Vol.2020-IOT-48 No.23
2020/3/3

ネットワーク環境に応じた HTTPバージョン選択によるコンテンツロード時間削減

大橋 滉也[1,a)]　北口 善明[1]　山岡 克式[1]

概要：IETF で仕様策定が行われている HTTP/3 は，UDP 上に輻輳制御や再送処理を再実装した QUIC を利用することで，主にモバイルネットワークなどの，固定回線に比べ低品質なネットワークでスループットが向上する．しかし，低遅延・低パケットロス率な高品質ネットワークでは，QUIC がユーザ空間で実装されていることなどにより，HTTP/2 よりも低い性能を示すことがある．そのため，いかなる場合でも HTTP/3 を利用することがコンテンツロード時間の削減につながるとは限らない．そこで本研究では，様々なネットワーク環境における HTTP/2 と HTTP/3 の性能比較を行った結果を元に，ネットワーク環境に応じてサーバ側での動的な利用プロトコルの制御を行い，コンテンツロード時間を削減する手法を提案する．これにより，最悪ケースにおけるコンテンツロード時間を 27.8%削減した．

HTTP Version Selection Method depending on Network Environment to Reduce Contents Load Time

Abstract: The HTTP/3, for which the IETF is developing specifications, uses QUIC, which re-implements congestion control and retransmission processing on UDP. Therefore, HTTP/3 increase throughput in low-quality networks, mainly mobile networks. However, HTTP/3 in a high-quality network such as low delay and low packet loss rate may show lower performance than HTTP/2 because QUIC is implemented in the userland. Thus, using HTTP/3 does not always reduce in the content loading time. We realized dynamic control of the protocol by server-side according to the network environment, based on the results of the performance comparison between HTTP/2 and HTTP/3 in various network environments. As a result, the content load time was reduced by 27.8% in the worst case.

1. はじめに

Web アプリケーションや Web サービスの発展により，Web を支えるプロトコルスタックは継続的に発展してきた．これらは，SNS やオンラインバンキングなどの普及によってセキュリティやプライバシーを守ることが重要になったことも関係している．

現在では，一般的な Web スタックは HTTP/2 over TLS over TCP [1–3] となっている．HTTP/2 は 1 つの TCP コネクション内でストリーム多重化などを実現することにより HTTP/1.1 から性能を向上させているが，未だ TCP でのコネクション確立の 1Round-Trip Time（RTT）に加えて，TLS セッション確立のために 2RTT が必要な点や，TCP に起因する Head of Line（HOL）ブロッキングなど，非効率な点が存在している．

また，現在インターネットの利用者は 41 億人を越え，人類の過半数がインターネットを利用するようになった [4]．特にスマートフォンの普及による，モバイルネットワークは急速に拡大が進んでおり，人口の 96.6%の地域をカバーし，全世界で 63 億以上のモバイル回線契約がされている [4]．モバイルネットワークは場所に依らず，移動しながら利用できるなどの利点がある一方で，場所や天候などにより通信遅延やパケットロスが生じ，通信品質が悪化することもある．

通信品質の悪化による Web ページのロード時間の増大は，ユーザ満足度に大きく影響し，ユーザはパフォーマンスの低い Web ページから離れる傾向にある．例えば，大手検索サービスでは検索結果の表示が 0.4 秒遅くなると 1 日あたり 800 万件の検索が行われなくなる可能性があると

[1] 東京工業大学
Tokyo Institute of Technology, Meguro, Tokyo, 152–8550, Japan
[a)] k-ohashi@net.ict.e.titech.ac.jp

いう報告 [5] があるほか，ページの読み込み時間が 2 秒以内の場合は平均直帰率[*1]が 9%であるのに対して，読み込み時間が 5 秒になると平均直帰率は 38%となるという調査結果 [6] もある．

これらの状況で，ページ読み込み時間を削減するために，現在 IETF のワーキンググループで仕様の策定が進んでいるのが QUIC である．QUIC では，これまで TCP 上で行われていた輻輳制御や再送処理が，UDP 上に実装されている．これにより，TCP よりも遥かに大きな柔軟性が得られるほか，実装を完全にユーザランドで行えるため，TCP と異なりプロトコル実装のアップデートが OS のアップデートに依存しないなどの利点がある．また，QUIC は TCP における 3-way handshake と TLS1.3 のハンドシェイクを組み合わせている．これにより，暗号化及び認証がデフォルトで提供されることになる上，実際にデータを送信し始めるまでの応答回数を減らすことができる．

