HTTP/2.0のインパクト

(株)レピダム
清水 一貴
(@kazubu)

HTTP/2.0って?

HTTP/2.0って?

HTTP/1.1の高効率版

これまで

- 2012/03にフランスで開催されたIETF83にて、HTTP/2.0策定への取り組みスタート
- GoogleのSPDYを参考にして一から作るイメージ
- その後3回のIETFミーティング、2回のInterimミーティングを経て今に至る

なぜHTTP/2.0を作るのか

- Webの普及に伴うトラフィック等の増加
- モバイル端末の普及・増加に伴う、高レイテンシネットワーク環境への最適化
  - TCPコネクションを沢山張るのはコストが高い！
- 一人当たりで使えるTCPポート数の減少

これらを少しでも解決するのが目的

HTTP/2.0のデザイン

パフォーマンス改善を目的とし、
HTTP/1.1とセマンティクスは変更しない

なので

- HTTP/1.1とHTTP/2.0は相互に変換できる
- GET, POST, PUT, … 等のメソッドに変更はない

具体的に何が変わるのか？

- ヘッダーのバイナリ化
- ヘッダーの圧縮
- 多重化(Multiplexing)
- 優先制御(Prioritizing)
- TCPコネクションの利用方針
- HTTP通信の開始方法
- など

具体的に何が変わるのか？

- ヘッダーのバイナリ化
- ヘッダーの圧縮
- 多重化(Multiplexing)
- 優先制御(Prioritizing)
- TCPコネクションの利用方針
- HTTP通信の開始方法
- など

ヘッダーのバイナリ化

- 今までのヘッダーの例(リクエスト)

  GET / HTTP/1.1¥r¥n
  Host: example.com¥r¥n
  Accept-Encoding: gzip, deflate¥r¥n
  User-Agent: Mozilla/4.0 (.....) ¥r¥n
  Cookie: hogehogehogehoge¥r¥n
  Connection: Keep-Alive¥r¥n

本当に必要なデータ(リクエスト)

