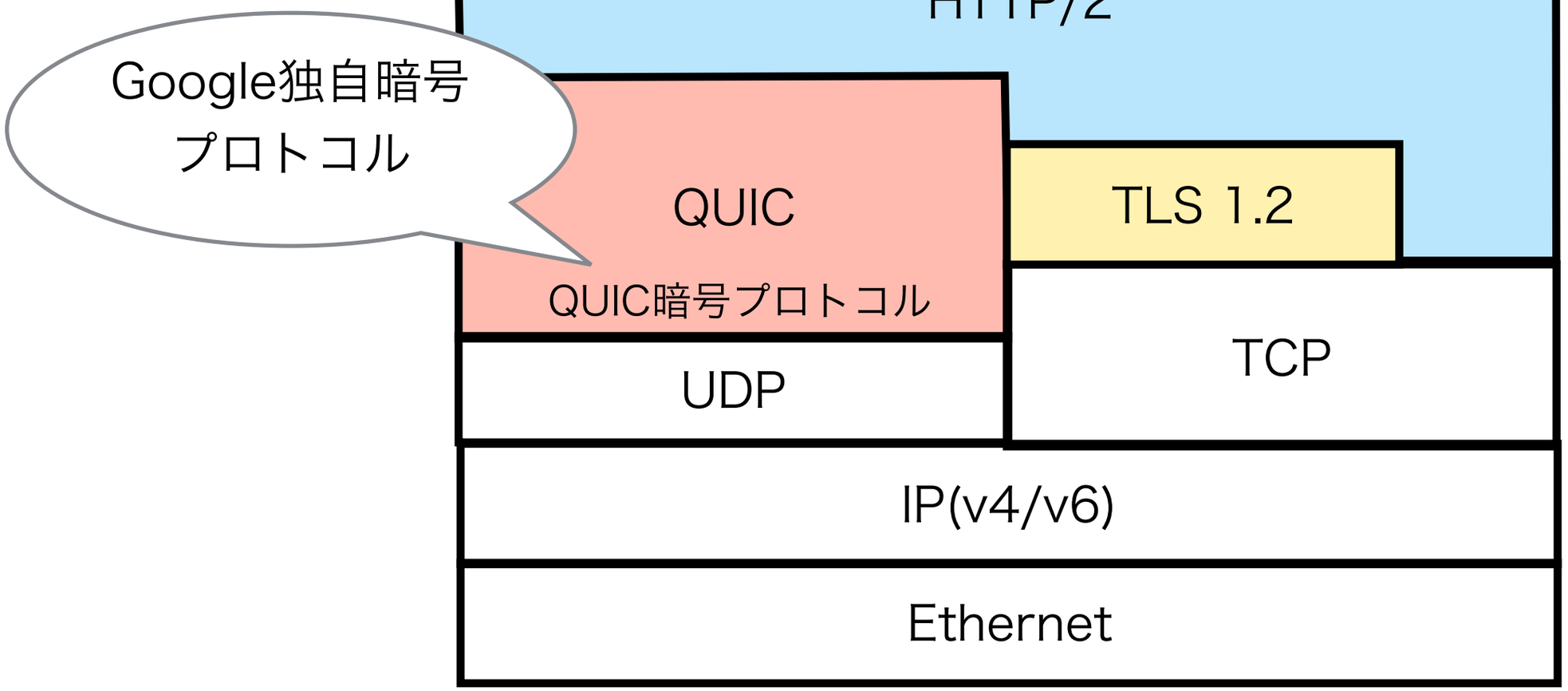
Webプロトコルの進化

HTTP/1.1からHTTP/2へ



Webプロトコルの進化

HTTP/2からQUICへ

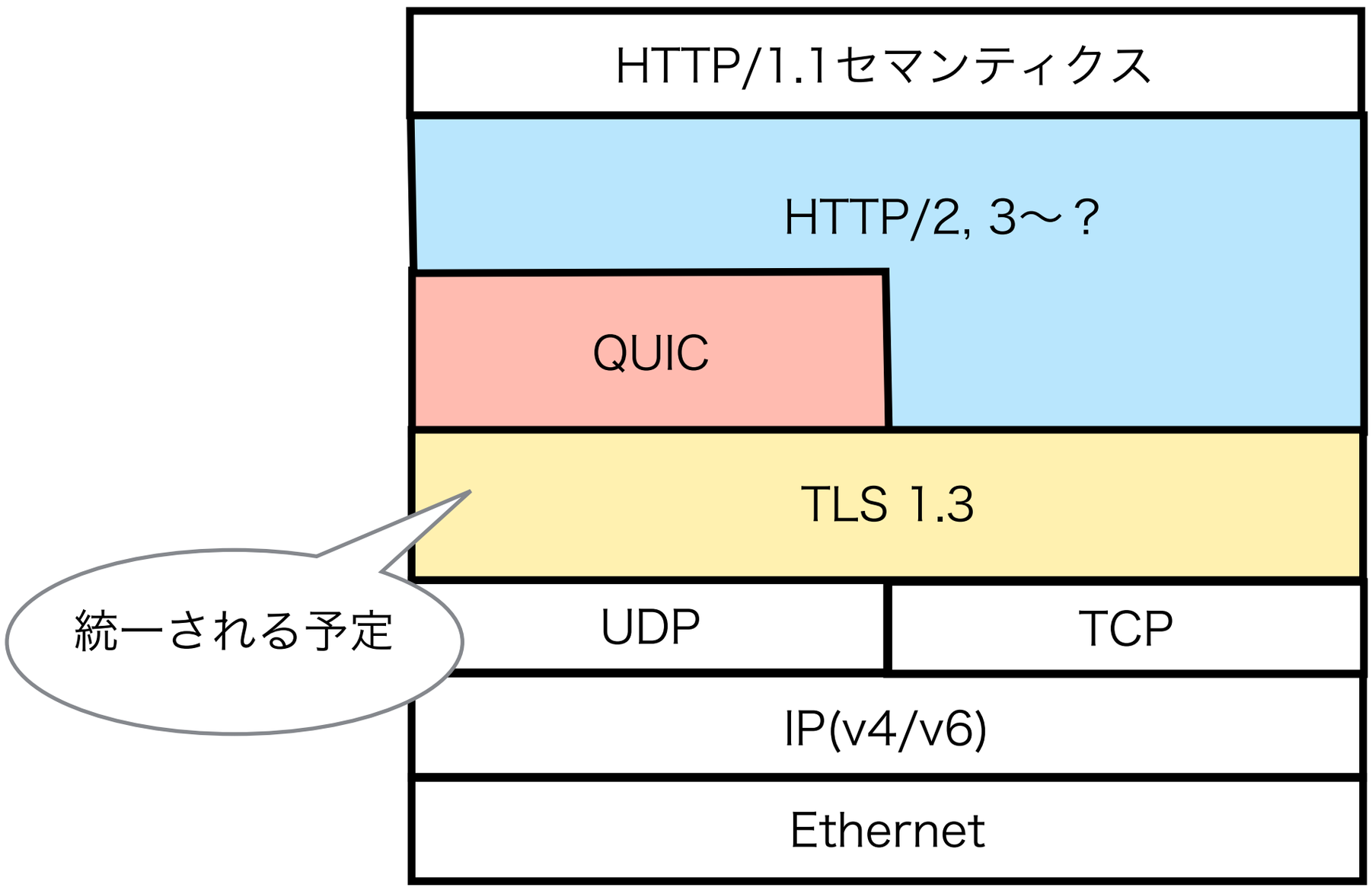


2013〜

(* HTTP/2 2015〜)

