HTTP/QUIC関連

後藤 (グリー株式会社)

自己紹介

- 後藤 浩行 (グリー)
- ISOC-JP インターネット標準化推進委員会
- 興味：Web, HTTP・QUIC関連
  - (Artなので, Transportは...)
- IETF
  - IETF94より、オフライン2回

HTTP関連 (HTTPbis)

IETF99 HTTPbis

IETF97以来、また2セッション開催に戻る
比較的落ち着いて来た、皆QUICに興味がありそう。ただしHTTPのメンテナンスの議論も

- 1セッション目
  - WG Draftの粛々とIssue 整理
  - Potential work / メンテナンスの話
- 2セッション目
  - まるまるQUIC

Documents

- State Management Bis (Mike West)
- Expect-CT (Emily Stark)
- Header common structure (Mark for Poul-Henning)
- Cache digest (Kazuho Oku)
- Random access and live content (Darshak)
- Replays in HTTP (MT)
- Origin (anything more to discuss?)
- BCP56bis (Mark)
- HTTPtre (Julian Reschke)

Cookies: HTTP State Management Mechanism Bis

Cookieの次期仕様、元々別々に提案されていた3つの仕様が一つにマージされた

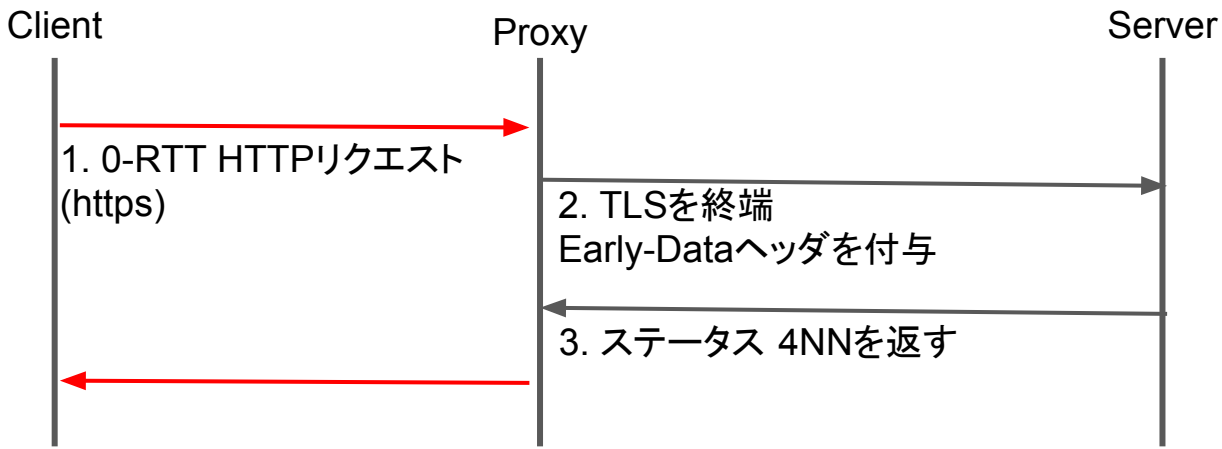
- Cookie Prefixes
- Same-Site Cookies
- Strict 'secure' Cookie (通称)

それぞれ、ブラウザに実装済みだが利用率は低く、バグなどもあったが引き続きフィードバック受付中

Using Early Data in HTTP

TLS1.3の0-RTT Early Dataには、リプレイ攻撃の可能性がある。TLS1.3側もSecurity Considaration が追記されている

0-RTT Early Dataで送信されたHTTPリクエストにEarly-Dataヘッダを付けるようにし、サーバがそれを許容できない場合に返すエラーコード4NN (Too Early)を定義する。



4NNステータスコード、418？？

4NN (Too Early)に空いている418を割り当てるか
と言ったところで議論が白熱

Hyper Text Coffee Pot Control Protocolで定義されている
「418 I’m a tea pot」を避けるか？

- 既に実装がある。避けるべきだ派
- IANAにも登録されていない、Joke RFCのために避けたくない派

利用状況などを調査/報告

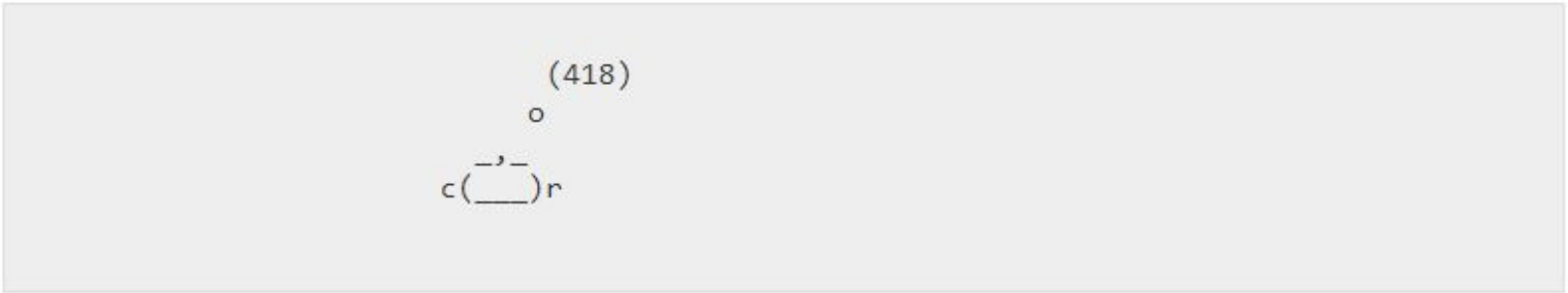
- https://www.google.com/teapot
- 各言語実装(Go, nodejs)でIssue報告、次のバージョンで修正予定

