情報処理学会論文誌 Vol.60 No.2 469–478 (Feb. 2019)

PR-SCTPを用いた分割ダウンロード方式における所要時間とブロック到達順序を考慮した要求方式

武田 和也[1,a)] 舟阪 淳一[1,b)]

受付日 2018年5月1日, 採録日 2018年11月7日

概要：パケット損失が頻発する劣悪なネットワーク環境において，バックオフによる待ち時間を低減させるPR-SCTPによるTTL制限つき再送機構とHead-Of-Lineブロッキングを緩和する順不同配送を組み合わせたダウンロード方式が提案されている．しかしながら，これまでに提案された方式ではブロックの到達順序について考慮していなかった．本稿では，HTTPストリーミング再生など順序を重視するアプリケーションでの利用も考え，再送する順序をまったく考慮しない再要求遅延方式と確率的に考慮する確率的再要求遅延方式を比較対象としてダウンロード所要時間と到達順序について評価した．実機とテストベッド環境において評価した結果，従来方式は到達順序において逆転が発生する可能性が低いが，バックオフ時間の上限を決めるTTLがサーバにおいて大きく設定されていると，ダウンロード時間が長くなることが分かった．また提案した再要求遅延方式と確率的再要求遅延方式はパラメータαによって順序逆転の度合いを調節できること，および再要求遅延方式は比較的短い時間で安定してダウンロードできることが分かった．

キーワード：PR-SCTP，分割ダウンロード，順不同配送，高ロス率ネットワーク

Requesting Schemes Considering Required Time and Arrival Order of Blocks for Segmented File Downloading Methods Using PR-SCTP

Kazuya Takeda[1,a)] Junichi Funasaka[1,b)]

Received: May 1, 2018, Accepted: November 7, 2018

Abstract: To yield better performance on highly lossy networks, a segmented file download method composed of both Time-to-Live (TTL) based PR-SCTP which limits the retransmission time and unordered delivery which alleviates the influence of Head-Of-Line blocking problem has been proposed. However, the existing method did not take the arrival order of blocks into account. In this paper, since the method can be applied to order-sensitive applications such as HTTP streaming, we evaluate the occurrence of out-of-order arrivals as well as the download time. Two methods, the delayed retrying method (DR) which does not worry the arrival order at all and the stochastic delayed retrying method (SDR), are proposed to be compared to the existing method. From the results obtained by the emulated network with real machines and prototype programs, the existing method rarely suffers from out-of-order arrivals while it requires longer time if the TTL, which limits the back-off duration, is set to a large time. It is also confirmed that SDR can adjust the possibility of out-of-order arrivals and DR can yield a stable performance to download a file in a short time.

Keywords: PR-SCTP, segmented file download, unordered delivery, highly lossy network

1. はじめに

IoT（Internet of Things）の概念が普及したことにより，さまざまな通信端末が有線だけでなく無線のリンクを通してもネットワーク接続することが一般的になった．この

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

際，特に無線リンクを利用すると，送受信端末の位置関係によりフェージングなどの影響を受け，パケット損失が発生することがある．このような環境で従来どおりのネットワークサービスをいかに提供するかが課題となっている．

インターネット経由でファイルを送信する場合など，1バイトも損失が許されない信頼性の高い通信では，一般的にトランスポートプロトコルとしてTCPが使われてきた．TCPは有線環境を想定して開発されてきたプロトコルであり，無線環境では性能低下につながる以下の問題が指摘されてきている．

(1) 損失パケットの再送があるまで後続が待たされるHead-Of-Line（HOL）ブロッキング問題
(2) バイナリバックオフにより再送タイムアウト時間が倍増するという問題
(3) パケット損失によりウインドウサイズが減少するという問題

問題(1)は複数の独立なデータを同時に送りたい場合には不要な待ち合わせの原因となる．従来，Webページにおける多数の埋め込み画像を同時に送りたい場合には，複数のTCP接続を用いてきた．しかしながら，単一のサービスのために複数のTCP接続を用いると，TCPポート番号が大量に同時使用されるためメモリ資源を浪費する．たとえばWebブラウザFirefox[*1]では同一サイトに対して最大で6本の並列接続を確立するため，この数をxとしクライアント数をnとすると，nxのポートが同時使用される．また複数のTCP接続が独立して輻輳制御を行って帯域を確保するため，1本のTCP接続がスループットtを獲得できる条件であれば，n本を利用するサービスはntの帯域を占有するのに対し，1本しか利用しないサービスはtの帯域しか利用できないため，帯域利用において不公平が生じる．問題(2)は輻輳発生時に送出レートを下げる機構として有効であるが，無線環境においてデータリンクの問題でパケットが損失した場合には，不要なレート低下となる．問題(3)も送出レートを下げる機構であるが，帯域確保TCP[1]を用いた解決策が多数，提案されてきている．

ファイルを高速に送信するため，ファイルを分割して並列に扱う手法も有効である．BitTorrent[2]に代表されるP2Pファイル分割共有では，ファイルを部分ファイルに分割し，複数のピアから同時に並行して取得することで，高速化を実現している．またDASH[3]に代表される適応型のHTTPプログレッシブダウンロードでは，動画の各部分ファイルに複数の異なる品質を用意し，帯域に応じて再生レートを変更可能である．このように，分割ダウンロードと総称される技術は開発・利用されてきているが，無線環境における性能評価はいまだ研究途上にある．

上述の問題(1)と問題(2)を解決するため，筆者らはPR-SCTPによる再送タイムアウト時間制限と順不同配送（unordered delivery）を用いた分割ダウンロード方式を提案してきた[4], [5]．PR-SCTP[6]は従来の高信頼トランスポートプロトコルと同様にタイムアウト時間を倍増させるが，損失したパケットが届くまで再送を繰り返すわけではない．たとえばRFC3758で*timed reliability*として定義された機構では，パケット単位で再送を繰り返す時間に制限を設けられる．この制限時間をTTL（Time to Live）と呼び，再送をあきらめたパケットはクライアント側からアプリケーションにより再送要求することで，ファイルの完成を保証する[7]．一方，順不同配送は同一コネクション内で送信されるパケットの順序を保証しない．前述のHOLブロッキング問題が緩和されるため，高速なファイル取得が実現される[5]．

