アルゴリズム 67－6
（1999．3．15）

高信頼マルチキャストのための テーブル依存制御アルゴリズム

戚 廼箴, 徳田 雄洋
東京工業大学大学院 情報理工学研究科 計算工学専攻

要旨

インターネット上で効率の良い高信頼マルチキャストを行うには、フロー制御と輻輳制御が重要である。ユニキャスト通信方式のTCPの場合は、スライディングウィンドウやスロースタートなどいろいろな制御方式があるが、マルチキャストの場合は応答爆発などの問題があるためTCPの方式は使えない。本稿では、テーブル依存制御アルゴリズムを提案し、高信頼マルチキャスト環境でのオーバーフローと輻輳に対して、ネットワークの部分的な転送レートの変更を行うことで、無駄なトラフィックを減少させ、全体的に効率の良い受信ができ、マルチキャストセッション全体のトラフィックを減少させる効果を示す。

A Table-based Control Algorithm for Reliable Multicast Protocols

Naizhen Qi and Takehiro Tokuda
Department of Computer Science, Tokyo Institute of Technology

Abstract

Reliable multicast on top of IP multicast requires flow/congestion control. There are two reasons. One is to adjust to the effective bandwidth which makes packet losses and retransmission minimize, and the other one is to share the link bandwidth with other traffic.This paper proposes a table-based control algorithm to realize flow/congestion control needed for bulk reliable multicast transport protocols.

Our table-based control algorithm makes transmit rate of packets locally changeable, which leads to less useless traffic, when overflow and congestion happen to the network. Packets are discarded according to their priority which makes control routers abandon as many identical packets as possible.

1 はじめに

ネットワーク上で、ネットワークの状況に合わせ、パケット通信を効率良く行うために、受信端末をオーバーフローさせないためのフロー制御と、ネットワークをオーバーフローさせないための輻輳制御を行う必要がある。受信端末の能力を超える送信を続けると損失パケット分のネットワーク資源や送信端資源を無駄に使い、未受信パケットの再送を行うとすれば再送によるトラフィックが多くなる。

また、輻輳制御を行わないと、ネットワークが混雑し、他の通信が滞ってしまう。そのうえ、輻輳によるパケットの損失があるにも拘らず送信を続けると、廃棄されるパケットを転送する無駄があり、更にパケットの損失に対して再送を行うとすれば、再送パケットにより更にネットワークが混雑する。効率の良い通信を行うためにはフロー制御と輻輳制御を欠かすことができない。このような理由で、ユニキャストの通信においてフロー制御と輻輳制御が導入されている。

例えばTCPでは、スライディングウィンドウやスロースタートなどいろいろな方式が検討、実用化されている。しかし、1対多のマルチキャストでは送信者が決める1つの送信レートがマルチキャスト受信ツリー全体に共通して適用される。ネットワークの一部にオーバーフローや輻輳が起きたときに、制御のため送信レートを一律に下げると、問題の起こっていないネットワーク部分では処理能力より低いレートでパケットを転送することとなり、マルチキャストセッション全体の所要時間が長くなってしまう。更に、ネットワークの資源を充分使用できない。

本稿では、高信頼マルチキャストのオーバーフローと輻輳を解決・改善する方法のテーブル依存制御アルゴリズムを提案する。まず、第2節でマルチキャスト環境下でのフロー制御と輻輳制御の実現機構上の問題点を述べ、次に第3節で優先順位を用いたテーブル依存制御アルゴリズムでフロー制御と輻輳制御を行う方法を説明する。最後に第4節でシミュレーションの結果と評価を行う。

2 マルチキャストのフロー・輻輳制御

2.1 ユニキャストのフロー・輻輳制御技術

TCPの場合、フロー制御は受信側のバッファの空き容量を含んだACKを送信側にフィードバックし、送信側ではそれに基づいて送信量を決める。また、効率をあげるため、誤り再送制御における先行送信の制御と組み合わされ、スライディングウィンドウ方式となっている。更にパケット損失を輻輳発生とみなして、損失発見後の先行送信量を小さく押えるスロースタート輻輳制御が組み合わされ、フロー制御、輻輳制御や再送制御が一体となって行われている。