QUIC はすでに大規模サービスにも導入されている．例えば，大手配車サービスではアプリケーションでのテールエンドレイテンシが 10%から 30%削減された実例や [7]，動画サイトなどのトラフィックが QUIC により配信されることでインターネット全トラフィックの約 7%が QUIC となっているという報告 [8] がある．

Web プロトコルスタックの進化とともに，インターネットインフラに関しても改善が続いている．大手回線速度測定サービスの報告によると，2018 年のモバイル回線及び固定回線の下り平均速度はそれぞれ 22.82Mbps，46.12Mbps で，それぞれ前年度から+15.20%，+26.40%向上しており，固定回線の国別下り速度が世界一位となっているシンガポールでは 175.13Mbps となっている．また，日本国内に目を向けてみると，固定系超高速ブロードバンドの世帯整備率は 2018 年 3 月末時点で 99.2%となっており [9]，都市部では複数の事業者が家庭向けに 10Gbps 接続サービスを開始もしくは開始を予定している [10–12]．

パケットロスや比較的大きな通信遅延が存在するネットワークでは TCP よりも高い性能を示す QUIC であるが，100Mbps 以上の非常に高い通信帯域幅かつ非常に低い通信遅延及びパケットロス率となるネットワークでは TCP よりも低い性能を示すことがあると指摘されている [8]．これは，QUIC がアプリケーションとして実装されているため，カーネルレベルで実装されている TCP に比べメモリや CPU などが非効率であるからである．

以上のことから，いかなる場合でも HTTP/3 が最適なプロトコルであるとは限らない．さらに，今後もインターネットインフラの改善は続いていくと考えられるため，HTTP/2 を利用することが最適であるネットワーク環境は増えていくと考えられる．そこで本研究では，通信遅延・パケットロス率・通信帯域などネットワーク環境を変化させ，HTTP/2 および HTTP/3 の性能評価を行い，各プロトコルが優位性を持つネットワーク環境を調査する．その結果を元にサーバ側で動的に使用するプロトコルを選択することでコンテンツロード時間を削減する手法を提案する．

[*1] あるサイトに対する全てのセッションの中で，ユーザが 1 ページのみ閲覧してそのサイトから離脱したセッションの割合．

2. 要素技術

2.1 HTTP/2

HTTP/2 は，HTTP/1.1 との後方互換性を維持しつつ，Web パフォーマンスの向上を図るプロトコルである．リクエストヘッダの圧縮方式や，明示的な要求がなくてもサーバからコンテンツをプッシュする仕様などが含まれている．HTTP/2 では，コンテンツのオリジンごとに単一の TCP コネクションを確立し，その上に複数のリクエストを非同期に多重化する．これにより，HTTP/1.1 の HTTP パイプラインに存在していたアプリケーション層での HOL ブロッキングを解消している．

2.2 QUIC と HTTP/3

QUIC [13] は暗号化されたトランスポートを介して多重化ストリームを提供する UDP ベースのトランスポート層プロトコルであり，HTTP/3(旧 HTTP over QUIC) [14] は QUIC 上で動作するアプリケーション層プロトコルである．HTTP/2 では，TCP 上のアプリケーション層プロトコルでストリームの多重化を実現していたが，各ストリームは TCP のコネクションを共有していたため，あるストリームでパケットロスが発生した場合，全てのストリームに影響が及ぶという問題があった．しかし，QUIC では UDP 上にストリーム多重化を再実装しており，各ストリームでのパケットロスが他のストリームに影響しない仕様となっている．これにより，トランスポート層での HOL ブロッキングを解消している．

HTTP/3 は，QUIC トランスポートを利用して HTTP/2 と同様のセマンティクスを提供するためのプロトコルである（図 1）．HTTP/2 では，トランスポート層を変更せず通信遅延を改善するために，HTTP/2 自体にストリーム多重化が導入されていたが，QUIC でトランスポート層のプロトコル側にストリーム多重化が含まれるようになったためそれらの機能は削除されている．

