広域分散ストリーミングのための精度保証スケジューリングアルゴリズムの開発

研究代表者　　藤　田　　聡　　広島大学大学院工学研究院　教授

1 はじめに

Cisco VNI の試算によると、2013 年のインターネット上のトラフィックのおよそ半分は VOD(Video on Demand)やライブストリーミングなどの動画データによって占められると予想されている。そのような大量のトラフィックを実現可能とするため、現在利用されている多くの動画配信サービスでは、その基盤アーキテクチャを C/S 型から P2P 型へと進化させつつある。実際、膨大な数の視聴者に向けて質の高い動 画を安定して提供するには、高い性能をもつ特定のサーバへの過度な負荷の集中を避け、多数のノードに配信に貢献してもらうことのできる P2P 型アーキテクチャへの移行は不可避である。これまでに、ZigZag に代表されるツリー型システムや CoolStreaming/DONet に代表されるメッシュ型システム、CoolStreaming に代表されるハイブリッド型システムなどが実装され、その性能が実証的に評価されてきた。しかしこれまでの研究の多くでは、比較的安定したノードたちからなるバックボーンネットワークを構築し、その上にストリームパケットをベストエフォート的に流すという手法がとられており、動画品質の保証にまでは至っていない。今後ネットワーク上で主流となると期待される課金型の P2P 動画配信をビジネスとして成立させるためには、動画の精度保証は重要な課題であり、そのための新しい技術の開発が強く求められている。

本研究ではこの課題に対し、スケジューリングアルゴリズムの改良によって精度保証を実現するというアプローチをとることとした。近似アルゴリズムやオンラインアルゴリズムなどの分野で得られている知見を応用・発展させることで、既存手法に比べて飛躍的な性能向上を目指すことを具体的な目標とした。

2 関連研究

P2P型ストリーミングシステムにおけるストリーミングレートの最大化は、多くの研究グループで取り組まれている重要な課題のひとつである。Kumarらは、完全結合されたネットワーク上での最大ストリーミングレートの上限に関する考察を行った[KUMAR07]。システム上にn台のノードの他にメディアサーバが存在するものとしよう。メディアサーバのアップロード容量をu_s、ノードiのアップロード容量をu_iとそれぞれ記すことにする。このとき、最大ストリーミングレートの上限r^{max}は

$$r^{max} = \min\{ u_s,(u_s+U(P))/n \}$$

のように与えられる。ここで$U(P)=\sum_i u_i$である。この式は、r^{max}がメディアサーバのアップロード容量を超えないことと、平均アップロード容量を超えないことを意味している。この上界を達成するためのいくつかの方式がこれまでに提案されている[KUMAR07,GU08,MASSOULIE07,LI04,NGUYEN08]。しかしこれらの方式の性能は、ノード数が増えるにしたがって、転送オーバーヘッドや遅延などの点で大きく悪化することが知られている。さらには、ネットワークが完全結合されているという設定は現実的ではなく、固定された次数のもとで同様の上界を実現することが強く求められている。

Liuらは複数のツリーをフルメッシュ型オーバーレイ上に埋め込むという方針を使ってP2Pビデオストリーミングシステムにおけるストリーミング容量の考察を行った。具体的には、下記の二つの次数に関する仮定のもとで考察を行っている：1) ひとつのツリーの中での各ノードの次数が定数で制限されていること。具体的なアルゴリズムとしてSnowballアルゴリズムが示されている[LIU08]。2) 各ノードの次数の合計が定数で制限されていること。具体的なアルゴリズムとしてBubbleアルゴリズムが示されている[LIU10]。LiuらはBubbleアルゴリズムとMutualcastの現実的な実装法として、Cluster-Treeアルゴリズムも提案している。このアルゴリズムは二つの階層からなっており、任意のεに対して最適ストリーミングレートμの1-ε倍のストリーミングレートを実現している。しかしこの手法ではツリーをメンテナンスするためにトラッカーが用いられているため、完全に分散型というわけではない。加えてこの手法では各クラスタの平均アップロード容量がμに十分近くなくてはならず、各クラスタのノード数は大きくなる場合が多い。そのため各クラスタ内でのファンアウト数の増大に伴う遅延が大きくなり、ストリーミングレートの最大化と遅延の短縮とのトレ

ードオフをどうするかが課題となっている。Zhaoらはソースから各ノードへのストリーミング容量が最後の1ホップでのアップロード容量によって決まり、各ノードが受け取る際の容量が高い確率で$(1-\varepsilon)\mu$となるようなアルゴリズムを示した[ZHAO11]。しかしこのアルゴリズムでは各ノードの次数制限はなされていない。

3 提案アルゴリズム１

3-1 基本アイデア

システム中の各ノードには 0 から n-1 までのユニークな ID が与えられているものとしよう。与えられたストリームの最大ビットレートを r^{max} とする。提案アルゴリズム 1 では、まず与えられたストリームを固定ビットレート s の$\lceil r^{max}/s \rceil$個のサブストリームに分割する(最後のサブストリームのレートは、s ではなく $s' = r^{max} - (\lceil r^{max}/s \rceil - 1)s \leq s$となる)。パラメータ s の決め方については後述する。そのようにしてつくられたサブストリームの数を N としよう。アルゴリズムでは、これらのサブストリームは異なるスパニングツリーを通して参加ピアたちに配信される。具体的には、サーバは N 個の辺素(edge-disjoint)なスパニングツリー$T_1,T_2,\ldots,T_N$をあらかじめ用意し、第 i 番目のサブストリームを T_i の根ノード v_i に向けて配信する。後述するように、これらのスパニングツリーは、各ノードのアップロード帯域が容量を超えず、かつ多数の子ノードに伴う転送遅延が効果的に抑えられることを意図して、ノードあたりのリンク数が定数で抑えられるように設計される。

