QoS向上のためのオーバーレイネットワークを利用した経路制御の提案

山本　正樹† 　上土井陽子†† 　若林　真一††

† 広島市立大学大学院 情報科学研究科　〒 731–3194 広島市安佐南区大塚東三丁目 4–1
†† 広島市立大学 情報科学部　〒 731–3194 広島市安佐南区大塚東三丁目 4–1
E-mail: †masaki@lcl.ce.hiroshima-cu.ac.jp, ††{yoko,wakaba}@ce.hiroshima-cu.ac.jp

あらまし　インターネット上で音声通信，TV 電話，TV 会議のようなサービスを実現するとき，現状のインターネットでは十分な *QoS* を確保できていないことが問題となっている．しかし，従来の *QoS* 制御技術では，*QoS* 制御を必要とする通信を行うすべての発信元，送信先間に *QoS* 制御をもつ IP インフラを整備する必要があり，これを実現することは困難である．そこで，既存のインフラストラクチャーの変更なしに複数の経路を考慮可能な，オーバーレイネットワークを利用した経路制御を用いたコールアドミッションを提案する．コールアドミッションとは，*QoS* の向上を計りつつ，コール (通信要求) を受理することによって得られる利益の合計を高めるためにコールの受理，拒否を決定することである．本稿では大容量データ通信において，IP(Internet Protocol) ネットワーク上でのコールアドミッション，経路探索，オーバーレイネットワークを利用した経路制御であるコールアドミッションについて述べ，*QoS* 向上を目的としたオーバーレイネットワーク上での経路制御方法を提案する．

キーワード　オーバーレイネットワーク，P2P，コールアドミッション，*QoS*，経路制御

A QoS Routing on an Overlay Network

Masaki YAMAMOTO†, Yoko KAMIDOI††, and Shin'ichi WAKABAYASHI††

† Graduate School of Information Sciences, Hiroshima City University
3–4–1, Ozuka-higashi, Asaminami-ku, Hiroshima, 731–3194 Japan
†† Faculty of Information Sciences, Hiroshima City University
3–4–1, Ozuka-higashi, Asaminami-ku, Hiroshima, 731–3194 Japan
E-mail: †masaki@lcl.ce.hiroshima-cu.ac.jp, ††{yoko,wakaba}@ce.hiroshima-cu.ac.jp

Abstract　When we offer services such as the voice-data communication, TV telephone, and TV conference on the Internet, it is not enough to guarantee quality of service. However the traditional method of *QoS* control requires that all network elements between a source and a destination implement *QoS* mechanisms, and it is difficult to achieve this requirement. Then, we propose the call admission with routing on an overlay network where two or more routes can be considered without changes in existing infrastructure. Call admission is to decide whether or not to accept calls to guarantee quality of service and obtain many profits by accepting calls. We study call admission and routing on an IP network and *QoS* routing on an overlay network for a mass data communication. We propose a routing method on an overlay network to improve quality of service.

Key words　Overlay Network, Peer-to-Peer, Call Admission, Quality of Service, Routing

1. はじめに

近年，インターネット利用者の爆発的な増加に伴いブロードバンド化が推進され，IP 電話，TV 電話，TV 会議のような帯域幅を大きく占有するデータをインターネット上で伝送するようになっている．利用者の増加と大容量データの伝送によって，すべてのコール (通信要求) を受け入れることは回線の混雑による *QoS* の劣化につながる可能性がある．*QoS* の劣化を避けるための従来技術として，通信で用いる帯域幅を確保する Intserv [5]，Diffserv [3] などが挙げられる．しかし，従来技術では，*QoS* 制御を必要とする通信を行うすべての発信元，送信先間に *QoS* 制御をもつ IP インフラを整備する必要があり，こ

れを実現することは困難である．

そこで，エンドホスト間で QoS 制御を行うため，既存の IP インフラを利用して実現可能なオーバーレイネットワークを利用した QoS 制御に注目する．本稿におけるオーバーレイネットワークとは，中継ノード (ルータ) 群からなる IP ネットワーク上に，エンドノード (ホスト) を中継ノードとして構築されるネットワークである．オーバーレイネットワークを利用した QoS 制御は，ネットワーク容量を超えるコールを受け入れることによる回線の混雑を避けるために，回線の状況から新しいコールを受理すべきかエンドホスト間により決定することである．QoS を保証しつつ，コールを受理することによって得られる利益の合計を高めるためにコールの受理，拒否を決定することをコールアドミッションと呼ぶ．また，コールが受理された場合，ネットワーク中の 1 つの経路をコールに割り当てるというルーティングを行う必要がある．受理したコールに割当てる経路は，ネットワーク資源の有効利用と割当てる経路が使用アプリケーション向きであるかを考えなければならない．本稿では，この経路の割当てをオーバーレイネットワークを利用した経路制御により行う．オーバーレイネットワーク上では，個々のエンドノード間では IP 経路制御が行われ，エンドノードを中継することにより IP 経路制御とは異なる複数の通信経路を考えることができる．オーバーレイネットワークを利用した経路制御は，計測した指標 (遅延，帯域幅，スループット) を用いて構築されたオーバーレイネットワーク上での特定のアプリケーション向きの通信経路を選ぶことを可能にする．