分割ダウンロードの利用形態において，ファイルを完全に取得してから利用する場合は，分割された部分ファイルはどのような順序で取得されても，すべてそろってから並べ替えて元のファイルを復元すればよい．しかしながら，ソフトウェアをダウンロードしながらの起動，プログレッシブダウンロード，または代理サーバを通したクライアントへのバイトストリームの提供のように，順序を重視する度合がまちまちのソフトウェアが多数存在する．たとえばファイルの先頭から動画再生が可能な場合には，TCPのように厳密な順序制御は必要ないが，利用できるバッファの大小に応じて，ある程度順序がそろっているとフリーズしにくい．このようなソフトウェアが分割ダウンロードを利用する場合，それぞれの基準で順序と所要時間を重視する比率を決定する必要があり，既存の研究では評価が不十分である．

そこで本稿ではPR-SCTPと順不同配送を用いた分割ダウンロードについて，ダウンロード速度と部分ファイル到達順序を評価する．この部分ファイルを以降，ブロックと呼ぶ．ブロックはアプリケーション層で扱われる単位でありトランスポート層における送信単位（セグメント）とは独立だが，本稿ではブロックをデータリンクの最大転送単位（MTU）より小さく設定することにより，セグメントとブロックを1対1で対応づけるものとする．これは複数のセグメントでブロックを構成する際の分割と再構成を考慮しないことで問題を単純化するためである．

本稿の次章以降は以下のように構成されている．2章では，本研究といくつかの関連研究との違いを紹介する．3章では，本研究における提案方式について述べる．4章では，テストベッドにおける評価実験の結果を議論し，5章において提案方式と既存方式の違いをまとめる．最後に，6章では本稿を要約し，今後の課題を示す．

2. 関連研究

分割ダウンロードにおいて，ブロックが順不同で到達して

[*1] http://www.mozilla.jp/firefox/

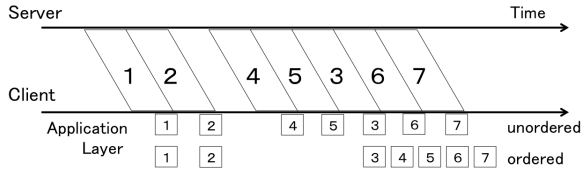


図 1 unordered delivery と ordered delivery の違い

Fig. 1 The difference between unordered delivery and ordered delivery.

も問題ない場合を想定し，順不同配送（unordered delivery）機能，および PR-SCTP によってバックオフ時間に上限を設ける機能の利用が検討されてきた．2.1 節では PR-SCTP を利用した方式を紹介し，2.2 節において PR-SCTP を用いた関連研究について述べる．

2.1 PR-SCTP を用いた分割ダウンロード

筆者らはすでに，パケット到達の順序保証を行わない順不同配送 [8] と，バックオフによる待ち時間を低減させるため SCTP の拡張オプションである PR-SCTP（Partially Reliable extension to SCTP）を用いて TTL を設定可能にするとともに到達を保証する再送機構を組み合わせた PR-SCTP を用いた分割ダウンロード方式を提案した [4], [5].

まず，パケット損失が発生した場合について，順不同配送（unordered delivery）と TCP のような順序どおりの配送（ordered delivery）の違いを**図 1** に示す．unordered delivery と ordered delivery ではトランスポート層からアプリケーション層へのデータの受け渡し処理が異なる．図 1 のようにパケット 3 が損失した場合，unordered delivery は後続のパケットが到着するとアプリケーション層にデータを受け渡し，次のリクエストを発行できる．一方，ordered delivery は後続のパケットが到着してもパケット 3 が到着するまでアプリケーション層で処理されず待ち合わせ時間が発生する．ソフトウェアなどのダウンロードではファイル全体を取得してからでないと利用できないので，到達順序は問題とはならず，順不同配送の導入によってダウンロード時間の短縮が期待できる．

次に，PR-SCTP における TTL を用いた配送方式を**図 2** に示す．クライアント側のアプリケーション層から要求（Request1）があるとトランスポート層を介してサーバに送信される．このときタイマを起動し，再送要求タイムアウト（以下，T_r）までサーバ側からクライアント側に要求したデータが届かなければ再びリクエストを送信する．サーバ側はクライアント側に要求された番号 1 のデータ（Send1）をトランスポート層を介して送信する．PR-SCTP ではサーバ側で設定された TTL の間で再送を繰り返すため，図 2 のようにトランスポート層における識別番号 5 のパケットが連続して損失した場合，TTL を過ぎた時点で再

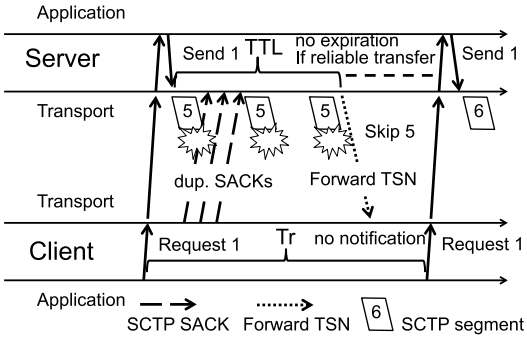


図 2 PR-SCTP における TTL を用いた部分的信頼性の確保とアプリケーションによる信頼性の補完

Fig. 2 TTL-based PR-SCTP delivery and request retransmission by application layer.

```
c=0;
for(i=0;i<Np;i++){
    request(c++);
    set_timeout(Tr, request_again());
}
while (new_block_received && unrequested_remain) {
    request(c++);
    set_timeout(Tr, request_again());
}
proc request_again() {
    foreach i (outstanding_requests) {
        if (expired(i)) {
            request(i);
            set_timeout(Tr, request_again());
        }
    }
}
```

図 3 T_r 自動調整方式の疑似コード

Fig. 3 The pseudocode of the existing method.

