情報処理学会論文誌 Vol.63 No.2 634–644 (Feb. 2022)

複数の経路において複数のTCP接続を用いるタイマ駆動のプログレッシブダウンロード方式

舟阪 淳一[1,a)]

受付日 2021年5月10日, 採録日 2021年11月2日

概要：複数の通信メディアを利用できる携帯端末が普及してきており，それらの帯域を同時に使い切ることができれば，プログレッシブダウンロードにおいて高品質の動画が再生できると考えられる．従来研究では複数の経路の上に複数の TCP 接続を確立してサーバから動画を取得する際，タイマ駆動により定期的に要求を発行することで，高品質の動画が再生可能であることを示したが，不均一な条件の複数経路においては非効率的な要求が問題となっていた．本研究では適応的に要求間隔を調整しながら，到着順序の逆転を緩和する方式を提案し，評価する．提案方式においては要求した部分ファイル（ブロック）の到着間隔を経路ごとに観測するとともに，各接続でブロックの取得に費やした応答時間を記録し，順序逆転を予防しながら適切な要求間隔を動的に求めることにした．提案方式を評価した結果，固定の間隔で要求する方式と比較して，高画質の動画を短い待ち時間で再生可能になることが分かった．

キーワード：プログレッシブダウンロード，複数経路，複数接続，順序制御，適応制御

Timer-driven Progressive Download Scheme Utilizing Multiple Connections on Multiple Paths

Junichi Funasaka[1,a)]

Received: May 10, 2021, Accepted: November 2, 2021

Abstract: Since a great deal of mobile terminals which can use multiple communication media such as WiFi and LTE are emerging, we can realize video playbacks in high quality by HTTP streaming, or progressive download utilizing their total bandwidth. In our previous work, we have proposed a progressive download method which establishes multiple TCP flows on each of multiple network paths and a timer-driven requesting scheme which does not need any information on bandwidth, round trip delay, and packet loss rate in advance. Although the proposal yields as high performance as the existing method which requires the advance information on network conditions, it is reported not to request the blocks (partial files) efficiently especially when the multiple paths have heterogeneous network conditions. In this paper, some adaptive adjustment methods of the requesting interval to avoid out-of-order arrivals are proposed and evaluated. In the proposal, the requesting interval is adjusted according to the observed block arrival rate on each path. In addition, our proposal proactively adjust the block ID for the next request according to the recorded response time of each connection. The proposal is evaluated and confirmed to be able to provide the video files with higher quality with a short waiting time before starting the playback.

Keywords: progressive download, multiple paths, multiple connections, order control, adaptive control

1. はじめに

高性能なモバイル端末が普及し，WiFi，LTE や WiMAX などいくつかの通信メディアに接続することにより，電話だけでなく高品質の動画視聴サービスなどが実現してき

[1] 広島市立大学情報科学研究科
Graduate School of Information Sciences, Hiroshima City University, Hiroshima 731–3194, Japan
[a)] funa@hiroshima-cu.ac.jp

ている．またインターネットを経由して信頼性のあるデータ通信を実現するため，従来よりTCPが使われてきている．ここで信頼性のあるデータ通信とはソフトウェアや科学データの配信などの1バイトのエラーも許されないものを指す．通信メディアは広帯域となってきているが，ネットワークにおいて遅延やパケットロスが発生すると，従来のTCPでは帯域を使い切れないことがある．たとえばパケットロス率と往復遅延時間（RTT）が与えられたときに，TCPの理論的なスループットが計算されている [1]．この研究によればTCPのスループットは往復遅延時間が大きくなるか，パケットロス率が大きくなれば，急激に低下することになる．

通信メディアの帯域が大きくなるのに対し，往復遅延時間は地理的距離に支配されるため短縮が難しく，結果として帯域遅延積（BDP: Bandwidth Delay Product）は大きくなる傾向にある．こうした高BDPリンクにおいて高い性能を得るため，TCPそのものの改良，あるいは複数のTCP接続の利用について研究がなされている．後者の例としてGridFTP-APT（GridFTP with Automatic Parallelism Tuning）がある [2]．GridFTP-APTはネットワークリンク上に複数のTCP接続を確立し，それぞれの速度を観測しながら，動的にTCP接続の本数を調整する．しかしながら今日，一般的になってきている複数の通信メディアを同時に使う複数経路の環境は考慮していない．

一方，インターネット経由で多く利用されているアプリケーションとして動画視聴がある．動画視聴にはパケットロスを許容して低遅延を重視する形態もあるが，昨今では信頼性の高いTCP接続を利用したプログレッシブダウンロード [3] が利用者を増やしている．プログレッシブダウンロードは動画のデータをダウンロードしながら再生する技術である．プログレッシブダウンロードは特別なソフトウェアを導入しなくてもブラウザで動画が視聴可能であり，CDNと呼ばれるコンテンツ配信サービスとも相性が良いことから，普及が進んでいる．またHTTP Adaptive Streamingにより，帯域に応じて複数の品質の動画部分ファイルを用意しておき，適切なものを動的に選択することも可能である [4]．しかしながら，1本の経路では低い品質の再生レートですら十分に得られない場合がある．そこで複数の経路に1本ずつTCP接続を確立してプログレッシブダウンロードを実現する方式が提案されている [5]．この方式はHTTPパイプライン要求も活用して各TCP接続の性能を十分に活用することを目指しているが，前述のようにTCP接続1本だけでは，ある経路の帯域が十分に使い切れない場合がある．

そこで我々の研究グループでは，このプログレッシブダウンロードに対し，帯域・遅延・パケットロス率が未知の場合に利用可能な方式として，タイマの利用により定期的に要求を送信するタイマ駆動型要求方式を提案してきた [6]．ここで定期的な要求間隔は固定して考えていた．

本稿ではこれまで段階的な改良と評価しか行われてきていないタイマ駆動型要求方式 [7], [8] に体系的な評価を加え，新たな視点でも考察を加えることで，本方式の有効性を改めて示すことを目的とする．想定する利用環境としてはインターネットのような共有ネットワークであり，空き帯域を使い切って高い品質の動画再生サービスを提供することを目標とする．他のユーザへの影響や公平性についてはmultipath TCP [9] の研究でも議論されている未解決問題であるので，本研究においても今後の課題とする．

本稿の以下の構成を簡単にまとめる．2章では関連研究を紹介し，本研究の位置づけを明らかにする．3章では提案する要素技術を紹介し，評価対象とする方式を説明する．4章では提案する要求方式について，高画質の動画再生の実現可能性の面から評価する．最後に5章において結論と今後の課題について述べる．