また，我々はコールの受理，拒否の決定にコールアドミッションアルゴリズムの 1 つである $SAAP\mu$ [4] を用いる．$SAAP\mu$ の特徴は，枝の負荷の増加に応じて指数関数的に増加するコストを枝に割り当て，経路の枝コストの総和に上限 (制約コスト) を設定し混雑している経路を避けることである．ここで，我々は制約コストを満たし，かつ **(a)** 枝コストの総和が最小，**(b)** ホップ数が最小な経路の中で枝コストの総和が最小という経路選択基準を考えた．基準 (a) では，経路探索手法として Dijkstra 法を用いる (従来手法)．本稿では，基準 (b) を満たす経路を探索するためのアルゴリズムを提案する．ただし，オーバーレイネットワークにおける枝はエンドノード間の IP 経路制御によって定められた経路，枝コストはエンドノード間の IP 経路制御によって定められた経路の IP ネットワーク上の枝コストの総和と定義する．提案経路探索法は，以前に探索した始点から各節点への全ての経路の中で枝コストの総和が最小となる経路を保持し，このコストを超える経路への探索と，制約コストを超える経路への探索を制限し条件付幅優先探索を行うことで，始点から終点へのコスト制約を満たす最小ホップ数の経路を探索する．オーバーレイネットワーク上での提案経路探索手法を用いたコールアドミッションの有効性を検証する．

2. オンラインコールアドミッション

音声，動画を対話的に通信するネットワークでは，ネットワーク容量を超えるコールを受け入れることによって回線の混雑が起こり，QoS の劣化につながる可能性がある．回線の混雑による QoS の劣化を避けるために，新たに発生するコールをネットワークの状況に応じて受理するか，拒否するかを決定するコールアドミッションが解決法の 1 つとして考えられている．受理されたコールの利益の合計を高めることを目的に逐次的に発生するコールに対して，コールが発生したときのネットワークの状況からコールの受理，拒否を決定することをオンラインコールアドミッションと呼ぶ．

以下では，本稿におけるコールアドミッション問題を定義する．本稿におけるネットワークは節点数が n である無向グラフとし，各枝 e は容量 $u(e)$ を持つとする．本稿では，基本的な場合を考えるためにコールの存続期間が無期限と見なせる長期存続コールであると仮定し，i 番目に発生したコールを $r_i = (s_i, t_i, b_i, p_i)$ と表し，s_i はコールの始点，t_i は終点，b_i はコール r_i のバンド幅要求，p_i はコール r_i を受理することによって得られる利益とする．本稿では b_i を 1 に固定する．

コールが発生したとき，受理，拒否を決定するためにネットワーク中の各枝 e にどれくらい負荷がかかっているか計算する必要がある．そこで，j 番目のコールを処理したときの枝 e に対する負荷率 $L_j(e)$ を以下に定義する．

$$\text{Ł}_j(e) = \frac{1}{u(e)} \sum_{\substack{k \in A_j \\ e \in P_k}} b_k \qquad (1)$$

ここで，A_j は j 番目までに受理されたコールの添字の集合，b_k は k 番のコールのバンド幅要求とし，P_k は $k \in A_j$ となるコールに割り当てられた経路を表す．

コールアドミッション問題とは，$L_{j-1}(e) + (b_j/u(e)) < 1$ を満たすようにコールの受理，拒否を決定し，受理したコールの利益の合計を最大にすることが目的である．本稿では，各コールの利益 p_i を一定値 p と仮定する．このとき，拒否率を最小にすることがコールの利益の合計を最大にすることを意味する．ここで，拒否率は (拒否したコール数/コールの総数) と定義する．また，コールが受理された場合，ネットワーク中の 1 つの経路をコールに割り当てるというルーティングを行う必要がある．受理したコールに割当てる経路は，ネットワーク資源の有効利用と割当てる経路が使用アプリケーション向きであるかを考えなければならない．

3. IP ネットワークにおける QoS 制御

IP ネットワーク上での QoS の劣化を避けるための従来技術として，通信で用いる帯域幅を確保する Intserv [5]，Diffserv [3] などが挙げられる．しかし，従来技術では，QoS 制御を必要とする通信を行うすべての発信元，送信先間に QoS 制御をもつ IP インフラを整備する必要があり，これを実現することは困難である．そこで，エンドホスト間で QoS 制御を行うため，既存の IP インフラを利用して実現可能なオーバーレイネットワークを利用した QoS 制御に注目する．

3.1 オーバーレイネットワークを利用した QoS 制御

一般的にオーバーレイネットワークとは，下位層のリンクを用いて上位層でそれぞれのサービスやアプリケーションの目的に応じて仮想的なリンクを形成することで構築されるネット

ワークである．本稿で対象とするオーバーレイネットワークとは，中継ノード (ルータ) 群からなる IP ネットワーク上に，エンドノード (ホスト) を中継ノードとする論理ネットワークを構築することにより，現状の IP で提供されないネットワークサービスをアプリケーション層で実現しようとするものである．オーバーレイネットワークを利用して実現されるサービスの例として，資源発見・共有・キャッシング，コンテンツ配信などが挙げられるが，本稿では，オーバーレイネットワークを利用したコールアドミッションと経路制御に注目する．オーバーレイネットワークを利用したコールアドミッションとは，ネットワーク容量を超えるコールを受け入れることによる回線の混雑を避けるために，回線の状況から新しいコールを受理するか，拒否するかをエンドホスト間により決定することである．

また，オーバーレイネットワーク上では，個々のエンドノード間では IP 経路制御が行われ，エンドノードを中継することにより IP 経路制御とは異なる複数の通信経路を考えることができる．オーバーレイネットワークを利用した経路制御は，計測した指標 (遅延，帯域幅，スループット) を用いて構築されたオーバーレイネットワーク上での特定のアプリケーション向きの通信経路を選ぶことを可能にする．今までにオーバーレイネットワークを利用した経路制御によって QoS を向上させることが考えられている [1], [2], [11]．しかし，従来手法で用いられているオーバーレイネットワークは，各オーバーレイネットワークノードがすべてのノードへのリンクを保持できる程度の規模しか想定していない．特に大規模なオーバーレイネットワークの利用を考える場合では，オーバーレイネットワークの維持に多くの資源を割くことになり現実的でない．本稿では，各ノードが一部のノードへのリンクしか保持しないような拡張可能なオーバーレイネットワークを扱い，大容量データ通信が行われる IP ネットワークにおけるオーバーレイネットワークを利用したコールアドミッションと経路制御方法を提案する．