2.3 Alt-svc

Alt-svc [16] は，ネットワーク上に現在アクセスしているサービスと同様のサービスが，別のプロトコルやオリジンで提供されていることをクライアントに示すための，HTTP ヘッダフィールドの仕様である．クライアントが Alt-svc ヘッダを含むレスポンスを受け取ると，Alt-svc ヘッダで指定されたプロトコルをサポートしておりかつそのプロト

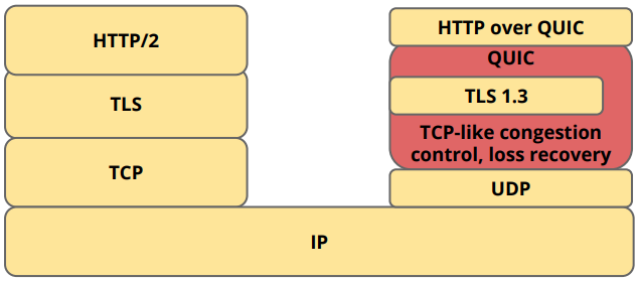

図 1　HTTP/2（左）と HTTP/3（右）のレイヤイメージ [15]

コルの利用を希望する場合に，Alt-svc ヘッダで指定されたプロトコルで指定されたオリジンに対してバックグラウンドで接続を行う．この接続が成功した場合，最初のコネクションの代わりにそのプロトコルを利用し通信を行う．

また，Alt-svc ヘッダによる別サービスが存在する情報は，ma パラメータでキャッシュの有効期限を指定することが可能である．ma パラメータが指定されない場合は，有効期限は 24 時間となる．

3. 提案手法

3.1 概要

提案手法では，サーバ側でクライアントとの間のネットワーク環境を推定し，接続時にサーバ主体で通信プロトコルを制御する．これにより，セッションごとにネットワーク環境に適したプロトコル選択が可能となり，コンテンツロード時間削減を実現する．

提案手法は，大きくネットワーク環境推定とプロトコル制御に分けることができる．

3.1.1 ネットワーク環境推定

各セッションの通信に関して，サーバはクライアントの IP アドレスをキーとして，各リクエストにおける送信パケット数，クライアントから Ack を返されたパケットの数，RTT，レスポンスを返すのにかかった所要時間を計測する．RTT としては，[17] における，min_rtt を用いる．送信したパケット数と Ack を返されたパケット数からパケットロス率を，各リクエストのレスポンスサイズとレスポンスの所要時間からスループットを求めることが可能である．以上から，サーバは対象のクライアントとの間のパケットロス率，通信遅延，スループットの情報を取得する．

3.1.2 プロトコル制御

2.3 節で述べたように，クライアントが接続先のオリジンに対するプロトコル情報を持たない時，クライアントはまず HTTP/2 でセッションを確立する．ここでサーバはそのクライアントとの間のネットワーク環境により，HTTP/2 もしくは HTTP/3 のどちらを利用するかを選択する．HTTP/3 の利用が選択された場合は，Alt-svc ヘッダで HTTP/3 での接続を促す．HTTP/2 の利用が選択された場合は，Alt-svc ヘッダをレスポンスに含めず HTTP/2 での通信を継続する．

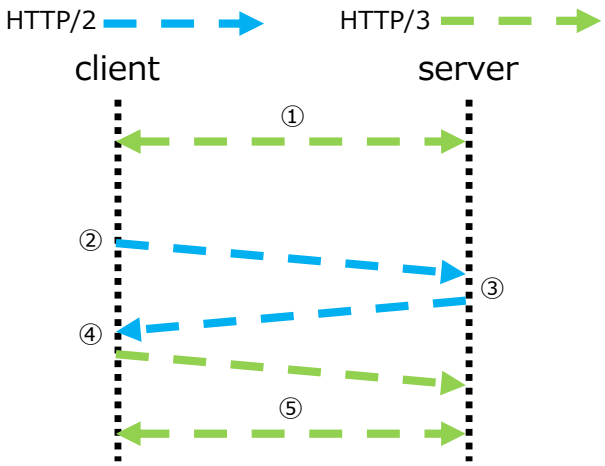

図 2　動作の流れ（HTTP/3 が選択された場合）

また，Alt-svc ヘッダの ma パラメータで指定するキャッシュの有効期限は 1 セッション程度の短い期間を設定する．これにより，次回セッション開始時に再度プロトコル選択が行われるようになる．