Webプロトコルの進化

QUICからTLS1.3へ



2017〜

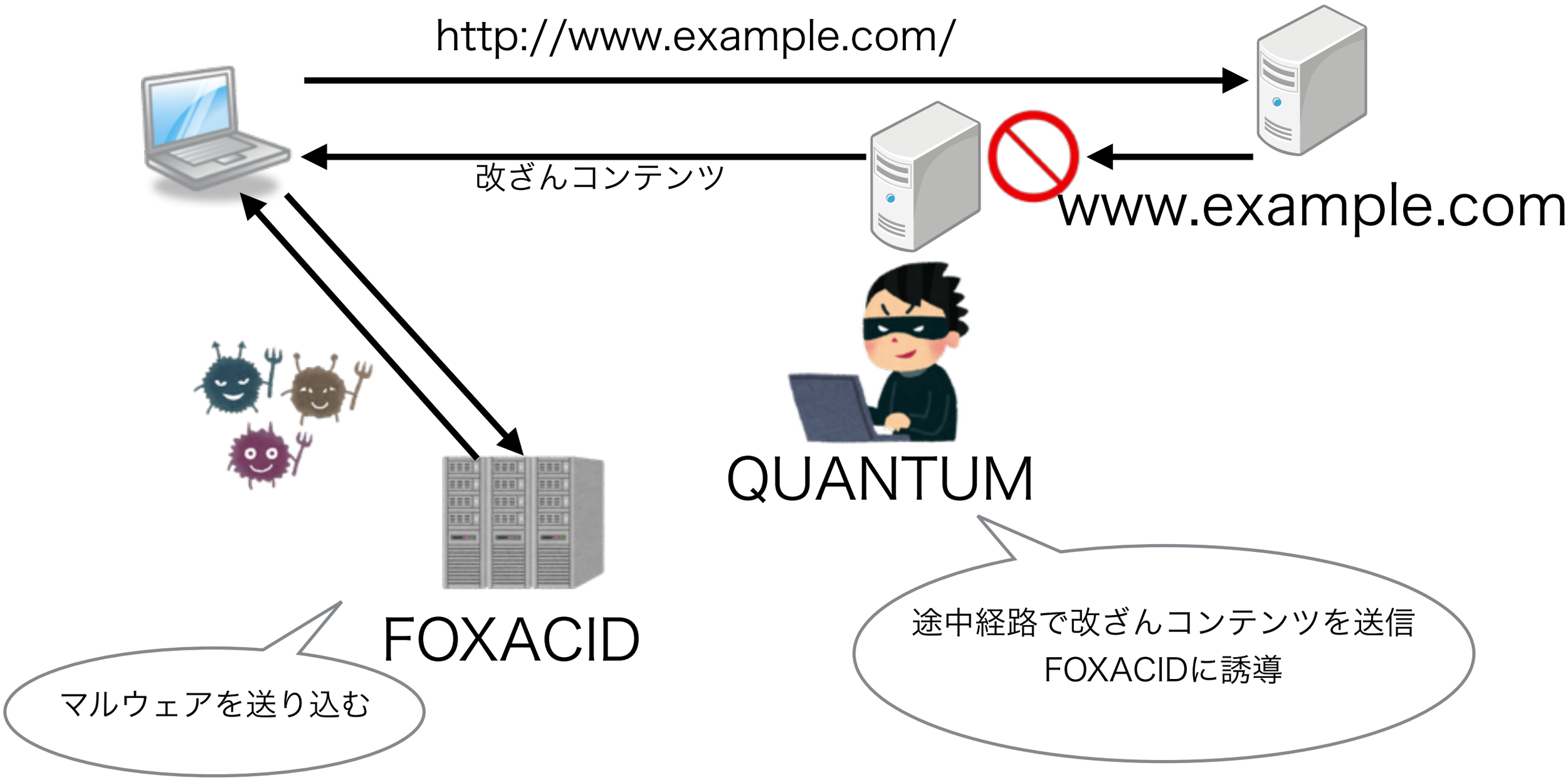
常時TLSへ至る道

Pervasive Surveillance
広範囲の盗聴行為

- 国家的な組織(米国NSAと英国GCHQなど)が莫大な予算で行う広範囲の盗聴行為
- 2013年6月 エドワード・スノーデンによってその活動内容がリークされる。

インターネット/電話の傍受・監視、データセンター内通信盗聴、暗号解読、暗号バックドア、サイバー攻撃 等

NSAによるサイバー攻撃の一例



https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html

プロトコル技術者の憂慮

- 従来大規模な設備と予算が必要で現実的には無理と見られてきた攻撃が実際に行われていた。

- 公衆無線LANの普及など通信の盗聴・改ざんが可能な環境が広がってきている。

- 幸い最新の技術でしっかり暗号化された通信まではまだ破られておらず、安全であろう。

検索サービス会社の憂慮

- 検索のページランクが高いサイト宛の平文通信は、攻撃対象として当然狙われる。
- 平文通信でユーザがコンテンツ改ざんやマルウェア感染によってDDoS攻撃の一端を担う恐れもあり(Githubへの攻撃例)。
- ネットコンテンツの健全性の低下は、長期的に検索サービスへの信頼性を損なうことになる。

IAB(*)によるインターネットの信頼性に関する宣言(2014/11)

- 新しくプロトコルを設計する際には、**暗号化機能を必須**とすべき。

- ネットワーク運用者やサービス提供者に**暗号化通信の導入を推進**するよう強く求める。

- コンテンツフィルターやIDS等平文通信が必要な機能については将来的に代替技術の開発に取り組む。

(* Internet Architecture Board)
https://www.iab.org/2014/11/14/iab-statement-on-internet-confidentiality/

Mozillaによる
安全でないHTTPの廃止宣言

1. ある時期から新規機能は、HTTPSだけ利用できるようにする。

2. 現在HTTP(平文通信)で利用できる機能で、ユーザのセキュリティやプライバシーにリスクを与えるものを削除していく

https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/

ChromeのHTTP上の機能廃止

Chromeでは、下記の機能をHTTP(平文通信)で利用禁止する予定

- 位置情報を取得 (廃止済)
- デバイスの動きや方向を操作
- 暗号化された動画音声の再生
- カメラ・マイクなどの操作
- アプリケーションのキャッシュ情報の操作

https://sites.google.com/a/chromium.org/dev/Home/chromium-security/deprecating-powerful-features-on-insecure-origins

HTTPはもはや

安全でないとユーザに通知

Chromium > Chromium Security >

Marking HTTP As Non-Secure

Proposal

We, the Chrome Security Team, propose that user agents (UAs) **gradually change their UX to display non-secure origins as affirmatively non-secure.**

The goal of this proposal is to more clearly display to users that HTTP provides no data security.

	Chrome 51	Chrome 52
Secure HTTPS	https://www.google.com	https://www.google.com
HTTP	www.example.com	www.example.com
Broken HTTPS	~~https~~://expired.badssl.com	~~https~~://expired.badssl.com

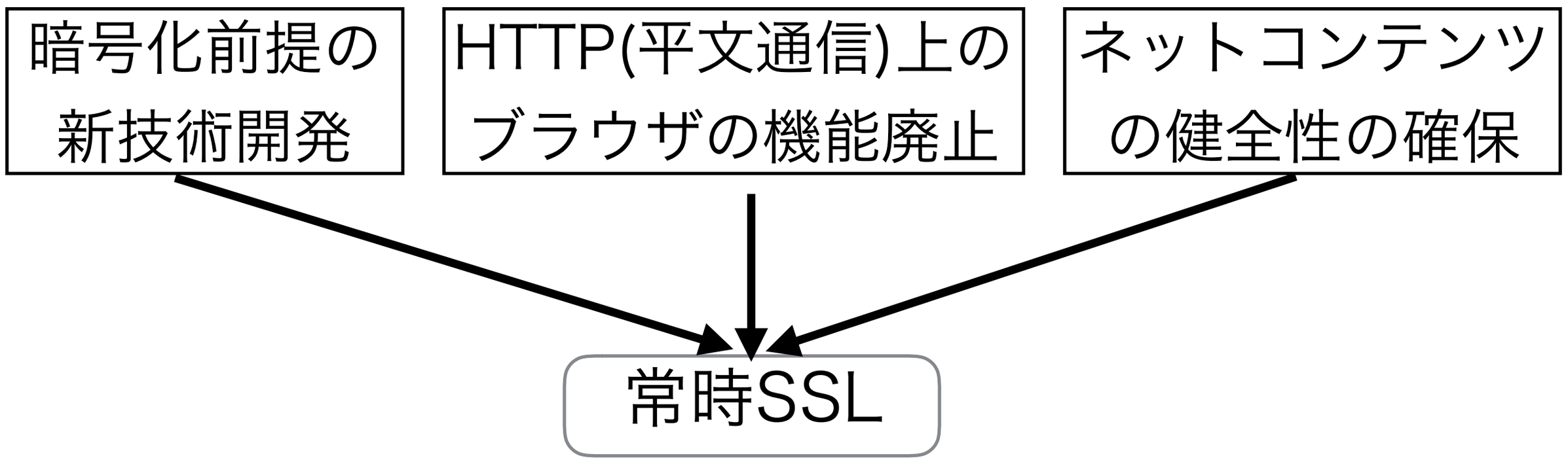
ⓘ HTTPS with minor errors uses the same indicator as HTTP.

徐々に非安全を強調

https://docs.google.com/document/d/1jlfCjcsZUL6ouLgPOsMORGcTTXc7OTwkcBQCkZvDyGE/pub

常時SSLへ至る道

国家レベルの広範囲な盗聴行為



常時TLSとHTTP/2

HTTPの年表

1990 1995 2000 2005 2010 2015

Webの始まり

HTTP/0.9

HTTP/1.0
RFC1945

HTTP/1.1
RFC2068

HTTP/1.1
RFC2616

HTTP-NG
中止

暗黒の時代

httpbis WG

HTTP/1.1
RFC7230-5

SPDY/2

SPDY/3

SPDY/3.1

HTTP/2
RFC7540

HPACK
RFC7541

HTTP/2サポート状況

HTTP/2 protocol - OTHER

Japan	66.46% + 19.7% =	86.16%
Global	63.07% + 6.8% =	69.87%

Networking protocol for low-latency transport of content over the web. Originally started out from the SPDY protocol, now standardized as HTTP version 2.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
9		45	49	9	Safari 9.1 Partial support Browser usage Global: 1.32% Japan: 2.33% Notes: 2 Only supports HTTP2 over TLS (https) 3 Partial support in Safari refers to being limited to OSX 10.11+			4.3	
		46	50					4.4	
11	13	47	51	9.1			8	4.4.4	50
	14	48	52	10				50	
		49	53	TP					
		50	54						

Notes | Known issues (0) | Resources (6) | Feedback

See also support for the SPDY protocol, precursor of HTTP2.