{:method=> :get,

:path => “/”,

:scheme => “http”,

:host => “example.com”,

:accept-encoding => [:gzip, :deflate],

:user-agent => “Mozilla/4.0 (.....)”,

:cookie => “hogehogehogehoge”,

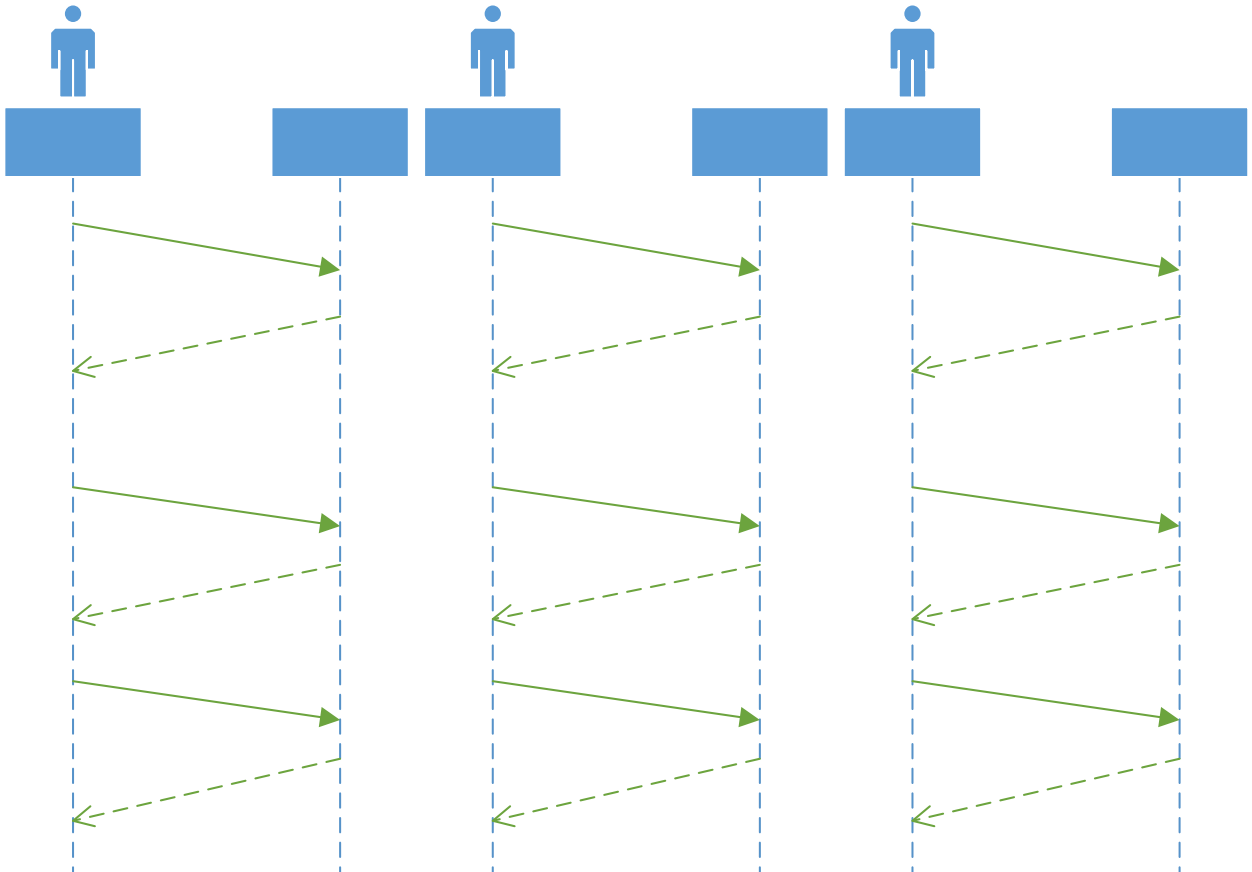
:connection => :keep-alive}

多重化(HTTP/1.1)

- 1つのTCPコネクション内では同時に複数のリクエストを行えない
- 同時に複数のTCPコネクションを利用する

HTTP/1.1

- 1接続につき1リクエスト、帰ってくるまで待つ

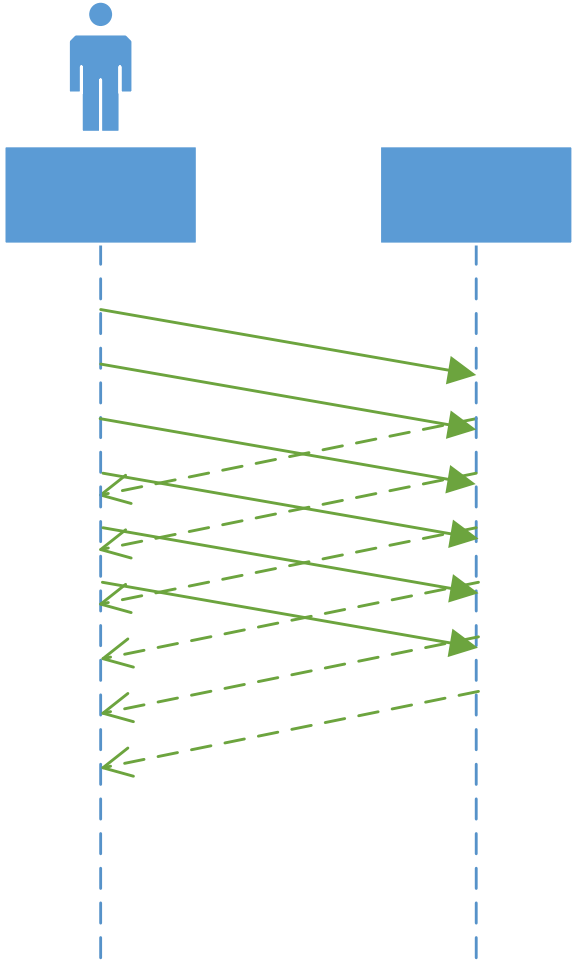


多重化(HTTP/2.0)