4. オーバーレイネットワークを利用したコールアドミッション

入力ネットワーク $N = (V, E)$ の各枝 e には容量 $u(e)$ と負荷が定義されているとする． ネットワーク N 上でのコールアドミッション問題に対し, 以下に示す $SAAP\mu$ [4] では, 枝の負荷の増加に応じて指数関数的に増加するコストを割り当て経路探索を行う．

4.1 オンラインアルゴリズム $SAAP\mu$

コールを受理して得られる利益を高めるためにコールの受理，拒否を決定するアルゴリズムをコールアドミッションアルゴリズムと呼ぶ．本稿では，コールアドミッションアルゴリズムの 1 つである $SAAP\mu$ について考察する．

$SAAP\mu$ は，コールが発生するとコールの受理または拒否を決定し，受理する場合はコールの始点と終点間の経路を決定する．$SAAP\mu$ の特徴は，枝の負荷の増加に応じて指数関数的に増加するコストを枝に割り当て，始点と終点間の経路上の枝コストの総和が式 (2) を満たし，経路上のすべての枝が式 (3) を満たす経路を選択することである．本稿では，経路上の枝の本数をホップ数とし，枝の容量がコールの要求するバンド幅以上空いているならホップ数 1 のコールを受理するものとする．つまり，式 (2) よりコールの利益を $p = b\mu$ と設定する．

$$\sum_{e \in P} b_j \mu^{L_{j-1}(e)} \leqq p \tag{2}$$

$$\mathrm{L}_{j-1}(e) + \frac{b_j}{u(e)} < 1 \tag{3}$$

コールアドミッション問題において，コストは枝容量の空いている度合いを表し，ホップ数は枝容量を帯域幅の分だけ占有する枝の数を表している．コストが低く，ホップ数の少ない経路を優先することは後に発生するコールの割り付けを効率的にすると考えられる．

ここで，式 (2)，(3) を満たす経路の中で，**(a)** 枝コストの総和が最小，**(b)** ホップ数が最小の経路の中で枝コストの総和が最小，という 2 つの経路選択基準を考える．基準 (a) では，経路探索手法として Dijkstra 法，基準 (b) では，条件付幅優先探索アルゴリズム [12] を用いる．

本稿では，オーバーレイネットワークを利用したコールアドミッションに $SAAP\mu$ を適用し，基準 (b) を満たす経路選択を行うことを提案する．図 1 に，オーバーレイネットワーク上での経路が IP ネットワーク上ではどのような経路となるか示す．オーバーレイネットワーク上でのコストが最小の経路を矢印，ホップ数が最小の経路を破線矢印で示している．IP ネットワークノード間の遅延を 20ms と仮定したとき，それぞれの経路の遅延時間に着目すると，コストが最小の経路では 100ms，ホップ数が最小の経路では 60ms の遅延時間である．オーバーレイネットワーク上での 1 ホップは，複数の IP ネットワークノードを中継することを意味し，基準 (b) を満たす経路選択を行うことはオーバーレイネットワークにおけるコールアドミッション問題において，より有効であると考えられる．

4.2 提案コールアドミッションアルゴリズム

4.2.1 条件付ホップ数最小経路探索問題

本節では, オーバーレイネットワークのモデルネットワーク上での条件付ホップ数最小経路探索を定義する．ネットワーク $N_{ip} = (V, E, u)$ は各枝 e に容量 $u(e)$ と負荷を持つ IP ネットワークのモデルネットワークとする．ネットワーク N_{ip} を 4.1 節で取り上げた $SAAP\mu$ [4] により，枝の負荷の増加に応じて指数関数的に増加するコストを割り当て，枝にコストと空き容量をもつネットワーク $G_{ip} = (V, E, c, u')$ へ変換する．このとき，オーバーレイネットワークのモデルネットワークで枝にコストと空き容量をもつネットワーク $G_{over} = (OV, OE, oc, ou'), OV \subseteqq V$ を次のように定義する．ネットワーク G_{over} における各枝 $oe = (u, v)$ はエンドノード u, v 間の IP 経路制御によって定められた経路 IP_{oe} に対応し，枝コスト $oc(oe)$ はエンドノード u, v 間の IP 経路 IP_{oe} のネットワーク G_{ip} 上の枝コストの総和 $\sum_{e \in IP_{oe}} c(e) = oc(oe)$ と定義し，枝 oe の空き容量 $ou'(oe)$ は，IP 経路 IP_{oe} に属する G_{ip} 上の枝の空き容量の最小値 $Min\{u'(e) | e \in IP_{oe}\} = ou'(oe)$ と定義する．このとき，ネットワーク G_{over} 上での条件付ホップ数最小経路探索問題を以下