プロトコル選択部分について，本研究では事前に測定したネットワークを環境ごとの性能評価から判断を行っているが，同時に通信を行っている他のクライアントの通信状況や，利用者からの事前のデータ収集など別の判断手法を利用することも可能である．

3.2 動作の流れ

この節では，実際にサーバとクライアントが通信をするときの挙動（図 2）について述べる．ここでサーバからクライアントにダウンロードされるコンテンツは，一般的な Web ページを想定し，HTML に複数のサブリソースが記述されそれらが同一のオリジンから配信されることを想定する．クライアントは HTTP/3 での通信が可能なブラウザであるとする．

① ネットワーク環境測定

本手法では，あるセッションが始まる前に，対象 IP アドレスのクライアントとサーバ間でのネットワーク環境情報が必要であるが，ある IP アドレスから初めてアクセスがあった場合は，ネットワーク環境の情報を元にプロトコルを選択することができない．そのため，本研究では HTTP/3 を選択することとし，3.1.1 節で述べた項目について計測を行う．

② セッションの開始

①におけるセッションが終了し，ma パラメータで指定された有効期限が経過した後，再度クライアントがサーバに接続してくることを考える．クライアントは HTTP/2 で通信を開始し，GET /index.html のリクエストを送信する．

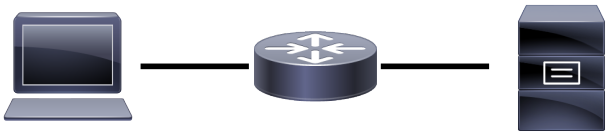

図 3　ネットワークトポロジー

表 1　仮想マシンのスペック

仮想マシン	CPU	Memory
サーバ	4Core	4GB
ルータ	2Core	1GB
クライアント	2Core	2GB

③ プロトコルの選択

リクエストを受けたサーバは，クライアントの IP アドレスをキーとしてネットワーク環境情報を取得する．その後，そのネットワーク環境情報から HTTP/2，HTTP/3 どちらのプロトコルを使うべきかを判断する．判断結果に応じてレスポンスヘッダを構築し，クライアントに対してレスポンスを返す．

④ プロトコルの変更

クライアントはレスポンスを受け取り，html の解釈を開始する．レスポンスヘッダで Alt-svc による HTTP/3 への接続が促されていた場合，並行して HTTP/3 でのコネクションの確立を開始する．

⑤ サブリソースの取得

選択されたプロトコルでサブリソースを取得する．

以上によって，ネットワーク環境に適したプロトコルを動的に選択することが可能となる．本手法は，サーバのみに導入することで実現可能であり，クライアント側に追加のソフトウェアをインストールする必要がない特徴を有する．また，クライアントが HTTP/3 に対応していない場合でも，Alt-svc ヘッダが無視されるのみで正常に通信を行うことが可能である．

4. 評価

本章では，HTTP/2 及び HTTP/3 の性能評価を行い，3 章に述べた提案手法について，実装して評価を行った．

4.1 評価環境

評価環境として，ESXi 上にサーバ，ルータ，クライアントの仮想マシンを構築し，インラインで接続した（図 3）．

各仮想マシンのスペックは表 1 の通りである．

OS は Ubuntu Server 18.04 で，カーネルバージョンは 4.15.0 を利用した．

ルータマシンで tc コマンドを用い，通信帯域制限，通信遅延追加，パケットロスなどを再現した．ESXi 上の VM にインストールした Ubuntu では標準の設定で，送信時にパケットのセグメンテーションを CPU の代わりに NIC が行う LSO（Large Send Offload）や受信時にセグメンテーションされたパケットの構築を CPU の代わりに NIC が行う LRO（Large Recieve Offload）が有効になっている．これらの機能が，有効になっていると tc コマンドの netem 機能でのパケットロスが正しく動作しないことが指摘されている [18]．この問題を回避するため，各マシンでのオフロード機能は無効化した状態で実験を行った．