1 Partial support in IE11 refers to being limited to Windows 10.

2 Only supports HTTP2 over TLS (https)

3 Partial support in Safari refers to being limited to OSX 10.11+

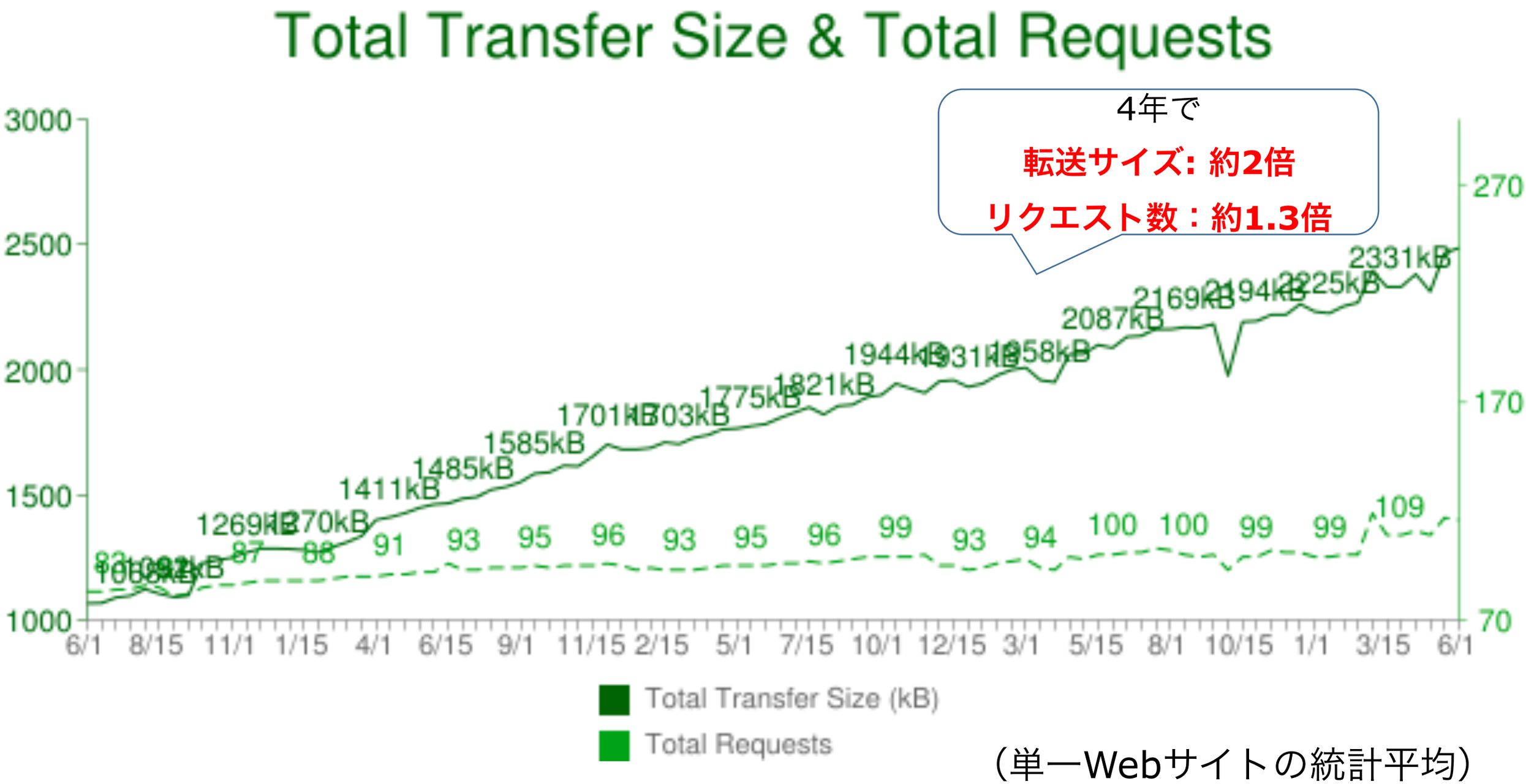
4 Only supports HTTP2 if servers support protocol negotiation via ALPN

ほとんどがTLSのみサポート

http://caniuse.com/#feat=http2

HTTP転送サイズとリクエスト数の遷移

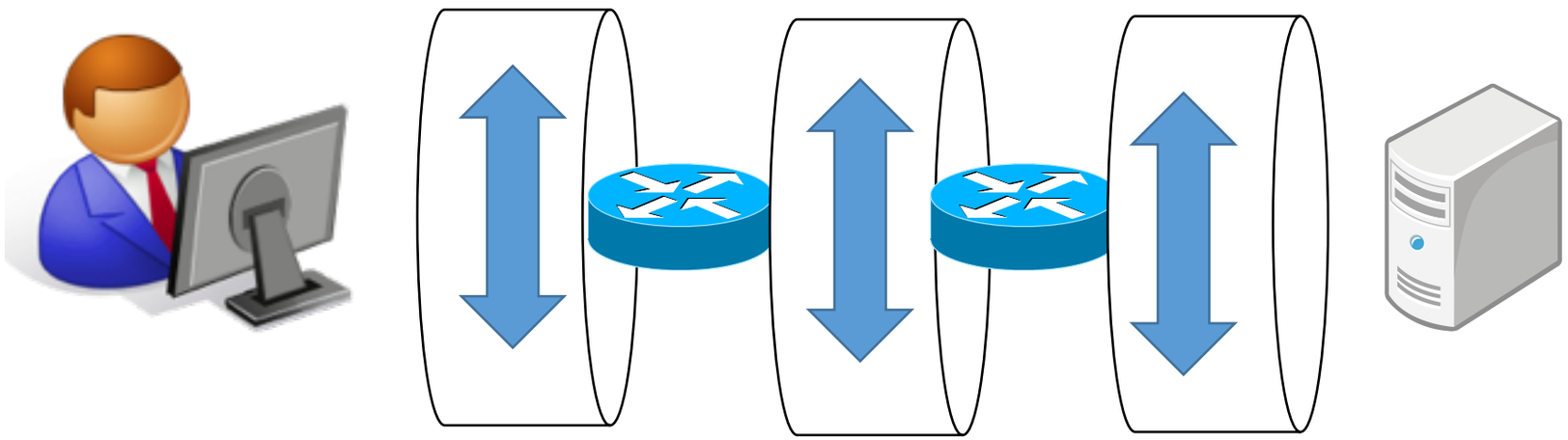
(2012/6/1～2016/6/1)

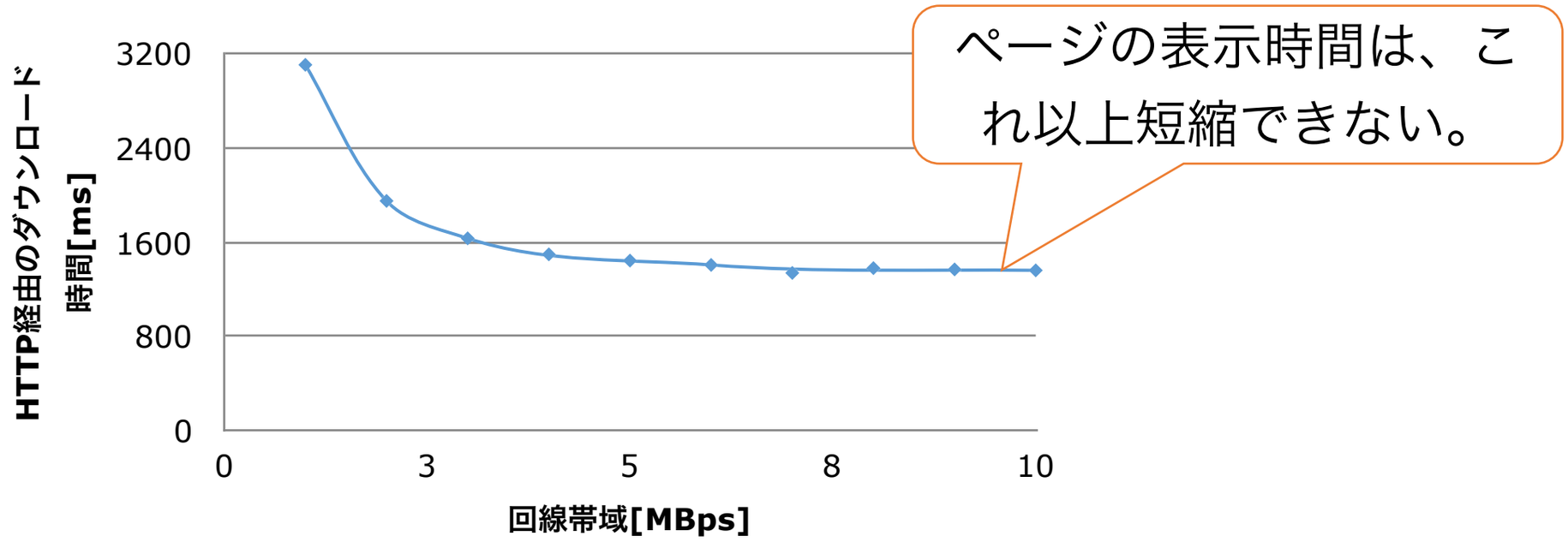


（単一Webサイトの統計平均）

http://httparchive.org/trends.php?s=All&minlabel=Jun+1+2012&maxlabel=Jun+1+2016#bytesTotal&reqTotal

