トランスポートプロトコルにおけるバッファ削減方式*

鹿間 敏弘*1

Buffer Reduction Schemes for the Transport Protocol

Toshihiro SHIKAMA

*1 Department of Electrical, Electronic and Computer Engineering

This paper studies the buffer reduction schemes for the Transport Protocol TCP. By using network simulations, it shows that a TCP connection requires large number of receive buffers to achieve high throughput. However, it also shows that theses buffers are not utilized efficiently. To improve utilization of the buffers this paper proposes common receive buffers for multiple TCP connections. Evaluation by simulations proves that the common buffer scheme can reduce the number of required buffers. Although segment losses are observed, its effect to the total throughput is allowable. This paper also studies outstanding buffers for the sending side of TCP and shows that large amount of outstanding buffers are held in provision for retransmissions. To mitigate this problem this paper proposes utilization of secondary storage for the outstanding buffers. Although extra delay is introduced for retransmissions, its effect to the total performance is limited.

Key Words : Transport Protocol, TCP, Reassembling, Buffer Management, Network Simulation

1. 緒　　言

ICT 分野ではあらゆる機器にコンピュータを組込み，無線通信を介してインターネット接続する方向に進んでいる．従来，組込み機器は単独で使用する場合が多かったが，機械と機械が通信を行う M2M やクラウドコンピューティングとの連携などネットワーク接続による広がりが拡大している．無線通信の速度向上も目覚ましく，無線 LAN および LTE や LTE アドバンストなど上りも 100Mbit/s を超える無線通信も実用化されようとしている．

インターネット接続するためにはプロトコルスタックとしてネットワーク層に IP を用いなければならないが，これに加え信頼性の高い通信を行うためには上位のトランスポートプロトコルとして TCP を用いる必要がある．TCP は複雑な通信プロトコルで組込み処理を行うマイコンの処理能力が要求されるが，近年は ARM 系マイコンなど処理能力が向上しており対応可能と考えられる．しかし，TCP には処理能力とともに送信用と受信用のバッファとして大量の主記憶が必要となる問題もある．特に，組込み機器が複数の TCP コネクションを同時に確立して高速通信を行う場合には，TCP のコネクションごとに送受信バッファが必要となるが，主記憶の容量が限られた組込み機器では高速な無線通信を十分に活用できない恐れがある．

Windows や Linux などパソコンやタブレットの OS ではハードディスクなどの補助記憶を利用した仮想記憶方式が採用されており，これにより広大な主記憶空間が利用可能であるため TCP のバッファによる主記憶利用の問題は深刻ではない．しかし，一般に仮想記憶を持たない組込み機器では TCP の送受信バッファにより限られた主記憶が利用されるのは問題と考えられる．本研究では，組込み機器におけるソフトウェアが TCP/IP による通信により複数のコネクションを設定して送受信を行うことを前提に，TCP の送信バッファと受信バッファについて，最初に利用効率を調査し，主記憶を有効利用するための方式を検討する．次に主記憶を利用するこれらバッファの削減方式を検討し，シミュレーションにより評価を行う．

* 原稿受付　2014 年 2 月 26 日
*1 電気電子情報工学科
E-mail: shikama@fukui-ut.ac.jp

2. TCP における送信および受信バッファの問題点

TCP における送信および受信バッファの利用を説明するためのシーケンス例を Fig. 1 に示す．この図では重複 ACK 受信により高速再送[(1)(2)]が行われる場合を想定している．図において左側は送信，右側は受信を示す．TCP ではデータの送受信単位をセグメントと呼ぶ．セグメントは下位のネットワーク層でパケットに組み立てられネットワークで転送される．図で最初にセグメント S1 から S6 の 6 セグメントが送信されている．セグメント S1，S2 は正しく受信され，それらセグメントのデータは一旦受信バッファに格納されるが，直ちに上位のアプリケーション層に渡され，受信バッファは解放される．受信側では正しくセグメントを受信すると送達確認(ACK)を送信側に返す．送達確認 A1，A2 を受信すると送信側はセグメント S7，S8 を送信する．

セグメント S3 は受信誤り又はネットワークでの輻輳により喪失したものとする．後続のセグメント S4，S5，S6 は正しく受信されるが，これらは S3 が再送されるまで受信バッファに保存される．また，送達確認受信後に送信された S7，S8 も正しく受信されるが，これらも S3 を受信するまで受信バッファに保存される．このようにデータの順序を保存するために後続の受信データを保留する処理をリアセンブルと言う．Fig. 1 では A2 を 4 回受信すると重複 ACK 受信により S3 の再送が行われ，S3 が受信されるとこのセグメントのデータとともに，受信バッファに保留されていたセグメント S4 から S8 のデータが上位層に転送され，受信バッファが解放される．

TCP の場合，予め受信バッファを確保してこの大きさを告知ウィンドウとして送信側に通知し，送信側は送達確認を受けずにこのサイズ分のデータを送信できる．ウィンドウサイズは帯域と遅延時間の積に比例して大きくする必要があり，これに整合した受信バッファ量の確保が必要となる．有線ネットワークや無線ネットワークの高速化に伴い，それに見合うスループットを得るため確保される受信バッファの大きさは増大している．しかし，Fig. 1 から分かるように受信バッファが一杯に利用されるのはセグメントの喪失に伴うリアセンブリングの場合であり，セグメントの喪失が発生しない場合は確保されたバッファのほとんどは利用されず，計算機の主記憶が有効に利用されない問題がある．

通常，計算機内部では多数の TCP コネクションを設定しており，各コネクションに受信バッファが確保される．本研究では，複数のコネクション間で受信バッファを共有することにより受信バッファの利用効率を高め，少ない主記憶で多数の TCP コネクションの実現を狙う．