送が中止される．このとき Forward TSN が送信され，クライアントの累積確認応答番号が飛ばされる．T_r 経過するとクライアント側は要求したデータが送信されなかったと判断し，アプリケーション層において同じデータを再度要求する．そのため，再送中止されたデータも損失することなくデータの到着は保証される．ただし番号 1 のデータを再送要求した場合，トランスポート層では新たなデータとして扱われるため識別番号は異なる（図 2 内，SCTP segment 5 と 6）．

なお，既存方式（以下，T_r 自動調整方式）[4], [5] はブロックを要求する際，**図 3** の疑似コードに基づいて動作する．図に示すように，最初は同時リクエスト数 N_p までパイプライン的に要求するとともに，各ブロックの到達までにタイムアウトを設ける．また新しいブロックを受信した場合で，かつ未要求のブロックが存在するならば次の最小 ID を持つブロックを要求する．要求してから T_r を経過してもブロックが取得できない場合は，request_again プロシージャの中身を実行する．すなわち，要求を発行して取

得が完了していないすべてのブロックについて，指定した T_r を経過したか確認し，該当したすべてのブロックを連続して再要求する．このとき該当するブロックが複数あると，バースト的に要求が発行される．

また T_r 自動調整方式では直近 n_s 個（文献 [4], [5] では 100 個）のブロックを取得するごとに，その間に発生したブロックの重複取得の数を n_s で割った率 d_r，その間に発生した要求タイムアウトの数を n_s で割った率 t_r を求め，以下の式 (1) に従って T_r を調整する．

$$\begin{aligned} &T_r \leftarrow T_r \times 2 \quad (d_r > 2 \times t_r) \\ &T_r \leftarrow \frac{T_r}{2} \qquad \left(d_r < \frac{t_r}{2}\right) \end{aligned} \tag{1}$$

文献 [9], [10], [11] ではサーバにおいて設定される TTL の値が小さい場合，ダウンロード時間が短いことを示すとともに Forward TSN チャンクを含むパケットが過剰に送出される副作用についても指摘している．これは Forward TSN の送出条件が，本研究で想定している分割ダウンロードのようにサーバの送信バッファを常時充填する状況を考慮していないためであると考えられる．このようにサーバにおける TTL の最適な設定には不明な点が残されており，加えてさまざまなクライアントに対応する必要のあるサーバではクライアントごとのネットワーク条件を考慮することが難しい．本研究ではクライアントにおける要求方式に注目し，サーバにおける TTL の設定値は制御できないものとして，複数の設定値を考える．この前提は，さまざまな環境からアクセスするクライアントに対応して，サーバが個別に設定を変更するのは困難であることからも，妥当であると考える．

これまでの研究では分割ファイルをすべて受信してから取得ファイルを再構成するアプリケーションを想定しており，部分ファイル（ブロック）の取得順序については調査されてきていない．部分ファイルの到達順序は，ファイルの先頭から再生する HTTP ストリーミングなどでは重視する必要がある．

ブロックが順不同で到達する原因としては損失により再送されたブロックが遅れる場合に加え，経由するネットワークの遅延の差やその変動により逆転が生じる場合も考えられる．しかしながら方式の根本的な効果を評価するため，これまでの研究と同様に，本研究でもクライアントとサーバを直接接続した最も単純なトポロジを想定する．方式の効果についての確証が得られたのち，遅延差や変動が生じるネットワークでその効果がどのように変化するかについての評価は今後の課題とする．

また従来研究 [10] では並列して要求するブロックの個数を設定し，未到達のブロックについては再要求を繰り返すことで，到達を確認しながら次の新しいブロックを要求していた．したがって順序逆転は生じにくいが，何度も再要求することで冗長トラフィックが発生しやすく，待ち時間も長くなりやすい．一般に利用されているプログレッシブダウンロードなどのアプリケーションでは独自にバッファを設け，多少の順序逆転には対応可能であることが多い．このような条件においては，従来研究の要求方法は順序保証を過度に重視し，取得レートが十分に得られない可能性がある．

2.2 PR-SCTP を用いた関連研究

PR-SCTP の *timed reliability* サービスを活用した一例として，再生時間に制限のあるマルチメディアストリーミングに用いた方式が研究されている．たとえば，MPEG-4 動画ストリーミングのために PR-SCTP の有効期限付き再送を用いた研究がある [12], [13]．文献 [12] では，重要なフレーム（I フレーム）には比較的長い有効期限である 2,000 ms を設定し，それより重要度の低い B フレームや P フレームは再送を行わない方式が提案されている．また，文献 [13] では各フレームごとに異なる再送回数を設定した方式が提案されている．本稿で扱うアプリケーションではサーバから送信されるパケットの重要度はすべて等しく，有効期限切れのブロックはクライアントのアプリケーション層レベルで再要求する．したがって，アプリケーション層における到達保証の方針が異なる．そのほかにも RTP（Real-time Transport Protocol）と PR-SCTP を組み合わせることでバースト的なパケットロスに対応したリアルタイムストリーム伝送方式 [14] が提案されている．文献 [14] は中継ノードにおいて PR-SCTP を利用して部分的信頼性を保証する再送機構を採用しているため，本研究が取り組むサーバ・クライアント間を対象とした end-to-end の再送機構にはそのまま適用することができない．

3. 提案方式

本研究で対象としているダウンロード形態は以下の 2 つである．

(1) ファイルを非同期にダウンロードする場合（順序と所要時間をそれぞれ重視する比率 0 : 1）
(2) ダウンロードしながらソフトウェアを起動する場合（順序と所要時間をそれぞれ重視する比率 $1-\beta : \beta$，ただし $0 < \beta < 1$）

形態 (1) に関しては所要時間のみが短ければよいので，既存研究 [9], [10] からサーバの制御パラメータによりトランスポート層における再送回数を制限することで，飽和フローにおける最大の速度と同等な性能が得られる．しかしながら，形態 (2) に関しては既存研究ではファイル取得時間のみに着目していたため，取得順序と所要時間の関係は明らかにされていない．従来方式はウインドウをずらしながら，ウインドウ内のブロックが 1 つ取得完了するごとに 1 つ新しいブロックを要求するため，特にバックオフが長い場合に順序を重視しすぎる可能性がある．形態 (2) で

