ABR 上で用いるトランスポート層プロトコルの提案

3F－4

五十嵐 健 古賀 祐匠 坂井 達彦 重野 寛 松下 温

慶應義塾大学

1 背景

近年インターネットにおいてデータ転送に用いられる TCP/IP はコンピュータ通信を支える重要なプロトコル群となっている.

一方,1996 年 ATM フォーラムにおいてデータ転送用のサービスクラスとして ABR(Available Bit Rate) が提案された [3]. これによって, ATM の高速・広帯域網をデータ転送の世界へ導入することができるようになった. しかし, ABR 上で TCP を用いた場合, 効率, 公平性の部分で問題が生じることが報告されている [2]. 本稿では TCP over ABR の問題点について考え, 問題点を改善するための新たなトランスポート層プロトコルを提案する.

2 TCP の制御

TCP Reno では送信側 TCP は一度パケットが廃棄されると, 廃棄されたパケットが遅れて到着することも考慮にいれて, 後続のパケットの到着によりつくられる同じ内容の Ack(重複 Ack:duplicate Ack) を 3 つ待った後に廃棄パケットの再送を行う (fast retransmit and recovery). この後再送パケットに対する Ack を受信すると cwnd を半分にする. これによってネットワークへのデータの流量を制限する. しかし, 1 ウインドウ中で複数のパケットが損失した場合などは重複 Ack を受け取ることができないので, 計測された RTT(Round Trip Time) の 2 倍以上の値が設定されている timeout によって再送を行う.

TCP の制御の特徴は重複 Ack や timeout によってパケットの損失を間接的に検出し, そのような間接的な情報を用いてフロー制御を行うことである.

3 ABR の制御

ABR では図 1 の様に, ユーザのデータ cell と共に RM(Resource Management) cell をネットワーク中に巡回させることで, 途中のスイッチやエンド端末によって現在のネットワークの状態が CI(Congestion Indication) ビットや ER(Explicite Rate) 値として RM cell にセットされ, 受信側で折り返され送信側へと送られる. 送信側では受け取った RM cell の情報を参照することで次の最適な送信レートである ACR(Allowed Cell Rate) を決定することができる.

ABR の大きな特徴は RM cell によって網の現在の状態を取得することができるので, TCP の様に間接的な情報を利用してフロー制御を行うのではなく, 直接的な情報を利用してフロー制御を行えることである.

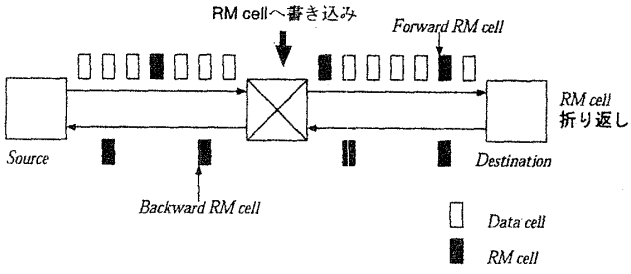

図 1: RM cell によるフィードバック

4 TCP over ABR の問題点

ABR 上で TCP を用いた場合, 2 つのフロー制御が存在する.

- TCP の間接的な情報を用いるウインドウベー

スのフロー制御

- ABR の直接的な情報を用いるレートベースのフロー制御

この両者の独立したフロー制御の衝突が性能低下に大いに関わってくる．

ABR を用いると CLR(Cell Loss Ratio) はかなり低く抑えることができる．しかし，ABR においてもコネクション開始時は RM cell を受け取るまでは網の状態に関係なく ICR(Initial Cell Rate) で送信するため，この間 (open loop) 網は加負荷状態となる．もし，頻繁にコネクションが開設されたり，途中のスイッチに十分なバッファが用意されていなければ ABR を用いていてもわずかながら cell 損を生じてしまう．一度 cell 損が発生すると TCP のフロー制御が働く．こうなると，この後 ACR が回復しても TCP によってデータの送信が制限されるので ABR が用意した帯域を利用できなくなってしまう．わずか 0.18 % の CLR でもスループットは 36 % も低下してしまう．さらに cell 損を起こした TCP コネクションはデータ送信が著しく制限されるため，公平性の面でも大きな問題を生じてしまう．