418をIANAでReservedにする方向に

mnot: Reserving the 418 HTTP Status Code
martin:The Teapot Exists (提出されず...)

2. Philosophical Considerations

We cannot be certain whether the teapot is cognizant of its own existence. Though Decartes [COGITO] used thought as the basis of his theory of existence, it's not established whether thought is a necessary precondition for existence.



Expect-CT Certificate Transparency Response Header

CT(Certificate Transparency)が使用していることを、クライアントに確認するように指示するレスポンスヘッダ

例「expect-ct: max-age=3600;report-uri=https://example.com/ 」

Issue

- Report時のCORSの扱い
  - 通常CORSでは、サーバが受け付けられるか確認するためにpreflightを送る。
  - Reportでも、preflightを送るか？(Chromeは現状送ってない。)
  - 送るなら、originヘッダは?null-origin?
  - whatwg のFetchで議論へ https://github.com/whatwg/fetch/issues/567

BCP56「**On the use of HTTP as a Substrate**」

多くのアプリケーションプロトコルでHTTPが使用される様になってきている。
HTTPを使うときのBest Current Practiceが RFC3205/BCP56で記述されている。

BCP56「On the use of HTTP as a Substrate」(2002年)

現在も多くのドラフトから参照されている。HTTPbisが使い方について記述するは有用

HTTPbis -> HTTPter ?

HTTPのErrata, Issueが溜まっている。いつアップデートを行うべきか？

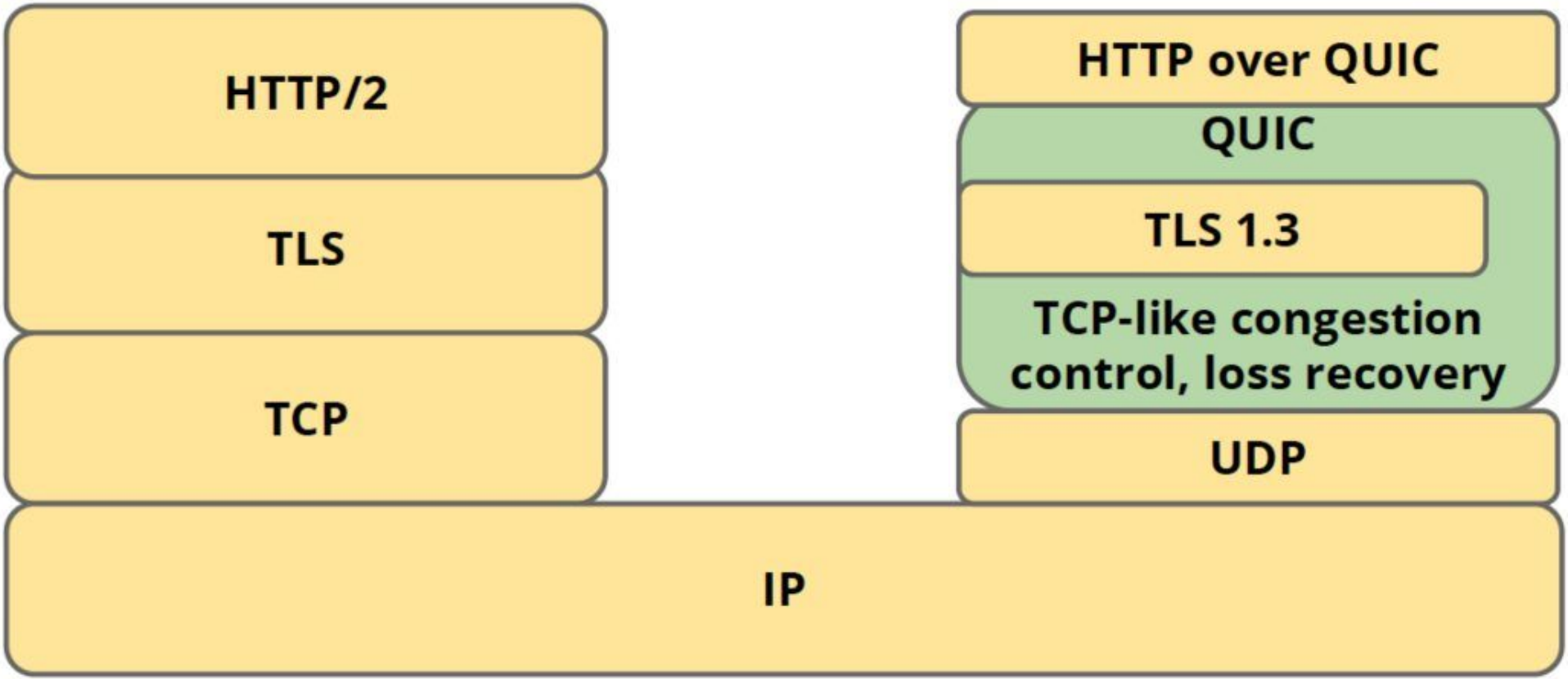
- メンテナンスは楽しい作業ではないが実施する必要がある。
- HTTPbis(RFC 726*)は重要だった
- ただし、一旦は今は実施しない