表 2　実験環境

コンテンツサイズ（MB）	10
通信帯域（Mbps）	50, 100, 200, 400, 600
パケットロス率（%）	0, 1, 2, 3, 4, 5
通信遅延（ms）	0, 5, 10, 15, 20, 25
計測回数（回）	400

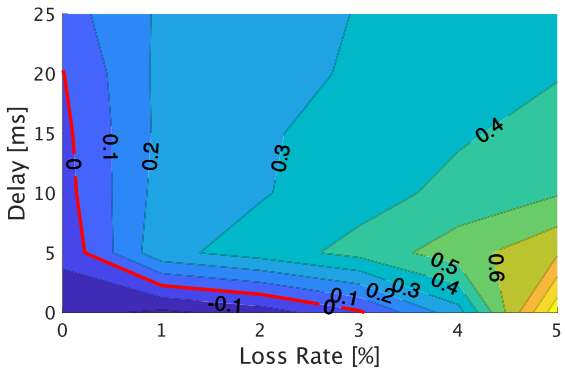

図 4　通信帯域:200Mbps での通信遅延 (Delay)・パケットロス率 (Loss Rate) に対する性能向上比 (Performance Gain)

HTTP/3 及び QUIC のバージョンは Draft 24 [13,14] を利用し，セッション間での鍵やパラメータの共有による 0-RTT は無効化した状態で通信を行った．また，TCP の輻輳制御アルゴリズムは Cubic を利用した．

4.2 ネットワーク環境ごとの HTTP/2 と HTTP/3 の性能比較

様々なネットワーク環境で，HTTP/2 および HTTP/3 が優位性を持つのかを知るために，通信帯域，通信遅延，パケットロス率などを変化させ，コンテンツロード時間を計測する実験を行った．

4.2.1 実験環境

サーバ，クライアントともに Go 言語の実装を利用した．HTTP/2 は Go 言語の標準ライブラリの実装を利用し，HTTP/3 は quic-go [19] を利用した．表 2 のようなネットワーク環境でセッション確立から 10MB のデータをダウンロード完了するまでの時間を計測した．

4.2.2 実験結果

各プロトコルのコンテンツロード時間を $load_{proto}$ としたとき，性能向上比（performance gain）を

$$\text{performance gain} = \frac{load_{HTTP/2} - load_{HTTP/3}}{load_{HTTP/3}}$$

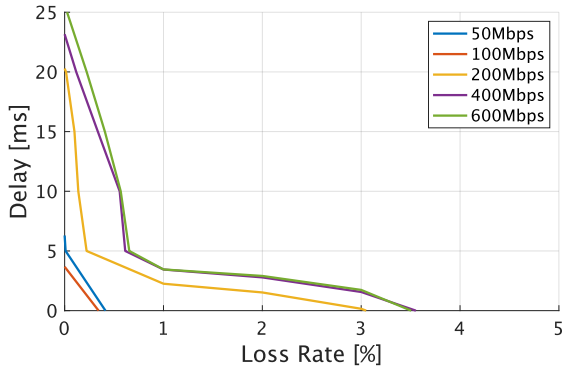

図 5 各通信帯域における性能向上比（Performance Gain）の値が 0 となる線

表 3 実験環境

コンテンツサイズ（B）	125(index.html) + 10M(image.png)
通信帯域（Mbps）	200
パケットロス率（%）	0, 2.5, 5
通信遅延（ms）	0, 12.5, 25
計測プロトコル	HTTP/2 固定，HTTP/3 固定，提案手法
計測回数（回）	400

として，計測結果を示す （図 4，図 5） ．

図 4 は，通信帯域を 200Mbps に制限した場合の性能向上比の分布を示している．性能向上比=0（図中の赤線）を境界として，性能向上比が正の値を取る領域では HTTP/2 に比べ HTTP/3 が性能優位となり，性能向上比が負の値を取る領域では HTTP/3 に比べ HTTP/2 が性能優位となることを意味している．