はアプリケーションごとに順序を重視する比率が異なるので，柔軟な調整が必要となる．そこで，本研究では取得順序と所要時間を考慮したブロックの要求方法の基礎検討として確率的再要求遅延方式，および再要求遅延方式を提案する．

本章では，まず 3.1 節において提案方式の確率的リクエスト法を紹介する．次に，3.2 節で提案方式で導入する受信確率推定法に基づくリクエスト方法について議論する．

3.1 確率的リクエスト法

本研究では，まず確率的再要求遅延方式（SDR: Stochastic Delayed Retrying）を提案する．なお，同時リクエスト数 N_p までパイプライン的に要求する挙動は T_r 自動調整方式と同様であるがタイムアウトは受信時に設定する．ブロックを受信したときとタイムアウトが発生したときの挙動も異なるため，図 3 に対応する範囲を**図 4** の疑似コードに示す．図 4 より，SDR 方式はブロックの正常受信時，およびタイムアウト発生時には同じ挙動（next_request プロシージャ）を示す．なお rand() 関数は 0 から 1 の間の値を一様分布に基づいてランダムに与える関数であるとする．SDR 方式では，ブロックがすべて 1 度は要求済みとなるまでの間，図 4 で示す確率 *rate* で要求済みながら取得済みでないブロックを若い番号から順に要求し，確率 $1 - rate$ で要求済みでない最も若い番号のブロックを要求する．一方，すべてのブロックを少なくとも 1 度以上，要求しているのであれば，未取得のブロックを順に再要求する．途中に抜けがあってもすべてのブロックの 1 回目の要求をひととおり発行するため，連続して取得できているブロックは分散する．ここで *rate* は式 (2) で定義する．

$$rate = \frac{1 - \frac{i}{m}}{\alpha} \tag{2}$$

式 (2) における i は，取得したブロックの ID をファイル先頭から連続して昇順に確認したとき，最初に未取得となっているブロックの ID を示す．一方，m は現時点で取得できている最大のブロック ID を示す．すなわち，先頭に近い ID のブロックが未取得のままであり，それより大きな ID の取得が順調に進んでいるなら，上記の *rate* が大きくなるため，順序逆転の原因を作るこの未取得ブロックを高い確率で再要求する．なお α（> 0）は再要求する割合を下げる係数であり，本稿では 1 から 5 までの整数を想定した．

```
c=0;
for(i=0;i<Np;i++){
    request(c++);
}
set_timeout(Tr, next_request());

while (block_received) {
    next_request();
    set_timeout(Tr, next_request());
}

proc next_request() {
    if (unrequested_remain && rand()>rate) {
        request(c++);
    } else {
        b = next_incomplete_block();
        request(b++);
    }
    set_timeout(Tr, next_request());
    if (rand()<1-rr) next_request(); /*(a)*/
}
```

図 4 提案方式の疑似コード

Fig. 4 The pseudocode of the proposed method.

もう 1 つの提案として，再要求遅延方式（DR: Delayed Retrying）を考えた．DR 方式ではブロック正常受信時，およびタイムアウト発生時において，未要求ブロックの要求をつねに優先する．この方式はファイルを完全に取得してから利用する場合のダウンロード形態 (1) への適用を想定している．未要求ブロックがあればつねに要求するので，図 4 の *rate* は 0 となる．したがって，$\alpha = \infty$ に相当する．

3.2 受信確率推定法

図 4 で示したように本稿で提案する再要求遅延方式，および確率的再要求遅延方式では，next_request() イベントが発生する条件は以下の 3 つである．

(1) ブロックを受信した場合
(2) 再送要求タイムアウト T_r 経過した場合
(3) 受信確率推定法条件を満たした場合

パケット損失が多発するネットワーク環境において，条件 (1) だけでは next_request() イベント発生機会が減少することが予想される．したがって，本研究で想定するような劣悪なネットワーク環境では条件 (2) によりリクエストを発行するとよいと考えられるが，T_r 経過までの間は同時並列リクエスト数が減少することが予想される．ここで同時並列リクエスト数とは，ある時点において発行済みのリクエストのうち受信確認がないままのものの数を示しており，同時リクエスト数（図 3，図 4 中の N_p）とつねに等しいとは限らない．そこで提案方式では，同時並列リクエスト数を維持するために条件 (3) を追加し，受信確率推定法から算出された受信確率（以下，r_r）によりネットワーク状況に応じてリクエスト数を増加させる．図 4 の `/* (a) */` 部分を参照し，rand() 関数により与えられた 0 から 1 の間のランダム値と $1 - r_r$ を比較し，ランダム値が $1 - r_r$ より小さい場合に next_request() イベントを発生させる．すなわち，次の r_r が更新されるまでの間は算出された r_r が適用され，その間は next_request プロシージャごとに $1 - r_r$ の確率でリクエスト数を 1 つずつ増加させる．

次に受信確率推定法による r_r の算出方法を説明する．受信確率推定法では，直近に到達したブロック ID を重複なく n_h 個採取し，集合 B とする．集合 B の中で最大の ID を max_id とし，$lowest\_id = max\_id - n_h + 1$ と定義する．すなわち $lowest\_id$ から max_id までの範囲には n_h 個が含まれる．また集合 B，max_id，および $lowest\_id$ は重