- TCPコネクションの内部に複数チャネルを作成
- それにより、1つのTCPコネクションで同時に複数のリクエストが可能
- レスポンスはリクエストの順序に依存しない

HTTP/2.0

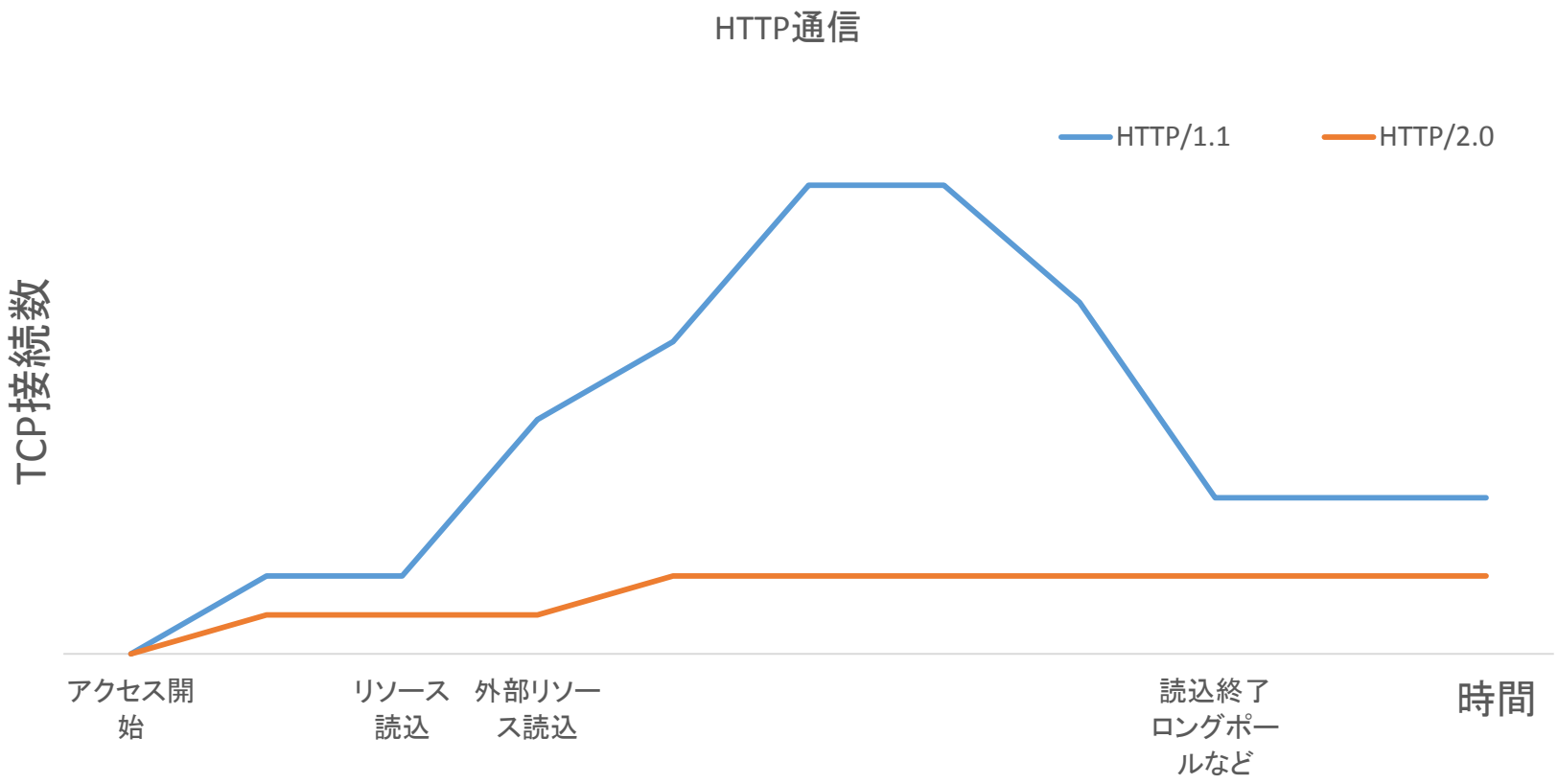
- 順番は入れ替わっても問題ない



TCPコネクションの利用方針

- HTTP/1.1
  - FQDNごとに2～6のTCP接続
  - コンテンツ受信が完了すると切断(実装依存)
    - 必要に応じて、JavaScriptによるLong-polling, WebSocket等
- HTTP/2.0
  - 多重化により、FQDNごとに1つ程度のTCP接続