2. 関連研究

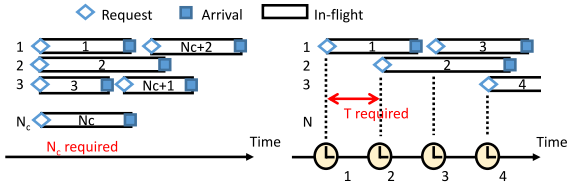
本章では，2.1節でTCPの問題に対処する関連研究を紹介し，2.2節でタイマ駆動型要求方式を導入したプログレッシブダウンロードを説明する．

2.1 TCPの問題に対処する研究

インターネットを通してファイルを取得する場合など，信頼性の保証が必要な場合はトランスポートプロトコルとしてTCPが使われてきた．しかしながら，TCPは1本の接続で運ばれるデータが完全に順序どおりであることを保証するため，パケットが損失した場合にその再送が完了するまで後続のパケットをアプリケーションに渡すことができないという問題（Head-of-Line（HoL）ブロッキング問題）が避けられない．この問題に対処するため，複数のTCP接続を確立してファイルを分割して取得されることが多い．複数のTCP接続を用いると，TCPそのものを修正することなく，TCP接続1本では使い切れない帯域の有効利用も可能になることがある．複数の経路を同時に利用するトランスポート層の技術としてはMultipath TCP [9]，SCTP [10]，SCTP-CMT [11] があげられるが，いずれもHoLブロッキング問題への対処が必要である．

また，複数のTCP接続を同時に用いた場合，適切な接続数を決定する必要がある．接続数が少なければ，すべての接続で合計してもネットワークの利用可能帯域を使い切れない可能性がある．一方，接続数が多いと，TCP接続がお互いに干渉し，合計のデータ転送速度がネットワークの利用可能帯域を下回る可能性がある．前述のGridFTP-APT [2] が提案されたものの，昨今の動画視聴環境，無線環境や，複数経路環境に対応させるための動的な接続数管理方式が必要である．

複数のTCP接続を用いてファイルをダウンロードする



(a) 受信駆動型 (b) タイマ駆動型

図 1 受信駆動型とタイマ駆動型の要求方式

Fig. 1 Reception-driven and Timer-driven requesting schemes.

場合，対象のファイルを細分して別々の接続から同時に部分ファイルを取得する分割ダウンロードが有効である．このとき，1 つの部分ファイルを受信すると，その空き接続に次の要求を割り当てる方式（受信駆動型要求方式）が提案されてきた [12]．受信駆動型の場合，要求したブロックが受信できた接続に対して次のブロックの要求を送信する動作を，すべてのブロックが受信できるまで繰り返す（図 1 (a) 参照）．ダウンロード開始時には，あらかじめ接続数 N_c を決めておき，すべての接続に順に要求を送信するのが一般的である．従来方式 [12] は経路ごとに事前に判明しているネットワーク条件，すなわち帯域遅延積およびパケットロス率から適切な TCP 接続数を得る式を提案している．この数式はさまざまな帯域，往復遅延，およびパケットロス率から近似によって帰納的に得られたものである．従来方式は複数の経路のそれぞれに適切な本数の TCP 接続を確立し，これらを同時に用いることで，プログレッシブダウンロードによる高画質の動画再生を目指す．しかしながら従来方式のネットワーク条件が既知であるという前提は現実的ではない場合も多く，特に昨今普及している無線リンクでは条件の変動もあって正確な把握が困難である．

2.2 タイマ駆動型要求方式

我々の研究グループでは複数の経路上で複数の TCP 接続を同時に用いてプログレッシブダウンロードを実現する一方式として，タイマ駆動型の要求方式を提案してきた [6]．ここでプログレッシブダウンロードは，動画ファイルを等分割し，再生順に番号づけられたブロックを取得すると仮定する．タイマ駆動型の要求方式 [6] は，パラメータとして要求発行間隔 T のみを必要とする（図 1 (b) 参照）．TCP 接続は 100 本など十分な数をあらかじめ用意しておき，接続を識別する番号を設定する．タイマにより定期的に要求を送信することとし，その時点で TCP 接続を識別子が小さいほうから順に走査し，最初にみつかった空き接続を利用する．なお複数の経路に対応する場合は，経路ごとにタイマ間隔を独立に設定できることを前提とする．タイマ駆動型ではブロックの要求機能とブロックの受信機能を独立して実装することができる．これらの擬似コードを図 2 に示す．要求機能は以下のように動作する．プログラム実行開始時に経路ごとに確立されたすべての接続を番号順に調べ，空きをみつけたらその接続を通してブロックを要求する．このとき次の要求対象ブロックを選ぶ方法によって順序制御が実現できる（図 2 (a) の 6 行目，`next_block_id` 関数に相当）．この要求機能は動的に変更可能な要求間隔ごとに実行する．一方，受信機能については，受信したブロックを受信済みとし，すべてのブロックが取得できていたらプログラムを終了する．要求の送信時刻を記録しておけば応答時間が分かるので，この応答時間を使って要求間隔を更新することができる（図 2 (b) の 6 行目，`update_intvl` 関数に相当）．

```
1  function req( path_id ) {
2      num = num_of_conn(path_id); #コネクション数
3      for (i=0; i<num; i++)
4          conn[i]が空きならbreak;
5      if (num < i) {
6          b = next_block_id( ); #次の要求対象ブロック
7          path_idのconn[i]でブロックbを要求;
8      } #空きコネクションがなければ送信しない
9      現在時刻＋送信間隔の時刻にreq( path_id)呼び出し;
10 }
```

(a) requesting function

```
1  function recv(path_id) {
2      b = recv_block_id( ); #受信ブロックID
3      r = update_comp_stat( b ); #受信済ブロック更新
4      if (r == COMPLETED) exit();
           #全ブロック取得でプログラム終了
5      intvl = block_resp_time(b); #応答時間
6      update_intvl( intvl ); #送信間隔更新
7  }
```

(b) receiving function

図 2 要求と受信に関する擬似コード

Fig. 2 Pseudo-code for requesting and receiving functions.

