社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE

予備親探索機能を有したアプリケーションレベルマルチキャスト

國近 洋平† 甲藤 二郎† 大久保 榮††

† 早稲田大学大学院理工学研究科 〒169-8555 東京都新宿区大久保 3-4-1
E-mail: †{yohei, katto}@katto.comm.waseda.ac.jp,

†† 〒169-0051 東京都新宿区西早稲田 1-3-10
E-mail: ††sokubo@waseda.jp

あらまし 本稿では単一ソースノードからツリー状に構成されるアプリケーションレベルマルチキャストにおいて、各ノードが親ノードの離脱に備えて効率よく予備親を探索するアルゴリズムを提案する。従来の手法ではマルチキャストツリーから離脱するノードがそのノード以下のサブツリーに対してメッセージを送信し、新たな親の探索を促すことになる。これに対して、各ノードがあらかじめ親の候補となるノード情報を保持しておくことにより、効率的なツリー再構築を行うことを目標とする。

キーワード アプリケーションレベルマルチキャスト Peercast

Application Level Multicast with Backup Parent Searching Function

Yohei KUNICHIKA† Jiro KATTO‡ Sakae OKUBO††

† Graduate School of Science and Engineering,Waseda University
3-4-1 Okubo,Shinjyuku-ku,Tokyo,169-8555 Japan
E-mail: †{yohei,katto}@katto.comm.waseda.ac.jp

†† 1-3-10 Nishi-Waseda, Shinjuku-ku, Tokyo, 169-0051 Japan.

E-mail : ††sokubo@waseda.jp

Abstract In this paper, an efficient algorithm to look for backup parents in preparation of parent leaving is proposed for application level multicasting whose topology is constituted in the shape of a tree from a single source node. In conventional methods, a parent node which leaves from a multicasting tree transmits a message to all the nodes in its sub tree, and each child node starts searching for its new parent. In our proposal, each child node aims at performing efficient tree reconstruction proactively by holding the node information which indicates parent candidates.

Keyword Application Level Multicast Peercast

1. はじめに

近年、ADSL や FTTH などブロードバンドの普及に伴い動画や音声などのマルチメディアコンテンツが増加してきた。これらはテキスト主体のコンテンツに比べサイズが大きいものが多く、サーバやネットワークにかかる負荷が懸念されている。従来のクライアント-サーバモデルではサーバは要求を受けた数だけフローを流す必要があるからである。

この問題を解決する手段として IP マルチキャストが知られている [1]。IP マルチキャストはネットワークの有効利用という点に関しては非常に優れている。しかし、現在のインターネットのインフラストラクチャの対応の面から考えると、ネットワーク全体で使用するに至るまでの普及は難しいという問題がある[2]。

これに対して近年、アプリケーションレベルマルチキャスト（以下 ALM と呼ぶ）という技術が注目を集めてきている [2,3,4,5,6,7,8]。メンバ管理、パケットの複製、転送はすべてオーバーレイネットワーク上で仮想的に接続されたホストがユニキャストで行う。このため、既存のネットワークの変更は不要であり、IP マルチキャストと比べて実現が容易であるというメリットはあるが、その一方では、ピア同士のリレーでパケットが転送されるために接続が不安定になるというデメリットがある。

具体的なデメリットの１つにホストの離脱が挙げ

られる。あるホストがネットワークから離脱してしまうと、そのホストが中継しているすべてのホストはデータの受信が不可能になってしまう。このとき、該当ホストはネットワークの再構築を試みるが、これまでに提案されている多くの ALM では、上位ホストの離脱が確定してから新たな中継ホストの探索を開始している [2,3,4,5,6,7,8]。

これに対して、本稿では、各ホストがあらかじめ予備の親となるホスト情報を保持しておくことにより、上位ホストの離脱によるツリー再構築を効率よく実現するためのアルゴリズムを提案する。

2. Peercast

ALM には大別してツリー型とメッシュ型がある [2,3,4,5,6,7,8]。ツリー型は単一ソースノードによる動画配信などに用いられ、メッシュ型はテレビ会議などに用いられる。本稿では、ツリー型の ALM を検討対象とする。