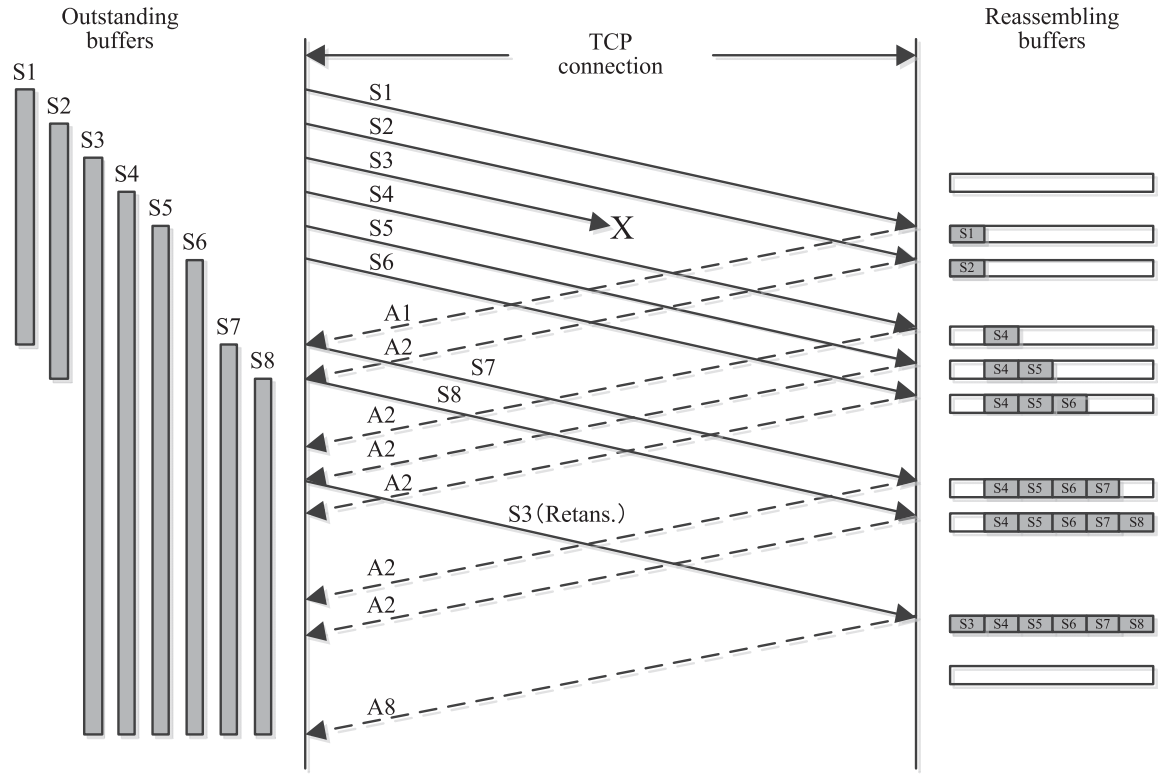

Fig. 1　Example of TCP sequence.

上記のようなTCPの受信側で行われるセグメントのリアセンブリング制御に関しては，これと等価な選択再送プロトコルの順序制御による遅延とその解決方式[4][5]や順序制御による出力トラヒックのバースト性の問題とその解決方式[6][7]などの研究が行われているが，順序制御に必要となるバッファ量に関して研究は行われていない．

Fig. 1 で左側は送信バッファが使われるシーケンスを示している．送信バッファはセグメント送信後，そのセグメントの送達確認が行われるまで送信データを保留するために使用される．これは，再送を行う場合に前に送ったデータが必要となるためで，送達確認された時点で再送の可能性がなくなるので解放される．図から分かるように，送信バッファは最低でもラウンドトリップ時間は保留され，主記憶のような高速のバッファを用いる必要はない．これから，ラウンドトリップ時間が大きい場合，補助記憶に送信データを移し，主記憶を解放することにより主記憶による送信バッファを削減することが可能と考えられる．本論文では上記のようなメカニズムにより主記憶と設ける送信および受信バッファを削減する方式を研究する．

3. TCPにおける送信および受信バッファの平均使用量

3.1 シミュレーションモデル

ここではTCPにおいてどの程度の送信および受信バッファが有効に使用されているかを把握するためシミュレーションにより評価を行う．本論文ではネットワークシミュレータとしてns-2[3]を用いた．TCPのバッファ使用量に関する性能評価を目的としているため，シミュレーションモデルはFig. 2に示す簡単なネットワーク構成とした．Table 1にシミュレーションの条件およびパラメータを示す．帯域が100Mb/sの回線に1本または複数本のTCPコネクションを設定し，送信可能な限りデータの送信をおこなう．TCPのバージョンは標準的なNewReno[1][2]とした．TCPのコネクション数は1および10本で行った．回線の遅延時間は国内のインターネット環境を想定して20msとした．これからラウンドトリップ時間は40msとなり，帯域と遅延時間の積は4Mbitとなる．これは512KBに相当するため，この値をTCPのウィンドウサイズとした．通常のTCPのウィンドウサイズは64KBであるが，ネットワークの高速化が進み，帯域と遅延時間の積が64KBを超える環境が一般的となったため，ウィンドウサイズの拡大を可能とするウィンドウスケール・オプション[8]の利用が一般的となっている．セグメントのデータサイズを1460Bとすると，512KBのウィンドウサイズは351セグメントに相当する．回線ではパケット損失率に従ってランダムにパケット廃棄が発生するものとした．

TCPの受信側ではリアセンブリング処理により，再送などによりセグメントの順序が乱れて受信した場合，受信したセグメントをバッファに保留する．Fig. 3はFig. 1のシーケンス例において受信側で保留されるバッファを示している．セグメントS3が失われた後，セグメントS4からS8を受信するが，これらのセグメントはバッファに保留される．ここで，失われたセグメントS3用のバッファも確保されており，保留バッファ数は6と勘定する．送信側の送達確認待ちバッファ数は，新規のセグメントを送信するとセグメント数分増加し，送達確認を受信すると確認されたセグメント数分減少するものである．シミュレーションではこれら受信側で保留されるバッファ数と送信側の送達確認待ちバッファ数を時系列として測定し，平均と最大値を求めた．ここで，TCPコネクションが複数存在する場合は，複数コネクションでの総和の平均と最大値を求めた．シミュレーションは乱数の初期値を変えた16回の独立した試行を行い，各試行で得られたバッファの平均値と最大値について，平均と標準偏差を求めた．さらに標準偏差から95%信頼度区間を計算した．