図 5 は，制限した通信帯域ごとの性能向上比=0 のラインをプロットしたものである．ここから低通信帯域では，殆どの環境で HTTP/3 が優位となるが，通信帯域が広くなっていくにつれ，低通信遅延・低パケットロス率な環境で HTTP/2 が優位となる環境が多くなることが確認できる．

4.3 提案手法評価

3 章で述べた提案手法について評価を行った．

4.3.1 実験環境

サーバ，クライアントともに Go 言語を利用した．HTTP/2 は Go 言語の標準ライブラリの実装を利用し，HTTP/3 は quic-go [19] を利用した．

ダウンロードするコンテンツは，index.html（125Byte）と，画像を想定したバイナリファイル（以下 image.png）（10MByte）とする．表 3 のようなネットワーク環境で，次の流れの通り計測を行った．

HTTP/2 固定・HTTP/3 固定の測定の流れ

(1) 所定のプロトコルでセッションを開始する
(2) index.html のダウンロード
(3) image.png のダウンロード
上記の 1 の直前から，3 の完了までの時間を測定する．

提案手法の測定の流れ

(1) サーバのネットワーク情報の初期化
新規のクライアントとして評価を行うため，サーバで保持している各クライアントとの間のネットワーク情報を破棄し初期化を行う．
(2) HTTP/3 でセッション開始
(3) image.png のダウンロード
サーバでのネットワーク情報計測のために，image.png をダウンロードする．
(4) 2 のセッションを終了
(5) HTTP/2 でセッション開始
(6) index.html のダウンロード
(7) プロトコルの選択
index.html のレスポンスヘッダの Alt-svc を解釈し，プロトコルの選択を行う．HTTP/3 が選択された場合は HTTP/3 でのセッションを開始し，セッションが確立するまで待機する．
(8) 選択されたプロトコルで image.png のダウンロード
上記の 5 の直前から，8 の完了までの時間を測定する．

4.3.2 プロトコル選択関数

本実験手法では，プロトコル選択のための関数として，図 4 の性能向上比の値が 0 となる折れ線を利用し，この線より左下の領域では HTTP/2 を選択，右上の領域では HTTP/3 を選択するようにした．

4.3.3 実験結果

400 回計測したものの平均をまとめたもの及び，提案手法において HTTP/3 が選択された割合を表 4 に示す．

HTTP/2 固定及び HTTP/3 固定の結果を見ると，実験番号 1，2，4 では HTTP/2 が優位なネットワーク環境，それ以外では HTTP/3 が優位なネットワーク環境であることがわかる．実験番号 1，4 では 400 回全てで HTTP/2 が選択されており，HTTP/3 固定に比べそれぞれコンテンツロード時間が削減されている．

一方で，実験番号 2 では HTTP/2 が選択されるべきだが，74%の測定で HTTP/3 が選択される結果となっている．これは，4.2 節の実験で，パケットロス率 0%の環境であっても，実際には通信帯域の制限により輻輳が発生し，その影響によるパケットロスも発生していた可能性があるが，プロトコル選択関数の導出においては，ネットワーク環境の設定値しか用いておらず，実際のパケットロス率に基づいていなかったためであると考えられる．したがって，プロトコル選択関数の導出時にはパケットロス率の実測値を用いるべきであったと考えられる．

HTTP/3 優位な環境は実験番号 3，5〜9 である．これらのネットワーク環境では全ての環境で HTTP/3 を選択できている．特に実験番号 9 では，提案手法は HTTP/2 に比べて 27.8%コンテンツロード時間を削減している．しかし，実験番号 3 では提案手法が最もコンテンツロード時間

表 4　実験結果

実験番号	ネットワーク環境	HTTP/2 固定	HTTP/3 固定	提案手法	提案手法における HTTP/3 選択率
1	0%, 0ms	446.74ms	535.73ms	448.23ms	0.0%
2	0%, 12.5ms	669.02ms	766.43ms	749.81ms	74.0%
3	0%, 25ms	993.20ms	972.62ms	1056.89ms	65.8%
4	2.5%, 0ms	636.40ms	746.84ms	649.24ms	0.0%
5	2.5%, 12.5ms	29700.82ms	22463.81ms	22698.84ms	97.3%
6	2.5%, 25ms	57416.07ms	44329.03ms	44605.80ms	97.8%
7	5%, 0ms	1640.68ms	858.72ms	1092.63ms	99.8%
8	5%, 12.5ms	49729.07ms	33812.26ms	34058.50ms	99.3%
9	5%, 25ms	92879.67ms	66450.00ms	67105.30ms	98.0%