5 TCP と ABR の連携制御

上述したような問題を解決するために新たなトランスポート層プロトコルを提案する．ABR 上で用いるトランスポート層プロトコルとしては次の様な機能が必要になる．

フロー制御: ABR に連携した制御．但し，プロトコル部分のバッファに関するフロー制御が別途必要

損失回復: 損失は少ないことが予想されるため Ack ベース → Nack ベース

誤り検出: ビットエラーは AAL 5 でチェックするので上位プロトコルによるチェックは必要なし

公平性: ATM コネクション間の公平性は ACR に従ってデータを送信している限り ABR によって保たれる

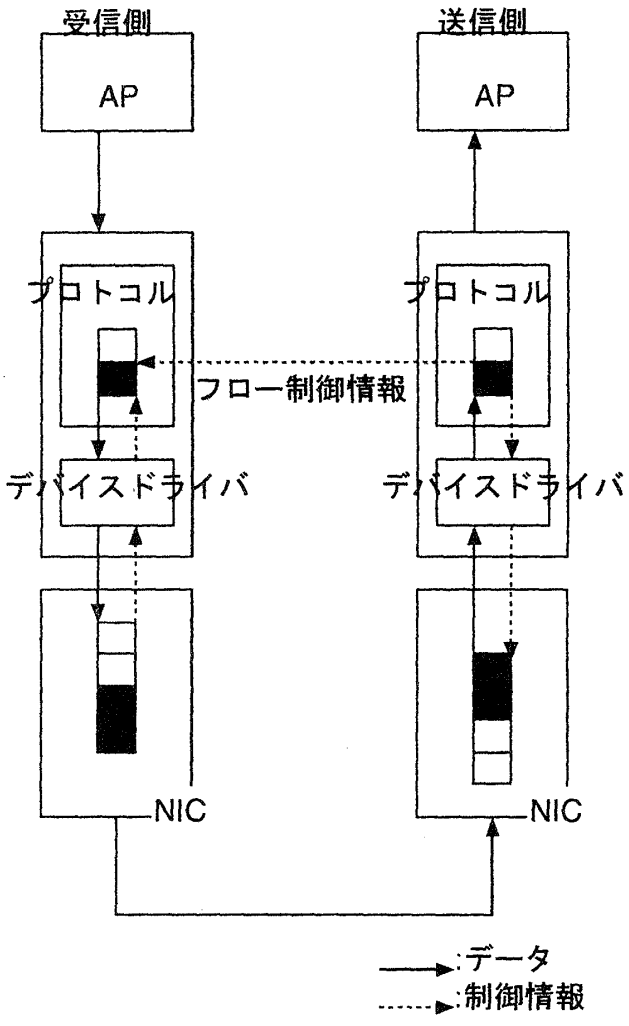

図 2: 機能ブロック図

5.1 フロー制御

ACR を API(Application Program Interface) を用いてトランスポート層に通知する方式の場合，ACR 値を通知するタイミングが大きな問題になる．

- RM Cell 到着毎 → 上位はソフトウエア処理中心のため処理が追い付かない
- 上位のソフトウエア処理が追い付く速さで ACR を通知 → いくつかの ACR の変化が無視される

どちらの方式をとってもトランスポート層側が ACR の急激な変化を見逃す恐れが生じる．

そこであらかじめ NIC バッファにデータを用意しておく方法を提案する．各プロトコルが処理可能なタイマ精度を Ta とすると，Ta の間に ABR が送信することのできる cell の最大数 Nm は，R を最大スループット，Nrc を Ta の間に送られる RM cell の数とすると

$$Nm = \frac{R \times 10^6 \times Ta}{53 \times 8} - Nrc$$

となる．つまり Ta 時間の間にいつも NIC バッファに Nm 以上のセルを溜めておけば，ABR 側では常に送信待ちのデータが存在することになり,ABR が用意した帯域 (ACR) を満たした送信が行える．

データ送信開始時に Nm 以上のデータを NIC のバッファ内に溜めた後は，Ta 時間毎に各コネクションについて送信したデータ cell の数を計測しておき，トランスポート層では送信した分のデータを NIC バッファに書き込めばよい．