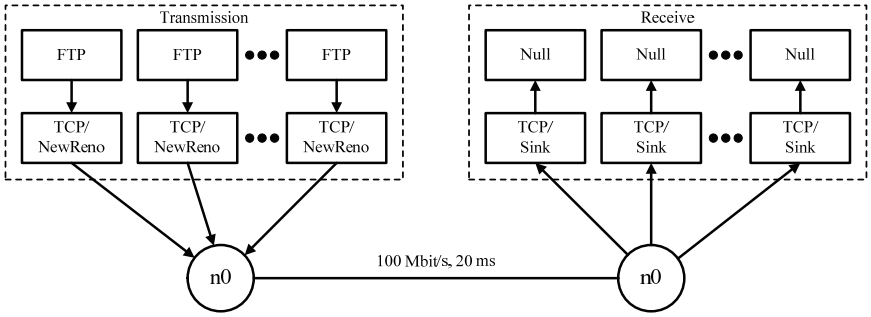

Fig. 2 Simulation model.

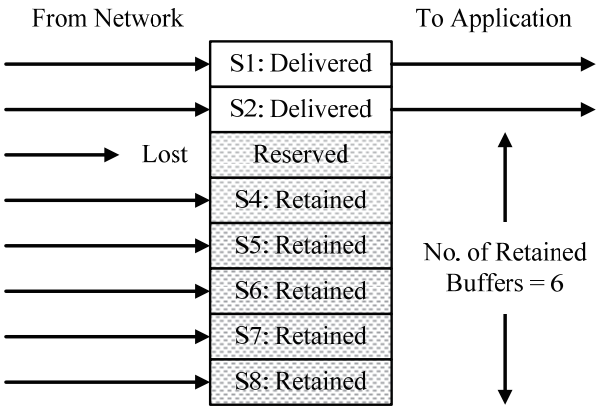

Fig. 3　The number of retained buffers by the receiver.

Table 1　Simulation conditions and parameters.

Version of TCP, its window size	NewReno, Window size:　351 segments (512 KB)
Segment size	1460 B
Characteristic of packet loss	Random
Packet loss rate (P_L)	0.00001, 0.00002, 0.00005, 0.0001, …, 0.05, 0.1
Volume of buffers at a node	Unlimited
Bandwidth and delay of the link	Bandwidth: 100 Mbit/s, Delay: 20 ms
Number of TCP connections (N_f)	1, 2, 5, 10
Simulation time	10 sec
Number of simulation runs	16

3.2　スループットと受信バッファに関するシミュレーション結果

Fig. 4 はシミュレーション結果で，TCP のコネクション数 (N_f) を 1，2，5，10 と変化させた時のパケット損失率とスループットの関係を示している．図において，“95% Conf.” は平均値の 95%信頼度区間である．TCP のコネクション数が 1 の時，パケット損失率が極めて小さい場合にスループットは 100Mb/s に近い値となるが，パケット損失率が大きくなるにしたがってスループットは低下する．TCP はパケット損失が起こる原因をネットワークの輻輳によりルータなどでバッファ溢れが発生したものとしている．このため，パケットの損失を検出すると再送を行うとともに，ネットワークの輻輳を防ぐため輻輳ウィンドウと呼ばれるパラメータを小さくする．これにより新規に送信するデータ量を抑え，輻輳を回避しようとするがスループットも低下する．

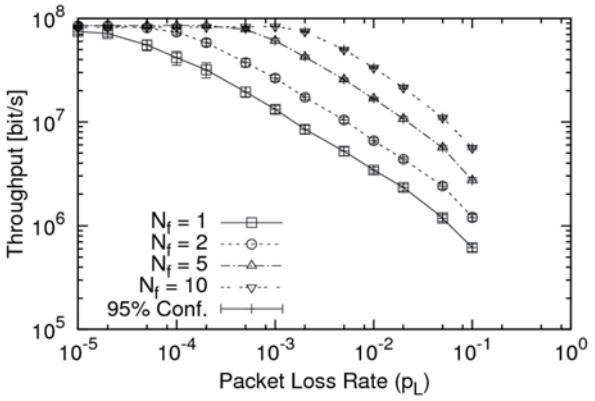

Fig. 4　Packet loss rate vs. throughput.

TCP のコネクション数を増加させると，パケット損失率が大きい場合でもスループットが向上する．TCP のコネクション数が 10 の場合，パケット損失率が 0.001 より小さい場合は回線帯域 100Mb/s に近いスループットが得られる．これは，100Mb/s の帯域を 10 本の TCP コネクションで分割使用するため，TCP 1 本当たりのスループットが約 1/10 に抑えられ，輻輳ウィンドウが絞られることによる影響が少なくなるためである．パケット損失率が 0.001 より大きくなるとスループットの低下が見られる．

Fig. 5 は TCP コネクション数が 1 と 10 の場合について，パケット損失率を変化させた時の受信側で保留される平均バッファ数を示している．また，Fig. 6 は受信側で保留される最大バッファ数を示している．TCP コネクション数が 1 の時，パケット損失率が小さい場合に最大バッファ数は 350 とウィンドウサイズに近い値を取る．一方，平均バッファ数は 5 以下とウィンドウサイズに比べ小さな値である．これは，受信バッファが最大でウィンドウサイズ分使われることはあるが，その確率は極めて小さく，平均的には小さな数の受信バッファで済んでいることを示している．この場合，1 本の TCP コネクションは 351 セグメント分のバッファを用意しているが，平均的に使われるのは 5 / 351 = 1.4%である．受信バッファの利用効率が小さいと言える．Fig. 5 で平均バッファ数はパケット損失率が 0.005 付近で最大となっている．最大値が存在する理由は，この値よりパケット損失率が小さいと，受信バッファが保留される頻度が少ないために平均値が小さくなること，この値よりパケット損失率が大きいと，輻輳ウィンドウサイズが小さくなるために受信側で保留されるバッファが少なくなり，平均値が小さくなるものと考えられる．