3-2 ツリー集合の構成法

ノード集合を V={ 0,1,2,...,n-1 }とする。スパニングツリーの集合 Tは、以下の条件を満たすように構成される：

1)一部の例外を除き、各ノードが内部ノードとして参加するのはひとつのツリーだけであり、残りのツリーには葉ノードとして参加していること。

2)各ツリーの任意の内部ノードの次数が定数で抑えられていること。

その結果、各ノード i は、ひとつのサブストリームの $\eta(i)=u_i/s$ 個の隣接する下流ノードへの配信に責任をもつことになる。ここで u_i は i のアップロード容量、s はサブストリームのビットレートである。

提案するツリー集合の構成法では、まず根ノード v_1 をもつツリーT_1をつくる(v_1 は $\eta(v_1)$個の子ノードをもつことができるから、v_1 はメディアサーバから受け取ったサブストリームをすべての子ノードに単位時間で転送することができる)。具体的には、T_1 における v_1 の子ノードの集合は、

$$\rho(v_1) := \{i : v_1 < i \leqq v_1+\eta(v_1)\}$$

のように決められる。ここで+は n を法とする加法である。v_1 の最初の子であるノード v_1+1 は受け取ったサブストリームを集合 $\rho(v_1+1):=\{i : v_1+\eta(v_1)<i\leqq v_1+\eta(v_1)+\eta(v_1+1)\}$中のノードに逐次転送し、2 番目の子であるノード v_1+2 は、受け取ったサブストリームを集合 $\rho(v_1+2):=\{i : v_1+\eta(v_1)+\eta(v_1+1)<i \leqq v_1+\eta(v_1)+\eta(v_1+1)+\eta(v_1+2)\}$中のノードに逐次転送する(以下、同様の手順を繰返す)。もし自身のアップロード帯域を使い切ることなくツリーT の構成を終えるようなノードが残った場合は、帯域を使い切らなかったノードは、その次のツリーT'の根ノードとして選ばれ、T'では帯域が使い切れるだけの子ノードをもたされる。以下、同様の処理を繰返してスパニングツリーを順次構成する。

3-3 最適転送方式

もし $u_s=r^{max}$ であれば、メディアサーバは N 個のサブストリームを N 個のツリーの根ノードに送り、各ツリー上でそれぞれ転送を繰返すことで最大ストリーミングレート r^{max} を実現することができる。しかし $u_s>r^{max}$ のときは、サーバのアップロード帯域がフルには使用されていないことから(u_s-r^{max} 分の帯域が使われないままになっているから)、上述のような単純な方法では最大ストリーミングを実現することはできない。提案手法ではこの点を解決するため、$u_s>r^{max}$ のときには、メディアサーバの役割を次のような役割をもつ二つのサブサーバ σ_a、σ_b に分けて処理を行う。ここで、

- σ_a は N'個のサブストリームを N'個の異なるツリーの根ノードにビットレート s で送る(N'の定義は後述)。ツリー内部の配信は基本方式と同様に行われる。
- σ_b は残ったサブストリームをすべてのノードに自身の余剰アップロード帯域を使って直接送る(サブサーバを根とするスター状のツリー)。

N'の値は次のように決められる。σ_a と σ_b のアップロード帯域をそれぞれ u_a、u_b としよう。定義より $u_a+u_b=u_s$ であることに注意されたい。u_a の値は $u_a=(u_a+U(P))/n$ となるように決められる。したがって $u_a:=U(P)/(n-1)$ であり、ビットレート s で送られるサブストリーム数 N'は $N':=\lceil u_a/s \rceil$のように決められることになる。N-N'の値が一般にはとても小さいことに注意しておこう。

例 1 : $u_s=6$ とし、各ノードのアップロード帯域が$[u_0,u_1,u_2,\ldots, u_9] := [3,2,3,3,2,2,3,4,3,2]$のように与えられているものとしよう。$\sum_i u_i = 27$であるから、$r^{max}=\min\{ 6,(6+27)/10 \}=3.3$ となる。また $u_s > r^{max}$ であるから、メディアサーバは、アップロード帯域 $u_a = \lceil U(P)/(n-1) \rceil = 3$と $u_b = u_s-u_a =3$ をもつ二つのサブサーバに仮想的に分割される。各サブストリームのビットレートを s=1 のように決めたとしよう。$N' = \lceil u_a/s \rceil = 3$であるから、ビットレート r^{max} のストリームは、ビットレート 1 の 3 つのサブストリームとビットレート $r^{max}-(N-1)s=0.3$ のサブストリームに分割される（図 1 参照）。最初のツリーT_1 の根ノードをノード 0 としよう。$\eta(0)$ $(=u_0/s)= 3$ であるから、ノード 0 は T_1 中で(1, 2, 3 という)3 つの子ノードをもつ。加えて $\eta(1)=2$、$\eta(2)=3$、$\eta(3)=3$ より、T_1 は図の左に示すように構成される。ただしこの時点ではノード 3 の容量はフルには使用されていないため、ノード 3 は二つ目のツリーT_2 の根ノードとしても選ばれ、それ以降のツリーは、T_1 と同様の方法で順次構成される。T_3 の構成を終えた時点で、ビットレート $s'=0.3$ のサブストリームが未配信であり、サーバ上にはアップロード帯域 3(=6−3)が残されている。したがって最後に、サーバはそのサブストリームを図の右に示すように配信することができる。

