高遅延高パケットロス環境における HTTP/3 QUIC の通信性能に関する一考察

宮澤 航人† 明吉 舞世‡ 山口 実靖‡

†工学院大学 大学院 工学研究科 電気電子工学専攻 〒163-8677 東京都新宿区西新宿 1-24-2
‡工学院大学 情報学部 情報通信工学科 〒163-8677 東京都新宿区西新宿 1-24-2
E-mail: †cm19044@ns.kogakuin.ac.jp, ‡{j016006@ns, sane@cc}.kogakuiun.ac.jp

あらまし HTTP1.1 は，TCP コネクションを大量に確立しサーバに負荷がかかることによって生じる C10k 問題などの課題がある．そこで，HTTP/2, QUIC などが提案され，特に昨今は HTTP/3 の提案に注目が集まっている．従来の HTTP はトランスポート層において TCP を用いて動作していたが，HTTP/3 は QUIC を用いて動作する．QUIC は UDP 上で動作するが，内部では TCP と同様の誤り訂正やフロー制御，輻輳制御などを行っている．また，QUIC の輻輳制御アルゴリズムは CUBIC TCP を用いている．そのため，HTTP/3 は従来の CUBIC TCP 上で動作する HTTP と同様の問題点を抱えている可能性がある．本稿では，任意のパケットロス率を設定できるネットワーク環境において単一フローの HTTP/3 の性能を評価し，パケットロスが多発する環境において性能が劣悪化することを示す．次に，QUIC の最低輻輳ウィンドウサイズの設定を変更することで，単一フローHTTP/3 通信の性能向上手法を提案する．そして，性能評価によりその有効性を示す．

キーワード HTTP/2, HTTP/3, QUIC, CUBIC, 輻輳制御

1. はじめに

これまでに web データ転送用プロトコルとして,HTTP/1 [1] や/1.1 [2] が伝統的に使われて来たが,C10k 問題 [3] 等の課題が指摘され,新しいプロトコルの開発の重要性が主張されるようになってきた.そして 2015 年に HTTP/2 [4] が提案され,近い将来に HTTP/3 [5] が提案される予定である.HTTP/3 は TCP ではなく UDP を用いるなどの挑戦的な取り組みを含んでいるが,輻輳制御アルゴリズムとしては古典的な手法になった CUBIC TCP [6] が採用されており,その性能の評価や考察は重要であると考えられる.本稿では,高遅延高パケットロス環境に着目し,その環境における HTTP/3 の性能を評価し,考察を行う.

2. 関連研究

2.1. HTTP/2

HTTP/2 [4] は HTTP/1.1 [2] と互換性を持つ Hypertext Transfer Protocol である.効率的にデータを転送できるよう新たな仕組みが導入された.その 1 つであるストリームは通信を管理する為の単位であり,1HTTP リクエストに対して,1 つのストリームが生成される.HTTP/2 では 1 つの TCP コネクションで通信を行い,そのコネクション中にリクエストの数と同等のストリームを生成する.各ストリームは TCP コネクション内で多重化され,並列に転送されることによって後述する HoL ブロッキング問題を解消している [7].

しかし Saxce らの論文[8]によると，HTTP/2 の性能と HTTP/1.1 の性能と比較し，HTTP/2 の性能は セルラーネットワーク(携帯電話網)にて低くなることを示した.

2.2. HoL ブロッキング問題

HTTP/1.0 [1] では 1 つリクエストを送信するとレスポンスが返ってくるまで次のリクエストを送ることが出来なかった.HTTP/1.1 [2] では HTTP パイプラインが導入され,リクエスト送信後にレスポンスを待たず, 1 つの TCP コネクション上で複数のリクエストを送信することが可能となった.このことによりデータの転送時間が短くなった.

しかし HTTP パイプラインによって複数のリクエストを送信することが出来てもリクエストとレスポンスの順序が同期していなければいけない.その為,先頭のレスポンスが遅いとその後のレスポンスが待ち状態となり,全体としての処理が遅くなってしまう.この問題を HoL(Head of Line)ブロッキング問題という.この問題を解決するため HTTP 要求ごとに TCP コネクションを確立するという方法があるが,サーバに対し,クライアントが複数の TCP コネクションを確立することでサーバに負荷がかかり好ましくない [7] .この問題は C10k 問題として広く知られている.