2.2 マルチキャストのフロー・輻輳制御技術

マルチキャストにおいてはTCPの場合と異なり、多数の受信者から一つの送信者への応答が集中する応答爆発という問題がある。これにより送信者とその回りのネットワークに、パケットの転送や応答処理の大きな負担がかかる。従って、TCPのフロー制御や輻輳制御方式をマルチキャストにそのまま使うことができない。

マルチキャストのフロー制御と輻輳制御としては、いろいろな研究が行われている。具体的に次のような方法がある。

1. 送信者にフロー制御情報や輻輳制御情報をフィードバックし、その情報を元に送信量を加減する方法。送信量制御をすることにより、無駄なトラフィックを減少させることができる。応答爆発問題を避けるため、応答を分散処理 [4] したり応答を遅延させる方法 [8] などがある。しかし、フィードバック情報が送信者まで転送されて、送信量制御が始まるまでの遅延が大きく、その間はネットワークに過大負荷が続く。

2. マルチキャスト経路を利用した方法 [7]。データを複数のバンドを用いて、各々を異なる重なった受信ツリーに送り出す。受信者は、全てのグループに参加して受信すれば元のデータを全て受信できる。この方法は、パケットの重複受信の可能性があり、その上、無駄なトラフィックも多い。

3. オーバーフローや輻輳のためパケットの損失が一定レベルを上回ると、端末がマルチキャストグループから退去し、それによって状況を軽減させる機構を持つ。この場合、退去した受信者への送信は後からユニキャストにより行う。そして、オーバーフローや輻輳状況が改善された場合、受信者は受信を再開する。従って、マルチキャスト受信ツリーが頻繁に変わる可能性がある。しかし、一般にマルチキャストの経路制御は、受信者の加入や退去を要求してから新しい受信ツリーが正しく設定されるまでに時間がかかる。小範囲の変更は制御にすぐ反映できるが、広範囲の変更は遅れがかなり大きくなる。

3 テーブル依存制御アルゴリズム

3.1 基本となる考え

テーブル依存制御アルゴリズムでは、送信者だけがフロー制御と輻輳制御を行うのではなく、マルチキャスト受信ツリー全体がフロー制御と輻輳

制御を行う。オーバーフローになった受信端末によってあるいは転送途中で輻輳が起こったリンクの直前のルータによって従来廃棄されていたパケットが、上流でより早い段階で廃棄する方法である。そして、グループの各パケットに優先順位という概念を導入し、フロー制御と輻輳制御を行う各制御ルータが、低い優先順位を持つパケットを廃棄することにより、各ルータができるだけ同じパケットを廃棄することが可能になる。その結果、マルチキャストによる再送をより効率良くさせる効果がある。

テーブル依存制御アルゴリズムは二つのアルゴリズムからなる。パケット先捨てアルゴリズムとCTテーブル制御アルゴリズムである。パケット先捨てアルゴリズムは転送レート制御を行う制御ルータを選ぶためのアルゴリズムである。CTテーブル制御アルゴリズムは、制御ルータがパケットのCTテーブルに依存した優先順位に基づき、廃棄するパケットを決めるためのアルゴリズムである。CTテーブルはパケットのシーケンスナンバーとパケットの優先順位を関連させるテーブルで、それを持つ必要があるのは、グループの送信者だけである。グループセッションのパケット総数はセッションが始まる前に、マルチキャスト受信ツリー全体に知らせる。

3.2 パケット先捨てアルゴリズム

パケット先捨てアルゴリズムでは、各インターフェースが整数となるレート制御変数 *limit* を用いて制御を行う。リンク L_0 が輻輳が起こったとき、直接の上流のルータ R_0 が輻輳状況を改善するため、L_0 への出力インターフェースのキューを空にする。ルータ R_0 が上流に向かって転送レート制御の要求メッセージ (CRメッセージ) を送り出し、以下のアルゴリズムに従って上流での制御ルータを決める。