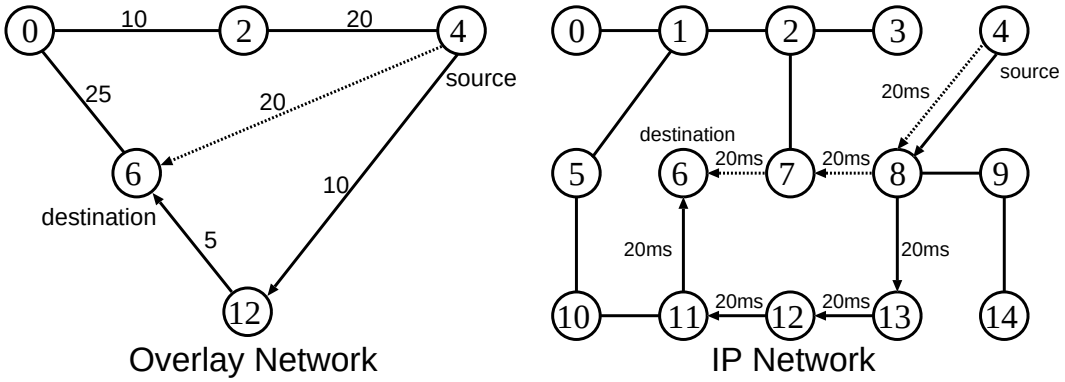

図 1　オーバーレイネットワーク上の経路と IP ネットワーク上の経路の対応

のように定義する．

コール r_j が発生したとき，

$$\sum_{e \in P} b_j \mu^{L_{j-1}(e)} \leqq p$$

上式を満たす，つまり，上式右辺の p を制約コスト C_{limit} とし，$b_j\mu^{L_{j-1}(oe)}$ を枝 oe の枝コスト $oc(oe)$ とする．また，ホップ数 l 以下の始点 s から節点 v への経路の中で，経路の枝コストが最小の経路のコストを $MC_{l,v}$ と定義する．このとき，節点 v において，以下の条件 1, 2 を満たすとき，そのときに限り真となる述語 $X(l,v)$ を考える．

条件 1　$MC_{l,v} \leqq C_{limit}$

条件 2　$MC_{l,v} < MC_{l-1,v}$

条件付ホップ数最小経路探索問題では，述語 $X(l,t_j)$ が真である最小値 l におけるコール r_j の始点 s_j から終点 t_j までのホップ数最小経路 P を探索することを目的とする．経路を見つけた場合は経路を出力し，条件 1，2 を満たす経路がない場合は経路が存在しないことを出力する．

以降では，節点 s は始点，節点 t は終点，$EdgeC(v,w)$，$u(v,w)$ は節点 v，w 間の枝のコストと枝容量とする．

4.2.2　経路探索アルゴリズム

本節では，条件付ホップ数最小経路探索問題を解く条件付幅優先探索アルゴリズム [12] を説明する．条件付幅優先探索アルゴリズムは，以前に探索した始点から各節点への経路の中で枝コストの総和が最小のコストの経路のコストを保持し，このコストを超える経路への探索と，制約コストを超える経路への探索を制限し条件付幅優先探索をすることで，始点から終点へのコスト制約を満たす最小ホップ数の経路を探索する．

条件付幅優先探索アルゴリズムでは，ホップ数 i の始点 s から各節点 v への制約コスト C_{limit} を超えない探索経路中で最も小さいコストを持つ経路の枝コストの総和を $Cost(i,v)$ へ記録し，以前に探索した始点 s から各節点 v への経路の中の枝コストの総和の最小値を配列 $min[v]$ へ記録する．節点名，ホップ数の二項組を要素とし，後に延長される候補となる経路を表現する要素を FIFO 型のデータ構造キュー Q に保持する．また，キュー Q へ要素を入れるとき，同じ要素がすでにキュー Q へ入っている場合は古い要素を削除する．

条件付幅優先探索アルゴリズム

入力：枝 oe に容量 $ou'(oe)$ とコスト $oc(oe)$ をもつネットワーク $G_{over} = (OV, OE, oc, ou')$，始点 s，終点 t，コスト制約 C_{limit}

出力：$X(l,t)$ が真である経路が存在するならば最小ホップ数 l の経路，またはこの条件の経路が存在しなければ，経路が存在しないこと

初期設定

- $min[v] \leftarrow \infty$ $(\forall v \in V - \{s\})$，$min[s] \leftarrow 0$
- $Cost(i,v) \leftarrow \infty$ $(\forall v \in V - \{s\}, i = 0, 1, \ldots, n$ のすべての組合せに対して)
- $Cost(0,s) \leftarrow 0$
- $Q \leftarrow \emptyset$

STEP1: $i \leftarrow 0$，$u \leftarrow s$，

STEP2: 節点 u に隣接する各節点 w に対して，$C_{limit} \geqq Cost(i,u) + EdgeC(u,w)$，かつ $min[w] > Cost(i,u) + EdgeC(u,w)$ を満たすならば，1，2，3 を実行する．

（1）　$Cost(i+1,w) \leftarrow Cost(i,u) + EdgeC(u,w)$，$min[w] \leftarrow Cost(i,u) + EdgeC(u,w)$

（2）　要素 $(w, i+1)$ をキュー Q へ入れる．

（3）　節点 w が t なら ***STEP6*** へ．

STEP3: 終了判定：キュー Q が空なら求める経路が存在しないことを出力し，終了．

STEP4: キュー Q の先頭の要素のホップ数が i ならば，キュー Q の先頭の要素 (i,v) を取り出し，$u \leftarrow v$ とし ***STEP2*** へ．
キュー Q の先頭の要素のホップ数が i でなければ ***STEP5*** へ．

STEP5: $i \leftarrow i+1$，***STEP4*** へ．

STEP6: 節点 t に隣接する節点で $Cost(i,u) + EdgeC(u,w)$ が最小となる節点に対し，$Cost(i+1,t) \leftarrow Cost(i,u) + EdgeC(u,t)$ とし，$Cost(i+1,t)$ からホップ数とコストを参照して始点 s の $Cost(0,s)$ まで辿ることによって経路を求め出力し，終了．

条件付幅優先探索アルゴリズムにおいて以下の定理が成り立つ. [12]