ツリー型の ALM の 1 手法として Peercast が知られている [8]。Peercast は非営利サイトによって提供されている ALM ソフトウェアであり、高い認知度がある。本稿ではこの Peercast を従来方式として位置づけ、この Peercast に対して予備親探索機能を付加することによる効果を検証する。以下に Peercast の基本的な動作について述べる。

2.1 Join アルゴリズム

図 1 において、①新規にセッションに参加したいホストはまずソースノードへ接続要求である join メッセージを送信する。ソースノードがそのホストを受け入れられるならば子として受け入れ、セッションを開始する。②もし駄目な場合は現在接続している子の中から 1 つを選択し、親候補として接続要求してきたホストへその情報を返す。仮に接続要求を受信したホストが制限なしに接続を許すと、子の数が非常に多いホストの上り帯域は非常に圧迫されてしまう。そこで 1 つのホストが接続を許可する数に制限を設け、そのような問題を解決する。新規ホストを受け入れられない場合とは接続している子の数が上限に達している状態である。③,④,⑤以降はこの動作が繰り返される。⑤の join メッセージが受け入れられたとすると、ソースに対しひ孫の位置で接続されることになる。

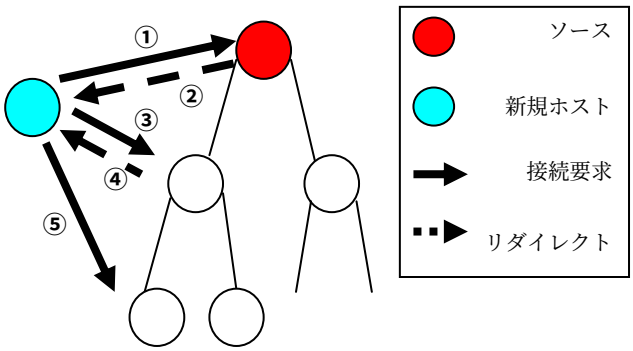

図 1: 新規参入ノードの接続までの流れ

2.2 Leave アルゴリズム

ツリー上のホストが離脱した場合、そのツリーが複数の部分木に分割されることになる。こうなると離脱したホスト以下の部分木はソースから分断されているのでデータを受信することができない。そのためあるホストが離脱するときは leave メッセージを子供に送信することにより離脱する旨を通知し、ツリーの再構築を促す。

この場合、Peercast では leave メッセージをすべての子孫に送信し、各々が独立に再 join を行う方法と、直下の子にだけ leave メッセージを送信、再 join させ、孫以下のホストは何もしないという方法がある。一般的には後者の特性が良いことが知られている。

また、再 join 先にどのホストを選択するか、という問題がある。Peercast はメンバ管理を行っている特殊なホストは存在しないため、確実に分かるホスト情報はソース、もしくは自分の親のみである。これも一般的には後者の利用が望ましい。図 2 において、あるホストが離脱する際には、①leave メッセージに自分の親の情報を載せて直下の子供に送信し、②受信したホストは祖父にあたるホストに再 join を行う。

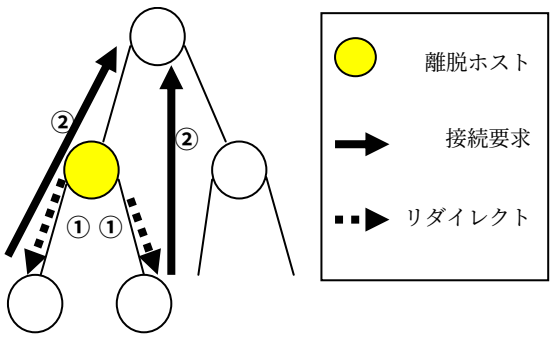

図 2: 離脱から再接続までの流れ

2.3 Dead アルゴリズム

ネットワークの輻輳などが原因で leave メッセージを正しく受信できなかった場合、離脱ノード以下の部分木はデータを受信できない状態が継続する。従ってこのような状態を早急に検知し、ツリー再構築を行うことが重要になる。Peercast では、接続関係にあるホスト間で定期的に HeartBeat メッセージ(以下 HB メッセージ)をやり取りすることで互いの存在を確認しあっている。