  - コンテンツ受信が完了しても切断しない(推奨)
    - タブのクローズ、ページ遷移等によって切断

TCPコネクション数予想(理想?)



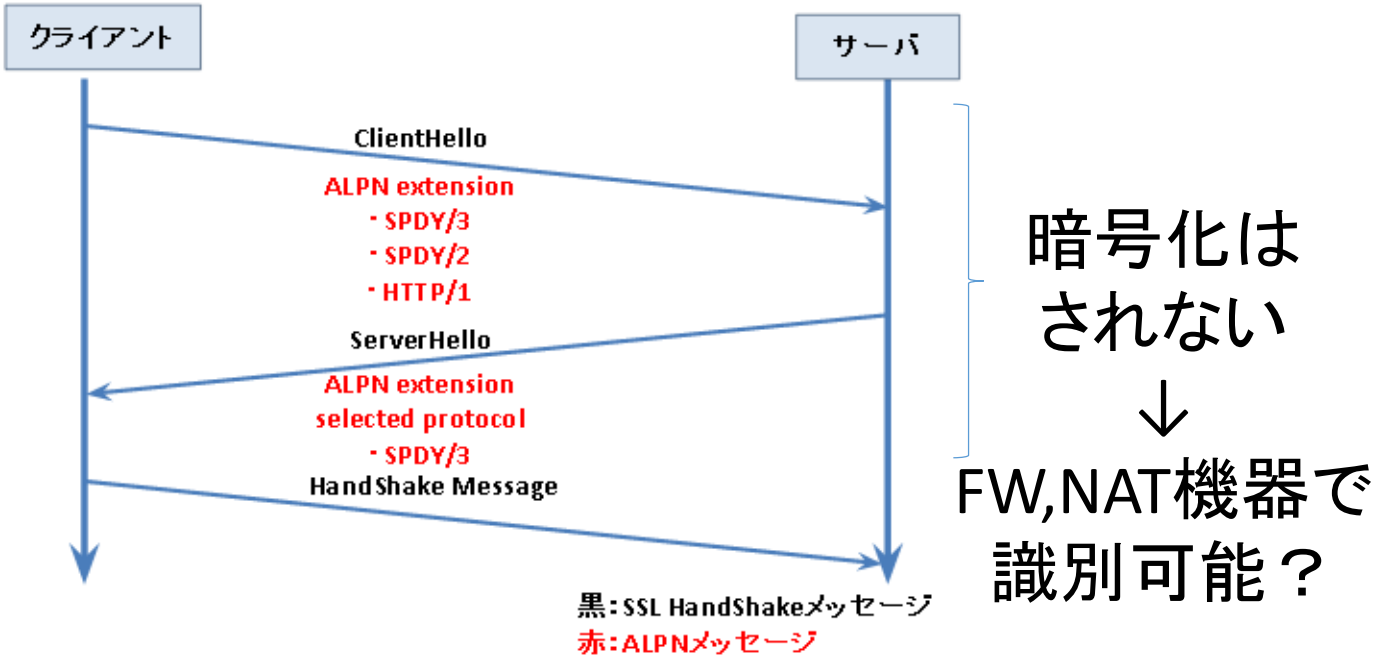
HTTP通信の開始方法

- HTTP/1.1とHTTP/2.0は、そのままでは互換性がない
- でも、80/TCP, 443/TCPは使いたい
  - http2://example.com/ ?…論外
  - 企業内ファイアウォールのポリシー等
- 通信先がHTTP/2.0に対応しているかを識別し、適切に選択する必要がある
- HTTPでは、HTTP/1.1にあるアップグレードヘッダーを利用する