[**定理 1**] 条件付幅優先探索アルゴリズムは，ネットワーク $G(V, E, c, u')$，始点 s，終点 t，制約コスト C_{limit} の入力に対して，$X(l, t)$ が真である経路が存在するならば最小ホップ数 l の経路を出力し，経路が存在しなければ存在しないことを出力する．□

5. 条件付幅優先探索の分散化

本節では，条件付幅優先探索の分散化をどのように実現するかについて述べる．条件付幅優先探索は，節点名とコストの組を要素とする経路情報をメッセージとしてブロードキャストすることによって行われる．メッセージの要素は，節点を経由する度に前の要素に連結して付加される．その探索例を図 2, 3, 4 に示す．探索例では，グラフの枝上の数字はコストを表し，制約コスト C_{limit} を 10 とし，始点 a から終点 d へのコールが発生したと仮定する．各節点 v は，始点 s から各節点 v へのあるホップ数の経路のコストの最小値を記録する．この記憶したコストを超える経路情報をもつメッセージが到着した場合，そのメッセージを破棄する．これを図中では破線のメッセージで表す．また，メッセージの経路情報にある節点へのメッセージの送信は行わない．これを図中では破線矢印で表す.

図 2 に示すように，まず最初に始点から接続しているすべての節点へメッセージをブロードキャストする．そして，メッセージを受け取った節点は，ホップ数 1 とコストを記憶する．

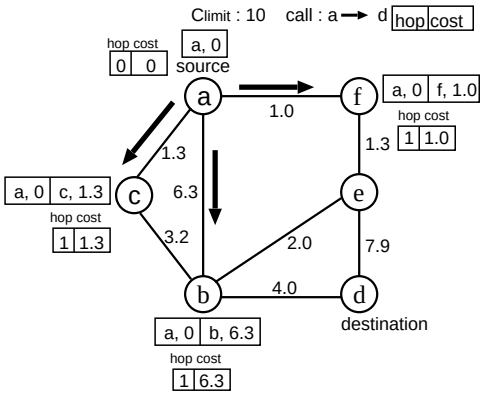

図 2 探 索 例 1

次に図 3 では，節点 c に注目する．メッセージの経路情報に節点 a が含まれているため，節点 c から節点 a へのメッセージ送信は行われない.

次に図 4 では，節点 b に注目する．節点 e は節点 b からメッセージを受け取るが，節点 e はホップ数 2，コスト 2.3 を記憶しているため，ホップ数 3，コスト 8.5 の経路情報をもつこのメッセージは破棄される．節点 d は節点 b からメッセージを受け取り，制約コスト C_{limit}：10 以下の経路情報をもつメッセージであるため，コールを受理し，この経路をコールに割当てる.

しかし，現実の IP ネットワーク上のオーバーレイネットワークにおいて条件付幅優先探索を分散アルゴリズムとして実現するには問題点がある．幅優先探索であるため，ホップ数の小

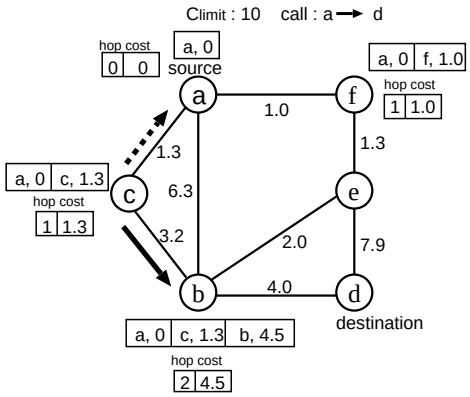

図 3 探 索 例 2

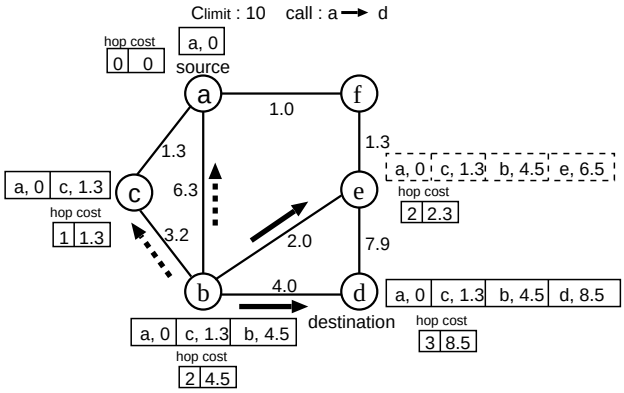

図 4 探 索 例 3

さい経路順にメッセージが到着すると仮定しているが，実際には遅延の小さい経路順にメッセージは到着する．例えば，ある節点へメッセージはホップ数 2 のメッセージよりホップ数 3 のメッセージが早く到着する可能性がある．この問題点を解決するために，各節点 v は始点 s から各節点 v へのあるホップ数 l の経路のコストの最小値をヒューリスティックに決定し，決定した後にメッセージの送信を行う．節点 v にホップ数 l のメッセージが到着した場合，すぐに隣接節点へメッセージを送信せず，ある一定の待ち時間は送信候補メッセージとして保持する．ホップ数 l，かつ，経路のコストがより小さい経路情報を持つメッセージが新たに到着した場合，新たに到着したメッセージを送信候補メッセージとして残し，以前に到着したメッセージを破棄する．ある一定の待ち時間が経過したとき，現在保持している送信候補メッセージの経路のコストを節点 v は始点 s から各節点 v へのホップ数 l の経路のコストの予測最小値と決定し，送信候補メッセージを隣接節点へ送信する．この後に，ホップ数 l，かつ，経路のコストが予測最小値より小さい経路情報を持つメッセージが到着しても破棄する.