先行研究 [6] ではネットワーク条件に応じた動的なタイマ間隔設定法や，タイマ駆動型要求方式に適した順序制御法は十分に検討されていなかった．

3. 提案方式

本章では，複数の経路が異なる帯域，往復遅延，パケットロス率を持ち，一部共有するリンクを持つ場合と持たない場合の典型例を対象に，プログレッシブダウンロードの待ち時間を短縮し，かつ，より高品質の動画を視聴できる方式を提案する．

まず 3.1 節で提案方式を構成する要素技術について説明する．次に 3.2 節で評価対象とする方式について説明する．

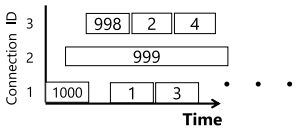
3.1 提案方式を構成する要素技術

本稿で提案する方式を構成する要素技術として，重複再要求機能（連続取得に基づく方式，連続性を問わない方式），経路ごとの動的要求間隔調整機能，初期冗長要求機能，初回限定逆順要求機能，要求ブロック ID 調整機能がある．それぞれについて以下で説明する．

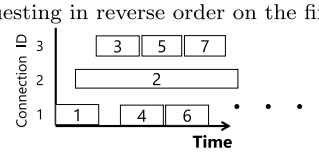
- 重複再要求機能
  特定の接続を通して要求したブロックが急な性能低下により取得に長時間を要する場合，後続のブロックを

取得していても動画を再生することができない．このため条件を満たしたら同じブロックを別の接続で要求する重複再要求機能を導入する．図 2 (a) の擬似コードでは 6 行目の `next_block_id` 関数において，最初の未要求ブロックではなく，要求済みで未取得のブロックのうち，条件を満たしたブロックの ID を返すことによって実現する．先行研究 [6] では，未取得ブロックの直後 20 個の連続した番号のブロックがすべて取得済みになった際，この未取得ブロックを次のリクエスト送信時に再度要求する機能を評価した．これを提案方式の一部に取り入れ，連続取得に基づく方式と呼ぶ．一方，既存研究 [12] では，ある未取得ブロックよりも後方の番号で取得されたブロックがあれば，その連続性は問わず，合計がしきい値（50 ブロック）を超えれば再要求する機能を評価していた．これを提案方式の一部に取り入れ，連続性を問わない方式と呼ぶ．

- 経路ごとの動的要求間隔調整機能 [7]
  ブロックの受信レートを観測し，これをタイマ間隔として採用する．ここでは安定した挙動となるように，経路ごとに過去 n_h 回の受信時刻を履歴として持ち，n_h 回の平均受信間隔を要求間隔として決定する．この履歴はブロックを 1 個受信するごとに更新する．履歴の個数については性能を左右する可能性があるが，適切な履歴数については今後の課題とする．
- 初期冗長要求機能 [7]
  順序どおりに再生するプログレッシブダウンロードにおいては，再生順の早いブロックの到達が全体の再生に与える影響が大きいにもかかわらず，ダウンロード開始直後は十分なフィードバック情報がないためネットワーク性能が未知である．そこで一定割合 r_i%のブロックを最初に 2 回ずつ要求し，グッドプットを犠牲にしても順序の逆転は発生しにくくなる効果を狙う．冗長要求する個数の適切な値についての検討は今後の課題である．
- 初回限定逆順要求機能 [8]
  それぞれの接続において最初に要求するブロックを再生順の最後から逆順とする（**図 3** (a) 参照）．動画ファイルの最初の部分は早期に再生を開始するために重要であるが，獲得できるスループットが未知である未使用接続に割り当てると，低速だった場合に再生開始が大幅に遅れることになる．この問題を解決するため，たとえば 1,000 分割のファイルであれば，1,000，999，998 の順で各接続の最初の 1 回の要求においてブロックを選択していく．比較的高速であることが判明した接続についてはブロック 1，2 を先頭から順に割り当てていくことにより，動画の開始部分が早期に取得できる可能性が高くなる．
  一方，従来は再生順にブロックを選択していたため，



(a) requesting in reverse order on the first time



(b) requesting in order

図 3 初回限定逆順要求機能（文献 [8] より引用）
Fig. 3 Requesting in reverse order on the first time (cited from Ref. [8]).

たとえばブロック 2 の取得が遅れた場合，ブロック 3 以降が取得できていてもすぐには再生できない状況に陥る（図 3 (b) 参照）．
初回限定逆順要求機能は図 2 (a) の 6 行目にある

```
b = next_block_id( )
```

を

```
if (conn[i] で初めての要求)
  b = reverse_block_id( )
else
  b = next_block_id( )
```

に変更する．`reverse_block_id` 関数ではブロック ID の末尾から逆順に番号を返すとともに，いままでに要求した最も若い番号を（`next_block_id` 関数とは別に）記憶する．なお初期冗長要求と目指すところが類似するため，初回限定逆順要求機能を有効にする際には初期冗長要求を無効にすることとした．
初回限定逆順要求機能を用いると動画の開始部分の取得が後回しになることで再生開始がやや遅れる可能性はあるが，特に共有リンクでは激しい競合により TCP のウインドウサイズを十分に拡張できない接続が発生するため，低速接続による再生開始の遅延を回避する効果のほうが重要であると考えた．また動画ファイルの末尾部分が再生終了に近い時間までバッファを占有し続けるという問題はあるが，再生済みのブロックを解放することでバッファ容量は確保が可能であると考えている．

- 要求ブロック ID 調整機能 [8]
  各接続の過去の応答時間より今後のブロック到達順序を予想し，あらかじめ要求するブロックの番号を動的に調整する．それぞれのブロック取得が完了した際，完了時刻から記録しておいた要求時刻を差し引くことで応答時間が計算できる．それぞれの接続で過去の応答時間を記録することにより，今後の応答時間を予測する．ブロック取得が完了した接続に対し次のブロックを割り当てる際，従来は未要求のブロックの中で最も再生順の早いブロックを選択していたが，順序逆転の発生をできるだけ避けるため，当該接続で次に取得

が完了する時刻よりも前に他の接続で取得できる個数を予測し，その個数だけ要求する未要求ブロックの番号を後方にずらす．たとえば**図 4** で接続 2 へ次に割り当てるブロックを選択する場合を考える．未要求で最も再生順の早いブロックは 7 であるが，次に接続 2 で取得するまでの間に接続 1 と接続 2 が 6 個のブロックを取得できると予想されるため，この個数を加えたブロック 13 を要求する．それぞれの接続の性能がしばらくの間，大きく変化しない状況であれば，この機能により到着ブロックの順序逆転は緩和されることが期待される．

提案方式はアプリケーション層の対応のみで実現できるという長所を有する．したがって OS の更新をせずに，アプリケーションのインストールのみで実装が可能である．一方，従来の TCP における HoL ブロッキング問題を解決する方法としては TCP の再送制御アルゴリズムを改良するものや，新たなトランスポートプロトコルを提案するものがあるが，OS の修正が必要なため普及に難がある．

3.2 評価対象の方式

比較対象となる従来方式と，本研究で提案する方式 1，2，3 を以下に紹介する．

- 従来方式（Existing）[6]
  動画ファイルの分割されたブロックはすべての経路において同一の固定間隔で要求される．すなわち図 2 (a) 9 行目の送信間隔は固定であり，図 2 (b) 6 行目の update_intvl 関数は無効化される．もし未取得のブロックがあり，その直後から再生順に連続して c_d 個のブロックが取得済みとなるならば，その未取得ブロックは他の接続から重複して要求する（連続取得に基づく重複再要求機能）．
- 提案方式 1（Proposal 1）[7]
  既存方式の固定要求間隔ではなく，経路ごとの動的要求間隔調整機能を用いる方式である．
- 提案方式 2（Proposal 2）[7]
  提案方式 1 の重複再要求機能を連続性を問わない方式に変更する．未取得ブロックより再生順後方に，連続かどうかにかかわらず d_d 個以上の取得済みブロックがあれば，この未取得ブロックを他の接続からも重複して要求する．また全ブロック数の r_i%に相当する数について初期冗長要求機能を適用する [7]．
- 提案方式 3（Proposal 3）[8]
  提案方式 3 は提案方式 2 の初期冗長要求機能のかわりに初回限定逆順要求機能を利用し，新たに要求ブロック ID 調整機能を含めた方式とする [8]．

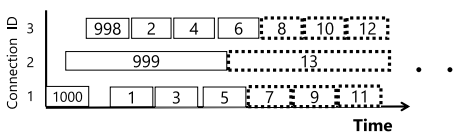


図 4 要求ブロック ID 調整機能（文献 [8] より引用）

Fig. 4 Adjust the ID of next block to be requested according to the history of response times (cited from Ref. [8]).