- [*step*0] : L_0 の上流のルータ R_0 が送信者である場合、送信者がリンク L_0 のインターフェースに対し、送信レート制御を行う。
- [*step*1] : L_0 の上流のルータ R_0 が受信者である場合、R_0 で転送レート制御を行う。中間ルータ (受信ツリーにある送信者以外のルータのこと) である場合、[*step*2] へ。
- [*step*2] : R_0 のマルチキャストグループに使われる下流リンクが L_0 だけの場合、R_0 の上流のリンクを L_0 とし、[*step*0] へ。下流リンクが2つ以上ある場合 [*step*3] へ。
- [*step*3] : R_0 のマルチキャストグループに使われる下流リンクの転送レートのうち一番大きいものを $Max_{rate} = Max(Rate)$ とする ($Rate$ は各リンクの転送レートの集合である)。Max_{rate} を上流のリンク L_1 で R_0 の上流のルータ R_1 に送って、[*step*4] へ。
- [*step*4] : R_1 が下流から受け取った Max_{rate} と L_1 の転送レート $rate$ と比べて、$Min(Max_{rate}, rate)$ を $rate$ とする。[*step*5] へ。
- [*step*5] : R_1 のマルチキャストグループに使われる下流リンクが一つしかない場合、L_1 を L_0 とし、[*step*0] へ。違う場合、R_1 を R_0 とし [*step*3] へ。

受信端末がオーバーフローとなる場合でも、その端末の上流リンクを L_0 とし、同じアルゴリズムを使い制御ルータを決める。

制御ルータがそのインターフェースを介して下流からのフロー制御と輻輳制御の要求を受け取るたびに、要求を受け取ったインタフェースに対し、下で説明するレート変更手順により *limit* の値を変化させ、転送レート制御を行う。

3.2.1 レート制御変数の変更手順

CRメッセージを受けとった制御ルータがそのインターフェースの *limit* の値を今までの半分にし、転送レートを下げる。次のCRメッセージを受けとらないままタイムアウトしたら、つまり今の転送レートはネットワークの処理能力以下の可能性があるときに、*limit* の値を1だけ増やし、リンクの転送能力あるいは端末の受信能力に適した転送レートに少しずつ合わせていく。

ネットワーク上のトラフィック状況はダイナミックに変わるので、今まで空いていたリンクが急に混雑することがある。上流の *limit* が下流のリンクのそれより小さいときに、下流のリンクの *limit* の値をそれ以上いくら増やしても、実際

の転送レートが増えないためにオーバーフローや輻輳が起こらないが、その後上流のリンクの *limit* の値が増え転送レートが急に増えると、下流ではまた輻輳が起こり、*limit* の値を半分にしただけでは制御ができない状況も起こりうる。ネットワーク上のオーバーフローと輻輳に対しては素早く対応する必要がある。上流のリンクが急な転送レートの変更に対応できるように、上流からの転送レートが小さいときに、下流の制御ルータのインターフェースの *limit* の値が大き過ぎる値に増えることを防ぐ必要がある。上流からの *limit* が増えないときに、リンクインターフェースの *limit* 値を最初にタイムアウトする直前の値の倍までしか増やさないように制限する。こうすれば、上流の *limit* の値が下流のそれより小さくなってくるときに、下流の実際の転送レートの値が前の値の倍より大きくなるのを防げる。逆に、上流の *limit* の値が大きくなるときに、下流の *limit* の値の増加に制限を加える必要はない。

3.2.2 パケット先捨てアルゴリズムの利点

パケット先捨てアルゴリズムにはお主に2つの利点がある。

1. パケットの先消去

   オーバーフローとなった端末や輻輳の起こったリンクの直前のルータでパケットを廃棄するのではなく、その上流にある一番近い受信者または分岐を持つ中間ルータでレート制御を行って転送レートを制御する。その結果、一律な送信レートでパケットを送信させるのではなく、他の受信にできるだけ影響を与えない制御ができる。その上、無駄なトラフィックが減少できる。