待ち時間をより長い時間に設定すれば，始点 s から各節点 v へのホップ数 l の経路のコストの最小値を正確に反映できるが，経路決定の時間が長くなってしまう．そのため，始点 s から各節点 v へのホップ数 l の経路のコストの最小値の正確な反映と経路決定時間の関係についてはあらかじめ調査しておく必要がある.

6. CAN を用いたオーバーレイネットワークの構築

シミュレーション実験において，オーバーレイネットワークの構築に用いた Content Addressable Network(CAN) [6] について説明する．CAN はインターネット規模のハッシュ表のような機能を提供する分散インフラストラクチャーである．CAN はスケーラブル，かつフォールトトレラントで，自律性を持っている．本稿では，CAN の節点が持つ接続関係をオーバーレイネットワークの構築に利用するため，以降では CAN の接続関係の構築部分についてのみ 4.1 節で説明する．また，CAN では IP ネットワークの近さが CAN 座標空間上での近さに反映されていない問題がある．この問題点を解決するために用いられているランドマーク手法 [7] について 4.2 節で述べる．

6.1 構　　築

まず，d 次元トラース上に仮想的なデカルト座標空間を展開する．この座標空間は完全に論理的で，物理的な座標システムとは無関係である．いずれかの時点で，座標空間全体はシステムのすべての節点間で動的に区切られ，すべての節点は個々の区切られた空間を所有する．CAN の節点は仮想的な座標空間を表現するオーバーレイネットワークを自己組織する．各節点は自身の座標ゾーンに隣接する座標ゾーンを所有する節点の IP アドレスを保持する．座標空間で直接隣接する節点の集合は，この空間の任意の節点間のルーティングを可能にする座標ルーティング表のような機能を提供する．

n 個のゾーンへ分割された d 次元空間において，構築されたオーバーレイネットワーク上での平均ルーティング経路長は $(d/4)(n^{1/d})$ ホップであり，個々の節点は $2d$ 個の隣接節点を持つ．d 次元空間における，これらのスケーリング結果は隣接節点数の増加なしで節点数を増加させることを示している．また，平均経路長は $O(n^{1/d})$ である．

6.1.1 節点の加入

CAN 座標空間全体は節点が所有する座標ゾーンによって分割される．CAN に新たな節点が加入する場合，CAN に参加する新しい節点は座標空間の一部をランダムに割り当てられる．これは既存の節点が割り当てられたゾーンを半分に分けて，半分をそのまま保持し，もう一方の半分を新しい節点へ渡すことによって実現される．その分割はある次元の順序でどの次元にそってゾーンを分割するか決められている．例えば，2 次元空間であれば，ゾーンが最初に X 次元で分割され，その次に Y 次元で分割する．この分割法則により，節点が離脱する場合，ゾーンは分割した次元にそって再度結合される．渡される半分のゾーンを得ると，新しい節点は前の占有者から座標が隣接している節点集合の IP アドレスを確認する．この集合はその占有者自身を加えた，前の占有者の隣接節点の部分集合である．同様に，前の占有者はもはや隣接していない節点を削除するために，隣接節点集合を更新する．最後に，新しい節点と古い節点の両方の隣接節点に対して空間の再配分が伝えられる．

新しい節点の追加は，座標空間のとても小さな領域にだけ影響する．節点が保持する隣接節点の数はただ座標空間の次元にのみ依存し，システム内の総節点数に依存しない．したがって，次元数を d とすると，節点挿入は $O(d)$ で隣接節点だけに影響する．これは膨大な数の節点をもつことを想定している CAN にとって重要な特性である．

6.1.2 節点の離脱

節点が CAN を離脱するとき，離脱する節点の占有ゾーンが残された節点によって引継がれることを保障する必要がある．引継ぎの手続きは，離脱する節点の占有ゾーンと関連する接続関係を隣接節点へ渡し，分割された次元にそって 1 つのゾーンへ再度結合することによって完了する．

6.2 ランドマーク手法

CAN では，加入する節点に対してランダムな座標空間を割り当てるため，IP ネットワークの近さが CAN 座標空間上での近さに反映されていない問題がある．この問題点を解決するために用いられているランドマーク手法 [7] について述べる．

ランドマークノードはネットワーク中のどのノードからも存在と IP アドレスなどが知られているノードとし，まず，加入する節点からランドマークノードへの遅延時間を測定する．測定された遅延時間によって，各ランドマークノードへの遅延時間の大きさと順序という 2 つの指標を元に節点をグループ分けする．例えば，3 個のランドマークノード *L1, L2, L3* が存在していると仮定すると，*L1:L2:L3*，*L1:L3:L2*，*L2:L1:L3*，*L2:L3:L1*，*L3:L1:L2*，*L3:L2:L1* なる 6 通りの順序に分けられる．また，大きさを 3 段階に分けて考えるなら，[0,100]ms をレベル 0，[100,200]ms をレベル 1，200ms より大きいをレベル 2 とすることができる．以降では，説明のため遅延時間の順序のみでグループ分けされると仮定する．例えば，3 個のランドマークノードを用いた場合，節点は順序数 3! 個のグループに分類される．

次に，m 個のランドマークノードを用いた場合，CAN 座標空間を $m!$ 個の同じ大きさの空間を持つグループに分割する．分割は決まった次元サイクル (例えば，xyzxyzx...) で行われ，まず初めの次元で m 分割され，次に 2 つ目の次元で $m-1$ 分割され，以降同様に続ける．新たな節点が加入するとき，加入節点は各ランドマークノードとの遅延時間を測定し，グループ分けされた空間上でランダムな座標空間を割り当てられる．このようにして，ランドマークノードを用いて，遅延時間の指標を元に節点のグループ分けを行い，加入する節点をグループ内の座標空間に割り当てることにより IP ネットワークの近接関係を CAN のネットワークへある程度反映させることができる．