2.3. HTTP/3

HTTP/3 [5] は，HTTP/2 API を QUIC プロトコル上で動かすためのプロトコルである．QUIC は，UDP 上で動く通信プロトコルである. QUIC は TCP とは異なり，接続の確立に 3-way handshake を必要としないため，データ提供開始までのオーバーヘッドが無い．また，TCP では送出されたパケットが順番通りに届くことが求められているが，QUIC では必ずしも順番通りに届

く必要がない．そのため，パケットの入れ替わりが発生しうるような混雑した経路上で通信を行う場合でも，比較的効率よく高速に通信を行うことができる．

2.4. CUBIC TCP

CUBIC TCP [6] は輻輳制御アルゴリズムであり,スロースタートフェーズと輻輳回避フェーズで構成されている.スロースタートフェーズでは，パケットロスを検出するまでウィンドウサイズを指数関数的に増加させる．ウィンドウサイズがスロースタートしきい値に達すると，輻輳回避フェーズに移行する．輻輳回避フェーズでは，以下に示す三次関数を用いた式に従って輻輳ウィンドウサイズを制御する．

$$\mathrm{cwnd} = \mathrm{C}(\mathrm{t}-\mathrm{K})^3$$

$$\mathrm{K} = \sqrt[3]{\frac{W_{max}\beta}{c}}$$

ただし，C 及びβはウィンドウサイズの増減速度を決めるパラメータであり，近年の Linux では主に C=0.4, β=0.3 が用いられている．上式において，cwnd は輻輳ウィンドウサイズ，t は最後のパケットロスから現在までの時間，W_{max}は最後のパケットロス時の輻輳ウィンドウサイズである．

3. HTTP/3 基礎性能調査

本章にて，HTTP/3 の基礎性能の評価を行う．

図 1 に示すような，3 台の計算機（サーバ PC，クライアント PC，ブリッジ PC）を用いてネットワークを構築した．すべての計算機には Linux OS をインストールした．ブリッジ PC は NetEm [9] を用いてネットワーク遅延およびパケットロスを擬似的に発生させる．

クライアント PC からサーバ PC へデータを要求し，それに応じてサーバ PC がクライアント PC にデータを送信する．ここで，送信するデータは 32KB, 1MB, 32MB, 1GB のいずれかの文字列データを用いた．また，ブリッジ PC の NetEm を用いて，遅延時間を 64ms，パケットロス率を 0.0001%, 0.001%, 0.01%, 0.1%, 1% のいずれかに設定した．

パケットロス発生率と通信中の平均スループットの関係を図 2 に示す．測定の結果から，データ量が 32KB, 1GB において，パケットロス発生率が 0.01%以上の場合，平均スループットが大幅に減少することがわかる．

4. HTTP/3 の動作の観察

パケットロス率が 0.1%，遅延時間が 64ms の環境で，1GB のデータを要求しそれに応じる通信を行った際の輻輳ウィンドウサイズの時間推移を図 3 に示す．HTTP/3 は CUBIC TCP のアルゴリズムと同様の輻輳制御を行っているため，パケットロスが発生すると輻輳ウィンドウサイズが減少する．図 3 に着目すると，実

図 1 実験環境

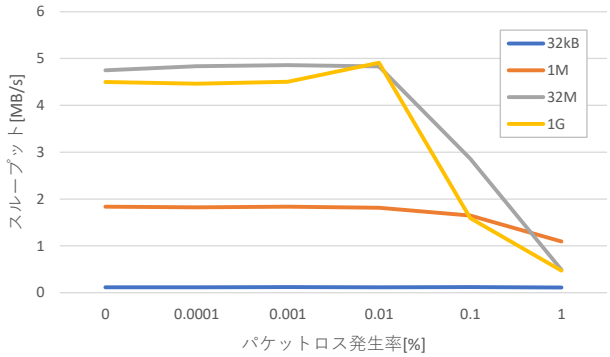

図 2 パケットロス発生率と平均スループットの関係

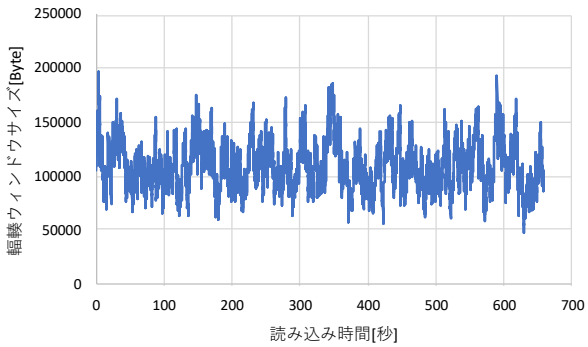

図 3 輻輳ウィンドウサイズの推移
(パケットロス率: 0.1%，遅延時間 64ms)