そして、一定期間内に子が親からの HB メッセージを受信できなかった場合、親を”dead”状態にあると判断し、新たな親の探索を開始する。この場合 2.2 節のような事前の通知はなく、子ノードは祖父の情報を持ち得ないため、ソースノードへ再 join を行う。

逆に、親ノードが子ノードからの HB メッセージを受信できなかった場合は、子を”dead”状態にあると判断し、確保しておいたセッション帯域を開放する。

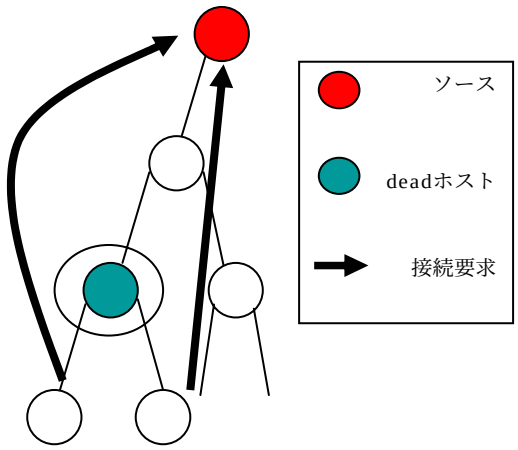

図 3: 親の dead から再接続までの流れ

3. 予備親探索機能を備えた経路管理

2 章に示したように Peercast では親ホストが離脱する直前、あるいは離脱した後に新たな親の探索を開始するため、ツリーの再構成に時間がかかるという問題がある。よって、もしネットワークが輻輳していた場合にはさらに大幅な遅延が発生してしまい、例としてビデオのストリーミング配信を行っている場合、受信端末のバッファ内再生可能時間の間に親を見つけることができずに、画面が頻繁にフリーズしてしまう、というケースが起こり得る。このような状況を回避するために、以下にあらかじめ適切な予備親を見つけておき、親ノードの離脱に対して迅速な経路の更新を実現するアルゴリズムを示す。

3.1 拡張 Join アルゴリズム

図 3 において、①新規参加ノードは Peercast 同様ソースノードへ join メッセージを送信する。受け入れが可能であれば従来同様の処理を行うが、②不可能な場合、接続している子の中から複数個を選択し、その情報を返す。③新規参加ノードはすべてのノードに対し join メッセージを送信する。それ以降は join メッセージを受け取った各々のノードが受け入れ可能ならばその旨を通知し、④不可能であれば接続しているこの中から 1 つを選んでその情報を返す。最終的にははじめにソースノードが選択した数の親候補が与えられることになる。

新規参加ホストは最初に受け入れ可能となったホストを親と見なし、改めて接続要求を送信し、それ以降受け入れ可能となったホストを予備親として登録しておく。こうすることにより複数個の中からより早く接続可能なホストを親にしつつ予備親を見つけることができる。

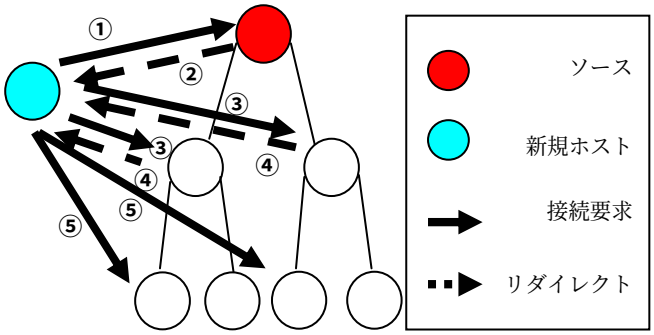

図 4: 新規参入ノードの接続と予備親探索の流れ

3.2 拡張 Leave アルゴリズム

親ホストの leave 及び dead 時の再 join 先は登録しておいた予備親となる。しかし予備親を探索していた時と実際に予備親へ join メッセージを送信する時では時間が経過しているため、予備親が実際には受け入れられないという場合も起こり得る。そのときは予備親が接続している子の中から 1 つを選択し、その情報を返し、与えられたホストに対して再 join を行う。