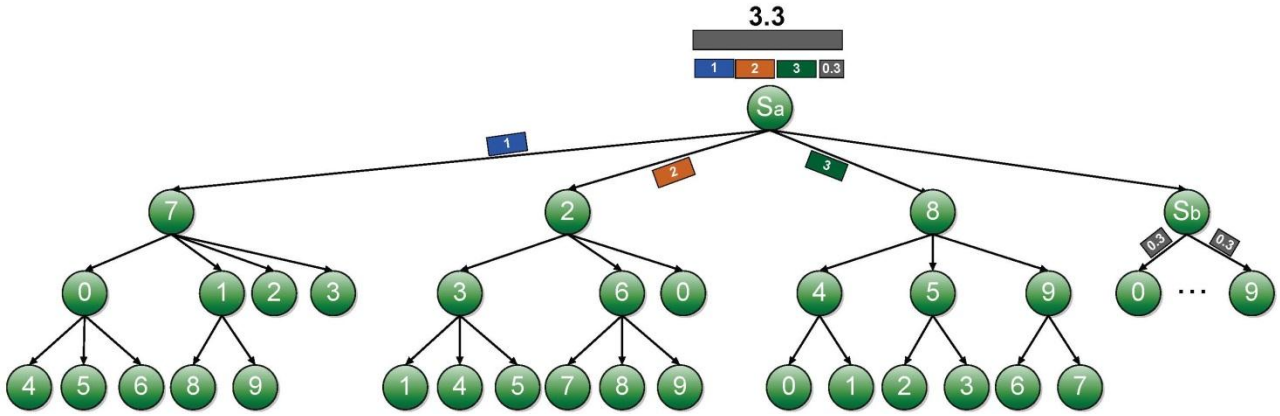


図 1　例 1

提案手法の最適性は、$r^{max}=u_s$ (Case 1)のときと $r^{max} < u_s$ (Case 2)のときに分けて証明することができる。

Case 1: サーバは N 個のサブストリームを N 個のツリーのルートに送り、各ツリー内でサブストリームは n-1 個のノードに転送される。提案手法では、ノード i はサブストリームを $\eta(i) = u_i/s$ 個の子ノードに送っている。したがって、$\sum_i \eta(i) \geqq N(n-1)$が言えれば証明ができたことになる。実際、$r^{max}=u_s$ であるから、

$$n\times u_s/s \leqq (u_s+U(P))/s = u_s/s + U(P)/s$$

となる。$u_s = N\times s$ かつ $U(P)=\sum_i u_i$であるから、$n\times N \leqq N+\sum_i u_i/s$ が言え、

$$(n-1)N \leqq \frac{\sum_i u_i}{s} = \sum_i \eta(i)$$

となる。よって題意が成り立つ。

Case 2: メディアサーバは、u_a、u_b のアップロード帯域をもつ二つのサブサーバ σ_a、σ_b に仮想的に分割される。ここで、1) σ_a は N'個のサブストリームを N'個のツリーの根ノードに送り、2) σ_b はひとつのサブストリームをすべてのノードに帯域 $u_b(=u_s-u_a)$で直接配信していた。以下のことを示す必要がある：

- $\sum_i \eta(i)$ $(=U(P)/s) \geqq sN'(n-1)$、すなわち、ノード集合が N'個のサブストリームを n-1 個のノードに送るだけの十分な容量をもっていること。
- $u_b \geqq (r^{max} - sN')n$、すなわち、サブサーバ σ_b のアップロード帯域が残りのサブストリームをすべてのノードに単位時間で送ることができる程度に大きいこと。

最初の言明は Case 1 より明らか。2 番目の言明は、$N'\geqq U(P)/s(n-1)$かつ $r^{max} = (u_s+U(P))/n$ より、

$(r^{max}\text{-}sN')n = u_s + U(P) - sN'n$

$\leqq u_s + U(P) - sn\ U(P)/s(n-1)$

$= u_s - U(P)/(n-1) = u_b$

のように示される。ここで最後の不等式は $u_a=U(P)/(n-1)$による。よって題意が成り立つ。

3-4 従来方式との比較

（１）平均遅延時間

同一のノードからなる P2P を考え、既存手法と提案手法の比較を行った。議論を簡単にするため、各ノードとメディアサーバのアップロード帯域を同一とし、各リンクの平均伝播遅延も同一であるとする。以下では、伝播遅延を t_p、転送遅延を t_s とそれぞれ記す。