TCP コネクション数が 10 の場合，受信側で保留される平均バッファ数および最大バッファ数は 10 本の TCP コネクション全体での和を取った値である．平均バッファ数は 60 と大きくなっているが，最大バッファ数は 1100 以下でコネクション数が増えても大きな増大は無い．この場合，各 TCP コネクションは 351 セグメント分のバッファを用意しており，10 本のコネクション全体で 3510 セグメント分のバッファが存在するが，使われるのは，最大で 1100 / 3510 = 31.3%，平均で 60 / 3510 = 1.7%である．このように TCP の受信側では高いスループットを達成するために大量のバッファが必要となるが，平均の利用効率は低く計算機のリソースとして有効利用されないことが分かる．

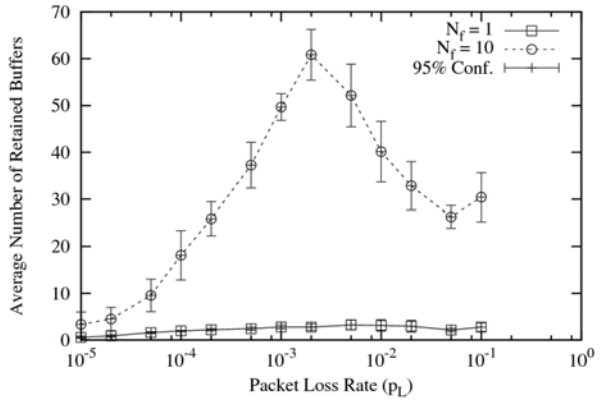

Fig. 5　Packet loss rate vs. the average number of retained buffers by the receiver.

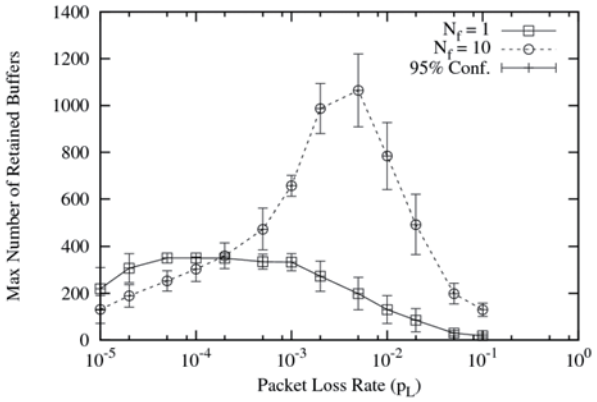

Fig. 6　Packet loss rate vs. the maximum number of retained buffers by the receiver.

3.3　送信バッファに関するシミュレーション結果

Fig. 7 は TCP コネクション数が 1 と 10 の場合について，パケット損失率を変化させた時の平均送達確認待ちバッファ数，Fig. 8 は最大送達確認待ちバッファ数を示している．TCP コネクション数が 1 の時，パケット損失率が小さい場合に最大送達確認待ちバッファ数はウィンドウサイズに近い 350 付近の値を取り，ウィンドウサイズ分一杯に送信していることが分かる．しかし，平均送達確認待ちバッファ数はパケット損失率が大きくなるにしたがい急速に小さくなる．これは輻輳ウィンドウによりデータの送信が抑えられることによるものと考えられる．先に説明したように TCP ではパケットの損失を検出すると，ネットワークの輻輳により損失が発生したものとし，

輻輳ウィンドウの大きさを絞ることにより新規セグメントの送信を抑える．TCP には別にウィンドウサイズがパラメータとして存在するが，二つのパラメータの小さい方が用いられる．

TCP コネクションの数が 10 倍に増えると平均送達確認待ちバッファ数も大きくなるが，10 倍となる訳ではない．これは，TCP コネクション 1 本当たりの平均的な帯域が減るためにウィンドウサイズ分先送りする必要がないことによると考えられる．Fig. 8 ではパケット損失率が小さい場合，最大送達確認待ちバッファ数は約 3000 で TCP コネクション数が 1 本の場合の 8.6 倍と 10 倍に近い値となっている．各 TCP コネクションで 300 セグメント程度，送達確認を待たずに先送りが行われていることを示している．送達確認バッファ数に関しては平均と最大の差は受信バッファの場合と異なり差は大きくなく，バッファの利用効率も小さいとは言えない．

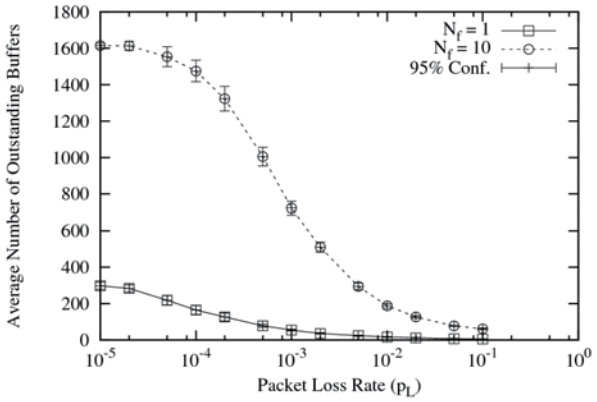

Fig. 7　Packet loss rate vs. the average number of outstanding packets.

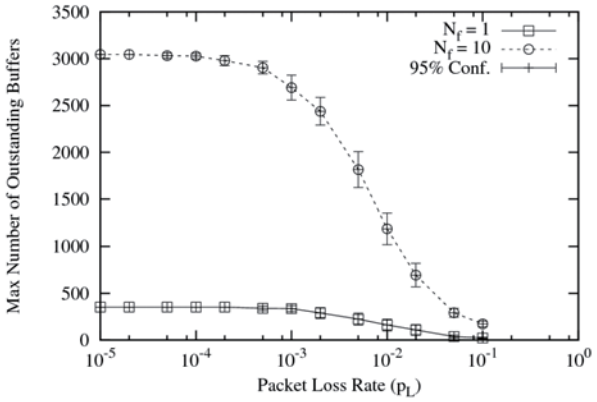

Fig. 8　Packet loss rate vs. the maximum number of outstanding packets.