7. シミュレーション実験

本節では，オーバーレイネットワーク上での提案経路探索法を用いたコールアドミッションの有効性をシミュレーション実験により検証する．実験 1 では，オーバーレイネットワーク上での 4.1 節の式 (2)，(3) を満たす経路の中で枝コスト総和が最小の経路を探索する Dijkstra 法 (従来手法) を用いたコールアドミッションと，式 (2)，(3) を満たす経路の中でホップ数が最小の経路の中でコスト最小の経路を探索する提案手法を用いたコールアドミッションの拒否率と実行時間を比較する．実

験 2 では，IP ネットワークをモデル化したネットワーク上に，CAN を用いてオーバーレイネットワークを構築し，より詳細なシミュレーション実験により提案手法の有効性を検証する．実験に用いた計算機の CPU は UltraSPARC- i 1062MHz，メモリは 2GB である．

7.1 実験 1：Dijkstra 法と提案手法を用いたコールアドミッションの性能比較

オーバーレイネットワーク上での Dijkstra 法を用いたコールアドミッション手法と，提案手法を用いたコールアドミッション手法を C 言語で実現し，枝密度 10% に対して節点数を変化させたときの拒否率と実行時間を計測した．

7.1.1 準 備

入力ネットワーク

オーバーレイネットワークのモデル化として，ランダムグラフ [9] を作成した．100×100 のグリッドを用いて，枝密度 10% に応じた枝数をもつ節点数 100, 150, 200, 250, 300 個の 5 種類のランダムグラフを作成した．また，枝 1 本の容量を 10 に設定した．

コールの入力系列

設定した種類は組合せ数の半分である．また, コールの数は総容量と同数とし，バンド幅要求を 1 とした．

7.1.2 実験結果と考察

枝密度を 10%に固定し節点数を変化させたときの Dijkstra 法と提案手法を用いたコールアドミッションの計測結果をそれぞれ表 1，表 2 に示す．ただし，Dijkstra 法において，始点 s から任意の節点への経路の最小コストを保持するためにヒープ木を用いている．節点名，経路の枝の合計コストの組をヒープ木の要素とし，ヒープ木の根は常にコストが最小の要素を持っている．Dijkstra 法における *in* はヒープ木に保持した要素の総数，*out* はヒープ木から取り出した要素の総数とする．提案手法における，*in* は，キューに保持した要素の総数，*out* はキューから取り出した要素の総数とする．

表 1 Dijkstra 法の節点数に対する拒否率と実行時間

節点数	拒否率	*Time*[秒]	*in*	*out*
100	0.799	0.25	85,295	47,049
150	0.631	2.30	575,095	318,201
200	0.528	9.68	1,717,78	950,438
250	0.520	23.8	3,465,228	1,902,272
300	0.499	51.5	6,313,553	3,424,950

表 2 提案手法の節点数に対する拒否率と実行時間

節点数	拒否率	*Time*[秒]	*in*	*out*
100	0.799	0.16	58,153	28,157
150	0.634	1.23	379,764	179,947
200	0.513	3.80	1,169,610	473,495
250	0.500	7.96	2,357,892	854,236
300	0.481	15.2	4,139,988	1,386,822

表 1，表 2 より，提案手法は Dijkstra 法に比べて，節点数 200, 250, 300 で拒否率の低い経路割当てを行っていることが確認できた．また，提案手法は Dijkstra 法に比べて，すべての節点数の入力で計算時間が短縮され，計算を 1.6〜3.4 倍に高速化することができた．計算時間の短縮については, Dijkstra 法では, 4.1 節の式 (2)，(3) を満たす経路の中で枝コスト総和が最小の経路を探索するため, 経路を求める計算時間が枝数に非常に依存してしまう．一方, 提案手法では式 (2)，(3) を満たす経路の中でホップ数が最小の経路の中でコスト最小の経路を探索するため, ホップ数とコストの値を保持することで探索する経路を大幅に減らすことができる．実際，Dijkstra 法と提案手法の *in*，*out* を比較すると，提案手法の探索している要素数は Dijkstra 法と比べてほぼ半分以下である．提案手法は, 経路の高速な探索を行い，コールの受理，拒否が効率的に行われていることから，コールアドミッションに適した経路探索法であると言える．

7.2 実験 2：モデル化した IP ネットワーク上に構築したオーバーレイネットワーク上での実験

実験 2 では，IP ネットワークをモデル化したネットワークと，モデル化したネットワーク上に CAN を用いて構築されるオーバーレイネットワークを入力ネットワークとした．まず, IP ネットワーク上での最小ホップ経路制御を用いたコールアドミッションを C 言語で実現し，拒否率を計測した．この拒否率は許容できない通信量を表している．また，Dijkstra 法によるコールアドミッションと，提案手法によるコールアドミッションを C 言語で実現し，拒否率，実行時間，受理されたコールにルーティングしたオーバーレイネットワーク上の経路のホップ数を計測した．

7.2.1 準 備

入力ネットワーク

IP ネットワークのモデル化として，Georgia Tech Internetwork Topology Generator (GT-ITM) [10] を用いて節点数 1000，枝数 2034 の Transit-Stub グラフ (T-S グラフ) を作成し，枝には T-S グラフの枝の種類に応じて容量を設定し，Transit-Transit 間の枝に 1500，Transit-Stub 間の枝に 1000，Stub-Stub 間の枝に 500 とした．また，T-S グラフ上に CAN を用いて節点数 50，枝数 209 のオーバーレイネットワークを作成した．各オーバーレイネットワークの枝 1 本に対して，IP ネットワークの枝数は平均 7.15 本であった．