AQCS における平均遅延時間は次のように見積もられる。サーバからツリーの根ノードへのチャンクの送信には、伝播遅延 t_p と転送遅延 t_s がそれぞれかかる。根ノードが子ノードにチャンクを送る際にかかる転送遅延は、最初の子ノードでは t_s、二番目の子ノードでは $2t_s$ のようになる。各転送には伝播遅延 t_p がそれぞれかかるから、各ノードがサーバから出されたチャンクを受け取るまでの平均遅延は $2t_p+((n+1)/2)t_s$ のようになる。具体的には、たとえばアップロード容量が 300[Kbps]でチャンクサイズが δ= 10 [Kbit]のとき、転送遅延は t_s= 1/30[s]となるから、伝播遅延が t_p=75[ms]のとき、各ピアまでの平均遅延は $ld_{AQCS}=0.15+(n+1)/(2\times30)\fallingdotseq0.0167n+0.167$ のようになる。

提案手法では、サブストリームのビットレートはチャンクを 1 秒で送ることのできるレートに固定されるから、このビットレートを δ[Kbps]と記すことにすると、各ノードの子ノード数は $\eta=t_s/\delta$ のように表現される。各ツリーは幅優先的に成長し、k-ary ツリーの深さ h までの部分木には$\sum_{i=0}^{h} k^i = \frac{k^{h+1}-1}{k-1}$ 個のノードが含まれる。したがって各ツリーの深さは高々

$$\left\lceil \log_{\eta}(n(\eta-1)+1) \right\rceil + 1$$

となる。ここで最後の+1 は、根ノードがほかのツリーの内部ノードでもある場合を反映している。以下では、つくられたツリーがすべて高さ d の完全 η-ary ツリーであると仮定して提案手法の平均遅延を評価する。任意のツリーT を考える。T 中の深さ h には η^h 個のノードが存在する。ノード u の i 番目の子ノードは、u が送信するチャンクを転送遅延 $i\times t_s$ で受け取るから、ノード u の子ノードたちの平均転送遅延は$((\eta+1)/2)t_s$ となり、期待値の線形性から、T の深さ h のノードたちの平均転送遅延は $h\times((\eta+1)/2)t_s+t_s$ となる(ここで最後のプラス t_s はサーバからツリーの根ノードまでの転送遅延である)。したがってすべてのノードの平均転送遅延は $(1/n)\ \sum_{h=0}^{d-1}\left(\frac{h(\eta+1)}{2}+1\right)t_s\times\eta^h$ となる。一方、深さ h のノードの伝播遅延は$(h+1)t_p$ であるから、平均遅延は

$$ld_{prop} = (1/n)\ (t_s/2)(\eta+1)+t_p)\ \sum_{h=0}^{d-1} h\eta^h + t_s + t_p$$

のようになる(導出の詳細は省略)。

（２）転送オーバーヘッド

パケットあたりの転送オーバーヘッドは 160 ビットのヘッダサイズである。最大転送ユニット(MTU)が 12 Kbit であることから、以下では最大ストリーミングデータサイズを 1 パケットで運べる最大データ長に対応する δ=10 に設定する。SFPS の転送オーバーヘッドは次のように評価される。任意のノード i は $s_i=u_i/U(P)\times r^{max}$ を n-1 個の隣接ノードに送る必要がある。ノード i は $p(i)=\lceil s_i/\delta \rceil$個のパケットを各隣接ピアに送るから、ピア i によって送られるパケット総数は$(n-1)\times p(i)$となる。よってサーバを含むすべてのピアの総転送オーバーヘッドは $to_{SFPS}=0.16\times(p(s)+\sum_{i=1}^{n}(p(i)\times n))/(r^{max}\times n)$のようにあらわされる。ここで $r^{max}\times n$ がシステム内で送信されるストリーミングデータの総量であることに注意されたい。前述の設定のもとでの具体的な値は、$to_{SFPS}= (0.16\times n^2\times300/(n\times10))/(300\times n)$となり、もし s=δ とすると、転送オーバーヘッドは 0.016 になる。SFPS の転送オーバーヘッドはノード数の増加に伴って大きくなっていく。ピア数が 500 に近づくと、SFPS のオーバーヘッドは 0.25 となり、これは送信されたデータの 1/4 を占める。

4 提案アルゴリズム２

4-1 準備

前述のアルゴリズムでは、すべてのノードがひとつのサブストリームを他のノードに配信するようにスパニングツリーがあらかじめ構成されていた。したがって、新しいノードが追加された際に、スパニングツリーの集合をはじめからつくり直さなくてはならず、オーバーヘッドの増加が避けられなかった。以下で述べるアルゴリズムでは、この点を解決するため、各ノードが自律分散的にオーバーレイの構造を変化させるというアプローチをとる。キーとなるアイデアは、各ノードが中継するサブストリームが、それぞれひとつになるように(それ以外のサブストリームに関しては葉ノードとして受信するだけになるように) 子ノードのつなぎ替えをすることである。

以下では、ツリーという言葉とサブストリームという言葉を同じ意味で用いる。Vをノード集合とし、N個のツリーの集合を$\mathcal{T}$であらわす。各ノードiに対して、そのノードがもつ**お金(money)**をあらわす変数m(i)と、各サブストリームkに関するそのノードの**価格**をあらわす変数$C_i[k]$をそれぞれ用意する。ここでm(i)はそのノードがもつことのできる子ノード数の最大値であり、$m(i) := u_i/s$のように初期化される。$C_i[k]$はk番目のツリーにおけるiの子ノード数に1を加えた値である。定義より、各kに対して$C_i[k] \geqq 1$であることに注意されたい。$c^*_i = \max_k C_i[k]$としよう。k番目のサブストリームは、$C_i[k] = c^*_i$のときiの**優占(dominant)サブストリーム**であるという。提案手法では、各ノードiは、自身の優占サブストリームに関する子ノード数を増やすとともに、それ以外のサブストリームに関する子ノード数を減らすように動作する。なお優占サブストリームの集合と最大価格c^*_iは、適宜更新される。