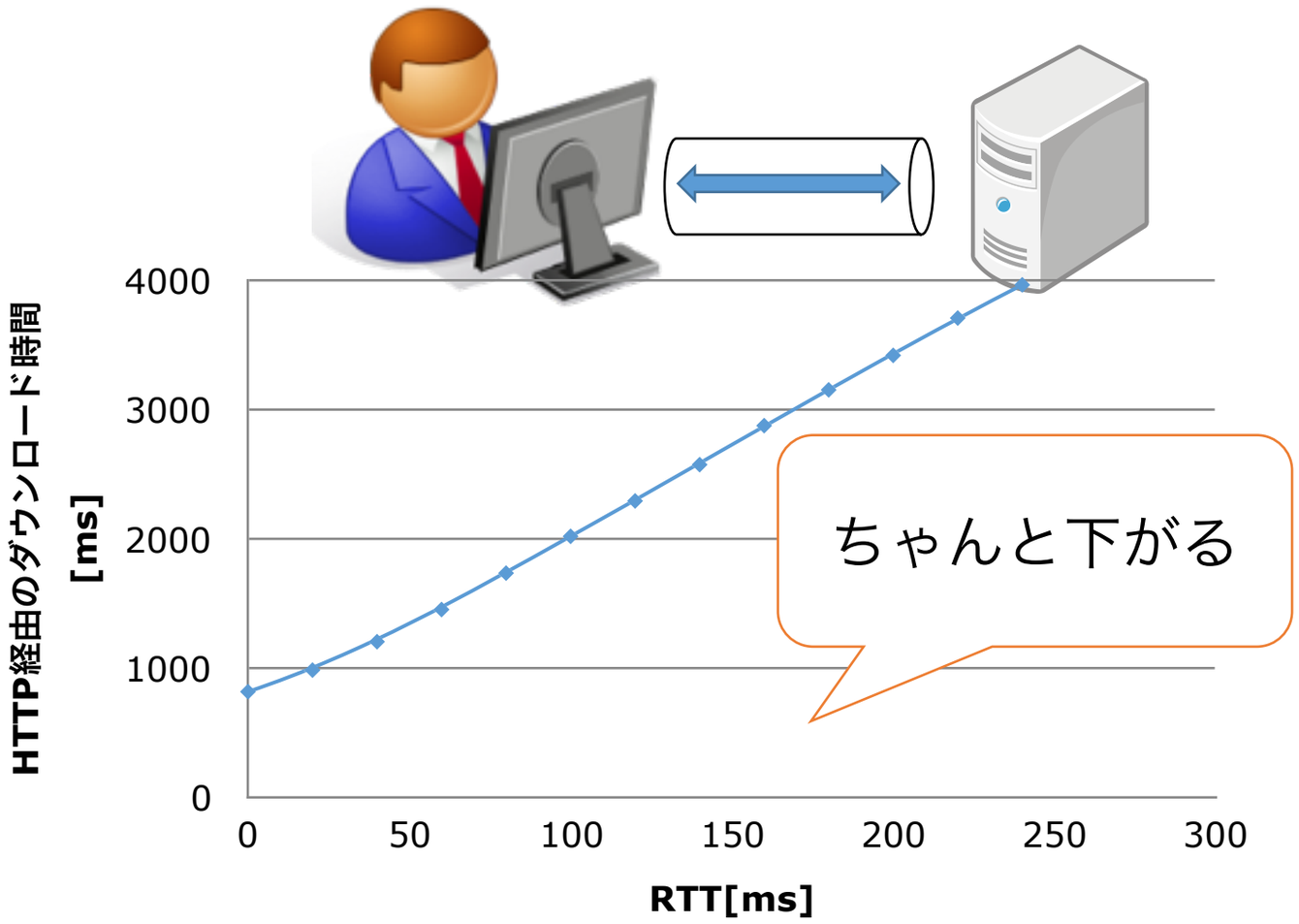
回線帯域を増速していくと





More Bandwidth does’nt matter よりデータ引用http://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

RTT(Round Trip Time)を小さくしていくと



Webページの表示速度を速くするには、回線速度増強よりRTTの影響を小さくするかが重要。でも物理的な制限で難しい。

HTTP/2はRTTの影響を受けるボトルネックの解消を狙ったもの。

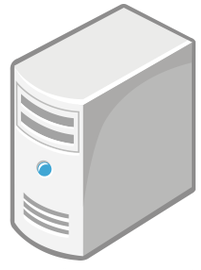
More Bandwidth does’nt matter よりデータ引用http://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2