4.　複数 TCP コネクションでの共通受信バッファ方式

4.1　共通受信バッファ方方式

3 章で TCP の受信側では高いスループットを達成するために大量のバッファが必要となるが，使用効率は低く計算機のリソースとして有効利用されないことを示した．これを解決するために複数の TCP コネクションで受信バッファを共有する方式が考えられる．TCP コネクションごとにバッファを設けず共有すればバッファの有効利用を期待できる．Fig. 9 はこの方式を説明する図である．各 TCP コネクションの受信側でバッファを共通にし，どの TCP コネクションもこのバッファを利用できるものとする．TCP の受信処理を行う前に，受信したセグメントの通番を調べ，飛びがあれば必要な保留バッファ数を計算する．もし，共通バッファに必要な保留バッファ数の空きがあればバッファを確保し，受信したセグメントを TCP の受信処理に渡す．もし，共通バッファが不足する場合は，受信したセグメントを TCP の受信処理に渡すことなく廃棄する．共通バッファ方式では TCP コネクション全体のウィンドウサイズの総和より小さい数のバッファを用意するので，条件によってはバッファ溢れが発生し，正しく受信したセグメントを廃棄することがある．しかし，このような廃棄が発生する確率が小さければバッファを共通化することにより受信バッファの有効利用を期待できる．

TCP では各コネクションにおいて受信側が保留可能なバッファ数を送信側に通知する告知ウィンドウの機能がある．共通バッファ方式の場合，各 TCP コネクションにどれだけ受信バッファがあるか確定しない．TCP の送信側は自身が管理するウィンドウサイズと告知バッファ数の小さい方を実際のウィンドウサイズとする．低い数値を告知バッファで通知すると共通バッファ溢れによるセグメント廃棄の発生を抑えられるが，ウィンドウサイズが抑えられるためスループットが低下する．ここでは，告知ウィンドウサイズは送信側が決めるウィンドウサイズより常に大きい値が通知されるものとする．

このような共通バッファ方式はデッドロックを生じる危険性がある．例えば Fig. 1 のシーケンス例で受信バッファが 5 個とした場合，S4 から S8 の 5 セグメントを保留するために 5 個の受信バッファを使用してしまうと，パケット損失により失われた S3 のセグメントを受信するためのバッファがないため，再送された S3 のセグメントが廃棄される．S4 から S8 の 5 セグメントは S3 のセグメントを受信しない限り上位層にデータを転送して受信

バッファを解放できないので，際限なく保留されることになる．このようなデッドロックを防ぐため，保留バッファ数には S3 分を予約として含める必要がある．上記の場合，S4 から S7 までのセグメントを受信すると，未受信の S3 を含め 5 個のバッファが保留され，S8 のセグメントは受信バッファがないため廃棄される．セグメント S3 が受信されると S3 から S7 までのセグメントのデータが上位層に渡され，保留されていた 5 個の受信バッファが解放されて再使用可能となり，デッドロックは生じない．Fig. 3 において，未受信で再送待ちの S3 セグメント分の受信バッファを保留バッファ数に含めるのはこのような理由による．

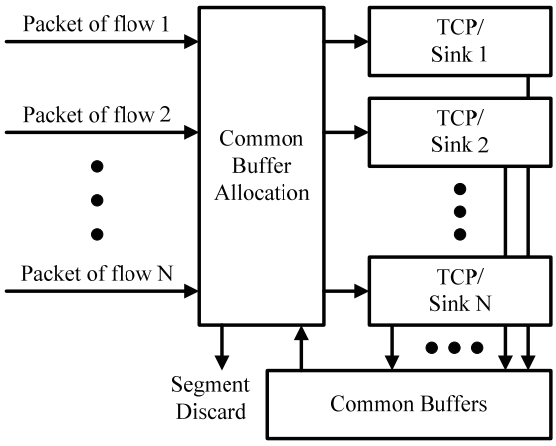

Fig. 9　Common buffers for the TCP receivers.

4.2　シミュレーションによる評価

ネットワークシミュレータ ns-2 を改造して，Fig. 14 に示す共通バッファの機能を組込み，共通バッファの効果を評価する．共通バッファ不足により生じるセグメントの廃棄率を評価項目として追加した．共通バッファの効果が期待できるのは，TCP コネクション数が大きい場合と考えられるので TCP コネクション数が 10 の場合について，共通バッファの大きさ (N_b) を 600，1000，1400 としてシミュレーションを行った．TCP コネクション 1 本当たりのバッファ数は 351 が必要で，TCP コネクション数が 10 であるから各コネクションで受信バッファを独立に確保すると 3510 個の受信バッファが必要であるが，共通バッファの大きさが 600 の場合は 17.1%，1000 の場合は 28.5%，1400 の場合は 39.9%に受信バッファ量を減少させていることになる．

Fig. 10 は共通バッファの大きさ (N_b) を 600，1000，1400 とした場合のパケット損失率とスループットの関係を示している．共通バッファ溢れによるセグメントの廃棄によりパケット損失率が 10^{-2} から 10^{-3} 付近でスループットの若干の低下が見られるが，共通バッファの大きさが 600 程度でもスループットへの影響は小さいと結果となっている．

Fig. 11 はパケット損失率と共通バッファ溢れによる受信セグメントの廃棄率を示している．パケット損失率が 10-2 から 10-3 付近でセグメント廃棄が発生していることが分かる．測定点を結ぶ線が垂直となっているのは，隣接する測定値が 0 でこれを対数目盛でプロットしていることによる．共通バッファの大きさを増加させることにより，受信セグメント廃棄の発生が減少しスループットへの影響が小さくなることが分かる．共通バッファの大きさ (Nb) を 2000 にするとセグメント廃棄は観測されなくなり，スループット特性は Fig. 4 と一致する．

Fig. 12 は共通バッファの大きさ (Nb) をそれぞれ 600，1000，1400 にした場合のパケット損失率に対する受信側で保留される平均バッファ数を示している．Fig. 5 の共通バッファを用いない場合に比べ，保留される平均バッファ数が大きくなっている．これは，回線におけるランダムなパケット損失に加え，共通バッファ溢れによる受信セグメントの廃棄が発生し，受信側でのセグメント保留の頻度が増えたためと考えられる．しかし，共通バッファの大きさを変えても大きな傾向の変化は見られない．以上から，共通バッファを用いることにより受信セグメントの廃棄は生じるもののスループットへの影響は小さく，この方式が有効であると言える．