Dispatch

Web Packaging

- Webページを丸ごとパッケージングし、署名を付け再配布可能にする仕様
- もともとはW3Cで議論されていたがフォーマットの議論がIETFに持ち込まれる
- WebPackaging
  - HTTPリクエスト/レスポンスも含まれる
  - CBOR
  - 署名が付けられる(なくても不要)
- feedback
  - HTTPbisに持ってくる前にユースケースなどを整理スベき(W3Cでも議論)
  - ユースケースをはっきりさせてから、署名などの議論をスベき
  - ART ML? BoF?

QUIC関連


HTTP/2
TLS
TCP
HTTP over QUIC
QUIC
TLS 1.3
TCP-like congestion control, loss recovery
UDP
IP

QUIC WG

他のWGや、幾つかのテクニカル事項に議論が波及している
QUIC/HTTP WGとしてはCore Specに集中し、具体的な中身の議論が活発化している。Github上での議論も非常に活発...
（IETF99でハッカソンも実施。6月・10月にInterim実施。）

現在のWG Draft

Core

- QUIC: A UDP-Based Multiplexed and Secure Transport
- Using Transport Layer Security (TLS) to Secure QUIC
- QUIC Loss Detection and Congestion Control
- Hypertext Transfer Protocol (HTTP) over QUIC

Ops

- Applicability of the QUIC Transport Protocol
- Manageability of the QUIC Transport Protocol

QUIC 相互接続テスト

QUIC(04)の First Implementation に則って実装し、相互接続テストを行った。
基本的にはハンドシェイクまでの実装である (表のリンク)

client ↓ | server →	Minq	Mozquic	Quicly	Quant	ngtcp2	NGINX+ngtcp2	mvfst	picoQUIC
Minq	HDC	HDC	HD	HD	H			
Mozquic	HDC	HDC	HD	HD		H		
Quicly	HD	H	HD	HD	H			
Quant	HD	HDC	HD	HDC				
ngtcp2	H		H	H	HC	H		
mvfst							NONE	
picoQUIC		HC						HC

	Legend:				
	self-test	interop	known broken	unknown	N/A

To Test:	H = Handshake
	D = Stream Data (encrypted)
	C = Close

Second Implementation Draft

次の相互接続テストのテスト項目についての議論

- 0-RTT and Resumption
- Key Updates
- Post handshake Connection ID Changes (Migration)
- Loss Recovery beyond the exising 1-RTO retransmissions.
- Congestion Control

テストのためにはワイヤイメージを決定する必要がある。

アプリケーションプロトコルとしてGopher? HTTP/0.9?

Passive RTT measurement in QUIC

経路上でRTT測定し、通信の最適化を行う例がある。
QUICではACKも暗号化されており測定は出来ない。

選択肢

- 何もしない
- パケットナンバーをEchoする
- **1bit経路に露出し、お互い受け取った時にトグルする**

有用ではあるが、プライバシーや攻撃の手助けになりうると懸念。
議論には情報が必要

ECN support in QUIC

- Draft: draft-johansson-quic-ecn
- QUICにECN(Explicit Congestion Notification) をサポートする提案
- フィードバック
  - ACKフレームは既に複雑になっており、これ以上複雑にスベきでない
  - ECNのサポートは重要だという意見も
- hum
  - ECNを追加することに対する調査に時間を費やすのに反対が強い

Unidirectional Streams

- QUICのストリームを単方向にし、両方向のストリームを組み合わせて双方向通信を実現する方式
- ストリームの状態遷移が簡潔になる(Finも不要), 一方 バインディングが複雑になる(知見も多くない)
- 非常に白熱したが、MartinとEkrらが調査検討するまで議論を停止

HTTPバインディング

- Priority
  - HTTP/2の優先度処理ではIdle streamを活用する。QUICではIdle Streamが無い。プレースホルド用のid空間を定義するか?
- Error
  - TransportとApplicationのエラーの扱いと、エラーコード空間の議論
  - アプリケーション側が切断に対して責任をもつ
- Push
  - 同一ストリームIDでの複数回Push可能に
- ヘッダ圧縮
  - 現状はHPACK使い、コントロールストリームで送受信
  - HLoB改善のために
    - QPACK, QCRAMという2つの提案仕様がある

End