RTTの影響を受けるボトルネック

HTTP Head of Line Blocking

クライアント

HTTP/1.1
1本のTCP接続



サーバ

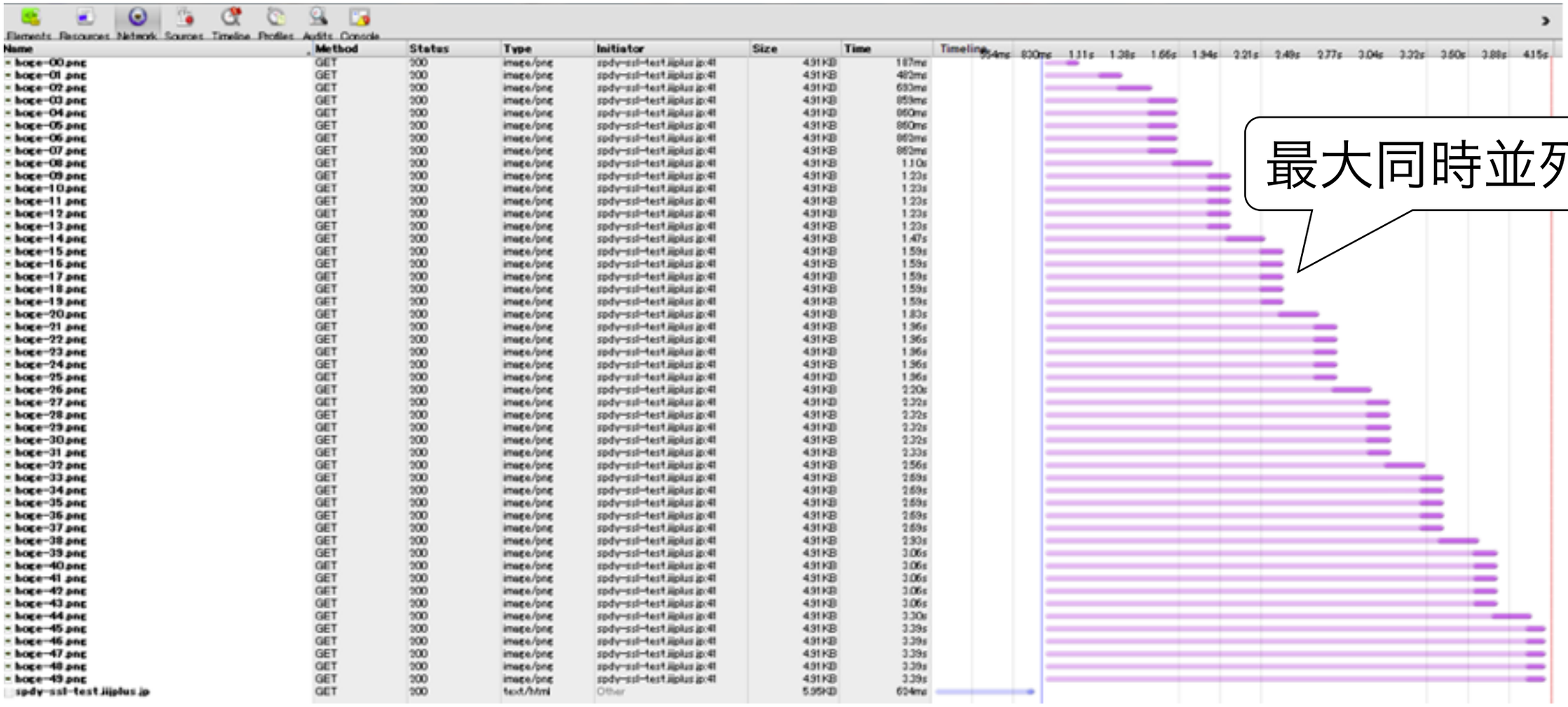
待ち時間

HTTP リクエスト

HTTPレスポンス

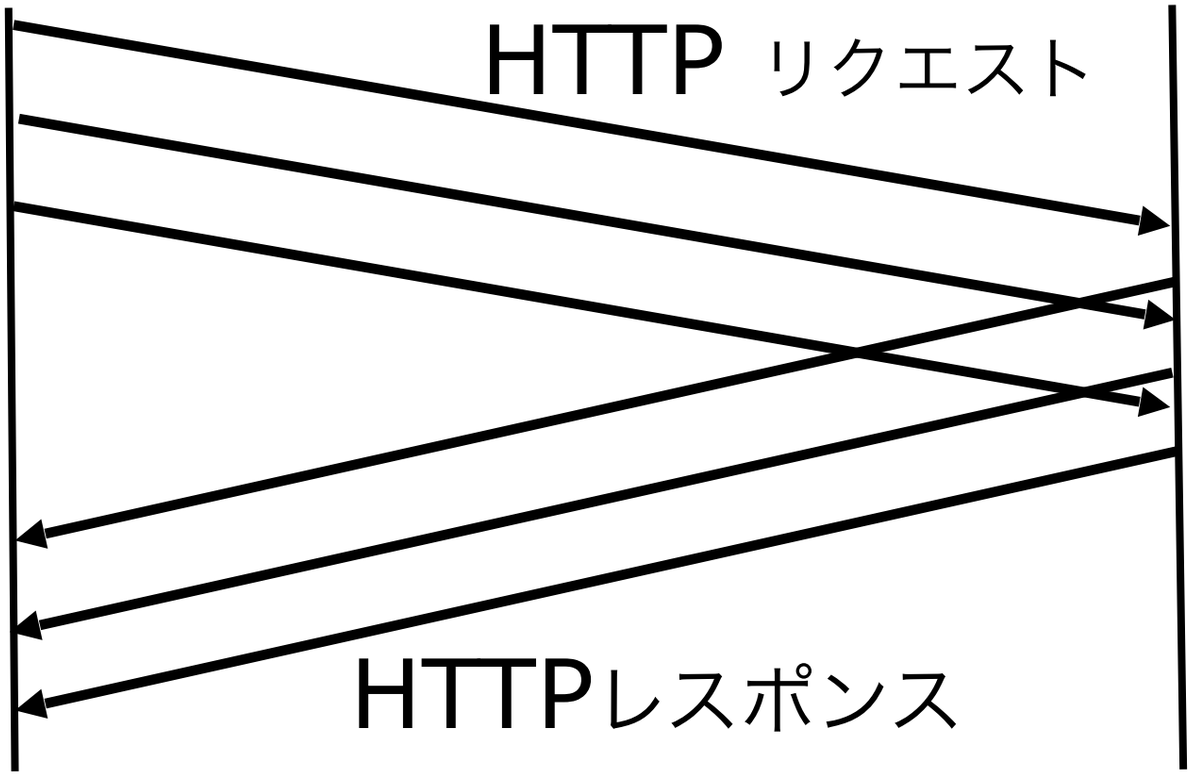
HTTP リクエスト

HTTP/1.1
ブラウザは最大同時4～6TCP接続に制限



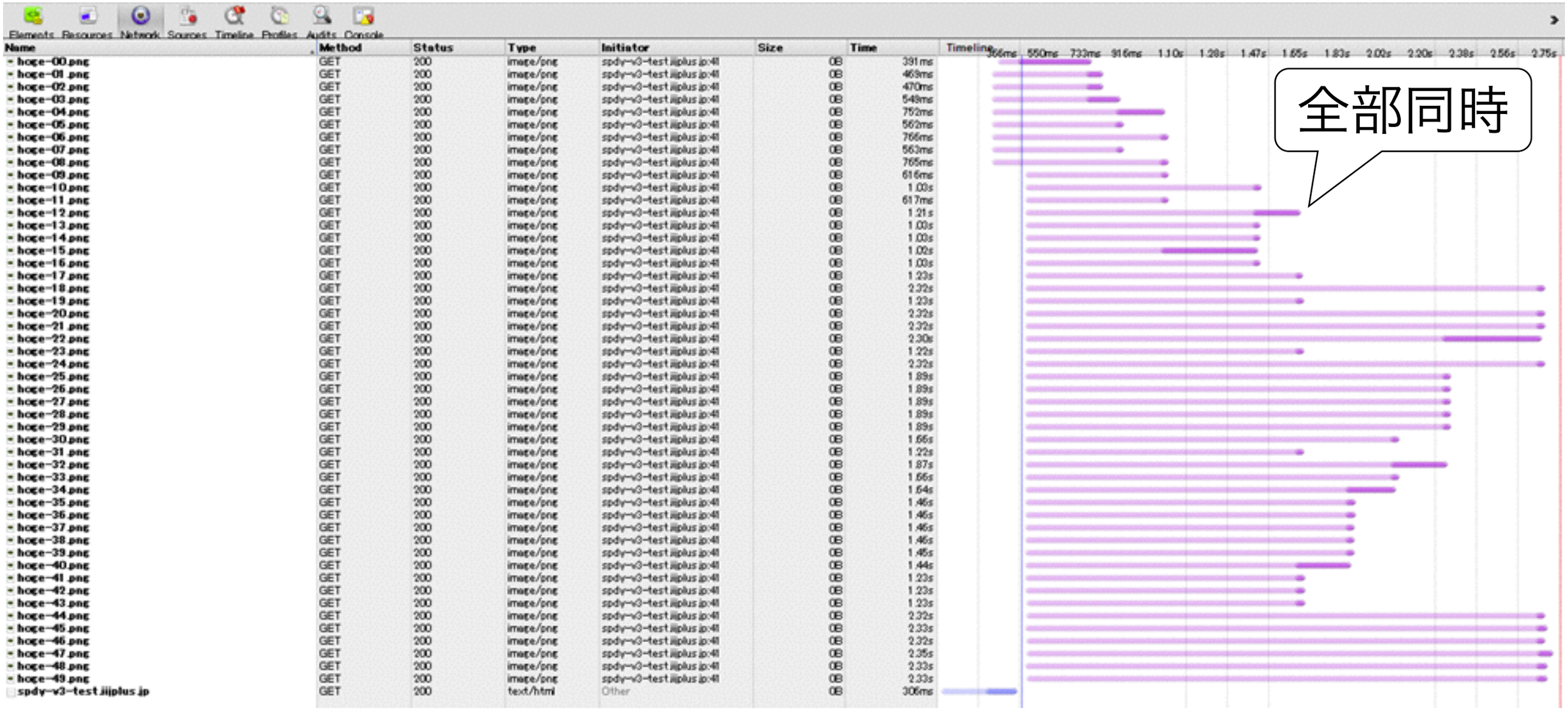
HTTP/2はHTTP Head of Line Blockingを解消

待ちが他のリクエストに引きずられない



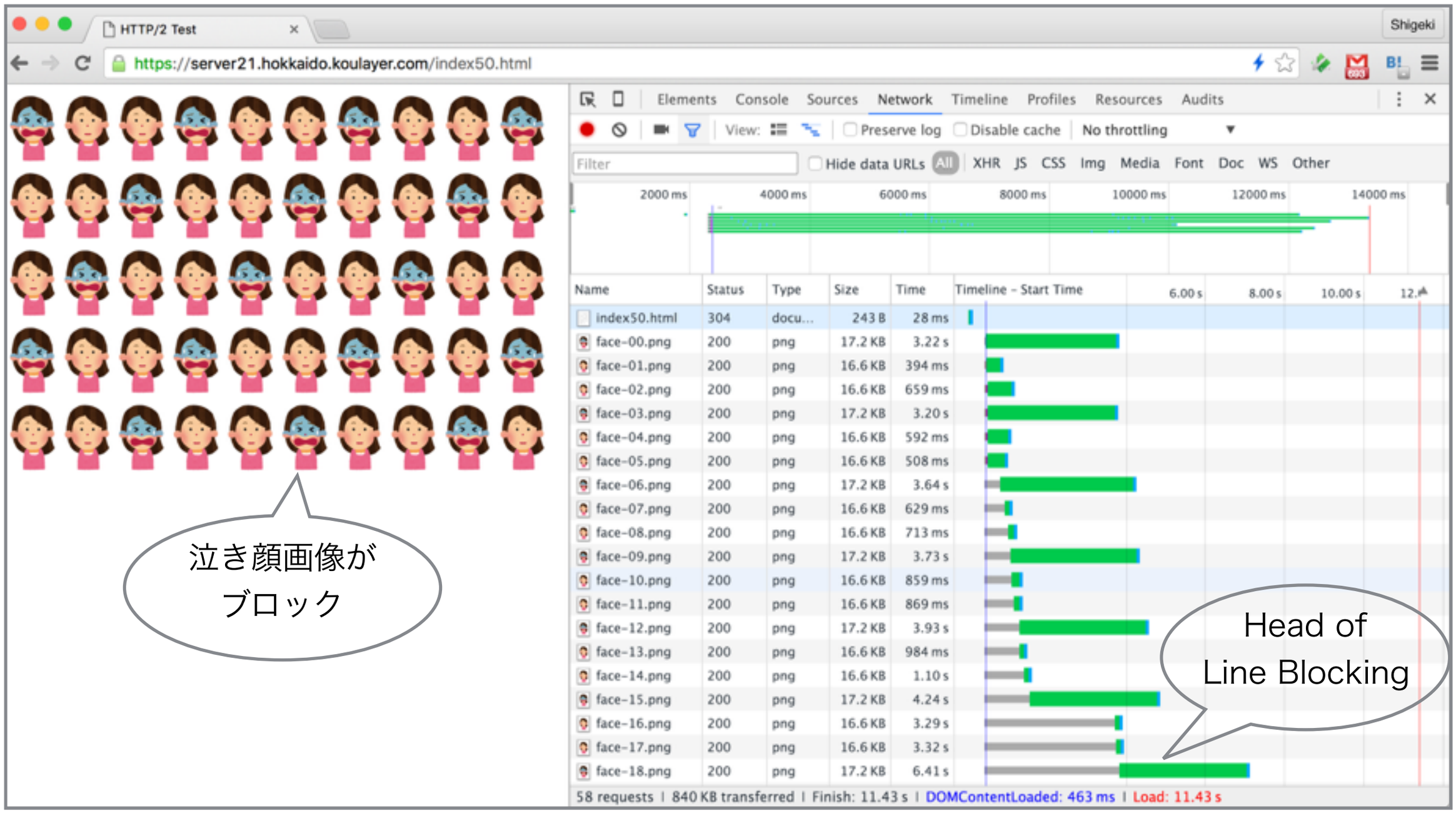
仮想的に多重化

HTTP/2
100以上の同時リクエストが可能

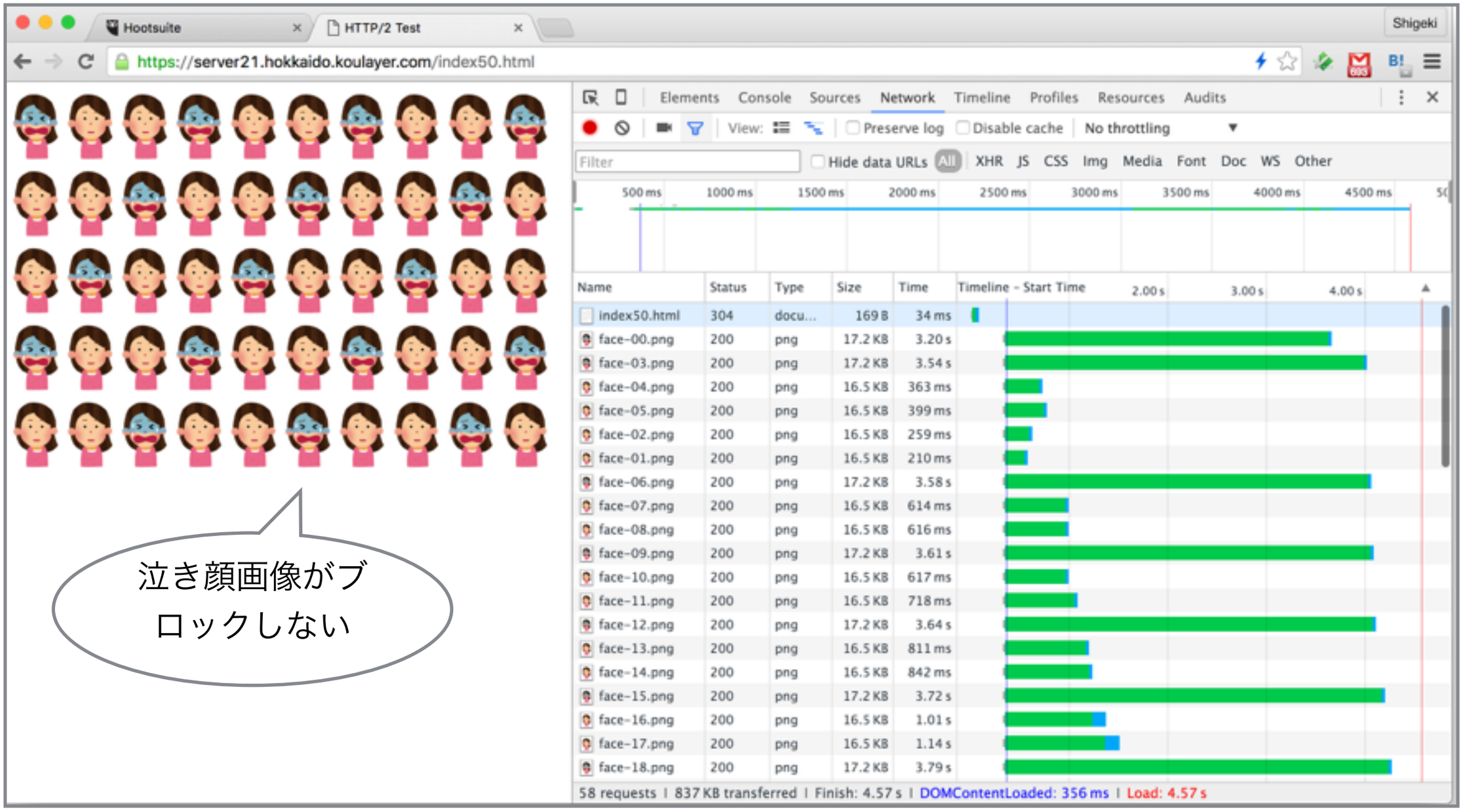


Name	Method	Status	Type	Initiator	Size	Time
hoge-00.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	391ms
hoge-01.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	469ms
hoge-02.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	470ms
hoge-03.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	549ms
hoge-04.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	752ms
hoge-05.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	562ms
hoge-06.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	766ms
hoge-07.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	563ms
hoge-08.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	765ms
hoge-09.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	616ms
hoge-10.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.00s
hoge-11.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	617ms
hoge-12.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.21s
hoge-13.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.00s
hoge-14.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.00s
hoge-15.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.02s
hoge-16.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.00s
hoge-17.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.23s
hoge-18.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-19.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.23s
hoge-20.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-21.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-22.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.30s
hoge-23.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.22s
hoge-24.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-25.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.89s
hoge-26.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.89s