つまり予備親として登録はするが確実に次の親になってくれるとは限らない。

3.3 予備親がいない場合の対応

3.1 及び 3.2 より、一度予備親に再 join を行うとそのノードの予備親はなくなってしまう。また、予備親と

して登録していたホストが離脱してしまう場合も十分に有り得る。もし予備親がいなかった場合、従来の Peercast のアルゴリズムをそのまま用いることとする。すなわちあるホストが離脱する際には親の情報を leave メッセージとして子に送信し、予備親がいる場合は予備親へ再 join、いない場合は祖父にあたるホストへ再 join することになる。

dead アルゴリズムも同様で、予備親がいない場合はソースノードへ join メッセージを送信する。ここで受け入れられなかった場合は 3.1 に示したように複数の候補を返し、新たな親と共に予備親も入手することができる。

4. 性能評価

提案アルゴリズムの性能評価のため、ns-2(network simulator ver.2)を用いて[8]において提案されている Peercast との比較を行った。具体的には join メッセージを送信してから実際にデータを受信するまでの遅延時間を計測した。join メッセージには

1. 新規参入時
2. 親ホストからの leave メッセージ受信時
3. 親ホストが dead と判断時

の 3 種類がある。

4.1 シミュレーション結果

4.1.1 ネットワークの設定

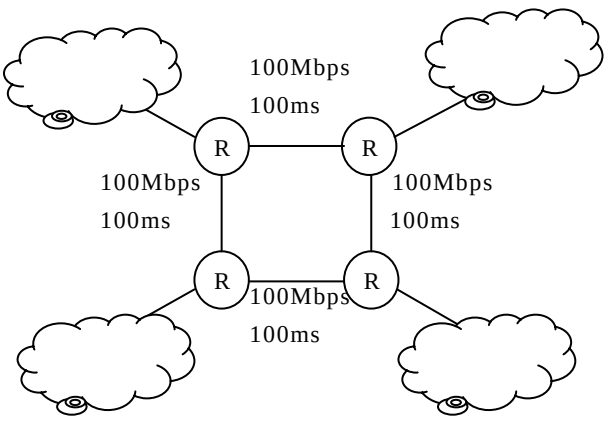

図 4; ネットワークトポロジ

図 4 のようなネットワークトポロジを用いる。R は中継ルータを表す。雲部はドメイン内部を想定してルータとホストで構成される。具体的なドメイン内部のパラメータは、

- ルータ数→5
- ホスト数→random(全体では固定)
- リンク容量→ルータ-ルータ 100Mbps
  ホスト-ルータ 10Mbps
- リンク遅延→ルータ-ルータ 10～50ms
  ホスト-ルータ 10ms

とした。また、

- セッション帯域→1Mbps
- 予備親の数→1

とした。

4.1.2 結果

join メッセージを送信してからデータパケット受信までにかかった時間の平均を図 5~図 8 に示す。本章の冒頭に示した通り新規参入時、親ホストの leave 時、dead 時の 3 パターンがある。

図 5、図 6 は子の上限 3、図 7、図 8 は子の上限 4 の結果である。また図 5、図 7 は参加ホスト数 100 での実験結果、図 6、図 8 は参加ホスト数 500 である。また、図 9 及び図 10 として、子の上限ノード数 4 の場合における制御メッセージの数を示す。

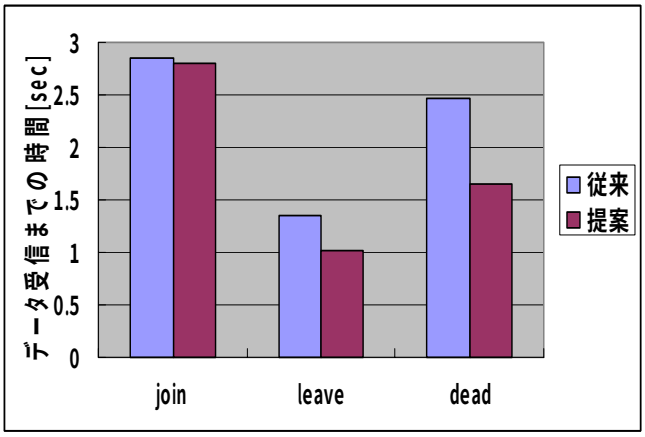

図 5: データ受信までの時間(上限 3 ノード,100 ノード平均)