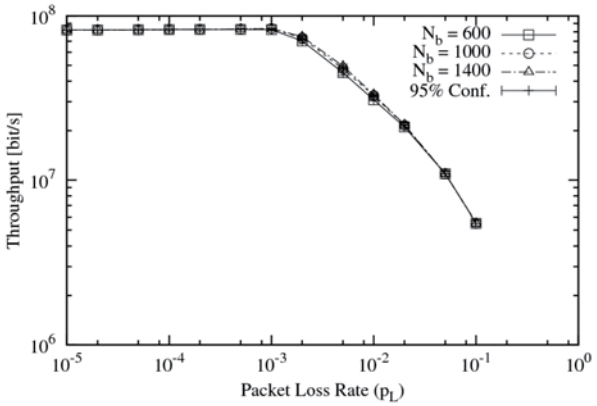

Fig. 10　Packet loss rate vs. throughput.

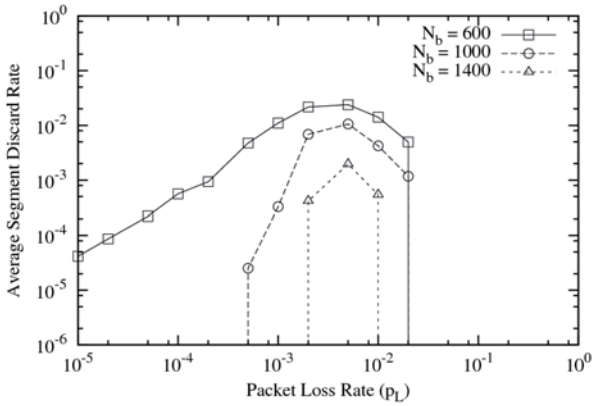

Fig. 11　Packet loss rate vs. segment discard rate.

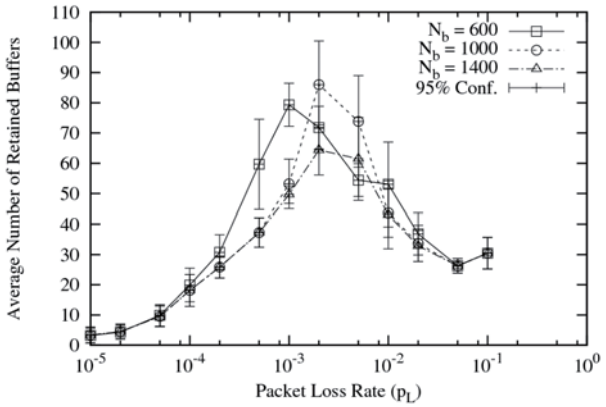

Fig. 12 Packet loss rate vs. the average number of retained buffers by the receiver.

4.　主記憶を使用する送信バッファ量の削減

4.1　補助記憶の利用による送信バッファ用主記憶の削減

この章では送信バッファに関して主記憶の利用削減を検討する．3 章で述べた受信バッファの場合，保留される受信バッファ数の最大値は大きいものの平均値は小さかった．このような場合，複数の TCP コネクションでパケットの廃棄が独立に生じるために統計的な多重効果を期待でき，受信バッファを共有することによりバッファ量削減効果が得られた．

送信側の送達確認に用いられる送信バッファの場合，Fig. 7 で示したように平均の保留数が大きく，最大値の差は少ない．このような場合，受信バッファのように複数の TCP コネクションでバッファの共有を行っても削減効果は期待できない．この理由は，送信側のバッファの場合，Fig. 1 から分かるように最小でも往復伝搬遅延時間（RTT）以上の間保留されることによる．送信側のバッファは再送に備えて保留されるので，再送によりバッファからの読み出しが必要となるのは新規のセグメント送信により送信バッファの保留が開始されてから，少なくとも RTT 以上経過してからとなる. 通常 RTT は 1ms 以上の値であり, 3 章で述べたシミュレーションでは 40ms である．このようにバッファへの書き込みから読み出しまでに ms 単位の時間を要することを考慮すると，送信バッファを高速の主記憶に設ける必要がないと言える．ここでは，Fig. 13 に示す補助記憶を利用する方式を検討する．組込みシステムにおいても最近はフラシュメモリを利用する場合が一般的となっており，送信バッファにフラッシュメモリのような補助記憶を利用することにより高速の主記憶の利用を削減できると考えられる．補助記憶を利用する場合，送信側では次のような処理を追加することになる．

(1) 新規のセグメントを送信する場合，主記憶の送信バッファのデータを補助記憶装置に書き込む.

(2) 主記憶にある最新の送信バッファはキャッシュとして残すが，新規セグメントの送信が一定数を超えたら一番古い送信バッファは解放し，主記憶の利用量を減らす．
(3) 再送が発生した場合，再送セグメントのデータがキャッシュに存在すれば直ちに再送を行う．キャッシュにない場合は補助記憶装置から読み出し再送を行うが，補助記憶装置からの読み出しに時間を要するため，再送が遅れる．
(4) 受信側からの送達確認を受信すると，補助記憶装置内で確認されたデータ用に使われていた領域を再利用可能とする．

このような補助記憶を利用する方式により，高速で小容量の主記憶を送信バッファに用いることを減少できると考えられるが，再送の場合に補助記憶装置から読み出すための時間が必要で，再送が遅れるために性能が低下する恐れがある．これを評価するためにシミュレーションを行った．送達確認用のバッファに補助記憶を利用する方式は従来考えられていないので，評価のために TCP の送信処理を改造する必要がある．