複のない新しいブロックを受信するごとに更新するものとする．次に，関数 $F(x)$ の x が $lowest\_id \leq x \leq max\_id$ を満たせば 1，そうでなければ 0 を得るものと定義し，式 (3) から r_r を求める．ここで B_k は集合 B の k 番目の要素とする（ただし $0 \leq k < n_h$）．式 (3) より，直近に到達した n_h 個のブロックの ID が max_id から連続していて抜けが少なければ r_r は 1 に近い値となり，抜けが多ければ 0 に近づくことから，到達率の指標となりうることが分かる．なお，本稿では r_r の初期値を 1.0 とし，サンプル数 n_h 個のブロック ID が蓄積した後に $1 - r_r$ は変動するものとする．

$$r_r = \frac{\sum_{k=0}^{n_h-1} F(B_k)}{n_h} \tag{3}$$

4. 実機実験

本章では，4.1 節で本研究の実験環境について説明する．次に，4.2 節では比較対象の方式を所要時間と到達順序の観点から評価し，考察する．最後に 4.3 節で適用環境に関する議論を行う．

4.1 実験環境

本実験では実機を用いたテストベッドネットワークを構築し，既存研究 [4], [5] で提案した T_r 自動調整方式と提案方式を用いた場合のファイル取得時間とブロック到達順序を評価する．実際に構築したテストベッドネットワークを**図 5** に示す．各リクエスト方式による純粋な性能を評価するためにサーバ PC 1 台とクライアント PC 1 台の間にネットワークエミュレータを介した単純なトポロジを想定した．本実験で使用するネットワークエミュレータはサーバ・クライアント間の帯域，遅延，およびランダムなパケット損失率を設定可能である．本実験で設定した値は無線 LAN 環境である IEEE802.11b を想定し，フェージングなどにより劣悪なネットワーク環境で通信する状況を考慮している．また PR-SCTP を用いた分割ダウンロード方式の制御に必要な実験パラメータを**表 1** に示す．取得ファイルはインターネットを介して配信される一般的なソフトウェアパッケージのファイルサイズに相当する 20 MByte を想定する．HTTP ストリーミングなどを長時間再生する場合は 20 MByte より大きいファイル容量が想定される

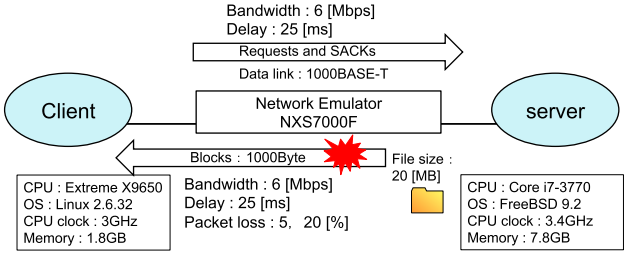


図 5 テストベッド
Fig. 5 Network topology.

が，取得するファイルサイズと取得時間の関係は比例するものと考える．1 つのブロックサイズは一般的に想定される MTU（Maximum Transfer Unit）である 1,500 Byte より小さい 1,000 Byte とし，パラメータの効果を細粒度高く評価できるように考慮する．PR-SCTP の *timed reliability* サービスにおける有効期限である TTL は 1 ms と 10 s の 2 通りに設定した．これは既存研究 [9], [10] において想定するネットワーク環境では TTL が 1 ms から 10 s の範囲（パケットロス率 20%の場合）で飽和フローが最大の速度を達成できることが判明したからである．なお，飽和フロー状態では，TTL = 10 s でもトランスポート層における再送率が 1%程度にとどまっていることから，パケットロス率 20%における上記の範囲はトランスポート層における再送を許容しない範囲といえる．クライアントからはサーバにおいて設定されている TTL を把握できないため，TTL が大きく設定されていると到達順序と所要時間を重視する比率を調整することは困難である．したがって，本研究では比較対象の方式を用いた場合，設定した TTL の範囲で飽和フローが最大の速度を達成できるかを評価するとともにブロックの到達順序についても評価する．また，クライアントの制御パラメータである同時リクエスト数 N_p はリクエストが十分に確保されている状態を想定して 100 に設定し，クライアントにおけるアプリケーションレベルの再送要求タイムアウト（T_r）は既存研究 [4], [5] で提案した T_r 自動調整方式の最小値である 1 s とした．

4.2 ファイル取得時間と平均順序逆転数

本実験では各リクエスト方式におけるブロック到達順序を平均順序逆転数によって評価する．平均順序逆転数を定義するために，先行未到達ブロック数を定義する．先行未到達ブロック数 p_i は，再生順が i であるブロック b_i に対し，到達時刻 t_i が分かっているとき，式 (4) で与える．

$$p_i = |\{b_k | k < i \ \wedge \ t_k > t_i\}| \tag{4}$$

すなわちブロック b_i についての先行未到達ブロック数は，b_i より再生順の早いブロックの中で，b_i よりも遅い時刻に到達したものの個数を示す．平均順序逆転数 $\bar{o}$ は式 (5) で定義する．

$$\bar{o} = \frac{\sum_{i=0}^{n_b-1} p_i}{n_b} \tag{5}$$

すなわちすべてのブロック（個数 n_b）についての先行未

表 1 実験パラメータ
Table 1 Parameters for experiments.

File size（MB）	20
Block size（B）	1,000
TTL（s）	0.001，10
同時リクエスト数 N_p	100
T_r（s）	1

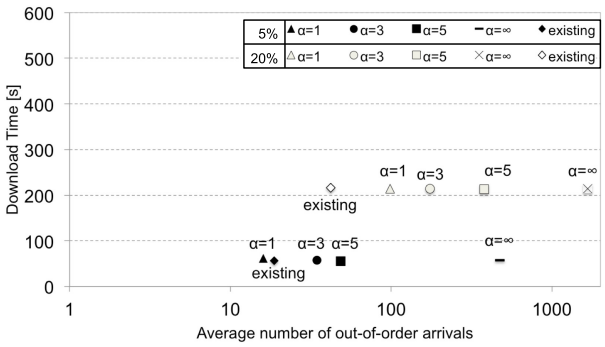


図 6 平均順序逆転数 × ファイル取得時間（TTL = 1 ms）
Fig. 6 ANOA × download time (TTL = 1 ms).