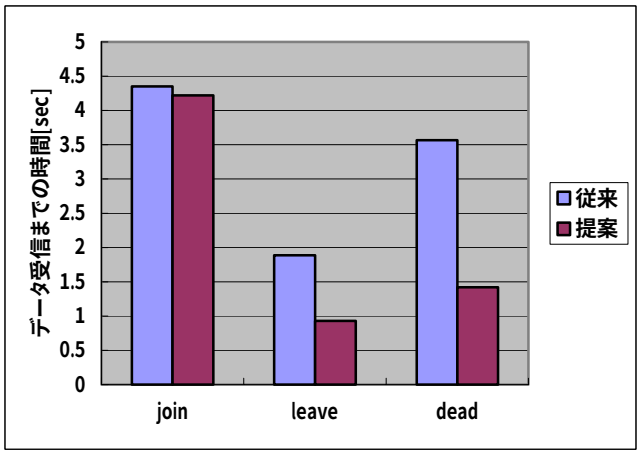

図 6: データ受信までの時間(上限 3 ノード,500 ノード平均)

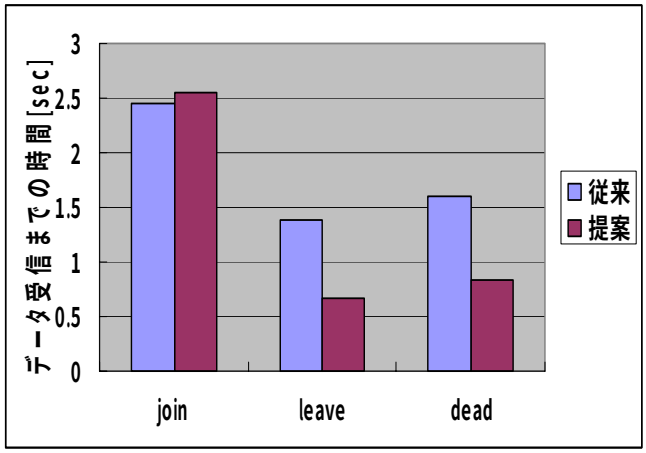

図 7: データ受信までの時間(上限 4 ノード,100 ノード平均)

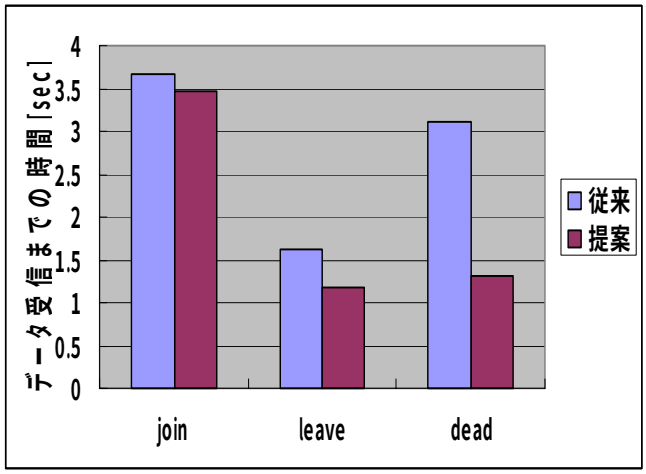

図 8: データ受信までの時間(上限 4 ノード,500 ノード平均)

