修士論文

クラスタリングを用いた オーバーレイネットワークの構築

平成 22 年度修了
三重大学大学院 工学研究科
博士前期課程 情報工学専攻

小林　悠二

目次

はじめに

近年，固定的なインフラストラクチャーがない環境下でも，移動端末同士により自律的にネットワークが構築されるモバイルアドホックネットワーク（mobile ad hoc network，MANET）が注目されている．MANET は，会議やイベント会場等で一時的な通信手段として用いられたり，災害が発生した場合等の緊急時のネットワークとして利用されるなど，様々な環境下での用途が期待されている．一方，ファイル共有などに利用されている P2P のネットワーク上で，ファイル検索を高速に行うために分散ハッシュテーブル（DHT）という技術が利用されている．MANET 環境下では，端末同士がネットワークを構築するので，中央サーバが存在せず，ネットワーク上にあるコンテンツ（情報）を探索する方法が必要となってくる．そこで，MANET 上でオーバーレイネットワークを構築し，DHT を利用することで目的のコンテンツ探索の効率化を図る研究が盛んに行われている [1][2][3]．その中に，ノードの物理的な位置を考慮し，クラスタリングを行うことで無駄な物理ホップを削減している MADPastry[4] がある．

MADPastry では検索を行う際，DHT の Pastry[5] の経路表を使い，目的ノードがいるクラスターにたどり着く．クラスター内では，ノードの ID 値の前後のノード情報（リーフセット）ならびにブロードキャストを用いて検索を行う．参加しているノードが少数のネットワークなら，一つのクラスターの大きさがあまり大きくならず，既存のクラスター形成でも検索範囲が狭いので問題ないが，大規模なネットワークになると検索におけるホップ数が大きくなる．

そこで本研究では，クラスタリングを階層化することで，大規模なネットワークにおけるコンテンツ検索の効率化を目的とする．

以下，第一章では MANET について述べる．第二章ではオーバーレイネットワークについて述べる．第三章では，分散ハッシュテーブルについて述べる．第四章では，比較対象である MADPastry について述べる．第五章では MADPastry の問題点を述べた後，その解決法である二段階クラスタリングを説明する．第六章では，提案手法の有効性を示すために，シミュレーション実験を行い，MADPastry と比較・評価を行う．

第１章
MANET について

近年，ノート PC や携帯端末等の無線機器が普及しており，それに伴い無線ネットワークが使われている．無線機器の動作モードには，「インフラストラクチャーモード」，「アドホックモード」の二種類がある．インフラストラクチャーモードでは，アクセスポイントを介して他の機器と通信を行う．一方，アドホックモードでは，無線端末同士が直接通信を行う．図１はインフラストラクチャモード、図２はアドホックモードによるネットワーク形成を示す．

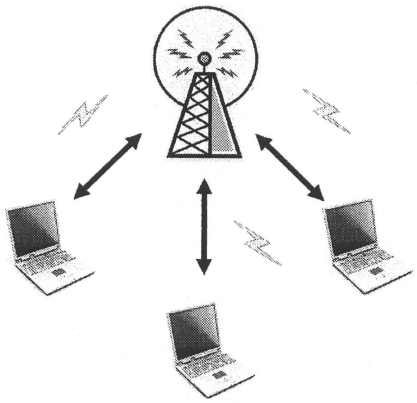


図１　インフラストラクチャーモード

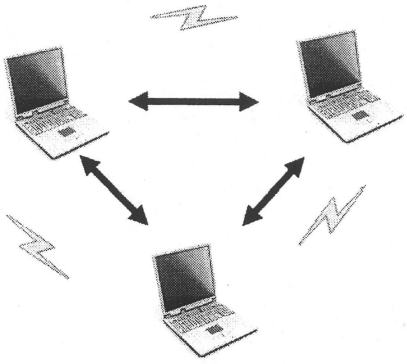


図２　アドホックモード

1.1 MANET

図 3 のように，無線端末同士がアドホックモードで直接通信を行うことで構築されるネットワークのことを MANET と呼ぶ．端末同士で自律的にネットワークを構築するので，地震等の災害が起こった際の緊急時のネットワークとして利用されたり，会議やイベント会場等で一時的な通信手段として用いられるなど，様々な環境下での用途が期待されている．

MANET のルーティングプロトコルには，Proactive 型プロトコルと Reactive 型プロトコルがある．

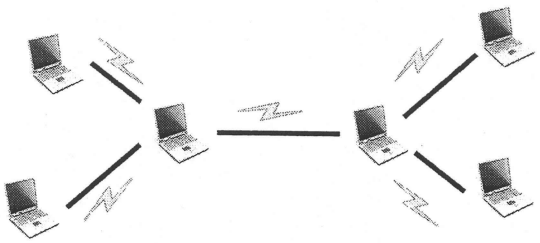


図 3　MANET

1.1.1 Proactive 型プロトコル

Proactive 型プロトコルでは，各端末が常に経路を維持しており，通信要求があった際にすぐにデータを送ることができる．しかし，経路維持のために制御パケットを常に出しているので，電力の消費が激しい．代表的な Proactive 型プロトコルに OLSR(Optimized Link State Routing) がある．

1.1.2 Reactive 型プロトコル

Reactive 型プロトコルでは，通信要求が出てから経路探索を行う．そのため，経路維持のための制御パケットが減らせるので，消費電力が抑えられる．しかし，通信要求があってから経路を探すので，遅延が発生する．代表的な Reactive 型プロトコルに AODV(Adhoc On-demand Distance Vector) がある．