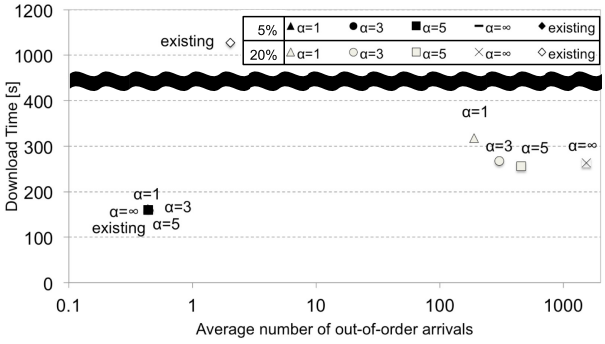


図 7 平均順序逆転数 × ファイル取得時間（TTL = 10 s）
Fig. 7 ANOA × download time (TTL = 10 s).

到達ブロック数の平均である．この値はすべてのブロックを取得後に得られるものであり，再生順が後のブロックが先に取得されるほど大きくなることから，直近の再生には不要であるにもかかわらずバッファ内に滞留したブロックがダウンロード試行を通して平均的にどれくらいの影響を与えたかを事後に見積もる尺度であると考えられる．

たとえば再生順のブロック ID が 1，2，8，5，3，4，9，6，7 の順で到達したとする．ブロック 5 が到達した時点，および 9 が到達した時点では，それぞれについて再生順が前にもかかわらず未到達のブロックが 2 個（3 と 4，および 6 と 7）ある．またブロック 8 については先行未到達ブロック数が 5（3 と 4 と 5 と 6 と 7）となる．それ以外のブロックについてはこの値が 0 であることから，平均順序逆転数はこれらを 9 で割った平均であるので，1 である．

図 6，**図 7** にそれぞれ TTL = 1 ms，10 s における平均順序逆転数とファイル取得時間の関係を示す．それぞれの図では横軸にブロックの到達順序を表す平均順序逆転数（ANOA: Average Number of Out-of-order Arrivals），縦軸にファイル取得時間を示す．なお，確率的再要求遅延方式は典型的な $\alpha = 1$，3，5 の結果のみに限定し，再要求遅延方式は $\alpha = \infty$ とする．図 6 より TTL = 1 ms に設定した場合は T_r 自動調整方式も確率的再要求遅延方式（$\alpha = 1$，3，5）と同程度のファイル取得時間を示すことが分かる．このファイル取得時間を既存研究 [9], [10] で示した飽和フローで最大の速度を発揮した場合と比較した結果を**表 2** に示す．飽和フローにおけるファイル取得時間と比較した上昇率はパケットロス率 5%の場合の $\alpha = 1$ を除き，105%以下にとどまっている．平均順序逆転数に関してはパケットロス率 5%の場合，確率的再要求遅延方式（$\alpha = 1$）が最も小さく，パケットロス率 20%の場合，T_r 自動調整方式が最も小さいことが分かった．ただし，パケットロス率 20%の場合，確率的再要求遅延方式（$\alpha = 0.5$）を試行した結果，T_r 自動調整方式と同程度の平均順序逆転数を示し，ファイル取得時間も T_r 自動調整方式と比較して 104%の上昇率にとどまっている，したがって，確率的再要求遅延方式では α の値を小さくすることによってファイル取得時間の上昇率をおさえつつ，平均順序逆転数を減少させることができる．この結果から，3 章冒頭で述べたダウンロード形態 (1) に適用する場合，TTL = 1 ms において確率的再要求遅延方式（$\alpha = 5$）が飽和フローにおけるファイル取得時間の上昇率が最も小さいことが分かり，望ましい方式であるといえる．またダウンロード形態 (2) に適用する場合，ファイル取得時間の上昇率をおさえつつ，平均順序逆転数を減少させることができる確率的再要求遅延方式において小さい値の α を用いることが望ましい．

表 2 飽和フローの場合に対するファイル取得時間の上昇率（%）
Table 2 Rate of download time to saturated flow case (%)

TTL	Loss	$\alpha = 1$	3	5	∞	existing
1 ms	5%	112.46	104.66	101.22	104.92	102.97
	20%	103.96	103.85	103.54	104.03	104.71
10 s	5%	135.69	133.43	133.4	132.14	134.88
	20%	143.84	121.35	115.87	119.13	478.08

次に，図 7 より TTL = 10 s に設定した場合について議論する．ロス率 5%の場合，平均順序逆転数はいずれの方式も 1 個以内におさまる結果となった．またファイル取得時間も表 2 の飽和フローにおけるファイル取得時間の上昇率からみて同性能であることが確認できる．TCP を用いてファイルを取得する実験を別途実施した結果，取得時間の違いは 10 s 程度であることが確認できている．このことからパケットロス率に対して TTL が長いと TCP に近い挙動を示す．一方，ロス率 20%の場合，T_r 自動調整方式と提案方式では異なる傾向が見える．まず 3 節冒頭で述べたダウンロード形態 (1) に適用する場合，確率的再要求遅延方式（$\alpha = 5$）が飽和フローにおけるファイル取得時間の上昇率が最も小さいため，望ましいといえる．またダウンロード形態 (2) に適用する場合，確率的再要求遅延方式の α の値によって順序と所要時間の比率 $1 - \beta : \beta$ の変更が可能である．一方，T_r 自動調整方式は平均順序逆転数は 2 個以内におさまるがファイル取得時間が長くなる傾向にある．このときのリクエスト数は 25,000 前後におさまっていることから，T_r 自動調整方式では T_r の長さを調整し

つつ，トランスポート層のバックオフアルゴリズムによる再送の恩恵を受けている．

4.3 適用環境に関する議論

昨今のネットワークでは複数のサービスが同一リンクを利用してインターネットにアクセスしていることが多い．その場合，提案方式で制限したバックオフにより，過剰にネットワーク資源を占有する可能性がある．既存研究 [7] では背景に TCP Reno が何本か存在するリンクにおいて方式を評価し，背景に与える影響は少ないと報告している．文献 [7] で提案する方式はストリーム数 10 としており，本研究の順不同配送（unordered delivery）との性能差はあまりない [5]．また TCP Reno や CUBIC などの現在普及しているトランスポート技術では，20%といった高いパケットロス率において帯域を占有できるほどのスループットが確保できないと考えられる．したがって現状では，輻輳の発生ではなく，データリンク起因のパケットロスを主な対象として評価しても現実的であると考えた．ただし，今後，無線ネットワーク上でも高いスループットが得られるトランスポート技術が普及してきた場合は評価し直す必要がある．さらに常時 20%のパケットロス率という環境はまれであると考えられるが，一時的にこのように悪化するケースは考えられるため，平常時と極端な高ロス率を評価しておくことで，時間変動のシナリオで実験した場合の結果を解析する際には参照値とすることができる．