が長くなっている．これは，HTTP/3 がプロトコル変更を伴わないのに対して，提案手法では HTTP/3 のセッション確立を待つ分オーバーヘッドが生じているからである．

5. 今後の課題

本章では，本研究の今後の展望について述べる．

5.1 他実装での検討

本研究では，性能評価や提案手法評価を Go 言語による QUIC 実装を用いた．しかし，QUIC は実装言語や実装者の違う多くの実装が存在する．これらは，IETF の QUIC WG で相互通信の検証は行われており，プロトコルの仕様を満たしていることは確認されているものの，言語特性や最適化により性能が大きく異なる可能性が考えられる．そのため，他の QUIC 実装でも評価を行い，同様の特徴が見られるか検討する必要がある．

5.2 通信帯域の考慮

本研究の提案手法評価では，制限通信帯域が 200Mbps での性能評価結果を元にプロトコル選択関数を構築し，制限通信帯域を 200Mbps で固定した環境で提案手法の評価を行った．これは，提案手法においてサーバで計測可能なパケットロス率・通信遅延・スループットのうち，パケットロス率と通信通信遅延に関しては性能評価におけるネットワーク環境設定と直接比較可能なためである．制限通信帯域が変化する場合，通信帯域によってプロトコル選択関数に用いる性能向上比の閾値のグラフを切り替えるなどの方法が考えられる．しかし，計測によって得られるスループットは制限通信帯域だけではなく，パケットロスと通信遅延が関与するため，どの通信帯域の性能向上比グラフを用いればよいか判断が難しい．とはいえ，実際のインターネットではクライアントとのスループットはそれぞれのクライアントごとに異なるため，通信帯域の考慮は必ず必要となってくる．そのため，今後はより詳細に性能評価を分析するなどして，通信帯域を考慮してプロトコル選択を行うことを検討する．

5.3 HTTP/2 から HTTP/3 への切り替え

本研究の提案手法実装では，ネットワーク環境を測定するのは HTTP/3 とした．HTTP/3 はユーザ空間のアプリケーションのみで実装されているため，リクエストごとのパケットの送信数やパケットロスの検知などを比較的容易に行えることから採用している．一方，HTTP/2 で同様の測定を行おうとした場合，TCP スタックがカーネルに実装されておりシステムコールを用いてパケットキャプチャを行う必要がある上，HTTP/2 で制御されるストリームとの対応付けも必要となり非常に複雑になってしまうことから実装を見送った．

これにより，本提案手法評価の実装では，HTTP/2 が選択された場合，その後ネットワーク環境が悪化し HTTP/3 が優位な状況になったとしても，HTTP/2 から HTTP/3 へのプロトコルの切り替えがサーバ側から促されないという問題が残っている．そのため，今後は HTTP/2 でもネットワーク環境の測定を行う仕組みを実装し，HTTP/2 から HTTP/3 への切り替えを実現する方法を検討する必要がある．

6. おわりに

本研究では，様々なネットワーク環境での HTTP/2 および HTTP/3 の性能評価を行い，ネットワーク環境ごとにいずれのプロトコルが高い性能を示すかを精査した．その結果，低通信帯域のネットワークでは殆どの状況で HTTP/3 が高い性能を示す一方，低遅延・低パケットロスな環境では広通信帯域になるにつれ，HTTP/2 が高い性能を示すことを確認した．

さらに上記性能評価の結果を元に，HTTP/3 が主に高品質な広通信帯域ネットワークにおいて，HTTP/2 よりも低い性能を示すことに注目し，サーバ・クライアント間のネットワーク環境に応じて動的に利用するプロトコルをサーバ側主体で選択することで，多くのネットワーク環境でコンテンツロード時間を削減する手法を提案した．提案手法を実装評価した結果，最悪ケースにおけるコンテンツロード時間を 27.8%削減可能であることを示した．