2. 受信影響の局所化

   直接の上流に分岐を持つ中間ルータが存在する場合、制御によって受ける影響をできるだけ小範囲に制限できる。他の分岐の下流の受信をそのまま維持できるために、最小範囲で転送レートの変更を行い、処理能力の不均一なネットワークでも、高い処理能力を持つリンクや高い受信能力を持つ端末を十分利用できる。従って、ネットワーク資源を充分活用させる。

3.3 CT テーブル制御アルゴリズム

マルチキャスト方式による再送を行うため、各制御ルータができるだけ同じパケットを廃棄した方が、無駄なトラフィックが少なく、より良い効率が得られることは明らかである。CT テーブル制御アルゴリズムでは、同じパケットを廃棄させるために、パケットに優先順位と言う概念を導入し、各制御ルータである優先順位以下のパケットを廃棄するようにする。

送信者はパケットを送信する時に、シーケンスナンバー以外に、CT テーブルを用いて決めた優先順位もパケットにつける。受信端末ではパケットのシーケンスナンバーを使ってファイルを組み立て、制御ルータではパケットの優先順位に基づき、パケットを廃棄するか転送するかを決める。

各制御ルータがパケットを廃棄するか下流に転送するかを決めるため、各ルータの各インターフェースが、転送レート制御変数 *limit* を持ち、シーケンスナンバーが CT テーブルサイズ分のパケットの中 (実際に送られていなくても) に、*limit* の値以下の数のパケットしか転送しない。そして、優先順位で判断するため、転送可能の優先順位の下限を表す *limit* の値が大きいルータが廃棄するパケットは *limit* の値の小さいルータでも必ず廃棄される。例えば、CT テーブルの大きさが8で受信者 A の上流リンク X のインタフェースの *limit* の値が8で、A の下流リンク Y のインターフェースの *limit* の値も8のとき、シーケンスナンバーが16から23までの8つのパケットを全部受信者 A の下流に転送できる。そして、X の *limit* の値が6になって Y の *limit* の値が4となっているとき、16から23までの8つのパケットの中の低い優先順位を持つ2つのパケットが X で廃棄され、受信者 A に6つのパケットを受信できる。しかし、Y 以下のネットワーク部分に転送するのは X とは関係なく、16から23までの8つのパケットに対して廃棄する4つのパケットを決めるが、すでに2つのパケットが X の時点で廃棄されていたので、Y では2つのパケットだけを廃棄する。

このようにCTテーブル制御アルゴリズムでは、廃棄するパケットを相対的に決めるのではなく、絶対的に独立的に廃棄するパケットを決める。それによって、各制御ルータで同じパケットを廃棄することが可能となっている。パケットが転送レート制御を行うルータに到着する前に、転送するパケットと廃棄するパケットを決めることができる。

CTテーブルのサイズをLとすると、*limit*変数の値の範囲は$0 \leq limit \leq L$のようになる。

次のコードで、2をCTテーブルのベースにしたCTテーブルを作ることができる。そのとき、CTテーブルのサイズは$2^N = L$となる。(c << 1 : cを2倍する ,i >> j: iをj ビットシフトする)

```
for (int i = 0; i < L; i ++) {
    c = 0;
    for (int j = 0; j < N; j ++) {
        c = (c << 1) | ((i >> j) & 1);
    }
    CT[i] = c;
}
```

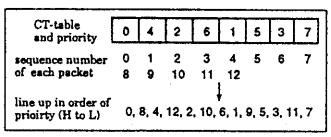

図 1: サイズ8のCTテーブル

このアルゴリズムを用いて、大きさ8のCTテーブルの中身を図1に示す。

送信者はCTテーブルを用いてパケットのシーケンスナンバーに基づいて、$CT[seq \bmod L](= P(seq)$ とする) 式を用いて優先順位を決める。

まず、1つの定義がある。

- シーケンスナンバー*seq*に対応した$P(seq)$の値が小さければ、小さい程このシーケンスナンバーを持つパケットの優先順位が高い。つまり、もし$P(a) > P(b)$ならば、シーケンスナンバーaを持つパケットの優先順位がシーケンスナンバーbのパケットの優先順位より低い。