シーケンス図提供元: http://d.hatena.ne.jp/ASnoKaze/20130207/1360249692

HTTP通信の開始方法(HTTPS)

- TLSのネゴシエーション時に、アプリケーションレイヤで利用するプロトコルを選択するALPN(Application-Layer Protocol Negotiation) TLS拡張を使用

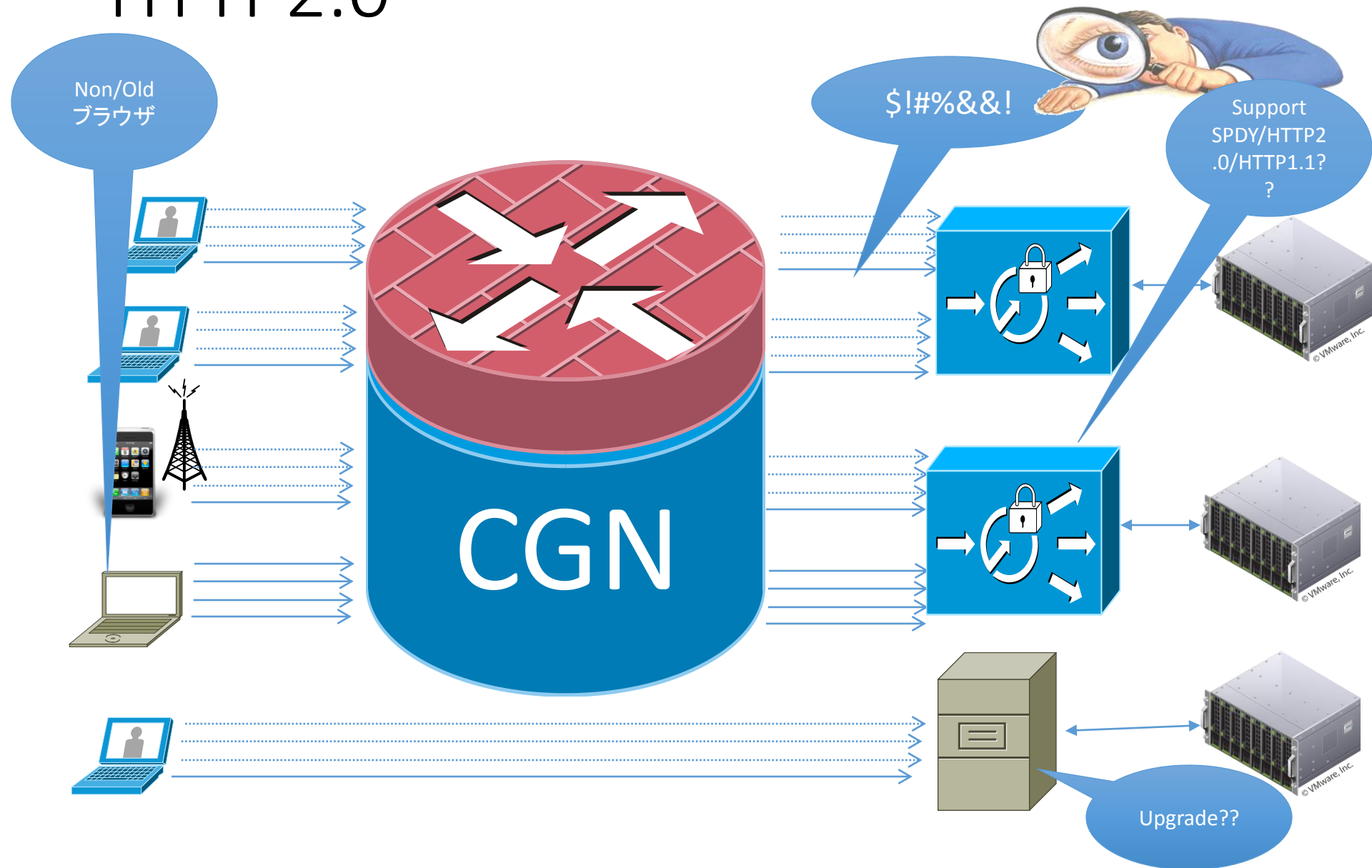


低レイヤの視点から

- バイナリ化・多重化等による可読性低下