提案手法では、オーバーレイを更新する際に更新操作の対象となるノードを効率的に発見するため、FSとNS_kという大域変数を用意する。FSは空き容量をもつノードの集合をあらわす変数である。以下でも述べるようにFSは、次のようにつくられる：

- FS中の各ノードは、アップロード容量が尽きた時点で自身をその集合から削除できること。逆に、アップロード容量に空きができた時点で自身を追加できること。
- 任意のノードがFSに属するノードを容易に発見できること。

一方、NS_kはk番目のツリーに関して飽和していないノード集合をあらわす変数である。ここでk番目のツリーに関して飽和しているとは、C[k] = m(i)+1であり、かつk以外のすべてのhについてC[h] = 1であるときをいう。

4-2 サブストリームの取得方法

提案手法で各ノードiがサブストリームkを取得する方法は以下の三つである。サブストリームkをすでに取得しているノードjを考えよう。一つ目の方法は、単純にjの子ノードとなることである。jは空き容量をもっていなくてはならず、取得されるサブストリームはjの優占サブストリームであることが望ましい。二つ目の方法は、k番目のサブストリームをjからお金を払って得ることである。 支払総額がm(≧1)ならば、iはjの代わりにサブストリームをjとそのm-1個の子ノードに配信する権利を得る。 三つ目の方法は、他のノードと子ノードのスワップ(交換)を行うことである。次のような二つのピアi、jがいる状況を考える: 1) iはk番目のサブストリームをすでに取得し空き容量がある。 2) jはk番目のサブストリームに関して少なくとも一つの子ノードをもっており、しかもそのサブストリームはjの優占サブストリームではない。このとき、iはjに対して、jの代わりにk番目の サブストリームを配信する権利を単位量の金額を支払うことで取得する。

4-3 スケジューリングアルゴリズム

ノードiが取得していないサブストリームの集合を変数P_iであらわす。提案手法では、新規ノードを含む任意のノードiは,ノード集合U_i中のノードと通信することでサブストリームを取得していく。また、もし空き容量の不足などのためにスケジューリングが完了しなければ、集合FS中の各ノードに順次問い合わせていくことで取得を完了させる(U_iの計算法とFSの管理法については後述する)。各ステップで各ノードiは、P_i中の各サブストリームqと各U_i中の各ノードについて、ノードjのサブストリームqに関する価格をチェックし、もし価格が2以上であり(すなわちjが少なくともひとつのqに関する子ノードをもち)、しかもqがjにとっての優占サブストリームでないならば、qの**頻度**を1増やす(頻度の高いサブストリームは、より多くのスワップ候補を含んでいることに注意)。 P_iの中でもっとも頻度の高いサブストリームをq^*とし、 q^*の頻度にかかわっているノード集合をU_i^*とする。U_iを使ったスケジューリングの具体的なステップは以下の通りである。

Step 1: ピアiはもっとも頻度の高いサブストリームq^*を最短のホップカウントをもつノードj∈ U_i^*からお金を支払って取得する。支払額は、m(i)<2でない限り2以上でなくてはならない。これは、iがq^*とjの優占サブストリーム of jを含む少なくとも2個以上のサブストリームを取得するためである。

Step 2:ノードiは各j∈U_i^*に対して、ノードjの子ノードとのサブストリームq^*に関するスワップを単位量のお金を支払って行う。ただしq^*はiとjによって共通に取得されているサブストリームであり、jの優占サブストリームではないものとする。

Step 3:もし空き容量をもつノードj∈ U_iが存在し、iがjの優占サブストリームqを取得していなければ、iはqに関するjの子ノードになる。この処理は、そのようなjがU_i中になくなるまで繰返される。

Step 4: もしm(i)≧1ならば、ノードiは各q∈ P_iに対してqのためのもっとも価格の安い非飽和ノードをU_iから探し、最大でmax{ 1, {m(i)}/{|P_i|}}支払って購入する。この処理は、そのようなqがなくなるかm(i)=0になるまで続けられる。

Step 5: この時点でノードiの所持金額は尽きているので、これ以降は、空き容量をもつノードをつかったサブストリームの取得のみが可能である。ノードiはまずU_iの中にそのようなノードがあるかどうかをチェックし、もしなければ最後の手段として、ノードiはFS中のノードに対してP_iが空になるまで問い合わせを行う。

Step 6: もしノードiが空き容量をもっており、しかもU_i^*が空でなければ、ノードiはサブストリームq^*に関するできるだけ多くの子ノードをあつめるため、q^*に貢献しているノードたちU_i^*との間で子ノードの交換を行っていく。この処理が通常のスワップの特別な場合であることに注意されたい。