送信者だけが持つCTテーブルには主に二つの特徴がある。CTテーブルの大きさをLとするときに

1. $\forall x,\ 0 \leq x < \frac{L}{2}$のとき,$CT[x] = CT[x+\frac{L}{2}] - 1$が成り立つ
2. $a, b \in seq, a > b$ かつ $CT[a \bmod L] = CT[b \bmod L] \Longrightarrow P(a) > P(b)$
3. CTテーブルの中に同じ優先順位は2つ以上存在しない。

パケットが13個ある場合、優先順位降順で並べた結果を図1に示す。パケットの優先順位が*limit*の値より大きくなければ、パケットが下流に転送される。下の疑似コードを参照。

```
if (CT[seq mod L] < limit)
        transmit(seq)
else
        discard(seq)
endif
```

3.3.1 CTテーブル依存アルゴリズムの特徴

2をベースにした大きさ8のCTテーブルを使い、8つのパケットを送信する場合、*limit*の値にしたがって、廃棄するパケットと転送するパケットは図2のようになる。この図から以下のようなことが分かる。

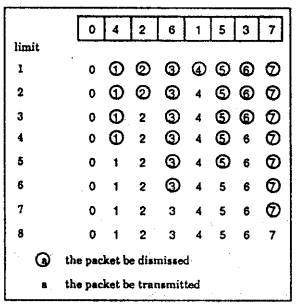

図 2: limitの値と廃棄するパケット

1. CTテーブルの中身は、優先度の値の各Pが偏在することなく全体的に平均にばらまかれ

ていて、バランスのよい数字の配列となっている。例えば、制御変数 *limit* が CT テーブルの半分となるときに、上流から転送されてくる 2 つのパケットのうち一つを下流に転送し、一つを廃棄することとなる。

2. 各制御ルータのインターフェースの *limit* の値が同じ場合、同じパケットが廃棄されることになる。*limit* の値が違っても、値の大きいところで廃棄されるパケットは *limit* の値の小さいところでも必ず廃棄される。

3. 各制御ルータのインターフェースの転送レートがこのインターフェースの *limit* の値だけに依存し、上流リンクからの転送レートと関係なく、独立に廃棄するパケットを事前に決められる。

本テーブル依存制御アルゴリズムでの転送レート制御変数 *limit* は上流の転送レート制御変数の値の変更とは独立で、上流からの転送レートが下流のリンクの処理能力を越えたときに、各リンクに合うような転送レートで通信を行うことが簡単にできる。依存性をもつ制御変数を用いる場合では、下流の転送レートは上流の転送レートに従って変わる。下流のリンクがレート制御をしたときに、上流のリンクもレート制御を始めると、下流のリンクの転送レートがそのリンクの処理能力以下になってしまい、ネットワーク資源が無駄になってしまう。

3.4 テーブル依存制御アルゴリズムの再送機構

3.4.1 再送タイミング

同じ優先順位を持つパケットが同時に同じマルチキャスト受信ツリーに転送されるのを防ぐために、再送はノーマルセッションが終了してから行う。複数のパケットが同じ優先順位を持つと、制御ルータで全部廃棄されあるいは全部転送されるため、フロー制御と輻輳制御に混乱を起こす可能性があるためである。

マルチキャスト受信ツリーの葉となる受信者は、通信に使うトラフィックを無駄にしないために、グループセッション[1]のパケットの受信が全て終ったら、直ちにマルチキャストグループから退去し、マルチキャストされた再送パケットをその受信者がこれ以上受けとらないようにする。

[1]グループセッション: ノーマルセッションと再送セッション全体のこと

3.4.2 再送順序

オーバーフローあるいは輻輳が起きたときに、制御ルータではテーブルに依存した優先順位に基づいて廃棄するパケットを決めるため、パケットの優先順位が低ければ、このパケットの再送を必要とする受信者が多い。そのため、再送はシーケンスナンバー順ではなく、低い優先順位を持つ (廃棄される可能性が高い) パケットから順序に再送を行う。シーケンスナンバーと共に再送順序から得た優先順位をつけて再送パケットをマルチキャストする。