hoge-27.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.89s
hoge-28.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.89s
hoge-29.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.89s
hoge-30.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.66s
hoge-31.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.22s
hoge-32.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.87s
hoge-33.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.66s
hoge-34.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.64s
hoge-35.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.46s
hoge-36.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.46s
hoge-37.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.46s
hoge-38.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.46s
hoge-39.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.45s
hoge-40.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.44s
hoge-41.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.23s
hoge-42.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.23s
hoge-43.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	1.23s
hoge-44.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-45.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.33s
hoge-46.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.32s
hoge-47.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.35s
hoge-48.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.33s
hoge-49.png	GET	200	image/png	spdy-v3-test.iijplus.jp:443	0B	2.33s
spdy-v3-test.iijplus.jp	GET	200	text/html	Other	0B	306ms

HTTP/1.1の表示デモ



HTTP/2の表示デモ



見え方の違いを比較

HTTP/2に関する大きな誤解

- HTTP/2は**銀の弾丸ではありません**。
- 多くのサイトは、HTTP/2にしたら単純に速くなる・遅くなるというものではない。HTTP Head of Line Blockingに困っているサイトが速くなります。
- 今まで使ってきたHTTP/1.1はなくなりません。半永久的にサポートされるでしょう。
- HTTP/2はわかりません。今後のバージョンアップで廃止される可能性もあります。

HTTP/2とTLSの関係

時期が悪かった

TLSのみ

2009 GoogleがSPDYプロトコルを開発・公開

平文とTLSで仕様策定開始

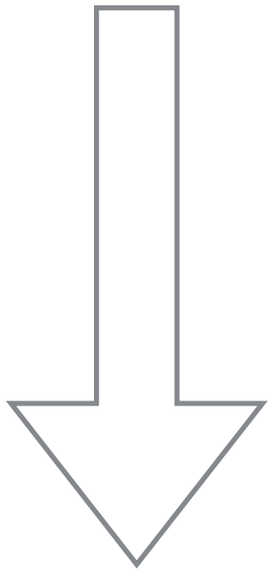
2012 HTTP/2仕様化開始

2013 **6月 エドワード・スノーデン事件**

常時SSLへの動きが本格化

ブラウザはTLS
のみサポート

2015 HTTP/2仕様化完了(RFC7540)



これからどう変わっていくのか？

TLS1.3とQUIC

TLSの簡単な歴史

	SSL 1.0(未発表)
1994年	SSL 2.0 DROWN
1995年	SSL 3.0 POODLE
1996年	IETF TLS WG スタート
1999年	TLS 1.0 BEAST
2006年	TLS 1.1
2008年	TLS 1.2
2013年	TLS 1.3検討スタート
2016年	TLS 1.3仕様化完了？