例えばプロトコルが処理可能なタイマの精度 Ta = 1 ms ,R = 155 Mbps,,Ncr = 11 の場合，

$$\begin{aligned} Mc &= \frac{155 \times 10^6 \times 10^{-2}}{53 \times 8} - Nrc \\ &= 354 \end{aligned}$$

つまり 1 ms 毎に 常に Mc = 354 セル以上をプロトコルバッファに溜めておけば良いことが分かる．

しかし，この方式を用いると常にバッファにデータを溜めておくために RTT の揺らぎが大きくなってしまう．結果，timeout による誤再送も引き起こす可能性が生じる．そこで，なるべく timeout に頼らない損失回復機能が必要になる．

5.2 損失回復

ABR 上で用いるトランスポート層プロトコルの損失回復としては，

- 前述の影響を少なくするために timeout 再送に頼らない損失回復が求められる →Sack(Selective Acknowledgement) option(図 3 参照) を用いた再送方法
- 送信端末が ACR に従ってデータを送信している場合、セル損率は非常に少ないため損失回復処理を軽量化することができる → NACK ベースの損失回復 (図 4 参照)

5.2.1 TCP Sack option

RFC2018[1] で提案されているもので，図 3 にフォーマットを示す．この様にして，今までに受け取ったパケットをブロックとして Ack のデータ部分に記述することで，送信側は本当に損失が起こったパケットのみを再送することができる．

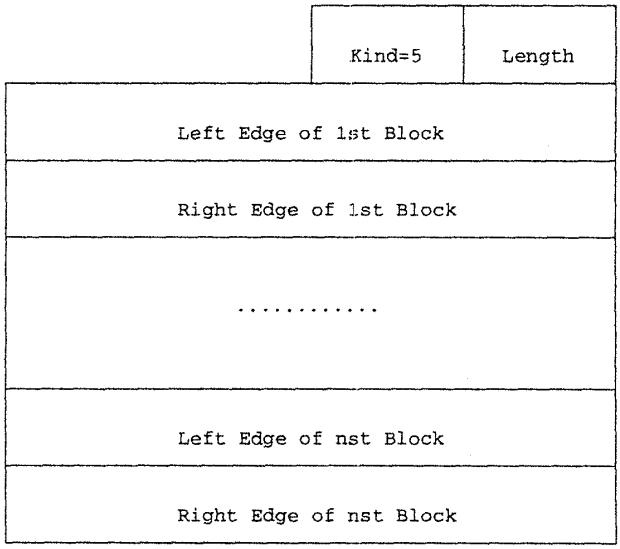

図 3: TCP Sack option

しかし，今回の提案では主に，送信側が受け取った Nack がどのパケットによって作られたものであるかを判断するために Sack option を用いた．

5.2.2 損失回復 (提案)

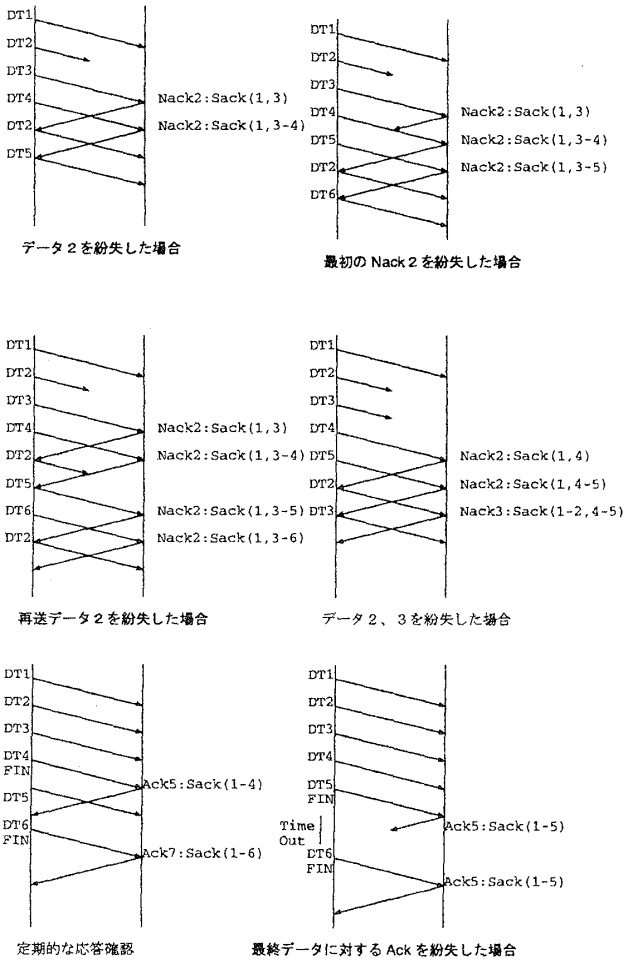

図 4: 損失回復 (提案)

図 4 に提案する損失回復順序を示す.