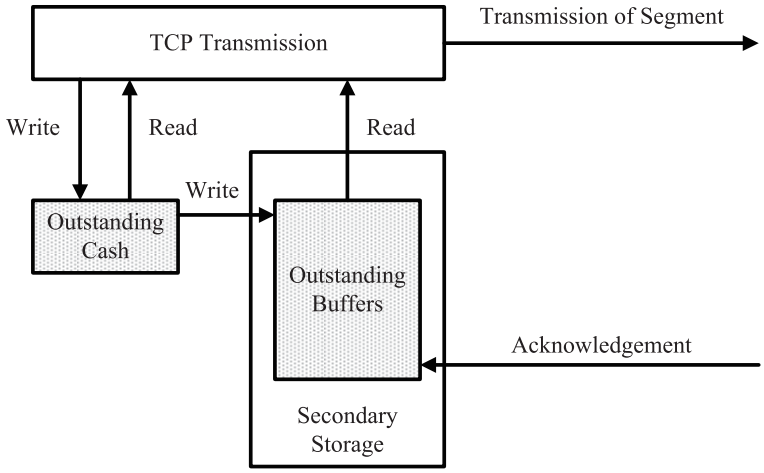

Fig. 13　Utilization of the secondary storage for TCP outstanding buffers.

4.2　シミュレーションによる評価

Fig. 14 は補助記憶を利用する方式を評価するためのシミュレーションモデルにおける追加部分を示している．

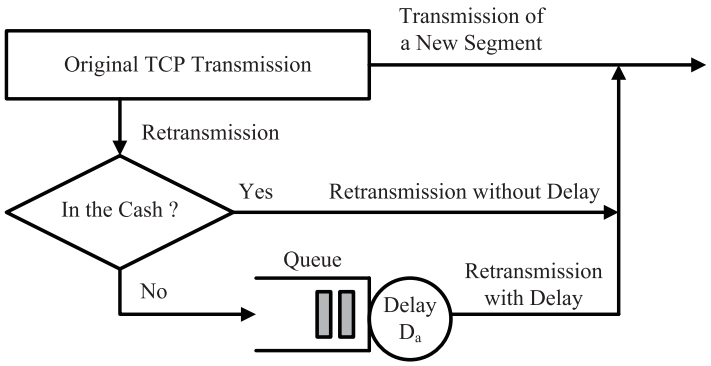

Fig. 14　Functions added to the simulation model for the scheme using the secondary storage.

ネットワークシミュレータ ns-2 の改造を少なくすため，オリジナルの TCP の送信処理においてセグメントの再送を行う場合に以下の機能を追加した．

(1) 新規送信したセグメントの内，最新の一定数 (Nc) のみをキャッシュに保持する．
(2) 再送するセグメントがキャッシュに含まれていれば遅延なく送信する．

(3) 再送するセグメントがキャッシュに含まれていない場合は再送セグメントを待ち行列に入れ，タイマーを起動する．タイマーの時間が補助記憶からの読み出しに要する時間 (Da) に相当する．
(4) タイマーが満了したら待ち行列から再送セグメントを取り出し送信する．待ち行列に再送セグメントが残っている場合はタイマーを再起動する．
(5) タイマー起動中にさらに再送セグメントが発生した場合は，これを待ち行列に入力し，以前の再送セグメントの送信がすべて完了した後にタイマーの満了を待って送信する．

補助記憶を利用する方式は複数の TCP コネクション間で共用する方式ではないので, TCP コネクション 1 本の場合で評価する. Fig. 15 は TCP コネクション数を 1 とした場合のパケット損失率とスループットの関係を示しており, ここでキャッシュの大きさ (N_c) を 10 とし, 補助記憶からの読み出しに要する時間 (Da) を 0, 100ms, 200m と変化させた場合を示している．補助記憶からの読み出しに要する時間が大きくなるとスループットが低下するが，100ms 以内であれば性能の低下は小さいと言える．補助記憶としてフラッシュメモリを使用する場合，読み出しに要する時間 (D_a) は 100ms よりも小さいと考えられ，補助記憶を用いて主記憶の利用を削減する方式は有効と言える．

Fig. 16 は補助記憶からの読み出しに要する時間 (D_a) を 200ms として, キャッシュの大きさ (N_c) を 10, 50, 200 と変化させた場合のパケット損失率とスループットの関係を示している．キャッシュを大きくするとスループットが向上するが，キャッシュの大きさとして 50 程度であれば十分な性能が得られることが分かる．補助記憶を用いない場合，主記憶に設ける送信バッファはウィンドウサイズ分の 351 個必要であるが，補助記憶を用いる場合は主記憶のバッファを 14.2%程度に削減でき，補助記憶を用いる効果があることを示している．

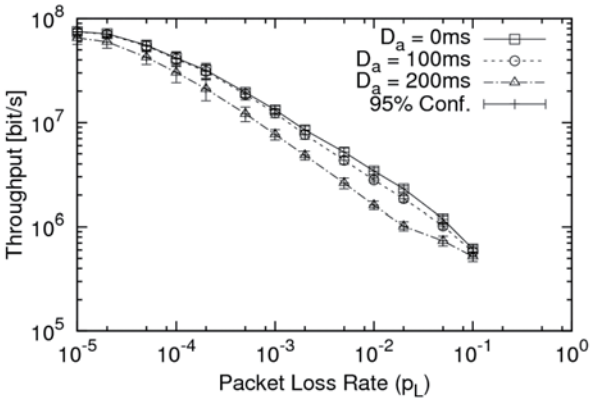

Fig. 15　Packet loss rate vs. throughput (N_c = 10).

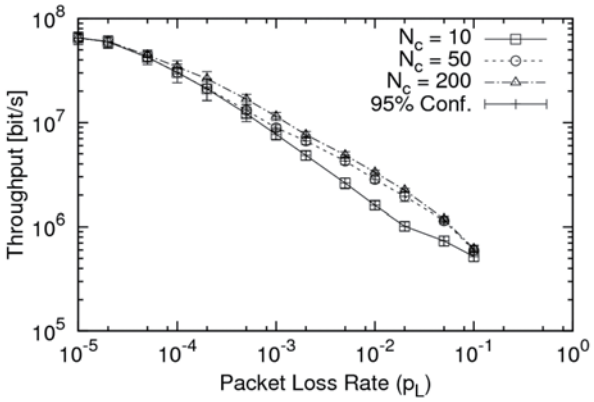

Fig. 16　Packet loss rate vs. throughput (D_a = 200ms).