また文献 [15] で評価された結果から Forward TSN の仕様についても修正が提案されており，この成果とあわせて今後，評価していきたい．トポロジについては現状では単純なクライアント・サーバ間のモデルのみであるが，PR-SCTP を用いた分割ダウンロード方式の基礎的な性能評価を行うためには複雑性を除去する必要があった．より現実的なトポロジで評価することも今後の課題である．

5. 比較評価

本章では提案方式を他の方式と比較評価する．**表 3** に評価対象となる方式の機能の違いをまとめる．比較項目について，損失待ちとは送信データが損失した場合に待ち合わせるかどうかである．リアルタイムストリーミング [12] は再生に間に合わないデータは損失してもよいのに対し，データのダウンロード，および HTTP ストリーミングはすべてのデータの到達を保証するので，我々のこれまでの研究ではすべて有となる．この有にはトランスポート層が放棄した待ち合わせをアプリケーションで補う場合を含むことに注意が必要である．続く順不同受信とはセグメントの到達が順序どおりでないときにトランスポート層で待ち合わせなくてよいことを示す．マルチストリーム機能がストリーム単位では順序を保証するのに対し，unordered delivery はセグメントの中身でしか順序を保証しない．T_r はアプリケーションが再要求まで待ち合わせる時間であり，トランスポートの部分的信頼性をアプリケーションで補う方針である我々が提案してきた方式にしか定義されない．TTL はセグメント単位で設定できるが，データ送信元（サーバ）において固定値として設定するのが簡単である．なお T_r と TTL はそれぞれの方式を提案した文献において評価の際に採用された値であり，この値しか使えないわけでも，これがつねに最適というわけでもないことに注意を要する．

表 3 各方式の機能比較表
Table 3 Comparison among existing and proposed methods.

	損失待ち	順不同受信の単位	T_r	TTL
MPEG4 ストリーミング [12]	部分的に有	セグメント	-	2,000 ms or 0 ms
マルチストリーム [7]	有	ストリーム	1 s	500 ms
T_r 固定 [4], [5]	有	セグメント	1 s or 10 s	500 ms
自動調整 [4], [5]	有	セグメント	1 s–10 s	500 ms
自動調整 [9]	有	セグメント	1 s–10 s	1 ms–1,500 ms
飽和フロー [9]	有	セグメント	理想値	1 ms–100 s
提案方式	有	セグメント	1 s	1 ms, 10 s

PR-SCTP をファイルの分割ダウンロードに応用した最初の方式 [7] では，特定のストリームに所属するパケットは順序が保証されるが，ストリームが異なるデータについては順不同でトランスポート層からアプリケーション層に渡すことができる．文献 [4], [5] が提案した方式では SCTP の順不同配送（unordered delivery）を採用しており，セグメント単位で順不同に処理できる．既存の方式はすべて，ファイルのダウンロードをアプリケーションとして想定してきたので順不同の受信を許容する．一方，本研究では完全な順序どおりの受信から順不同まで，アプリケーションについて広範囲のポリシをパラメータ α によって想定できるのが特徴である．

TTL はサーバ側で設定される値であり，本研究ではクライアントの修正のみでの性能向上を目指している．T_r 自動調整方式 [4], [5] は T_r を受信データの観測により 1 s から 10 s の間で自動調整するもので，当初の文献では TTL を 500 ms に固定して評価されている．文献 [9] ではクライアントの再要求待ち時間である T_r を自動調整した方式について，同時リクエスト数と TTL を変更して評価している．飽和フローとはサーバのバッファがつねに送信データで満たされている状態を指すが，応答に応じて次のブロックを要求する我々の想定ではパケットロスするブロックの特定などが事前には困難であるため，理想的な状態にあたる．

これらのうち，T_r 固定方式と T_r 自動調整方式 [5]，およ

表 4 各方式のダウンロード時間（s）
Table 4 Comparison of download time (s) among existing and proposed methods.

	TTL	パケットロス率 5%	10%	20%
T_r 固定 [4], [5] 1 s	500 ms	108.3	132.5	395.5
T_r 固定 [4], [5] 10 s	500 ms	146.2	214.6	482.3
T_r 自動調整	1 ms	56.6	103.1	215.7
	10 s	162.1	281.2	1,055.2
提案方式 $\alpha = 1$	1 ms	61.8	104.1	214.2
	10 s	163.0	250.5	317.5
提案方式 $\alpha = \infty$	1 ms	57.6	107.1	214.3
	10 s	158.8	227.2	262.9
飽和フロー	1 ms	55.3	100.1	209.4
	10 s	119.7	135.6	222.4