SSLは旧ネットスケープ社の私的プロトコル

大人の事情で名称変更

SSL3.0と基本設計は大きく変えず改良

様々な機能拡張

近日WGラストコールを目標
まだわかりません

TLS1.3が求められる背景

1. 常時SSL時代を迎えるにあたって、しっかりしたプロトコルが必要

2. TLS1.2の限界

    - 様々な技術負債の蓄積

長期に使えるより安全で高性能なTLSプロトコルを作る

TLS1.2の限界

- 現在のTLS1.2で定義されている機能の一部は、既に利用すると危険である。

- 過去様々なTLSの攻撃手法や脆弱性が公開され、その都度対策が取られてきた。

- しかし一時的な対応で根本的・抜本的な対策になっていないものも多い。

本来はなくて済むのが望ましい

TLS1.3の特徴

1. 様々な機能、項目の見直し・廃止

   時代に合わなくなったもの、より効率的に変更修正できるものをTLS1.2から機能・項目を数多く廃止

2. よりセキュアに

   平文通信が必要な部分を極力少なくして情報を秘匿

   これまで攻撃対象となった機能を極力排除し将来的な攻撃に備える

3. 性能向上

   初期接続の短縮による性能向上

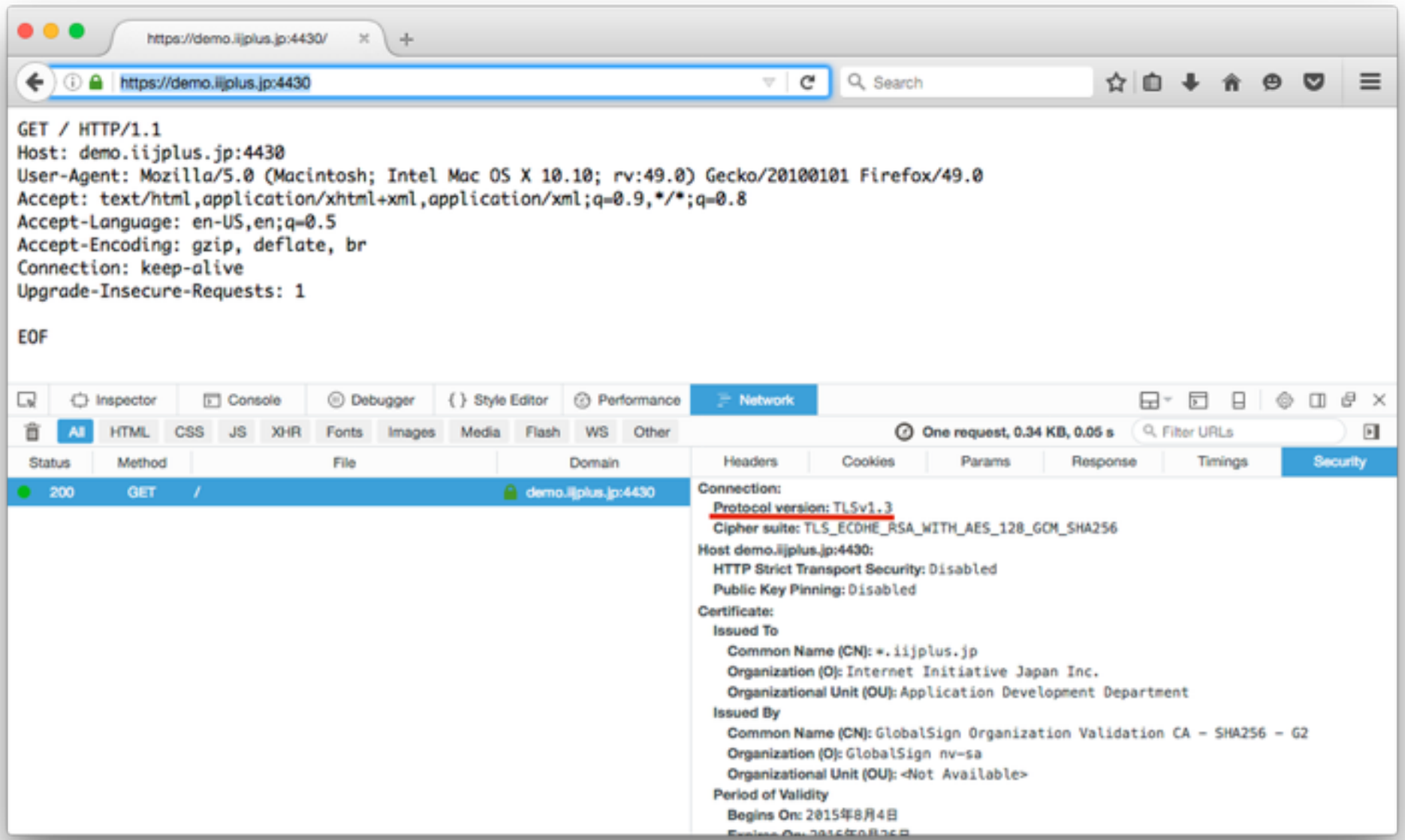
中身的にはTLS2.0レベルの大変更

TLS1.3絶賛開発中

Mozilla: 試験実装をFirefoxに導入済み

Google: 試験実装をChromeに導入済み

Microsoft: Edgeでサポートを表明



TLS1.3でどうなるか？

ユーザ： 全く気づかない(少し早くなったぐらい)

サーバ、ネットワーク管理者：

- 既存のサーバ・ネットワークとの互換性に注意
  (TLS1.3の接続を切断する恐れがある。TLS1.3未対応の場合はTLS1.2を利用することが必要)

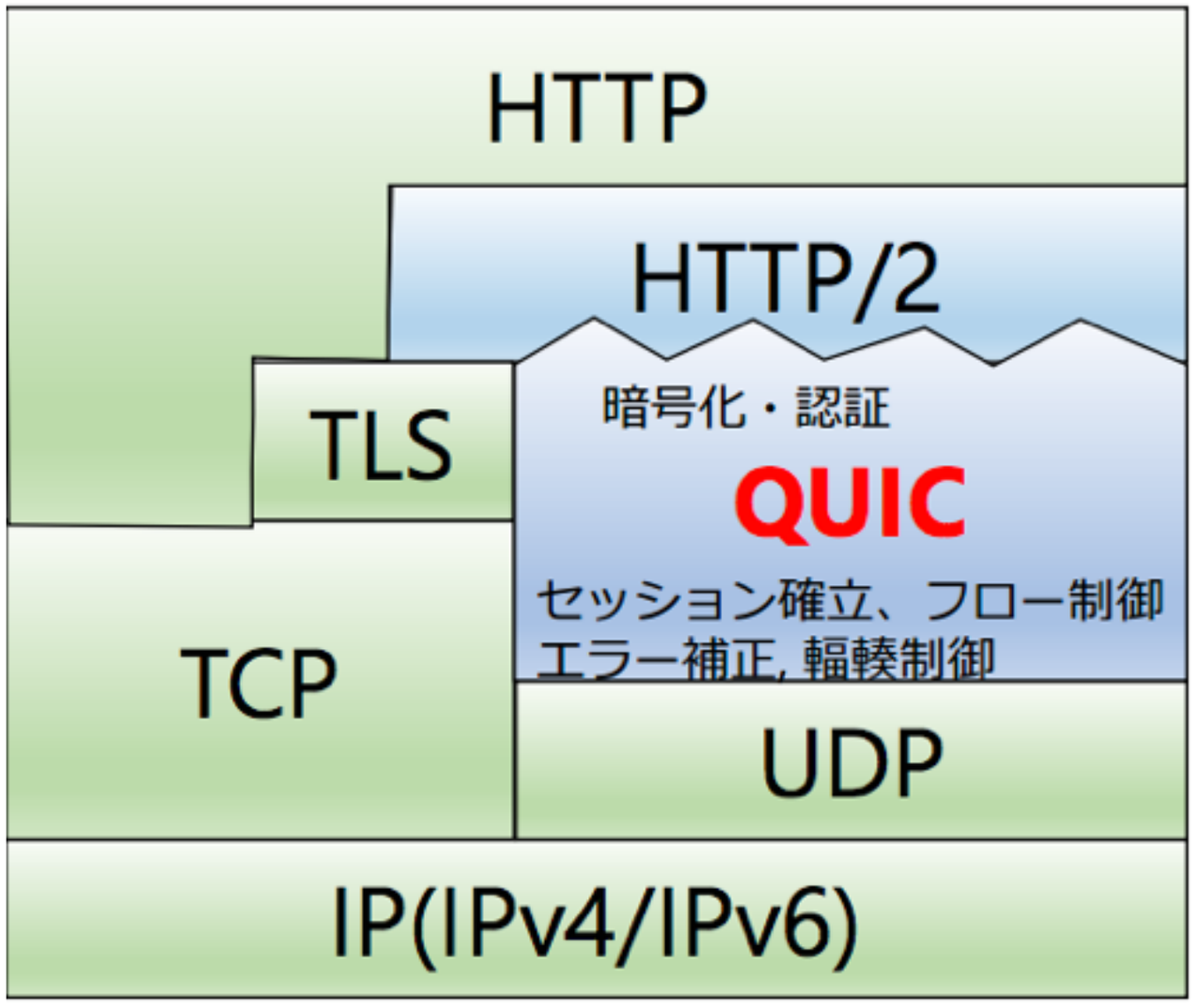
- いつTLS1.3を導入するか? 将来の新しい技術プロトコルはTLS1.3上で実現される見込み。

QUIC

Quick **U**DP **I**nternet **C**onnection

Quick UDP Internet Connection

- Googleが独自に開発・運用している新しいプロトコル
- UDP上で動作
- TCP/TLS相当の機能を実装
- 既に全サービスで運用中。
  Android/デスクトップ
  Chromeで利用中。



QUICによる性能向上

- 92%でQUICが利用できている(デスクトップ+モバイル環境)
- 平均5%のページ読み込み時間の短縮
- 速度下位1%の接続で1ÿ秒の短縮 (接続環境が悪い程効果が高い)
- 最適化しているGoogle検索ページでも平均3%の読み込み時間の短縮
- Youtubeで rebuffer を30%短縮(ビデオ停止時間が短い)
- 75%が 0-RTTの恩恵を受けている(QUICによる性能向上効果の半分が0-RTT)

Google DUO

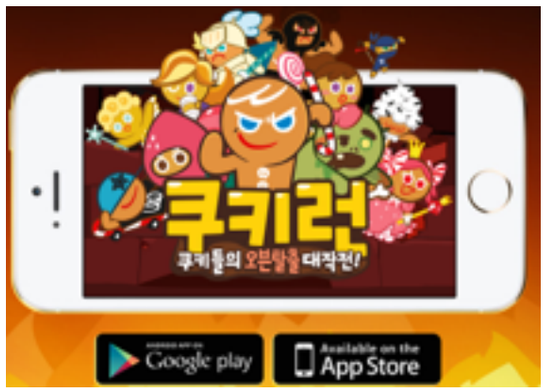
WebRTC over QUIC?

http://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html

https://docs.google.com/presentation/d/15e1bLKYeN56GL1oTJSF9OZiUsl-rcxisLo9dEyDkWQs/