  - WiresharkのプラグインやNetcatのようなツール類は適宜作成していく予定 (*)
- CGN等との相性？
  - HTTP/1.1と比べてTCPコネクションの寿命が長いため、NATタイマ等による問題が出るかも？
  - 概ね問題無いと思われる
- 暗号化？
  - 今までのHTTP/HTTPSの関係と同等

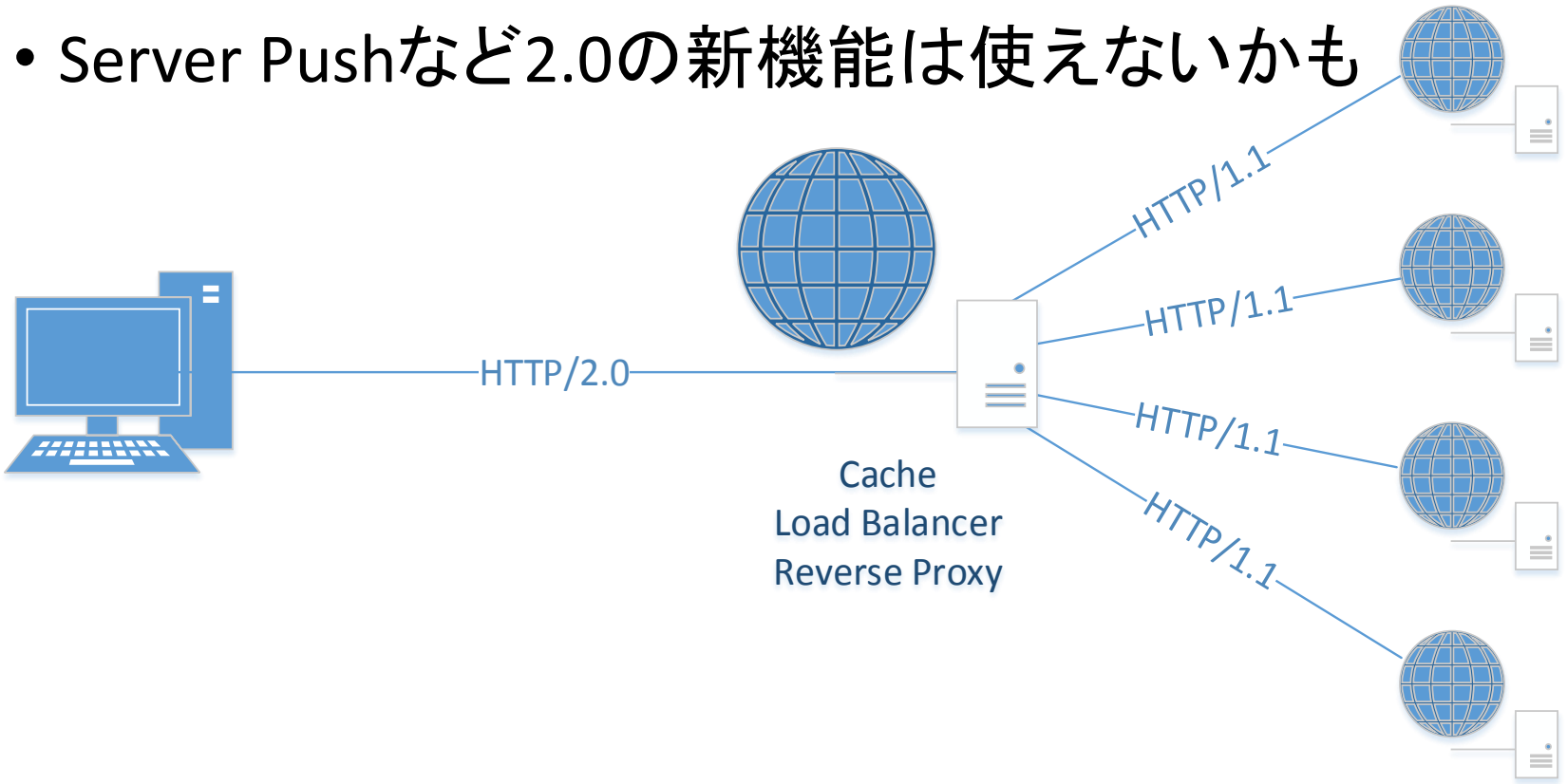
*: https://github.com/http2/wg_materials/blob/master/interim-13-01/minutes.md