4. 実験結果と考察

本章では，まず 4.1 節で実験環境と評価対象となる方式について説明する．次に 4.2 節で互いに素な 2 経路において従来方式と提案方式を比較評価する．さらに 4.3 節で共有リンクを持つ 2 経路における評価を示す．最後に 4.4 節において関連する問題点について議論する．

4.1 実験環境

図 5 に ns2 [13]（バージョン 2.35）を用いて実施したシミュレーション実験のネットワークトポロジを示す．2 本の経路を Path 1，および Path 2 とし，それぞれ帯域を 10 Mbps，および 4 Mbps，往復遅延を 100 ms，および 12 ms とした．ただし図 5 (c) のみはリンクにおける 0 ms の遅延を避けるため，0.001 ms の遅延が付加されている（往復では 0.002 ms）．ここで Path 1 は，地理的に近接した場所に設置されたサーバに対し，ローカルに設置された WiFi アクセスポイント経由で接続する経路を想定しており，Path 2 は LTE 接続でインターネットを経由し同じサーバに到達する経路の想定である．Path 2 では時間帯などの条件によりさまざまな性能で接続することになると考えられるため，パケットロス率を 0.5%，および 1%から 10%まで 1%刻みに変更して実験する．ここでパケットロス率を変更するのは，パケットロス率が高いほど速度低下を経験する TCP 接続が増えることにより全体的な速度低下や順序逆転が発生しやすくなるため，高品質の動画再生にとって不利な環境でも方式が効果的であるかを確認したいからである．一方，Path 1 は安定していることを想定し，パケットロス率を 0.5%とした．ここで簡単のためパケットロスはサーバからクライアントの方向にのみ発生するとし，パケットロス率はシミュレーション実験の開始から終了まで一定であるとする．

実験パラメータは**表 1** のとおりである．ファイルはソ

表 1 実験パラメータ

Table 1 Parameters for download experiments.

Simulator	ns2 (ver. 2.35)
TCP version	Reno [14]
ファイルサイズ (MB)	20
ブロックサイズ (KB)	20
分割ブロック数	1,000
パイプライン要求数	1
試行回数	10

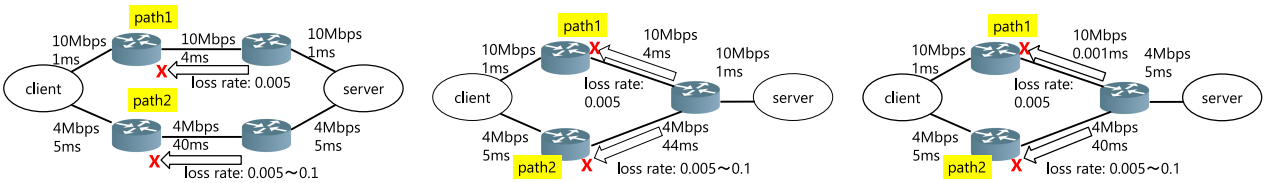


図 5 シミュレーションに用いたネットワークトポロジ．(a) 互いに素，(b) 10 Mbps リンクを共有，(c) 4 Mbps リンクを共有（文献 [8] より修正引用）

Fig. 5 Simulated network topology. (a) disjoint, (b) sharing broad link, (c) sharing narrow link (adapted from Ref. [8]).

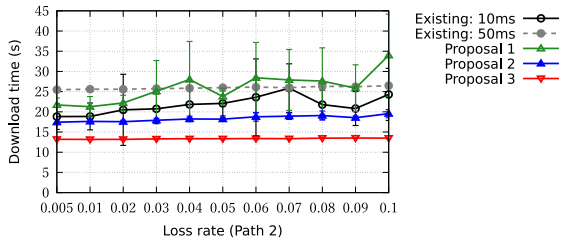


図 6 各方式の平均ダウンロード完了時間

Fig. 6 Download completion time for each method.

フトウェアなどで一般にみられるサイズとして 20 MB とした[*1]．このファイルを 1,000 分割して，プログレッシブダウンロードを実行する．HTTP による要求と応答を想定し，パイプライン要求は行わず，1 つの要求に応答が来るごとに次の要求を送出する．シミュレーションは各場合に 10 回ずつ試行し，平均を算出する．ns2 で用いる TCP のバージョンは基本的な評価をめざして Reno [14] とした．

評価対象とする方式について，従来方式では固定の要求間隔を 10 ms または 50 ms とする．従来方式と提案方式 1 で用いられる重複再要求機能のパラメータ c_d は 20 とする．提案方式 3 種については動的に要求間隔を変更するが，要求間隔の初期値は 10 ms としている．到着間隔の履歴数 n_h は 100 とし，提案方式 2，3 では n_h 個の履歴が得られていない初期段階でもその平均値を用いる．提案方式 2 および 3 の重複再要求機能のしきい値 d_d は 50 とした．初期冗長要求の割合 r_i は 10（%）としているため，1,000 分割のブロックのうち最初の 100 個を冗長に要求する．

4.2 互いに素な 2 経路における評価

本節では図 5 (a) に示される互いに素な 2 経路において実験を行った．**図 6** に各方式によるダウンロードにかかった時間を 10 回試行の平均として評価した結果を示す．横軸は Path 2 のパケットロス率を表しており，縦軸はそのときの平均値である．エラーバーは標準偏差を示しているが，マーカに隠れるほど小さなデータも含まれる．以降のグラフではすべて同様のエラーバーを表示している．