QUICのメリット

初期接続

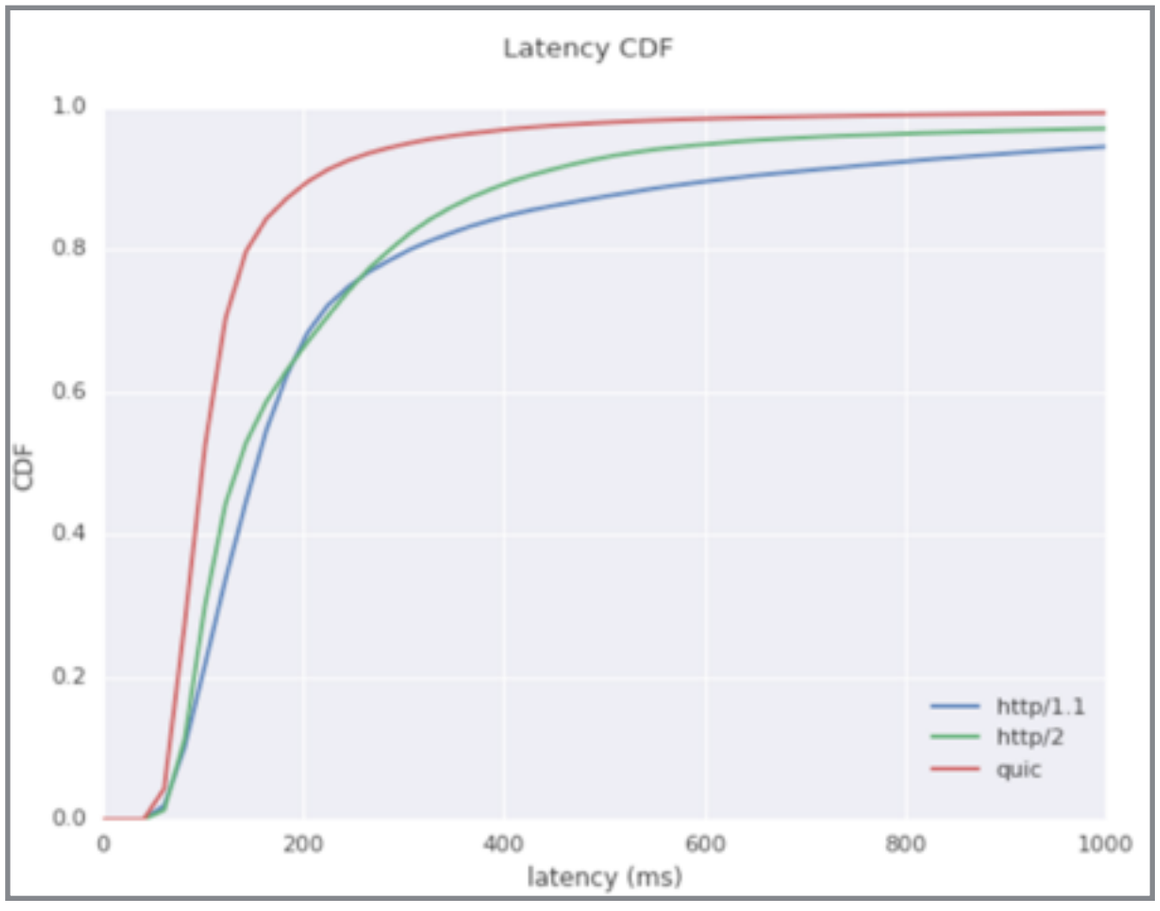


韓国devsistersのcookierunゲーム

- UDP上で動作するのでハンドシェイクは必要ない

- QUIC独自のセッション管理

- 以前の接続時に交換した鍵を使いいきなり暗号データを送信。0-RTTを実現。

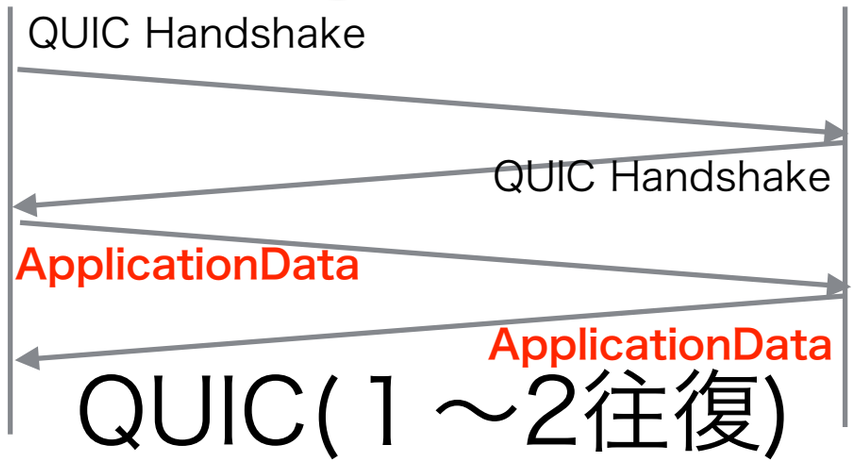


平均値：http/1.1(1.43s), h2(0.64s), quic(0.41s)

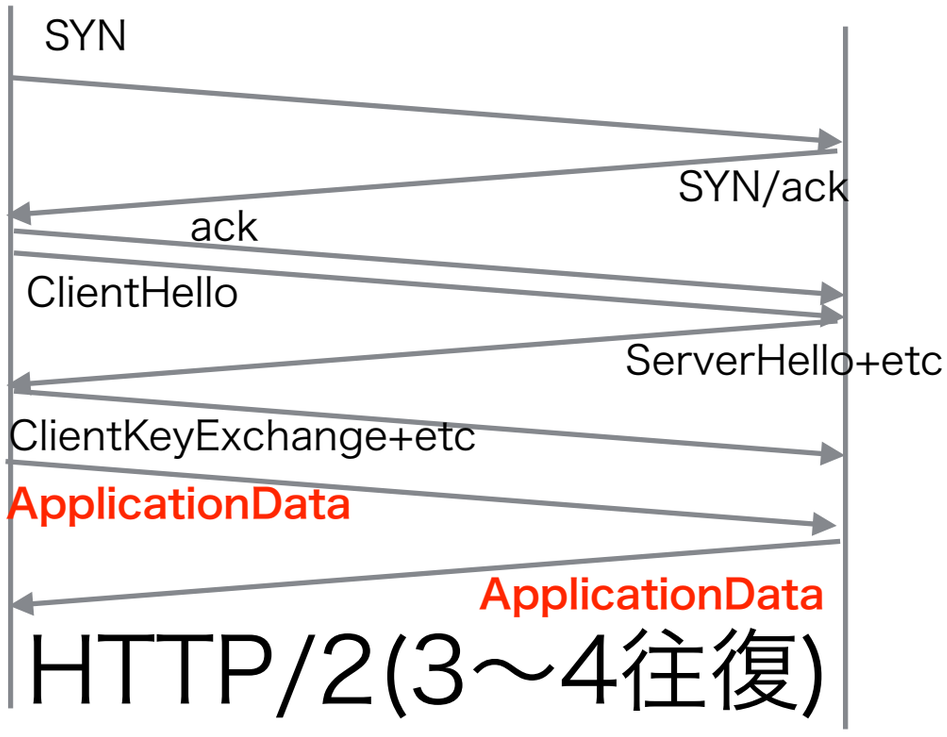
https://groups.google.com/a/chromium.org/d/msg/proto-quic/lewtuSYH1vs/anySxbHSCgAJ

QUICデモ

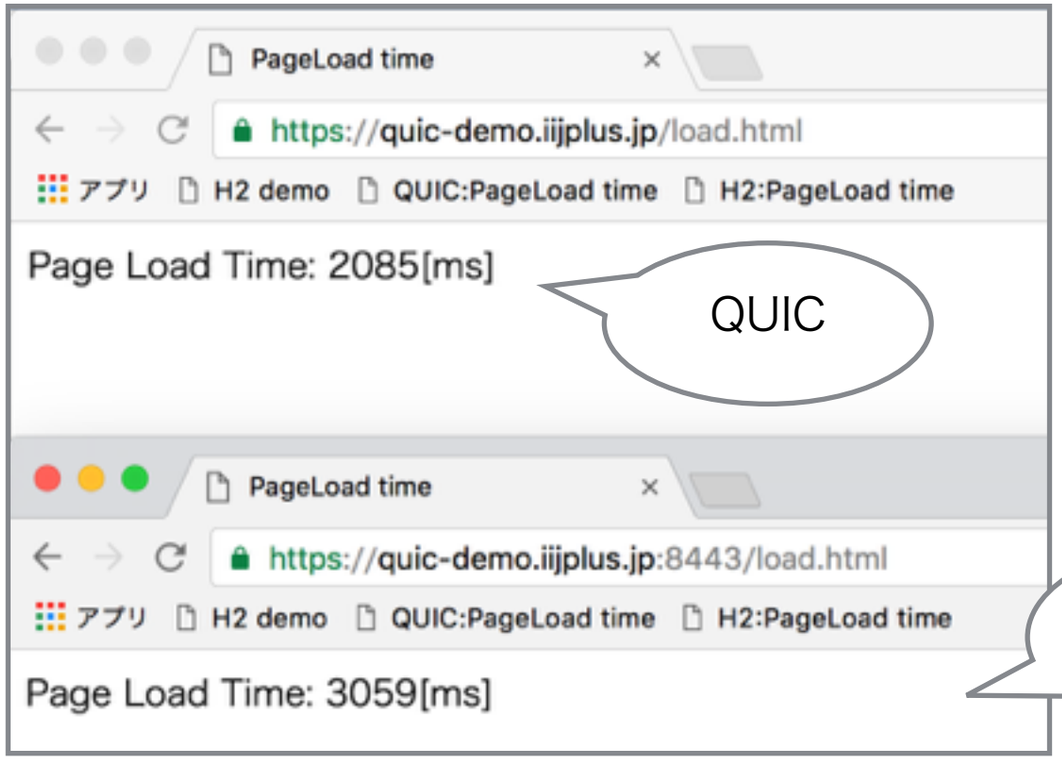
PageLoadTimeの比較



QUIC(1～2往復)



HTTP/2(3～4往復)



QUICのメリット
パケットロスに強い

回線輻輳

サーバ

落ちたのを再送

パケット3

パケット5

パケット4

順番を気にせず待たされない

パケット2

パケット1

パケット3

ドロップ

QUICデモ

- 人為的にパケットロスを発生させたネットワーク環境で HTTP/2 と QUIC で見え方がどう違うか