5 オーバーレイの収束性に関する実験的評価

5-1 セッティング

C++でシミュレータを記述し、提案アルゴリズムによるオーバーレイの収束の様子をシミュレートした。ノード数を 1000 とし、オーバーレイの構造は、新しいノードの参加に伴って順次変化していくものとする。実験では二つの設定を考えた。 一つ目は、すべてのノードが同一のアップロード容量 u=384 [Kbps]をもつような同種ネットワークである。この設定では、サーバのアップロード容量を u(s)=768 [Kbps]とし、サブストリームレートを s=64 [Kbps]とした(これは 6 個のツリーに対応し、r^{max}=384 [Kbps]であることを意味する)。二つ目の設定は、異なる三つのタイプのノードが存在するような異種ネットワークである。ここでは各タイプのアップロード帯域を 128, 384, 960 [Kbps]とし、その比率を 46%, 39%, 15%とした。またサーバの容量は u(s)=640 [Kbps]とし、サブストリームレートを s=32 [Kbps]とした(これは 11 個のツリーに対応し、r^{max} =352 [Kbps]であることを意味する)。以下では各設定のもとで、各ツリーを管理する異なる最大ディーラ数(MDPT)を設定する。

5-2 集中的な環境

トラッカーがすべての情報を管理する集中的な環境での飽和ノード比率を表２にまとめる。いずれの設定においてもMDPT=1のときにほとんどのノードが飽和しているが、大きなMDPTを用いることで、異種ネットワークにおける結果が改善されることがわかる。しかしMDPTの増加には、通信や処理のコストの増加に繋がるというデメリットもある。設定２のもとでの平均ホップ数と最適ホップ数の比較を図２に示す。図より、MDPTを大きくとることで平均ホップ数が短くなることがわかる。

表2　集中的な環境での平均飽和ノード比率

MDPT	1	2	3
設定１（同種）	99%		
設定２（異種）	92%	96%	97%

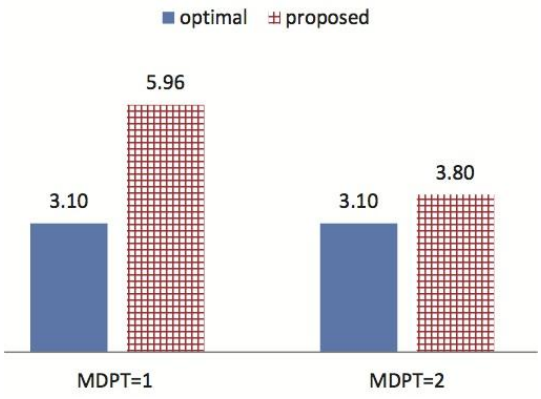


図2　集中的な環境でのホップ数（異種ネットワーク）

5-3 分散的な環境

トラッカーが存在しない分散的な環境での平均飽和ノード比率を表３にまとめる。新規ノードが通信相手をランダムに選んでいるにもかかわらず、提案アルゴリズムはいずれの設定でもよい性能を示している。同種・異種の二つの設定でわずかな違いがでている理由は、多数を占めるアップロード容量の小さなノードがそもそも飽和しやすいためである。このことを確かめるため、ノードタイプ別の飽和ノード比率を調べた。結果を図３に示す。図からわかるように、アップロード容量128 [Kbps]をもつノードの飽和比率がもっとも高い。図４はノードの参加に伴う飽和比率の変化の様子を示している。また図５には、分散環境におけるホップ数を示している。U_i中のノード数の増加に伴って平均ホップ数が減少していることがわかる。加えて、設定１の同種ネットワークではほぼ最適であり、異種の場合もMDPT>1のときは十分よい性能を示していることもわかる。