[*1] これより大きな動画でもブロックサイズが同じならば個数が増えるのみであるため，TCP が定常状態に入るために十分な大きさを想定した．

図 6 より，従来のタイマ駆動間隔を固定値とする方式では，適切な値である 10 ms に設定したとき（Existing: interval 10 ms）は高速であるものの，長い値（50 ms）に設定してしまうと安定して所要時間が大きくなることが分かる．一方，提案方式 1（Proposal 1）では従来の固定間隔 50 ms に設定した場合と同等程度の性能しか得られていない．ただし，適切な間隔は事前に知らされていない不利な条件であることに注意が必要である．提案方式 2（Proposal 2）では同じ不利な条件ながら，従来の固定間隔の 10 ms の場合とほぼ同等の性能が安定して得られている．要求間隔を到着間隔から見積もる点は提案方式 1 および 2 で共通しているが，提案方式 2 のほうが重複再送要求を頻繁に発行することで，より早くブロックを集めることができたと考えられる．さらに提案方式 3（Proposal 3）では，より短いダウンロード時間が実現できている．これは提案方式で頻繁に発行した重複再送要求を初回限定逆順要求機能と要求ブロック ID 調整機能によって起動しにくくしたためであると考えられる．重複再要求が減少したことは後に紹介する 4.3 節の表 3 でも確認できる．

ダウンロード時間を支配する要因について考察するため，各方式について全ブロックの 95%（本実験では 1,000 ブロック中 950 ブロック）を取得するまでの時間を計測した．これらの値は Path 2 のパケットロス率によってはあまり変化しないことが分かったので，図 6 の横軸に相当する 0.005〜0.1 の 11 例について集計し，全体の平均と標準偏差として**表 2** に示す．表 2 のトポロジ (a) より，提案方式 2 は初期冗長要求の影響もあり，95%のブロックを取得するまでの時間が長くなっている．それにもかかわらず既存方式や提案方式 1 より全ブロックのダウンロード時間が短くなっているのは，頻度の高い重複再要求により，取得の遅れたブロックを高速な接続から確保できたことによると考えられる．一方，提案方式 3 は 95%のブロックを取得する時間を短縮し，取得の遅れたブロックの確保も早く済んでいると考えられる．

図 6 の結果より，提案方式 1 はパケットロス率が高いほどダウンロード時間が長くなる傾向があるが，他の方式においてはこのような傾向は顕著でない．これは提案方式 1 において重複再要求を実行する条件が厳しいため，パケッ

表 2 全ブロックの 95%を取得するまでの各方式の平均所要時間と標準偏差（カッコ内）（トポロジ (a)，(b)，(c)）

Table 2 Average and standard deviation of required time (s) to fetch 95% of all the blocks for each method (topology (a), (b), (c)).

方式 ＼ トポロジ	(a)	(b)	(c)
既存方式	12.65	17.03	41.40
（10 ms）	(0.14)	(0.19)	(0.60)
既存方式	24.10	24.10	41.69
（50 ms）	(0.10)	(0.09)	(0.67)
提案方式 1	12.80	17.23	42.19
	(0.12)	(0.17)	(0.54)
提案方式 2	15.72	25.04	58.77
	(0.05)	(0.17)	(1.73)
提案方式 3	12.50	18.4	48.86
	(0.11)	(0.17)	(0.95)

トロスにより発生した低速な接続で要求したブロックを，取得完了するまで待ち続けてしまうためであると考えられる．実際，表 2 より全ブロックの残り 5%の取得に長い時間が占められていることが分かる．各 TCP 接続の速度低下の度合いはまちまちであると考えられるので，ダウンロード時間が不安定になる一因ともなっている．また再要求が少ないため，終了が近くなるほど到着間隔が開き，要求間隔も長くなる傾向にある．一方，提案方式 2 や提案方式 3 は，低速な接続で要求したブロックを早めに別の高速な接続で取得できるため，パケットロス率によって大きな影響を受けていないと考えられる．

なお 2 本の経路の合計帯域である 14 Mbps を完全に使い切った場合のダウンロード時間は $20{,}000{,}000 \times 8\,(bits)/14{,}000{,}000\,(b/s)$ より，11.4 s である．また既存研究 [12] では同じ環境で約 13 s のダウンロード時間を達成しているが，これはネットワーク条件をあらかじめ知っているという有利な前提である．

次に，各方式の順序逆転について評価する．ここでは初期バッファリング時間を定義し，これを評価尺度とする．初期バッファリング時間とは，指定された再生レートでフリーズすることなく再生できる最も早い開始時間を指し，この時間だけ最初にバッファリングすればスムーズな動画再生が実現できていたという目安である．たとえば**図 7** では横軸に時間，縦軸に再生順に割り振ったブロック ID をとり，各ブロックの到達時刻をプロットしている．すべてのプロットに指定された再生レートの直線を重ねたとき，最も大きな X 切片が初期バッファリング時間である．

図 8 は各方式の初期バッファリング時間を 10 回試行の平均で評価した結果である．図 8 (a)，(b)，(c) はそれぞれ，再生レートを 100 KB/s，1 MB/s，10 MB/s とした場合である．たとえば 100 KB/s は 800 Kbps に相当し，これは YouTube の 480 p 動画に近い値である．提案方式 1 の

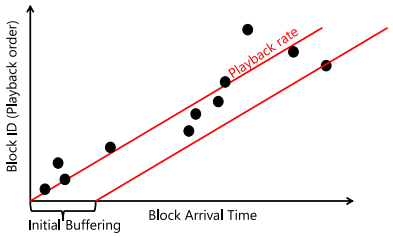
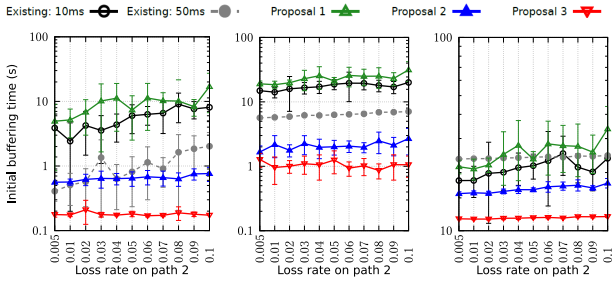


図 7 初期バッファリング時間

Fig. 7 Definition of initial buffering time.



(1) 100 KB/s (2) 1 MB/s (3) 10 MB/s

図 8 各方式の初期バッファリング時間（トポロジ (a)）

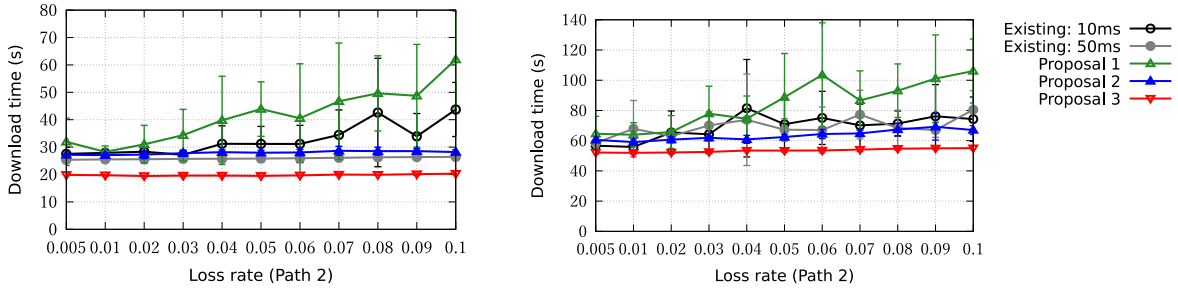
Fig. 8 Initial buffering time for each method (topology (a)).