オーバーフローと輻輳のため転送量制御を行っている制御ルータのインターフェースで、優先順位の低いパケットが共通に捨てられる可能性が高いため、マルチキャスト方式による再送において無駄なトラフィックを少なくできる。同じ領域で、何箇所かのリンクのインターフェースの *limit* の値が CT テーブルのサイズより小さければ、たとえそれぞれ違う値になっていても、*limit* の値が一番大きいインターフェースで廃棄されるパケットは他のインターフェースでも必ず廃棄される。つまり、小さい *limit* の値を持つところでは、大きい *limit* の値を持つインターフェースに比べ、より高い優先順位を持つパケットを廃棄している。グループセッションのパケットを全部受けとった受信者が直ちにグループから退去するため、*limit* の値が大きいインターフェースで廃棄されたパケットを早く再送したほうが、大きい *limit* を持つインタフェースの下流の受信者が早く全パケット受信を完了し、グループから退去できる。

3.5 CT テーブルサイズ

CT テーブルのサイズはノーマルセッションと再送セッションとは関係しないが、効率的なフロー制御と輻輳制御に関係する。

送信者がパケットを送り出す量を一定にすると、リンクの転送レート (R) は転送レート制御変数 *limit* の値と CT テーブルの大きさで決まる。

各パケットの大きさは1kバイトで、CTテーブルのサイズがLが総パケット数より小さく、一秒ごとにx個のパケットが届くとすると、

$$R = x \times \frac{limit}{L}(kB)$$

となる。各インターフェースの$limit$の値が、リンクの状況によって変化して端末とリンクの処理能力に適切な転送レートを求める。$limit$の値が最大になるとき、上流から流れて来るパケットをそのまま下流に転送しリンクの転送レートが一番大きくなる。CTテーブルサイズが大きければ、$limit$値の変化の余裕もあり、Rの値も細かくコントロールができ、$limit$の値をネットワークの処理能力に合わせた転送レートに変化させる可能性も高い。

より効率的に、より柔軟性を持ったフロー制御と輻輳制御を行うため、CTテーブルのサイズはマルチキャストグループのパケット総数より大きなものを使う。

つまり、ベースが2となる場合、CTテーブルのサイズLを

$$L \geq sum(packets),\ \exists n\ L = 2^n (n > 1)$$

を満足するLの最小値となるものにする。

4 シミュレーションによる評価

シミュレーションを行い、2をベースにした適切サイズを用いたCTテーブルを使う効果を示す。図3のようなシンプルな仮想ネットワークを使い、シミュレーションを行った。一番転送処理能力が高いリンクの転送処理能力を1とし、他のリンクの相対的な転送処理能力をリンクの上に振ってある。

4.1 シミュレーションの結果

テーブル依存制御アルゴリズムを利用したフロー制御と輻輳制御システムとして、CTテーブルサイズを16、及び適切サイズを用いた制御システムに対してそれぞれ、パケット総数が1000、2000、5000の場合について、それぞれシミュレーションを行った。

ノーマルセッションが終る時点で、半分以上の受信者に受け取られなかったパケット数の再送必要パケット総数に対する平均割合を図4、受信者全員がグループの全てのデータを受け取るまでのトラフィック総量の平均値を図5に示す。