び提案方式を，パケットロス率を変化させたときのダウンロード時間について比較する（**表 4** 参照）．T_r 固定方式の結果は文献 [5] より引用しており，同一条件で比較できるように，同時リクエスト数が 100 についてのみダウンロード時間を比較し，順序を評価対象としていない方式とは順序逆転について評価しない．文献 [5] では TTL を変化させた実験は行っていないが，同じ 500 ms の TTL を採用した文献 [7] では，TTL が無限大に相当する信頼性のある通信である SCTP に対し，PR-SCTP を用いた方式はパケットロス率 5%において 10%の高速化，およびパケットロス率 10%においては 30%の高速化を実現したと報告されており，TTL を設定した PR-SCTP の効果が得られていると考えられる．実際，この環境で SACK オプションつき TCP New Reno を使って分割ダウンロードした場合，パケットロス率 5%，10%，20%のそれぞれについて，ダウンロード時間は 170.6 s，298.3 s，616.3 s であり，信頼性（TTL 無限大相当）と順序をともにトランスポート層で保証する場合，ダウンロード時間の増大が認められる．提案方式については図 6 および図 7 の元データのうち α が 1 と ∞ のもののみを選択し，パケットロス率 10%の結果も加えている．飽和フローは理想の値を示しており，ダウンロード時間の下限であると考えられる．表 4 より，T_r 固定方式に比べ，T_r を動的に調整することにより，あるいは確率的に要求することにより高速にダウンロードできることが分かる．また理想的な値である飽和フローと比較しても，順序を問わないダウンロードであれば，$\alpha = \infty$ を採用することで，TTL の値によらず理想に近い速度が得られていることが分かる．提案方式はパラメータ α によって順序重視の度合いを変更することができるとともに，$\alpha = \infty$ を採用することでサーバ側の TTL 設定によらず順不同の高速なダウンロードが実現できることが分かる．

6. おわりに

本稿では従来方式と新たに提案する 2 方式について，ダウンロード所要時間とブロック到達順序を評価した．実機とテストベッド環境において評価した結果，既存方式である T_r 自動調整方式は到達順序において逆転が発生する可能性が低いが，バックオフ時間の上限を決める TTL がサーバにおいて大きく設定されていると，順序保証に重点が偏りダウンロード時間が長くなることが分かった．また提案した再要求遅延方式と確率的再要求遅延方式はパラメータ α によって順序逆転の度合いを調節できること，および再要求遅延方式は比較的短い時間で安定してダウンロードできることが分かった．

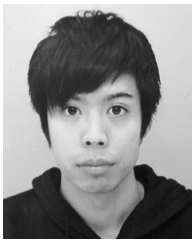
今後の課題として，実環境において提案方式を評価することが考えられる．動画の再生に応用した場合，実際のユーザが体感する品質も評価する必要がある．この際，複数の経路で SCTP を用いて動画再生を高品質化させる試みである CMT-CA [16] の成果も参考にしていきたい．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金（基盤（C）課題番号 JP26330109），および公益財団法人フジクラ財団研究助成により実施した．

参考文献

[1] 赤瀬謙太郎，小畑博靖，村瀬 勉，平野由美，石田賢治：無線 LAN を考慮した帯域確保 TCP のスロースタートしきい値設定方式，電子情報通信学会和文論文誌（B），Vol.J93-B, No.2, pp.153–165 (2010).
[2] Cohen, B.: Incentives Build Robustness in BitTorrent, *Proc. 1st Workshop on Economics of Peer-to-Peer Systems* (2003).
[3] Sodagar, I.: The MPEG-DASH Standard for Multimedia Streaming over the Internet, *IEEE Multimedia*, Vol.18, No.4, pp.62–67 (2011).
[4] 舟阪淳一，沖 達也：PR-SCTP を用いた分割ダウンロード方式の制御パラメータに関する検討，電子情報通信学会第 4 回ネットワークソフトウェア研究会，pp.3–9 (2014).
[5] Funasaka, J. and Oki, T.: On Control Parameters for Segmented File Download Using PR-SCTP Unordered Delivery, *Proc. ATC 2014*, pp.548–553 (2014).
[6] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M. and Conrad, P.: Stream Control Transmission Protocol Partial Reliability Extension, IETF RFC3758 (2004).
[7] Funasaka, J., Katsube, J. and Ishida, K.: Timeout Reduction Method for Multistream Segmented Download Using PR-SCTP, *Proc. 33rd ICDCS Workshops*, pp.380–385 (2013).
[8] Stewart, R. (Ed.): Stream Control Transmission Protocol, IETF RFC4960 (2007).
[9] 武田和也，舟阪淳一：PR-SCTP を用いた分割ダウンロード方式の制御パラメータ調整による高速化，電子情報通信学会技術研究報告 IN2015-59，Vol.115, No.IN-252, pp.35–40 (2015).
[10] Takeda, K. and Funasaka, J.: Performance Evaluation on Control Parameters for Segmented File Download Method Using PR-SCTP, *Proc. 3rd International Symposium on Computing and Networking* (*CANDAR '15*),

pp.299–302 (2015).
[11] 武田和也，舟阪淳一，石田賢治：PR-SCTP を用いた分割ダウンロード方式の飽和フロー状態におけるサーバの制御パラメータの解析—バックオフに与える影響について，電子情報通信学会技術研究報告 NS2016-193，Vol.116, No.NS-484, pp.205–210 (2017).
[12] Molteni, M. and Villari, M.: Using SCTP with partial reliability for MPEG-4 multimedia streaming, *Proc. European BSD Conf.*, pp.400–417 (2002).
[13] Wang, L. and Kawanishi, K.: Performance of MPEG-4 Transmission over SCTP Multi-Streaming in Wireless Networks, *IEICE Trans. Commun.*, Vol.E93-B, No.9, pp.2336–2347 (2010).
[14] 前田朋孝，小塚真啓，岡部寿男：PR-SCTP を用いた高信頼性ストリーミング伝送，電子情報通信学会技術研究報告 IA2008-74，pp.43–47 (2009).
[15] Rajiullah, M. and Brunstrom, A.: On the effectiveness of PR-SCTP in networks with competing traffic, *2011 IEEE Symposium on Computers and Communications* (*ISCC*), pp.898–905 (2011).
[16] Wu, J., Yuen, C., Wang, M. and Chen, J.: Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol.27, No.3, pp.710–723 (2016).



武田 和也

1992 年生．2015 年広島市立大学情報科学部卒業．2017 年同大学大学院情報科学研究科博士前期課程修了．PR-SCTP を用いた分割ダウンロード方式に関する研究に興味を持つ．

舟阪 淳一 （正会員）

1970 年生．1993 年東北大学理学部卒業．1995 年同大学大学院理学研究科博士前期課程修了．1997 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．1999 年同後期課程修了．同年広島市立大学情報科学部助手．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．IEEE，電子情報通信学会各会員．