データ 2 の紛失: この場合次のような損失回復を行う.

- データ 3 の到着によって データ 2 の損失を確認.
- SACK(1,3) を含む Nack2 を返す.
- データ 4 が届いた場合,Sack(1,3-4) を含む Nack2 を返す. Nack は抜け落ちたデータを受け取るまで送り続ける.
- 最初の Nack2 を受け取った送信端末はデータ 2 を再送.
- 次に受け取る Nack2 は Sack(1,3-4) から再送前に送られたデータによって作られた Nack だということが分かるのでそのまま, データ 5 を送信する.

最初の NACK2 の紛失: この場合, 次のような損失回復を行う.

- 2番目の Nack を最初の Nack として扱う.

再送データ 2 の紛失: この場合次のような損失回復を行う.

- Sack(1,3-4) を含む Nack2 は, 再送前に送られたデータによって作られたものなので無視される.
- 次に受ける Nack2 は Sack(1,3-5) より, 再送後に送られたデータによって作られた Nack なので, もう一度 データ 2 を再送.

連続してデータを紛失した場合: この場合次のような損失回復を行う.

- Sack(1,4) を含む Nack2 によって, データ 2,3 の損失を確認.
- データ 2,3 の再送.
- Sack(1-2,4-5) を含む Nack3 を受け取るが, 最初の Nack2 が最初の Nack3 の扱いになるため, 無視される.

定期的な応答確認: 応答確認が得られなければ再送キューをクリアできないので送信側は受信側に対して定期的に応答確認を要求する. 特に最終データの場合, この要求は必須である. このデータを受け取った受信側はそれに対する Ack を必ず返す.

最終データの Ack を紛失した場合: この後送られるデータが無いのでタイムアウト再送によって最終データを再送する.

このような方法を用いることで最終データの Ack の紛失以外には timeout を発生させずに損失回復を行うことができるため, 常にバッファにデータを溜めておくことによる RTT の揺らぎの影響を受けなくて済むようになる.

5.3 受信側のプロトコルバッファのためのレート制御

ABR の制御は受信端末の NIC までであるため，受信端末のプロトコル部のバッファオーバーフローを防ぐ仕組みが別途必要になる．具体的にはバッファサイズがある閾値を越えたらレートを下げるように通知，ある閾値未満ならレートを元に戻すように通知する方法が必要になる．

- 送達確認ポーリングに対する応答パケットで受信バッファの占有率を送信側に通知 → 結果が反映されるまでに時間がかかる
- 受信端末が ER 値を指定するような API を用いれば，プロトコル部受信バッファ占有率が大きくなったときに ER 値を MCR にセットする → 素早い対処を行える

6 まとめ及び今後の課題

今回の提案はネイティブ ATM 環境下での新たなトランスポート層について行った．しかし，インターネットを考えた場合，100 % ATM のような環境は考えにくい．将来においても現在の様に様々な速度のネットワークが混在した環境が想像できる．そうした場合，特に速度差のあるネットワークが継るエッジルータにおいてはオーバーフローを引き起こすことが予想されるため，速度差のあるネットワークにも対応できるようなトランスポート層プロトコルを考える必要がある．

参考文献

[1] M. Mathis, J. Mahdavi PSC, S. Floyd LBNL, A. Romanow Sun Microsytems. TCP Selective Acknowledgment Options. RFC2018, IETF, OCT. 1996.

[2] M.Ajmone Marsan,A.Bianco,R.Lo Cigno,M.Munafo. TCP over ABR:Some Preliminary Simulation Results. Technical report, Dipartimento di Elettronica,Politecnico di Torino-Italy.

[3] The ATM Forum Technical Committee. Traffic Management Specification Version 4.0. Technical report, The ATM Forum, Apr. 1996.