表3　分散環境における平均飽和ノード比率

MDPT	1	2	3	4
設定１（同種）	54%	72%	80%	85%
設定２（異種）	60%	71%	76%	81%

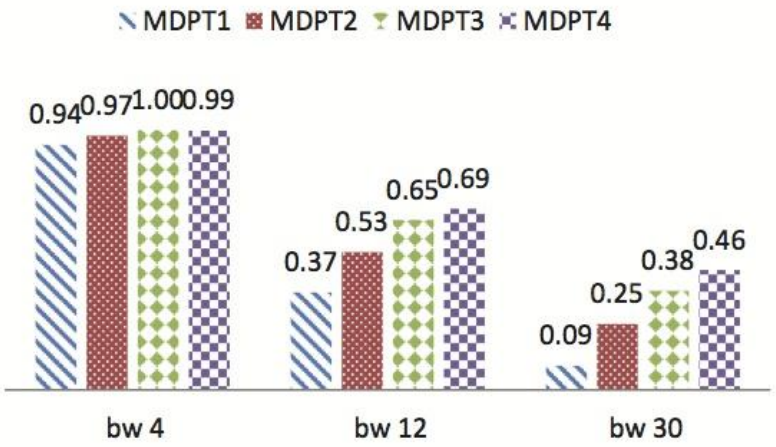
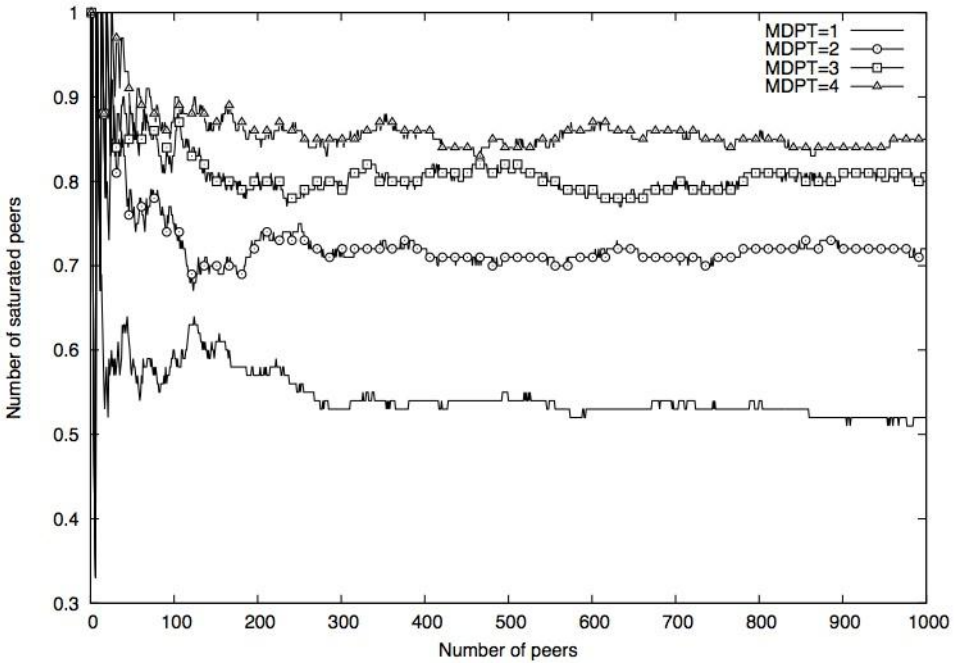
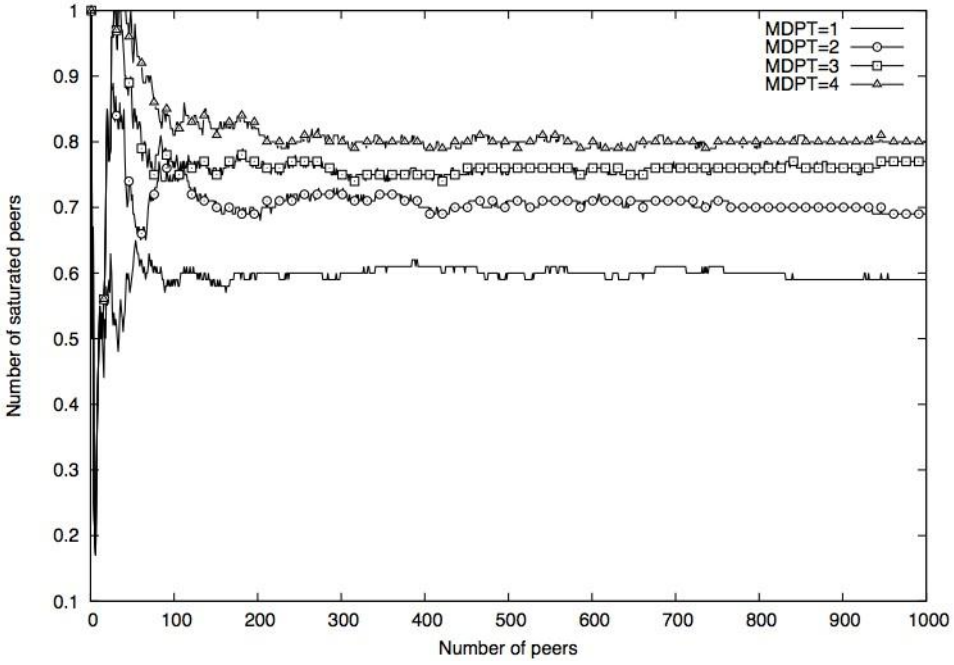