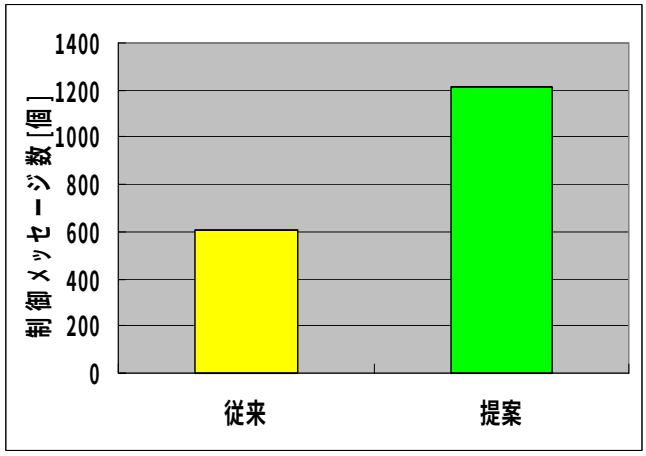

図 9: 制御メッセージ数の比較(100 ノード)

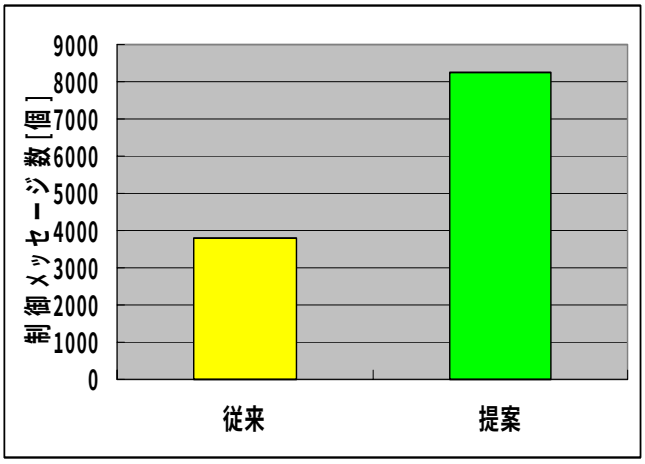

図 10; 制御メッセージ数の比較(500 ノード)

図 5～図 8 すべての場合において提案方式は従来方式と比較して、join 時の時間はほぼ等しく親ホストの leave 及び dead 時からデータを受信するまでの時間、すなわちツリー再構築にかかる時間が削減されている。以降はそれぞれの図を詳しく比較して考察する。

まず子の上限を変化させたときの結果を考察する。図 5 と図 7 はホスト数 100 のときの結果であるが、従来方式､提案方式ともに図 5 の時間が長くなっている。これは子の上限が少ないと接続要求を受け入れられない可能性が高まり、リダイレクション通知が増加傾向にあるからである｡リダイレクション数が増加すれば、親が確定するまでのメッセージのホップ数が増加するため、データ受信までの時間増加につながる。子の上限は多い方が良いが、その分上り帯域を多く使用することになる。

図 7 と図 8 は子の上限は同じで規模が違うときの結果だが、ほぼ同様の傾向が見られた。予備親を探索することにより親ホストの離脱時及び dead 時において再接続までの時間が削減できている。特に dead 時は 50%前後削減されている。これは予備親なしの場合、ツリーの最上位に位置するソースに join メッセージを送信するため、ツリーの長さに比例して実際に接続できるホストが見つかるまでのリダイレクション回数が増加する傾向があるためである。それに対して提案方式はツリーの各所に分散された予備親に接続要求を送信するので従来方式に比べて効率が良い。

また、leave 時も dead 時ほどではないが削減されている。従来方式では親ホストが離脱するときは祖父へ接続を試みる。親ホストが離脱するので少なくとも 1 つのホストはそのまま接続できることになるが、他のホストは上限を越えてしまう可能性が高い。それに対して提案方式はそれぞれのホストが異なるホストへ再接続を試みる。しかも予備親として登録した時点でそ

のホストは子を受け入れる余裕があったことは確定している。結果として提案方式の方が良い結果を示せている。

図 9 について、制御メッセージの数は従来方式と比べ 2 倍程度増加している。ここでいう制御メッセージとは join 及び leave に関するメッセージである。これについては提案方式は予備親を探すため、新規参入時で複数の join メッセージを送信する。予備親数 1 の場合はそれだけでおよそ 2 倍近く join メッセージが送信されることになる。それに加えて予備親として登録されたホストが離脱する際には子へ通知するなど、予備親の管理に関するメッセージがあるためにこのような結果となった。予備親の数が増加するごとに制御メッセージ数も比例的に増加していくため、予備親の数は慎重に選択する必要がある。