初期バッファリング時間が長く，不安定になる傾向にあるのは，図 6 と同様に低速 TCP 接続に割り当てられたブロックの到着が遅れていることによると考えられる．従来方式のいずれに比べても，また提案方式 1 および 2 と比べても，提案方式 3 を用いることにより，短い初期バッファリング時間を達成している．また図 6 より，提案方式 3 はダウンロード時間が短い．したがって提案方式 3 は図 6 と図 8 から分かるように高速にダウンロードできるだけでなく，順序制御も良好に働き，初期バッファリング時間が短くて済むといえる．

4.3 共有リンクを持つ 2 経路における評価

本節では複数の経路が共有しているリンクについて注目するため，いままで検討したネットワークトポロジ（図 5 (a)）に加え，異なる帯域の共有リンクを持つ 2 つのネットワークトポロジ（図 5 (b) および (c)）を考慮する．これらの帯域と遅延の条件を考慮するにあたり，3 つのネットワークトポロジがなるべく同じ条件となるように設定したが，比較的狭い共有リンクを設定したトポロジ (c) では Path 1 の帯域は Path 2 と同じ 4 Mbps がボトルネックとなる．既存研究 [12] ではネットワーク条件は事前に個別の経路ごとに計測されて与えられていたが，このような共有リンクではネットワーク条件が正確に把握できない可能性がある．ここでは共有リンクの影響について検討する．これらのトポロジは典型的な想定であり，一般的なネットワークへの適用可能性は今後の課題である．

本節では各方式を対象とし，共有リンクの影響を評価する．まずすべての部分ファイル（ブロック）を取得するま



(b) (c)

図 9 トポロジ (b) 広帯域リンクの共有 (c) 狭帯域リンクの共有，における各方式のダウンロード時間

Fig. 9 Download completion time for each method (b) sharing broad link (c) sharing narrow link.

表 3 各方式の延べ冗長取得ブロック数の平均値と標準偏差（カッコ内）（トポロジ (a)，(b)，(c)）

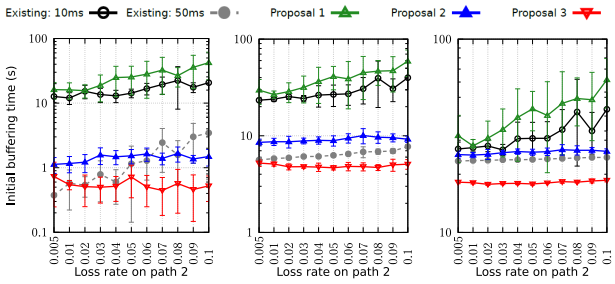
Table 3 Average number and standard deviation (in parentheses) of redundant block receptions for each method (topology (a), (b), (c)).

方式 ＼ トポロジ	(a)	(b)	(c)
既存方式	2.86	2.52	2.76
（10 ms）	(0.57)	(0.60)	(0.38)
既存方式	12.21	12.14	3.77
（50 ms）	(4.63)	(4.63)	(0.61)
提案方式 1	19.98	19.93	19.95
	(0.04)	(0.11)	(0.07)
提案方式 2	345.24	488.59	381.96
	(10.50)	(9.32)	(15.66)
提案方式 3	68.79	107.06	192.77
	(2.88)	(7.65)	(4.49)

での時間であるダウンロード時間はこれらの方式の有効性を示す尺度であると考えられる．図 5 (b)，(c) の各トポロジについて，複数の経路のボトルネック帯域はそれぞれ 10 Mbps，4 Mbps であり，20 MB のデータを取得する理想的な所要時間は 16 s，40 s である．

図 9 は共有リンクを持つ 2 つのネットワークトポロジ（図 5 (b)，および (c) 参照）において，各方式について，10 回試行したシミュレーション実験の平均ダウンロード時間を示す．図 9 より，共有リンクを持つトポロジにおいても提案方式 3 は短い時間でダウンロードを完了できていることが分かる．この主な理由は順序逆転を回避することで，重複再要求とそれにともなう冗長受信が削減できているためと考えられる．

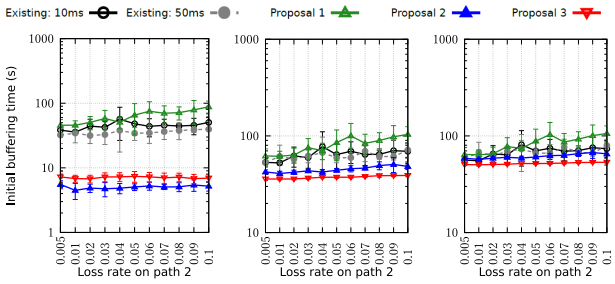
表 3 は各方式において冗長に取得された延べブロック数の平均値を示す．この値は異なるパケットロス率においても大きな変動がみられなかったため，Path 2 の各パケットロス率についての 10 回試行平均値を，図 6 の横軸に相当する 0.005～0.1 の 11 例について集計し，全体の平均と標準偏差として求めたものである．表 3 より，提案方式 3 の冗長受信ブロック数は提案方式 2 の 1/5 から 1/2 程度となっており，帯域の浪費が抑えられていることが分かる．



(1) 100 KB/s (2) 1 MB/s (3) 10 MB/s

図 10 各方式の初期バッファリング時間（トポロジ (b)）

Fig. 10 Initial buffering time for each method (topology (b)).



(1) 100 KB/s (2) 1 MB/s (3) 10 MB/s

図 11 各方式の初期バッファリング時間（トポロジ (c)）

Fig. 11 Initial buffering time for each method (topology (c)).