謝辞

本研究の一部は，科学技術振興機構事業研究成果最適展開支援プログラム A-STEP 機能検証フェーズの支援を受けて実施した．

参考文献

[1] M. Belshe, R. Peon, and M. Thomson, “Hypertext Transfer Protocol Version 2 (HTTP/2),” RFC 7540, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7540.txt

[2] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, RFC Editor, Aug. 2008. http://www.rfc-editor.org/rfc/rfc5246.txt

[3] J. Postel, “Transmission Control Protocol,” STD 7, RFC Editor, Sept. 1981. http://www.rfc-editor.org/rfc/rfc793.txt

[4] I.W. Telecommunication, “Statistics,” https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx, Oct. 2019.

[5] V. Green, “Impact of slow page load time on website performance,” https://medium.com/@vikigreen/impact-of-slow-page-load-time-on-website-performance-40d5c9ce568a, Jan. 2016.

[6] pingdom, “Does Page Load Time Really Affect Bounce Rate? - Pingdom Royal,” https://royal.pingdom.com/page-load-time-really-affect-bounce-rate/, Jan. 2018.

[7] E.G. Rajesh Mahindra, Vinoth Chandar, “Employing QUIC Protocol to Optimize Uber’s App Performance,” https://eng.uber.com/employing-quic-protocol/, May 2019.

[8] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, and etal., “The QUIC Transport Protocol: Design and Internet-Scale Deployment,” Proceedings of the Conference of the ACM Special Interest Group on Data Communication , p.183–196, SIGCOMM ’17, Association for Computing Machinery, New York, NY, USA, 2017. https://doi.org/10.1145/3098822.3098842

[9] 総務省，“ブロードバンド基盤の整備状況（平成 30 年 3 月末現在），” https://www.data.go.jp/data/dataset/soumu_20140909_0702/resource/2516c921-ba20-4e1e-b08f-296a7baa1b18，Jan. 2019.

[10] ソネット株式会社，“『NURO 光』、10 ギガサービスの提供開始について，” https://www.sonynetwork.co.jp/corporation/release/2015/pr20150601_3068.html，June 2015.

[11] KDDI 株式会社，“世界最速、超ひかり。「au ひかり ホーム 10 ギガ」を 3 月 1 日より受付開始，” http://news.kddi.com/kddi/corporate/newsrelease/2018/02/08/2947.html，Feb. 2018.

[12] 榊原 康，“NTT 東が 10 ギガのフレッツ光、ドコモなども同時期の提供が濃厚か，” https://tech.nikkeibp.co.jp/atcl/nxt/news/18/06793/，Jan. 2020.

[13] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet-Draft draft-ietf-quic-transport-24, IETF Secretariat, Nov. 2019. http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-24.txt

[14] M. Bishop, “Hypertext Transfer Protocol Version 3 (HTTP/3),” Internet-Draft draft-ietf-quic-http-24, IETF Secretariat, Nov. 2019. http://www.ietf.org/internet-drafts/draft-ietf-quic-http-24.txt

[15] QUIC WG, “QUIC Tutorial,” Internet-Draft slides-98-edu-sessf-quic-tutorial-00, Internet Engineering Task Force, March 2017. https://datatracker.ietf.org/doc/html/slides-98-edu-sessf-quic-tutorial-00

[16] M. Nottingham, P. McManus, and J. Reschke, “HTTP Alternative Services,” RFC 7838, RFC Editor, April 2016.

[17] J. Iyengar and I. Swett, “QUIC Loss Detection and Congestion Control,” Internet-Draft draft-ietf-quic-recovery-24, Internet Engineering Task Force, Nov. 2019. https://datatracker.ietf.org/doc/html/draft-ietf-quic-recovery-24

[18] K. Sasaki, T. Hirofuchi, S. Yamaguchi, and R. Takano, “An Accurate Packet Loss Emulation on a DPDK-Based Network Emulator,” Proceedings of the Asian Internet Engineering Conference , p.1–8, AINTEC ’19, Association for Computing Machinery, New York, NY, USA, 2019. https://doi.org/10.1145/3340422.3343635

[19] lucasclemente, “lucas-clemente/quic-go: A QUIC implementation in pure go,” https://github.com/lucas-clemente/quic-go.