このような送信バッファに補助記憶を用いる方式では，補助記憶から読み出す時間分再送が遅れるので，受信側での受信バッファの保留時間が長くなる．Fig. 17 は補助記憶からの読み出しに要する時間 (D_a) を 0, 100ms, 200m と増加させた場合のパケット損失率と受信側での平均保留バッファ数の関係を示している．ここでキャッシュの大きさ (N_c) は 10 と固定している．これから分かるように Da が大きくなるに従って平均保留バッファ数が大きくなる．このような傾向は 3 章で説明した受信バッファを共通化する方式に影響を与える恐れがある．

Fig. 18 は受信側の共通バッファの大きさ (N_b) を 1400 として，補助記憶からの読み出しに要する時間 (D_a) を 10ms と 100ms とした場合のパケット損失率とスループットの関係を示している．ここでもキャッシュの大きさ (N_c) は 10 と固定している．3 章の Fig. 10 では受信側の共通バッファの大きさを変化させても大きなスループットの差異は見られなかったが，Fig. 18 では補助記憶からの読み出しに要する時間 (D_a) が 100ms の場合には，スループットが大きく低下している．これから補助記憶からの読み出し時間は 10ms 程度以内に抑える必要があると言え，低速の補助記憶を用いる場合は受信バッファを共通化する方式との併用は避ける必要がある．

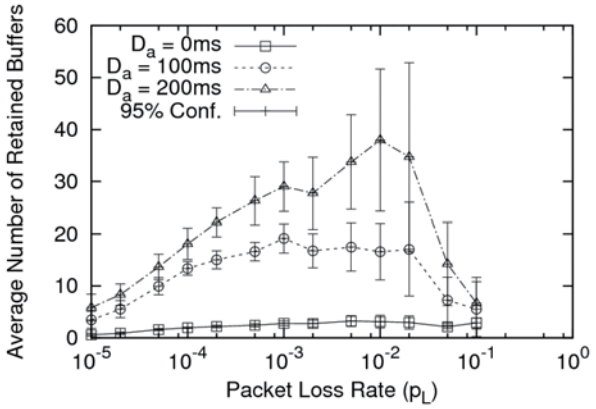

Fig. 17　Packet loss rate vs. the average number of retained buffers by the receiver (N_c = 10).

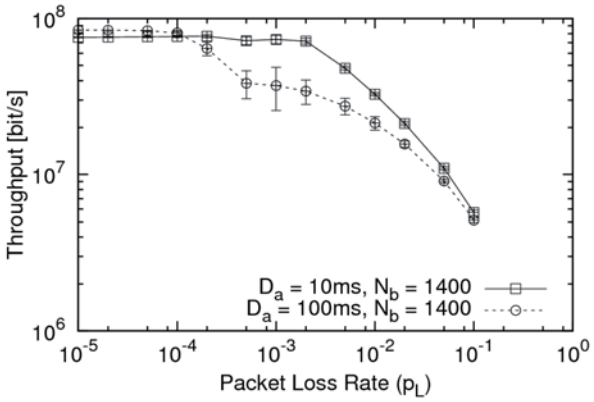

Fig. 18　Packet loss rate vs. throughput.

7. 結　　言

TCPにおいて高いスループットを得るためにはウィンドウサイズを大きくすることが必要でそのために大量の受信および送信バッファが必要となるが，主記憶容量の限られた組込み機器ではこれらバッファによる主記憶の利用が問題になると考えられる．本論文ではこれらバッファの利用効率について評価し，受信側のバッファについて有効利用されないことを示した．受信バッファの問題を解決する方式として複数のTCPコネクション受信でバッファを共通化する方式を検討し，シミュレーション評価により有効であることを示した．送信バッファに関しては保留される時間が往復伝搬遅延時間以上であることに注目し，補助記憶を利用する方式を提案し有効性をシミュレーションにより示した．

送信バッファの問題解決のために示した補助記憶の利用は受信バッファの問題解決にも有効と考えられ，今後方式を検討する予定である．本論文のネットワークシミュレーションはこれまで実績のあるns-2を使用したが，TCPの実装が厳密ではなくシミュレーション用に一部機能が簡略化されている．今後より実際のプロトコルに近いns-3[9]などによる評価を行う予定である．

謝　　辞

本研究は，福井工業大学特別研究費の助成を受けたものである．ここに記して謝意を表す．

文　　献

(1) W. Stevens, *TCP/IP Illustrated,* Volume 1: The Protocols (1994), Addison-Wesley.
(2) 松山公保，西田佳史，尾家祐二，トランスポートプロトコル, (2001)，岩波書店.
(3) “Network Simulator – ns-2”, http://www.isi.edu/nsnam/ns/ (参照日 2014 年 2 月 24 日).
(4) 鹿間敏弘，水野忠則, “Selective-Repeat ARQ における順序制御方式の提案と性能評価”，電子情報通信学会和文論文誌 B, Vol.J88-B, No.6 (2005), pp.1017-1028.
(5) T. Shikama, T. Watanabe and T. Mizuno, “Delay Analysis of the Selective Repeat ARQ Protocol with the Per-Flow Resequencing Scheme”, 情報処理学会論文誌, Vol.47, No.2 (2006), pp.369-381.
(6) T. Shikama, T. Watanabe and T. Mizuno, “A TCP-Aware Link layer Protocol Based on Selective-Repeat ARQ with No Resequencing”，情報処理学会論文誌, Vol. 47, No.2 (2006), pp.266-278.
(7) T. Shikama, “Evaluation of Mitigation to Bursty Packets by a TCP Proxy in a Wired and Wireless Network,” *International Journal of Informatics Society*, Vol.3, No.3 (2012), pp.95-102.
(8) V. Jacobson, R. Braden and D. Borman, “TCP Extensions for High Performance”, RFC 1323 (1999).
(9) “ns-3”, https://www.nsnam.org/ (参照日 2014 年 2 月 24 日).

（平成 26 年 3 月 31 日受理）