図3　飽和ノード比率（タイプごと）

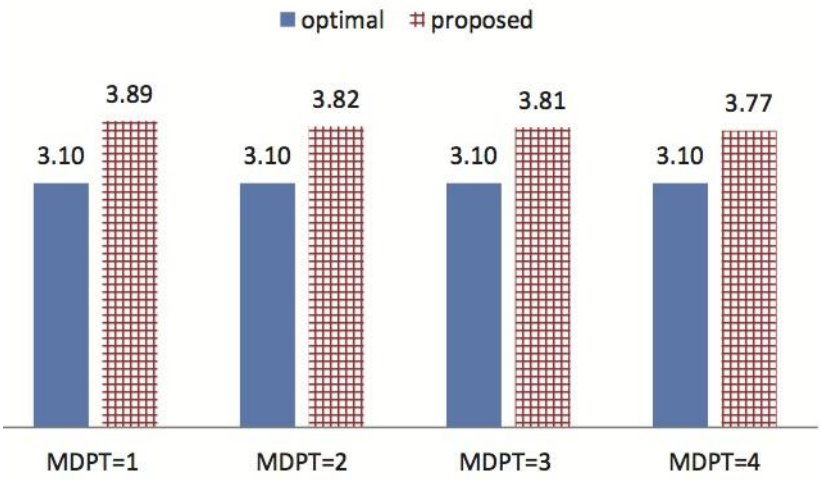


(a) 設定１（同種ネットワーク）

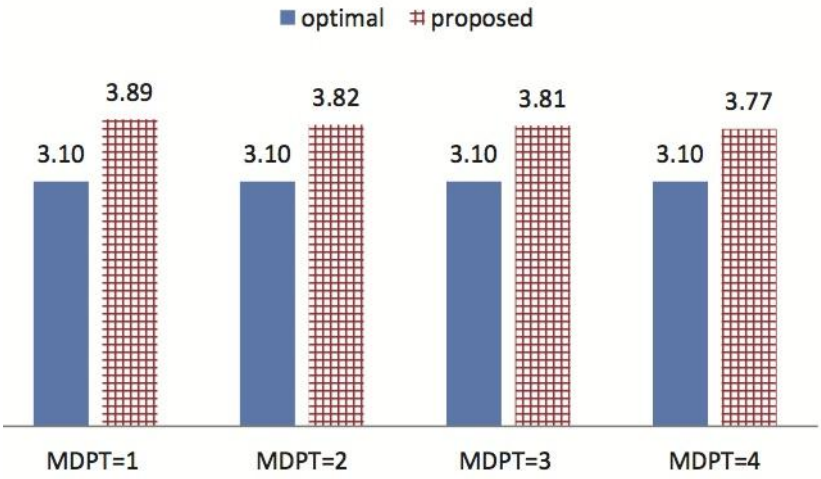


(b) 設定２（異種ネットワーク）

図4　飽和ノード比率



(a) 設定１（同種ネットワーク）



(b) 設定２（異種ネットワーク）

図5　分散環境におけるホップ数

5-4 FS への問合せ回数

FSへの問い合わせ回数を表４に示す(同種ネットワークでは問い合わせが全く起こらなかったため、この表には示していない)。表から、集中型と分散型とで問合せ回数に大きな違いがあることがわかる。この違いは、飽和に近いノードに関する情報をうまく利用できるか否かによっている。またMDPTの増加は問合せ回数にもよい影響を与えている。

表4　設定２のもとでのFSへの問合せ回数

MDPT	集中型	分散型
1	202	379
2	46	332
3	38	289
4	25	255

6 おわりに

本研究によって、P2P 型ストリーミングシステムに置いて最大ストリーミングレートを保証する具体的な手法があきらかとなった。本研究課題ではシミュレーションによる評価までを行ったが、今後は実際の分散システムで実証的な評価を行うことが望まれる。

【参考文献】

[KUMAR07] R. Kumar, Y. Liu, and K. W. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In INFOCOM 2007, pages 919-927, May 2007.

[GUO08] Y. Guo, C. Liang, and Y. Liu. AQCS: Adaptive Queue-Based Chunk Scheduling for P2P Live Streaming. In Proc. of IFIP Networking, pages 433-444, 2008.

[MASSOULIE07] L. Massoulie, A. Twig, C. Gkantsidis, and P. Rodriguez. Randomized decentralized broadcasting algorithm. In Proc. of IEEE INFOCOM, pages 1073-1081, 2007.

[LI04] J. Li, and P. A. Chou, and C. Zhang. Mutualcast: an efficient mechanism for content distribution in a p2p network. Microsoft Research, MSR-TR-2004 100, 2004.

[NGUYEN08] T. Nguyen, K. Kolazhi, R. Kamath, S. Cheung, and D. Tran. Efficient Multimedia Distribution in Source Constraint Networks. IEEE Trans. on Multimedia, Vol. 10, No. 3, pages 532-537, 2008.

[LIU08] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance bounds for peer-assisted live streaming. In Proc. of ACM SIGMETRICS, pages 313-324, June 2008.

[LIU10] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou. P2P Streaming Capacity under Node Degree Bound. In Proc. of ICDCS'10, pages 587-598, 2010.

[ZHAO11] C. Zhao, X. Lin, and C. Wu. The Streaming Capacity of Sparsely-Connected P2P Systems with Distributed Control. In Proc. of INFOCOM, pages 1449-1457, 2011.

〈発 表 資 料〉

題　　名	掲載誌・学会名等	発表年月
An Approximation Scheme for Burst Scheduling in Time Slicing Mobile TVs	13th International Conference on Parallel and Distributed Computing, Applications and Technologies	2012.12
Complementary Piece-Based Buffer Map for P2P VoDs Supporting VCR Operations	7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing	2012.11
Colluder Detection in Commercial P2P CDNs Using Reputation Information	7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing	2012.11
Network Coding を適用した P2P コンテンツ配信システムにおけるトラフィック増加量の実験的評価	平成 24 年度（第 63 回）電気・情報関連学会中国支部連合大会	2012.10
P2P ライブストリーミングにおける不正ピア検出法の提案	平成 24 年度（第 63 回）電気・情報関連学会中国支部連合大会	2012.10
P2P 型分散ファイルシステムの耐故障化手法の提案	平成 24 年度（第 63 回）電気・情報関連学会中国支部連合大会	2012.10
P2P ビデオオンデマンドシステムの待ち時間短縮化手法の提案	平成 24 年度（第 63 回）電気・情報関連学会中国支部連合大会	2012.10
P2P ライブストリーミングのための離脱耐性のある課税スキーム	電子情報通信学会コミュニケーションクオリティ研究会	2012.7
mTreebone に基づく離脱耐性のある P2P ライブストリーミング	電子情報通信学会コミュニケーションクオリティ研究会	2012.7
転送遅延を考慮した WMN 上のアクセスポイント削減手法	電子情報通信学会コミュニケーションクオリティ研究会	2012.7