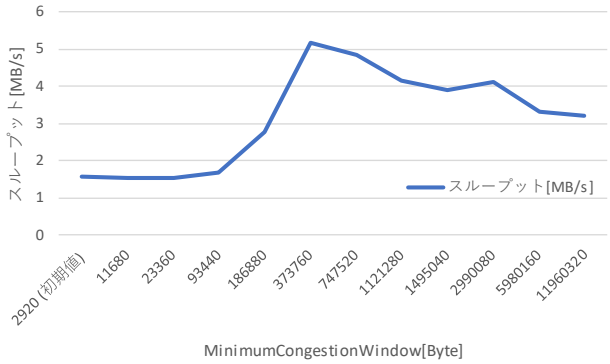

図 4 min_cwnd と平均スループットの関係

際にパケットロスに反応して輻輳ウィンドウサイズが減少している様子が確認できる．

5. 提案手法

MinimumCongestionWindow とは，QUIC において最小の輻輳ウィンドウサイズを定義しているものであり，オリジナルでは kDefaultTCPMSS の 2 倍の値を用いている．ただし，kDefaultTCPMSS とは 1 セグメントで転送可能なデータの最大長(KB 単位の Maximum Segment Size)であり，1460Byte である．

本稿では，MinimumCongestionWindow の値を kDefaultTCPMSS×n と定義し，n を拡大することにより，高遅延高パケットロス環境における単一フローのスループットを向上させることを提案する．具体的には，本稿では n=2~8192 に変更する．

以降，本稿では MinimumCongestionWindow を min_cwnd と表記する．

6. 性能評価

本章では，前章で述べた変更を適用させた QUIC を用いて性能評価を行った．

6.1. min_cwnd とスループットの関係

本節では，min_cwnd の n の値とスループットの関係について調査した．ここでは，n を 2~8192 の範囲で変更して性能評価を行った．なお，最小のウィンドウサイズは min_cwnd = n × 1460 Byte となる．パケットロス率 0.1%，遅延時間 64ms の環境において，クライアント PC が 1GB のデータを要求し，サーバ PC がそれに応じる通信を行った．

min_cwnd と通信中の平均スループットの関係を図 4 に示す．図 4 より，min_cwnd が 186880Byte 以上の場合,平均スループットが大幅に増加することが確認できた．また，min_cwnd の値と平均スループットとの間には一部相関があることが確認できた．

また, min_cwnd を 2990080 とした時の輻輳ウィンドウサイズの推移を図 5 に示す．図 3 と図 5 を比較すると, min_cwnd を 2990080 とした時の輻輳ウィンドウサイズは，オリジナルの設定の輻輳ウィンドウサイズと比べて約 30 倍増加していることがわかる．

6.2. min_cwnd と遅延時間の関係

本節では，変更された min_cwnd における，遅延時間とスループットの関係を調査した．ここで，min_cwnd は，6.1 節の実験で最も平均スループットが高かった 373760 Byte を用いた．パケットロス率は 0.1%,遅延時間は 64ms で測定を行った.測定環境は基礎性能調査と同一である.測定された平均スループットと，デフォルトの min_cwnd における平均スループットと遅延時間の関係を図 6 に示す．

図 6 より遅延時間 64ms 以上で提案手法によってスループットが向上しておることがわかる.しかし遅延時間 0~16ms では同程度またはやや劣る結果となった．

6.3. min_cwnd と遅延時間の関係

本節では，変更された min_cwnd における，パケットロス率とスループットの関係を調査した．ここで，min_cwnd は前節と同様に 373760 Byte を用いた．遅延時間は 64ms に設定し, データサイズ 1GB を転送した．測定された平均スループットと，デフォルトの min_cwnd における平均スループットとパケットロス率の関係を図 7 に示す．提案手法ではパケットロス率

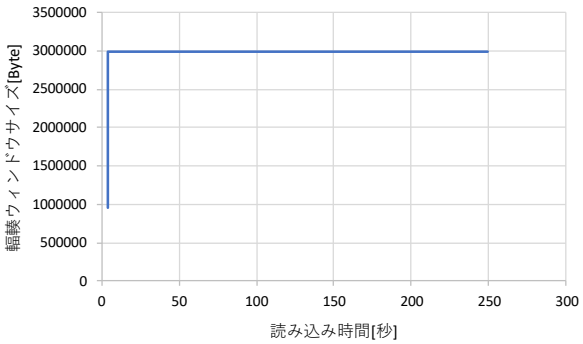

図 5 輻輳ウィンドウサイズの推移
(パケットロス率: 0.1%，遅延時間 64ms，
min_cwnd = kDefaultTCPMSS*1024)