図 9 (b) および (c) においても図 6 と同様，提案方式 1 のダウンロード時間はパケットロス率が高いほど，長くなる傾向にある．この理由はトポロジ (a) と同様と考えられる．

図 10 にトポロジ (b) における各指定再生レートの初期バッファリング時間を示す．また**図 11** にトポロジ (c) における同様の結果を示す．図 10 および図 11 より，いずれの送信レートにおいても，提案方式は既存方式よりも短いか同等の初期バッファリング時間を達成できることが分かる [8]．

4.4 議論

高い帯域遅延積のネットワークを克服する手段としては，TCP のウインドウ制御そのものを改良するアプローチもある [15]．しかしながら一般的に動画のプログレッシ

ブダウンロードのアプリケーションは軽微な順序逆転に対応できるバッファ機構を備えるにもかかわらず，TCP は厳密に First In First Out の厳密な順序制御を実現する．これは Head-of-Line（HoL）ブロッキング問題を引き起こしやすいので，複数の独立したデータを並行して送信する場合は TCP 接続を複数本確立することが望ましい．MMCFTP [16] も多数の TCP 接続を確立するアプローチだが，送信側，すなわちサーバ側でノンブロッキング I/O としてソケット API を利用して実現している．TCP による HoL ブロッキング問題を回避するアプローチとしては，UDP の上に構成される QUIC [17] も注目を集めている．

本稿のシミュレーションは無線を想定しているリンクにおいても，有線リンクに固定の帯域，遅延，およびパケットロスを設定して模擬している．無線 LAN などのメディアアクセス制御を模した実験は今後の課題である．なお TCP の複数接続についての研究 [18] において，IEEE802.11g 無線 LAN を考慮して TCP の複数接続を評価した結果，有線リンクの模擬と大差ない性質が得られている．

本稿の実験ではダウンロードの開始から終了まで，ネットワーク環境は変化しないことを想定している．しかしながら実際のネットワークでは変化が考えられる．従来の受信駆動型や既存方式では接続数や要求間隔といった値を固定で持つ必要があったため，変化前のネットワークで適切に設定されていても，変化後は適切な値に調整することができない．一方，本稿で提案する動的要求間隔調整機能は受信ブロックの到着レートによって要求間隔を変更できる．実験結果からも分かるように，たとえば帯域，往復遅延，およびパケットロス率が異なる 2 つのネットワーク環境で前半 20 MB，後半 20 MB のダウンロードを行った場合，それぞれで短いダウンロード時間と短い初期バッファリング時間が達成されるため，40 MB 全体の動画視聴では変化のある環境に対応できていると見なせる可能性は高くなる．ただし頻繁な条件の変更に対しては，追従するための時間が必要になるため，この点の定量的評価は今後の課題としたい．

本稿の実験では重複再要求のパラメータとして，$c_d = 20$ と $d_d = 50$ を採用してきた．c_d は文献 [6] でいくつかの値から選ばれたが，本実験の環境では適切とは限らない．そこで $c_d = 5, 10, 30$ の場合について提案方式 1 のダウンロード時間を評価した．Path 2 の各パケットロス率についての 10 回試行平均値を，図 6 の横軸に相当する 0.005〜0.1 の 11 例について集計し，全体の平均と標準偏差を求めた結果を**表 4** に示す．表 4 より，差は標準偏差より小さく有意とはいえない．このことから，提案方式 1 が $c_d = 20$ を採用したことで不当に低い性能となっている可能性は大きくないと考えられる．一方，提案方式 2 と提案方式 3 で採用している $d_d = 50$ は表 3 で分かるように重複再要求を多めに発行するが，提案方式 3 では初回限定逆順要求機能と要求ブロック ID 調整機能により重複再要求の原因となる順序逆転自体を減らすことができており，これ以上のチューニングは不要と考えている．

表 4 提案方式 1 で c_d を変更したときのダウンロード時間 (s) の平均値と標準偏差（カッコ内）（トポロジ (a)，(b)，(c)）

Table 4 Average and standard deviation (in parentheses) of download time (s) for each c_d value in Proposal 1 (topology (a), (b), (c)).

c_d ＼ トポロジ	(a)	(b)	(c)
30	25.26 (2.68)	39.99 (6.80)	84.63 (17.23)
20	25.98 (3.58)	41.53 (9.55)	84.17 (15.12)
10	27.55 (6.00)	39.70 (9.38)	83.65 (15.82)
5	25.89 (2.86)	41.13 (9.04)	85.55 (15.23)

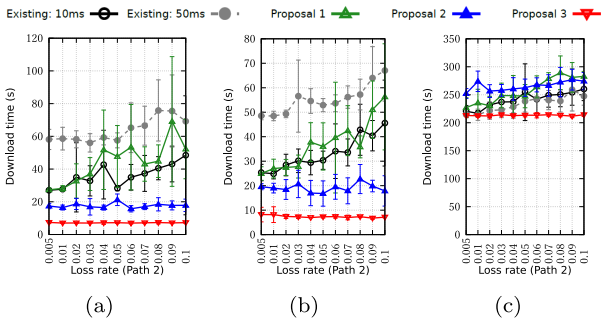


図 12 トポロジ (a) 互いに素 (b) 広帯域リンクの共有 (c) 狭帯域リンクの共有，における各方式のダウンロード時間（Path 1: 100 Mbps，Path2: 1 Mbps）

Fig. 12 Download completion time for each method (a) disjoint (b) sharing broad link (c) sharing narrow link (Path 1: 100 Mbps, Path2: 1 Mbps).

4.2 節および 4.3 節の実験では 10 Mbps と 4 Mbps の 2 経路を想定していたが，帯域の差がより大きい場合については考察していなかった．ここで図 5 (a)〜(c) のトポロジはそのままに，Path 1 の帯域を 100 Mbps，Path 2 の帯域を 1 Mbps とした変更のみを適用した場合の追加実験の結果を示す．**図 12** はダウンロード時間を比較する．帯域の差が大きい場合においても提案方式の優位が確認された．ただしトポロジ (a) と (b) では合計帯域が 101 Mbps と 100 Mbps であり，20 MB をダウンロードする時間は計算上，1.58 s と 1.6 s である．2 経路を扱うことで処理が複雑になることを考えると広帯域の 1 経路のみを利用し狭帯域の経路は利用しないことも考えられる．一方，トポロジ (c) では 1 Mbps が有効のため，計算上の所要時間は 160 s である．どのトポロジでも計算値より長くなっており，これは帯域の狭い経路に多数の TCP 接続を確立することで資源の競合が激しくなっているためであると考えられる．また**図 13** で指定レートごとの初期バッファリング時間を