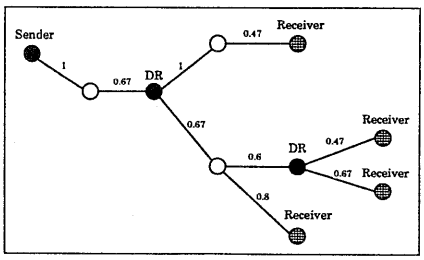

図 3: 実験用仮想ネットワーク

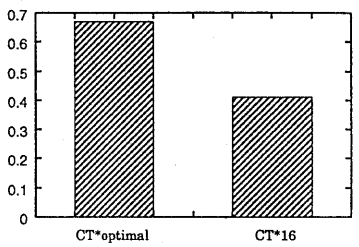

図 4: 半分以上必要とされるパケットの平均割合

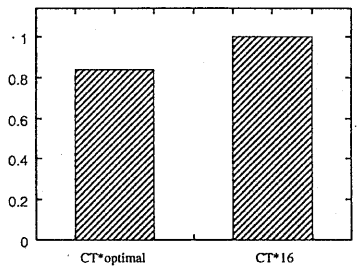

図 5: 平均総トラフィック

4.2 評価

シミュレーションを行った結果から、以下のことが分かった。

1. ノーマルセッションが終る時点で、再送すべきパケットの数がほぼ同じであるが、適切サ

イズを使う場合、全体的に再送セッションの65% ～ 70% を占めるパケットが半分以上の受信者に受けとられていないもので、サイズ16のテーブルを用いた場合の1.75倍となる。 従って、マルチキャストによる再送では、適切サイズを用いたテーブルの場合、無駄なトラフィックの生成が減少することが分かる。

2. CTテーブルの大きさが16のときに、半分以上の受信者から再送要求されるパケットの数は適切テーブルサイズを使う場合に比べ約40% 少ない。その原因は、CTテーブルサイズが16のとき、各リンクの転送レートが16種類しか選択できず、一定時間毎に転送レートが前の時点の転送レートより約6% づつ粗く増加されるため、適切サイズのCTテーブルを使う場合より、輻輳を起こしやすく、出力インターフェースのキューの中の全てのパケットの廃棄が頻繁に起こるからである。

3. セッション全体の総トラフィックを比較すると、適切サイズテーブルの場合はサイズが16の場合の約82% となる。小さいサイズのCTテーブルを使うと、各リンクの転送レートの変化幅が大き過ぎるため、リンクの転送処理能力あるいは受信端末の受信能力に合わせた転送レートでパケットを転送することが難しくなる。結局多くのトラフィックを使うことになる。

以上より、適切サイズを用いたCTテーブルのテーブル依存制御アルゴリズムの場合、柔軟性を持った細かい転送レートの変更ができ、無駄なトラフィックの少ないパケット転送ができることが分かる。

5 まとめ

ネットワーク上で、他のパケットと共存するため、マルチキャスト環境のフロー・輻輳制御が重要であるが、TCPで用いるフロー・輻輳制御方式はマルチキャストには適用できない。本稿では、マルチキャスト環境での従来の方法と欠点を分析し、テーブル依存制御アルゴリズムを利用した制御方法を提案した。より上流の制御ルータで、適切サイズを用いたCTテーブルに依存したパケットの優先順位を利用しパケットを廃棄することにより、効率的なマルチキャストによる再送ができ、無駄なトラフィックが減少できる効果を示した。

参考文献

[1] S.E.Deering, "Host Extensions for IP multicasting", Jul-01-1989,RFC 1112

[2] V.Jacobson, "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM '88, Pages. 314-328, August 1988

[3] B.N.Levine, S.Paul, and J.J.Garcia-Luna-Aceves, "Organizing Multicast Receivers Deterministically According to Packet-Loss Correlation",

[4] S.Paul, K.K.Sabani, J.C.Lin, and S.Bhattacharyya, "Reliable Multicast Transport Protocol(RMTP)" IEEE Journal on Selected Area in Communications, April 1997, Pages 407-421

[5] S.Pingali, D.Towsley, and J.Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols.", In Performance Evaluation Review, volume 22, pages 221-230, May 1994

[6] W.Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", Jan 1997, RFC 2001

[7] L.Vicisano, J.Crowcroft, "One to Many Reliable Bulk-Date Transfer in the MBone" HIP-PARCH 1997, June 1997 http://www.cs.ucl.ac.uk/external/L.Vicisano/index.html

[8] 山内長承, 城下輝治, 佐野哲央, 高橋修, "高信頼マルチキャストにおけるフロー・輻輳制御" http://www.trl.ibm.co.jp/projects/rmtp/index.htm