5. まとめ

本稿では、アプリケーションレベルマルチキャストにおける予備親探索アルゴリズムの提案を行った。提案アルゴリズムでは新規参入時にあらかじめ予備親情報を保持しておくことにより親ホストの離脱によるツリー再構築の時間削減を行う。従来の予備親を用いないアルゴリズムに対し新規接続の時間は同等のままで、leave メッセージを受信した場合で 20～50%、leave メッセージが受信できなかった場合は 50%前後の時間削減となり、提案アルゴリズムの有効性が確認できた。

今後の課題として、3.3 に示したように現在のアルゴリズムでは予備親がいない状況ができてしまうため、そのような場合に新たに予備親を探すアルゴリズムを追加したい。また、今回の実験では子の上限を全てのホストに対して一律に定めていたが実際の環境はホストごとに異なる場合もあり、上限に達する前に上りの帯域が飽和してしまう可能性がある。そうならないようにセッション帯域を小さくしてしまうと逆に帯域を持て余しているのに低レートでしかデータを受信できないホストが生じてしまう。解決策としてはホストごとに動的に子の上限を変えるアルゴリズムを適用する、あるいは階層符号化したデータをツリー構成に合わせて適応的に配信する[9][10][11]、などが考えられる。また、図 9 に示したように本提案では予備親を探索するためのメッセージが従来方式と比べて増えることになる。このことが数千、数万ノード規模のインターネット放送網を想定した場合、ネットワークにどの程度の負荷を与えるかの検証も必要になる。

それらを踏まえた上でさらに効率の良い予備親探索アルゴリズムを模索したい。

謝辞

この研究は、TAO 委託研究課題”通信ネットワーク利用放送技術の研究開発”のサポートによる。

参 考 文 献

[1] Dave Kosiur 著, 苅田幸雄訳: "マスタリング TCP/IP IP マルチキャスト編", オーム社開発局, 東京, 1999.

[2] J. Jannotti, D.Gifford, K.johnson, M.Kaashoek, and J.O’Toole: ”Overcast:Reliable multicasting with an overlay network”, *Proc. of the Fourth Symposium on Operating Systems Design and Implementation*, pp.197-212, 2000.

[3] S.Zhuang, B.Zhao, A.Joseph, R.Katz, and S.Shenker, “Bayeux:An architecture for scalable and fault-tolerant wid-area data dissemination” *Proc. of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 2001.

[4] Y.Chawathe, “Scattercast: An architecture for internet broadcast distribution as an infrastructure service” *PhD Thesis, University of California,Berkeley*, 2000.

[5] D.Pendarakis, S.Shi, D.Verma, and M.Waldvogel, “ALMI: An application level multicast infrastructure” *Proc. of the 3rd USENIX Sysposium on Internet Technologies and Systems (USITS)*, pp. 49-60, 2001.

[6] Y.Chu, S.G.Rao, and H.Zhang, “A case for end system multicast” *in Measurement and Modeling of Computer Systems*, pp.1-12, 2001.

[7] P.Francis, “Yoid: Extending the internet multicast architecture”, white paper, 1999. http://www.aciri.org/yoid/.

[8] H.Deshpande, M.Bawa, H.Garcia-Molina, “Streaming Live Media over Peers”, *Tech.Rep.2001-31, Stanford University*, 2001.

[9] S.McCanne, V.Jacobson and M.Vetterli, "Receiver-driven Layered Multicast", *Proc. of ACM SIGCOMM '96*, Aug.1996.

[10] 山口, 本間, 甲藤: "TCP 親和性を持つ受信者駆動型階層化マルチキャストとレート制御に関する一検討", 信学技報 IN2002-105, Nov.2002.

[11] G.I.Kwon and J.Byers: "Smooth Multirate Multicast Congestion Control", *Proc. of IEEE INFOCOM 2003*, Mar.2003.