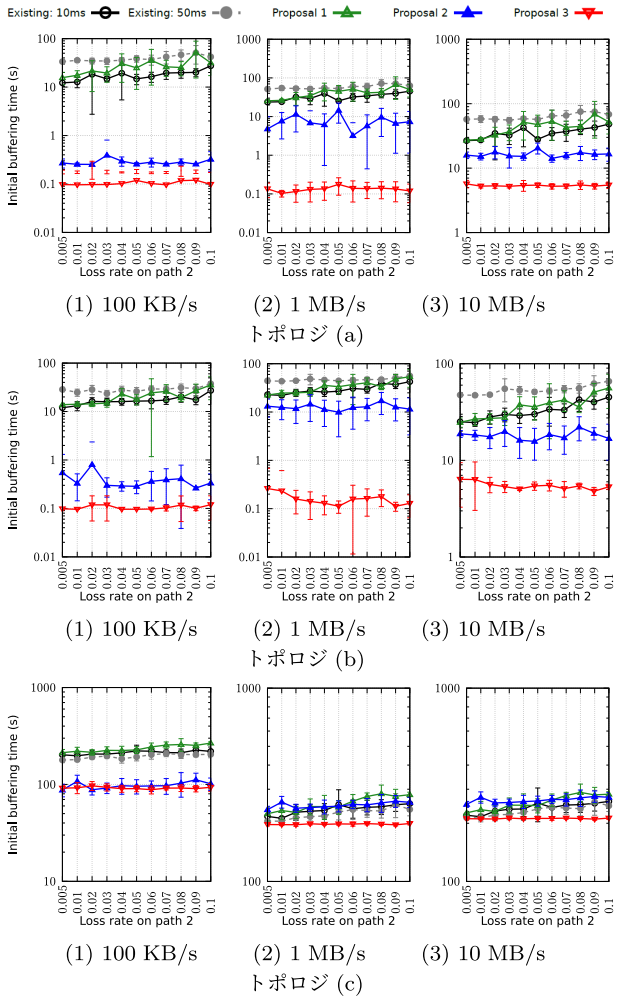


図 13 各方式の初期バッファリング時間（トポロジ (a)，(b)，(c)）（Path 1: 100 Mbps，Path2: 1 Mbps）

Fig. 13 Initial buffering time for each method (topology (a), (b), (c)) (Path 1: 100 Mbps, Path2: 1 Mbps).

比較する．初期バッファリング時間においても提案方式 3 が優位であるといえる．なお合計帯域が 100 Mbps 程度であるトポロジ (a) と (b) では，10 MB/s（80 Mbps）以下の指定レートで待ち時間がほとんどないのが理想であり，提案方式 3 は理想に近いと考えられる．一方，トポロジ (c) ではいずれの方式のいずれの指定レートでも初期バッファリング時間が 100 s 程度以上となった．これはボトルネック帯域が 1 Mbps のように狭い場合，2 経路のそれぞれで確立した多数の TCP 接続が集中することによる競合により各接続の速度が不安定になり，取得の遅れるブロックが多数発生するからである．ここでは例を増やしたにすぎないが，提案方式の適用可能なネットワーク条件については今後の研究で明らかにする必要がある．

5. おわりに

本稿では複数の経路の上で複数の TCP 接続を確立する方式において，タイマ駆動により要求を発行するプログレッシブダウンロード方式を提案した．比較評価の結果，提案方式により，従来の固定要求間隔を用いる方式と同等以上の高い品質の動画が短い初期バッファリング時間で再生できる可能性があることを示した．

今後の課題として，提案方式は 3 章の冒頭で示した想定以外の環境でも有効に働く可能性があるため，その適用領域を明らかにすることがある．また実装により実環境においても評価する必要がある．

謝辞 本研究の一部は，公益財団法人フジクラ財団研究助成の支援により行われた．

参考文献

[1] Padhye, J., Firoiu, V., Towsley, D. and Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation, *Proc. ACM SIGCOMM* (1998).
[2] Ito, T., Ohsaki, H. and Imase, M.: GridFTP-APT: Automatic Parallelism Tuning Mechanism for GridFTP in Long-Fat Networks, *IEICE Trans. Commun.*, Vol.E91-B, No.12, pp.3925–3936 (2008).
[3] Ma, K.J., Bartos, R., Bhatia, S. and Nair, R.: Mobile Video Delivery with HTTP, *IEEE Communications Magazine*, Vol.49, No.4, pp.166–175 (2011).
[4] Sodagar, I.: The MPEG-DASH Standard for Multimedia Streaming over the Internet, *IEEE Multimedia*, Vol.18, No.4, pp.62–67 (2011).
[5] Kaspar, D., Evensen, K., Engelstad, P. and Hansen, A.F.: Using HTTP Pipelining to Improve Progressive Download over Multiple Heterogeneous Interfaces, *2010 IEEE International Conference on Communications (ICC)*, pp.1–5 (2010).
[6] Horiba, H., Hiraoka, T. and Funasaka, J.: A Progressive Download Method Based on Timer-Driven Requesting Schemes Using Multiple TCP Flows on Multiple Paths, *Proc. 37th IEEE ICDCS Workshops*, pp.26–31 (2017).
[7] Funasaka, J.: Adaptability Enhancement for Progressive Download Methods Based on Timer-Driven Requesting Schemes Using Multiple TCP Flows on Multiple Paths, *Proc. 4th IEEE SmartWorld*, pp.1605–1610 (2018).
[8] Funasaka, J.: Evaluation on Progressive Download Methods Based on Timer-Driven Requesting Schemes on Multiple Paths with Shared Links, *Proc. 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, pp.14–20 (2020).
[9] Ford, A., Raiciu, C., Handley, M. and Bonaventure, O.: TCP Extensions for Multipath Operation with Multiple Addresses, IETF RFC6824 (2013).
[10] Stewart, R. (Ed.): Stream Control Transmission Protocol, IETF RFC4960 (2007).
[11] Iyengar, J.R., Amer, P.D. and Stewart, R.: Concurrent Multipath Transfer Using SCTP Multihoming over Independent End-to-End Path, *IEEE/ACM Trans. Netw.*, Vol.14, No.5, pp.951–964 (2006).
[12] Hiraoka, T. and Funasaka, J.: A Progressive Download Method Using Multiple TCP Flows on Multiple Paths, *Proc. BWCCA2015*, pp.318–324 (2015).
[13] ns2 project: The Network Simulator – ns-2 (online), available from ⟨http://www.isi.edu/nsnam/ns⟩ (accessed 2021-08-23).
[14] Allman, M., Paxson, V. and Blanton, E.: TCP Congestion Control, IETF RFC5681 (2009).
[15] Caini, C. and Firrincieli, R.: TCP Hybla: A TCP Enhancement for Heterogeneous Networks, *International*

Journal of Satellite Communications and Networking, Vol.22 (2004).
[16] 山中，阿部，漆谷：アプリケーションによる適応型トラフィック分散方式，信学技報，NS2015-239, pp.411–415 (2016).
[17] Cui, T. et al.: Innovating Transport with QUIC: Design Approaches and Research Challenges, *IEEE Internet Computing*, pp.72–76 (2017).
[18] Funasaka, J., Obata, H. and Ishida, K.: Number of TCP Connections to Saturate Bandwidth of Wireless Networks, *Proc. 12th International Conference on Mobile Computing and Ubiquitous Networking* (*ICMU 2019*), 6pages (2019).

舟阪 淳一 （正会員）

1970年生．1993年東北大理学部卒業．1995年同大学大学院理学研究科博士前期課程修了．1997年奈良先端科技大学院大情報科学研究科博士前期課程修了．1999年同後期課程修了．同年広島市大情報科学部助手．2007年同大学大学院情報科学研究科講師．2009年より同准教授．博士（工学）．インターネットにおける効率的なデータ交換方式の研究に従事．本会2019年度論文賞受賞．IEEE，電子情報通信学会各会員．