コールの入力系列

始点，終点に対応する 2 節点間にランダムな確率を与えてコールの到着間隔を計算し，ポアソン過程を用いて発生させたコールの系列を入力とした [8]．設定した種類は 25, 50, 100, 200, 400 である．また, コールの数は 8000 個とし，バンド幅要求を 1 とした．

7.2.2 実験結果と考察

IP 経路制御，Dijkstra 法，提案手法を用いたコールアドミッションの 3 つの計測結果を表 3, 表 4 に示す．ただし，表中の *Time* は実行時間を表し，$\overline{hop}$ は, 受理されたコールにルーティングしたオーバーレイネットワーク上の経路のホップ数の平均値である．

表 3 より，オーバーレイネットワークを利用した提案手法を

表 3　IP 経路制御と提案手法のコールの種類に対する拒否率

コールの種類	拒否率	
	IP	提案
25	0.537	0.107
50	0.444	0.081
100	0.397	0.063
200	0.393	0.063
400	0.387	0.060

表 4　Dijkstra 法と提案手法を用いたコールアドミッションの計測結果

コールの種類	経路制御方法					
	Dijkstra 法			提案手法		
	拒否率	$Time$[秒]	$\overline{hop}$	拒否率	$Time$[秒]	$\overline{hop}$
25	0.141	0.92	2.92	0.107	0.54	2.40
50	0.128	0.91	2.89	0.081	0.40	2.40
100	0.096	0.88	2.78	0.063	0.46	2.30
200	0.094	0.82	2.78	0.063	0.70	2.26
400	0.090	1.02	2.75	0.060	0.70	2.25

用いたコールアドミッションは，IP 経路制御を用いたコールアドミッションと比較して，拒否率は非常に低くなっている．特に，コールの種類が制限された場合，IP 経路制御ではホップ数最小の経路がすぐに飽和状態になってしまうことがわかる．それに対して，オーバーレイネットワークを利用した提案手法を用いたコールアドミッションでは，複数の経路を探索可能であり，コールを分散して割当てることができたことが拒否率の低い経路割当ての原因である考えられる．

表 4 より，提案手法によるコールアドミッションは Dijkstra 法によるコールアドミッションと比較して，経路の高速な探索が行われ，拒否率の低い経路割当てを行っていることが確認できた．計算時間の短縮については，提案手法ではホップ数とコストの値を保持することで探索する経路を大幅に減らすことができていることが原因であると考えられる．Dijkstra 法では枝コスト総和が最小の経路を探索するためホップ数による制約がないのに対して，ホップ数に制約をもつ提案手法はよりホップ数の小さい経路を選択する．オーバーレイネットワーク上での 1 ホップは，複数の IP ネットワークノードを中継することを意味する．そのため，提案手法は Dijkstra 法と比べて各コールの割当てにおいて枝の容量を占有する枝数が少なくできたことが拒否率の低い経路割当ての原因であると考えられる．具体的には，各オーバーレイネットワークの枝 1 本に対して，IP ネットワークの枝数は平均 7.15 本，コールの種類 25 のときの Dijkstra 法の平均ホップ数 2.92，提案手法の平均ホップ数 2.40 より (2.92-2.40)×7.15=3.718 であるから，提案手法と比較して Dijkstra 法は 1 つのコールにつき IP ネットワークの枝約 3.7 本分多くの枝の容量を使っていると予測される．

8. おわりに

本稿では，各ノードが一部のノードへのリンクしか保持しないような大規模オーバーレイネットワークを扱い，大容量データ通信が行われる IP ネットワークにおける，オーバーレイネットワークを利用したコールアドミッションと経路制御を提案した．また，提案手法を用いたコールアドミッションの有効性をシミュレーション実験により確認した．今後の課題として，より現実に近い IP 経路制御との比較による提案手法を用いたコールアドミッションの評価，提案手法を用いたコールアドミッションに適したオーバーレイネットワークの構築についての調査などが挙げられる．

文　献

[1] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris, “The case for resilient overlay networks,” Proc. of the 8th Annual Workshop on Hot Topics in Operating Systems, pp.152–157, 2001.

[2] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R.Morris, “Resilient overlay networks,” Proc. of 18th ACM Sumposium on Operating System Principles (SOSP), pp.131–145, 2001.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An architecture for differentiated services” RFC 2475, 1998.

[4] A. Borodin and R. El-Yaniv, “Online Computation and Competitive Analysis,” Cambridge Univ. Press, pp.226–237, 1998.

[5] S. Braden, D. Clark, S. Shenker, “Integrated services in the Internet architecture:an Overview,” RFC 1633, 1994.

[6] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, “A scalable content-addressable network,” Proc. of SIGCOMM2001, pp. 161–172, 2001.

[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” Proc. of INFOCOM 2002, 2002.

[8] S. M. Ross, “Simulation: Second Edition,” Academic Press, pp.74–77, 1996.

[9] L. Wei and D. Estrin, “The Trade-offs of Multicast Trees and Algorithms, ” Computer Science Department University of Southern California, pp.1–15, 1995.

[10] E. Zegura, K. Calvert, S. Bhattacharjee, “How to model an internetwork,” Proc. of 15th Annual Joint Conf. of the IEEE Computer and Communications Societies, 1996.

[11] 村田正幸, “サービスオーバーレイによるネットワークの高信頼化,” 電子情報通信学会誌, Vol.89, No.9, pp. 792–795, 2006.

[12] 山本正樹, 上土井陽子, 若林真一, “コールアドミッションアルゴリズムに適した経路探索法の提案,” , 情報処理学会アルゴリズム研究会, AL-110(8), 2007.