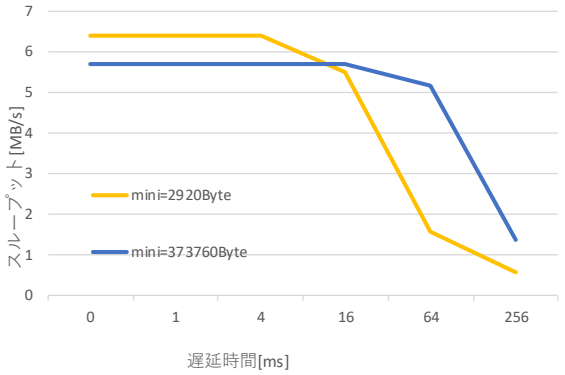

図 6 min_cwnd と遅延時間の関係

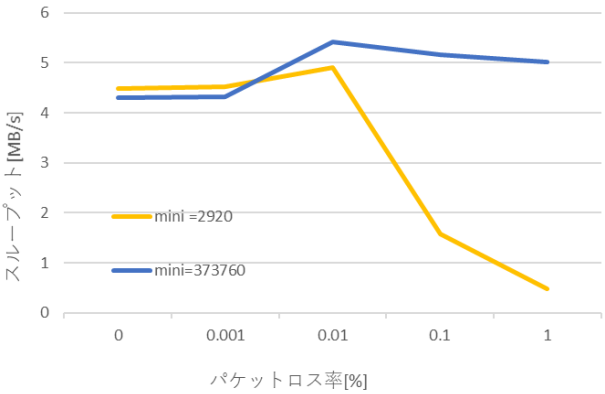

図 7 min_cwnd とパケットロス率の関係

0.01%以上の場合でスループット向上した.パケットロス率 1%の場合では min_cwnd=2920 の場合と比べてスループットがおよそ 10 倍となった．

7. 考察

高遅延，高パケットロス環境において，CUBIC TCP を輻輳制御アルゴリズム として用いた場合の HTTP/3 QUIC は，パケットロスを検出するたびに輻輳ウィンドウサイズを低下させるため，通信性能が劣悪化すると考えられる. そこで, min_cwnd 値を増やすことによって，パケットロス検出時の輻輳ウィンドウサイズの大幅な減少を抑えることができるため，スループット

を向上させることが出来たと考えられる．ただし，最低輻輳ウィンドウサイズ値を大きくする場合，多数の通信が行われているインターネットなどの環境で輻輳崩壊が発生する可能性がある．従って，本提案をインターネット等の公平性が必要とされる環境にそのまま適用するのは不適切であると考える．

8. おわりに

本稿では，新しい web データ用転送プロトコルである HTTP/3 の基礎調査と，最低輻輳ウィンドウサイズを変更して性能評価を行った．その結果，特にパケットロスが多発する環境において性能が劣悪化すること，最低輻輳ウィンドウサイズを大きくすることによって性能が向上することを示した．

今後は，実ネットワーク環境であることを考慮した上で，性能を向上させる手法を考案，提案していきたい．

参 考 文 献

[1] Berners-Lee, Tim, Roy Fielding, and Henrik Frystyk. "RFC 1945: Hypertext transfer protocol—HTTP/1.0."

[2] Fielding, R., et al. "RFC 2068: Hypertext transfer protocol–HTTP/1.1, Jan. 1997."

[3] Kegel, Dan. "The C10K problem." (2006).

[4] Stenberg, Daniel. "HTTP2 explained." *Computer Communication Review* 44.3 (2014): 120-128.

[5] Bishop, M., and E. Akamai. "Hypertext transfer protocol version 3 (HTTP/3) draft-ietf-quic-http-18." *Internet Requests for Comments, IETF Internet Draft, Tech. Rep*. (2019).

[6] Injong Rhee and Lisong Xu “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” *Proc. Workshop on Protocols for Fast Long Distance Networks*, 2005, 2005.

[7] Varvello, Matteo, et al. "Is the web http/2 yet?." *International Conference on Passive and Active Network Measurement*. Springer, Cham, 2016.

[8] de Saxcé, Hugues, Iuniana Oprescu, and Yiping Chen. "Is HTTP/2 really faster than HTTP/1.1?." *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2015.

[9] S. Hemminger, “Network Emulation with NetEm”, Proceedings of the 6th Australia’s National Linux Conference (LCA2005), April 2005, Canberra, Australia.