HTTP2.0


Non/Old
ブラウザ
$!#%&&!
Support
SPDY/HTTP2
.0/HTTP1.1?
?
CGN
©VMware, Inc.
©VMware, Inc.
©VMware, Inc.
Upgrade??

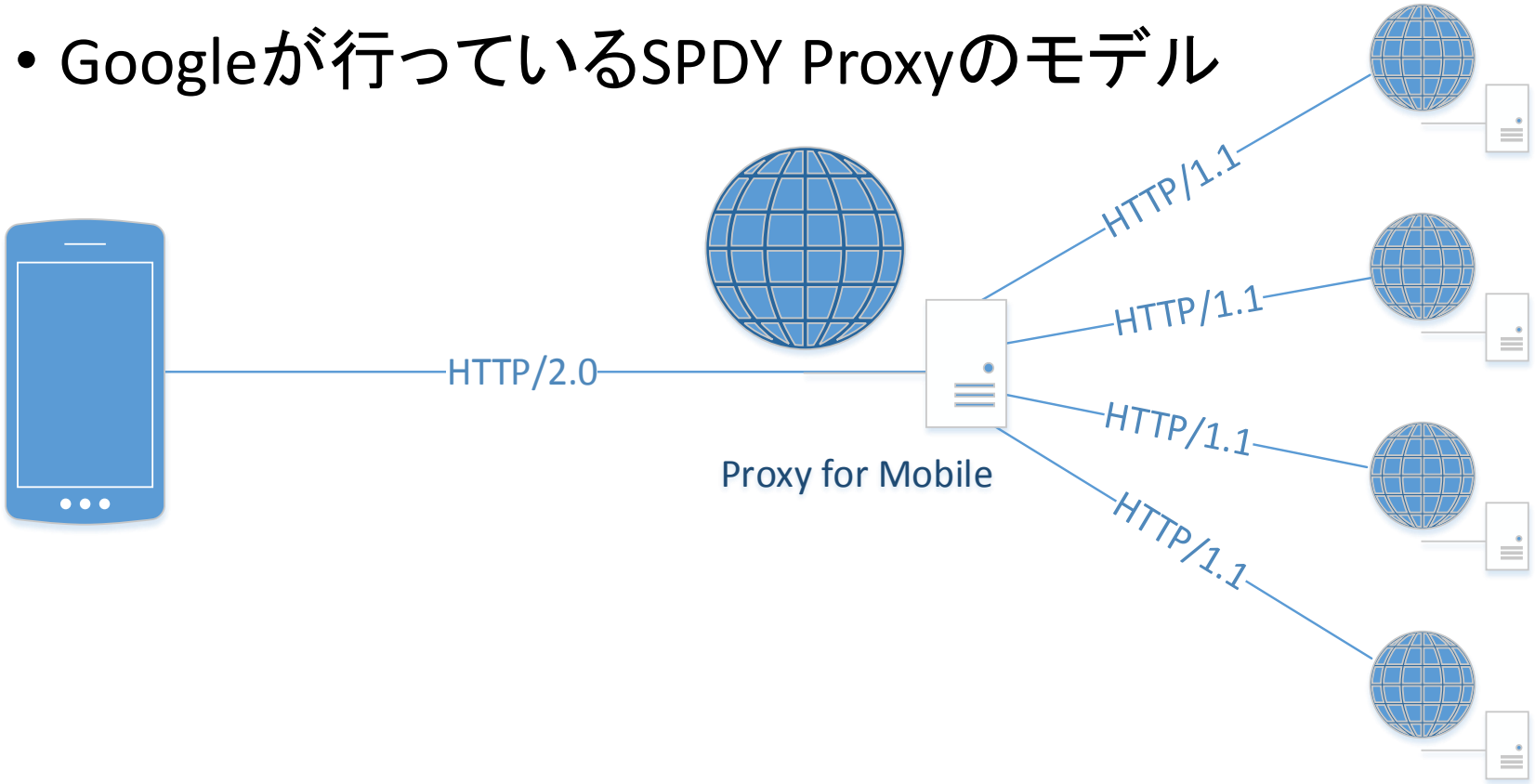
ユーザ・フロントエンド間のみ2.0

- バックエンドのコストはユーザに直接影響しにくい?
- 通信の解析が容易
- Server Pushなど2.0の新機能は使えないかも



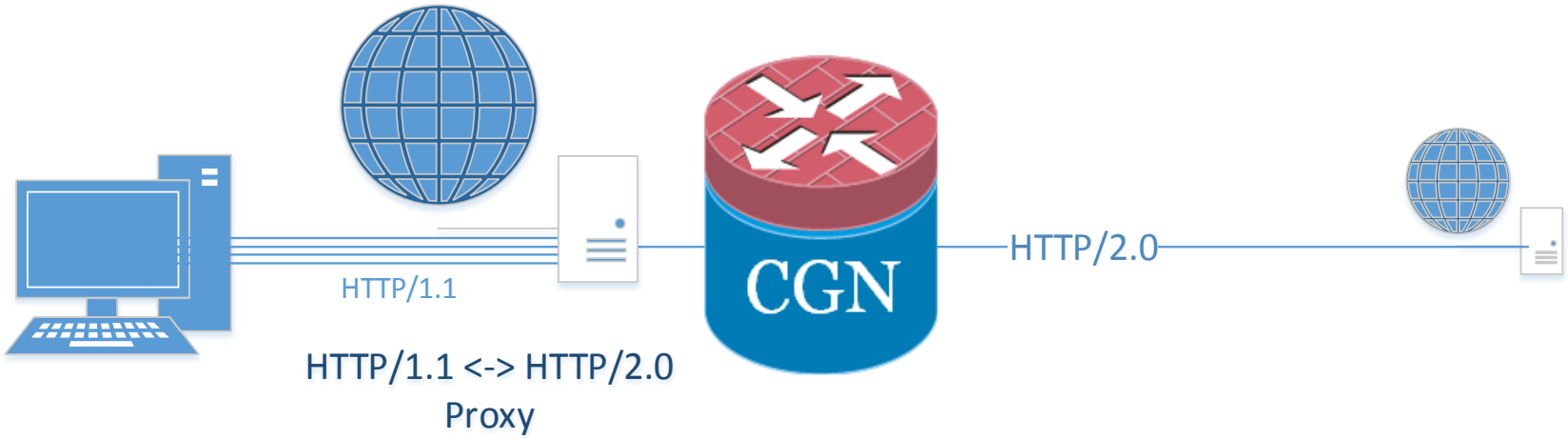
モバイル向け2.0プロクシ

- 高レイテンシなモバイル環境向け
- TCPセッション開始やDNSルックアップのコスト削減
- Googleが行っているSPDY Proxyのモデル



2.0非対応ブラウザ向けプロクシ

- クライアントからHTTP/1.1で受けてHTTP/2.0でリクエスト
- CGNの内側に置けばクライアント・プロクシ間のポート数は問題なし
- NATを通るTCPセッション数削減？



まとめ

- HTTP/2.0は効率を重視したHTTPの改良版
- HTTP/1.1と2.0は相互に変換可能(予定)
- 普及すればいいことあるかも？
  - TCPポート数やトラフィック、レイテンシ等の削減