第 2 章

オーバーレイネットワーク

オーバーレイネットワークとは，コンピュータネットワーク上に構築された別のコンピュータネットワークのことである． 図 4 のように，IP ネットワーク上に配置されたコンピュータがオーバーレイネットワークを構築すると，IP ネットワークの配置を気にせずに別のネットワークを構築する．

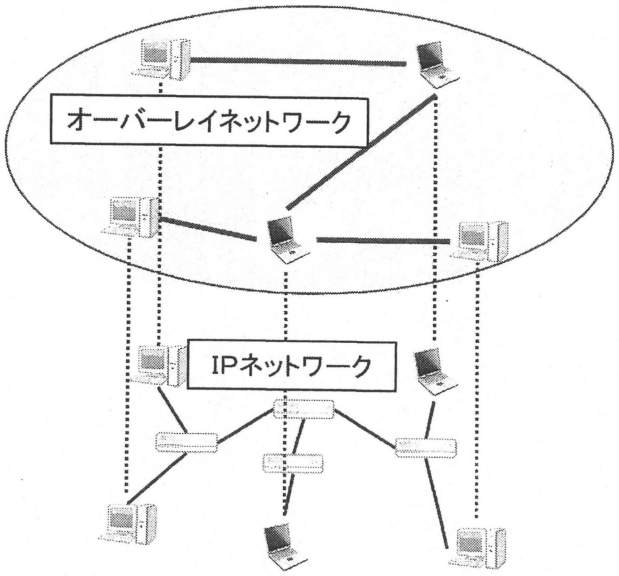


図 4　オーバーレイネットワーク

2.1　クライアント・サーバ型

ある特定の役割を集中的に担当するサーバに，クライアントが「要求」を出し，それに対応した処理がサーバによって行われて「応答」が返される．このように，役割が明確に決まっているシステムをクライアント・サーバ型と呼ぶ．図 5 はその例である．

クライアント・サーバ型では，クライアントの処理が小さい，システムの管理や制御，著作権処理やログイン処理がしやすいこと等が長所として挙げられる．一方，サーバの処理負荷が非常に高いので，高価なコンピュータを多数揃える必要がある．また，サーバに処理が集中するため，サー

バに繋がるための回線帯域を圧迫する恐れがあり，より大容量の回線が必要となる．これらが短所として挙げられる．

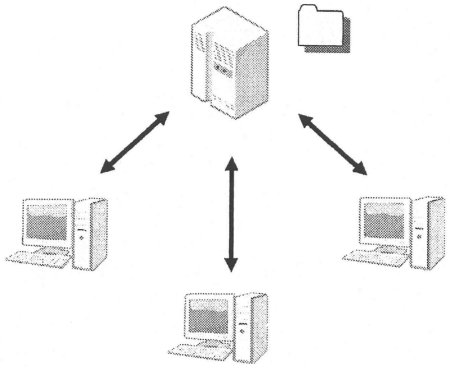


図 5　クライアント・サーバ型

2.2　P2P

P2P(Peer to Peer) の”Peer” とは，「対等の」という意味がある．P2P システムを構成する各ピアは，あるときはサーバとして動作し，あるときはクライアントの動作をする．このようにピアはサーバとクライアントの二役をこなし，P2P システムにつながっているコンピュータ同士は対等の立場になる．図 6 はその例である．

P2P 型の長所として，個々のコンピュータの性能や回線の容量がそれほど要求されないこと，匿名性が高いこと等が挙げられる．また，短所として，処理が分散するために管理や監視がしにくいこと等が挙げられる．

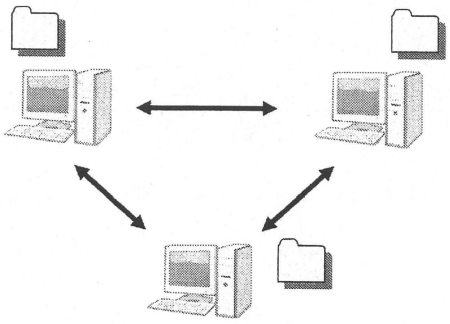


図 6　P2P 型

P2P では各ノードの役割の観点から，ハイブリッド P2P，ピュア P2P，スーパーノード P2P の 3 つに分類される．

2.2.1 ハイブリッド P2P

ハイブリッドとは，クライアント・サーバ型と P2P 型の「混合」という意味である．ファイルのメタデータをもったインデックス・サーバ (検索用サーバ) が存在し，ファイルそのものは各ノードが所持している．各ノードはデータが欲しいとき，インデックス・サーバにデータの在処を問い合わせる．インデックス・サーバから該当のデータをもつノードの情報を得られたら，直接 P2P 接続によりデータ交換をする（図 7)．

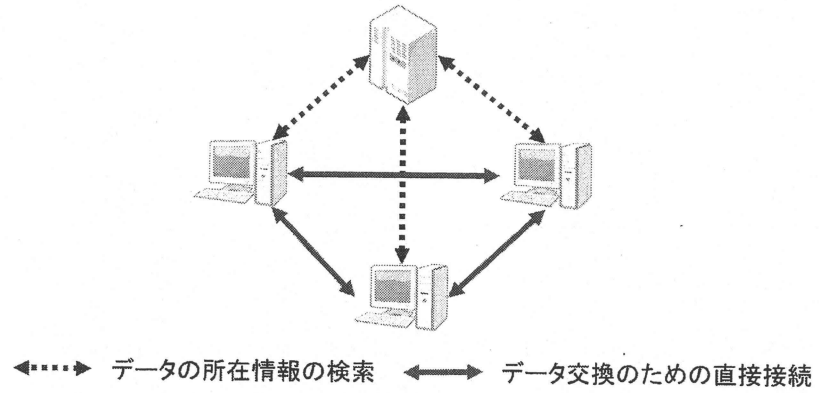


図 7　ハイブリッド P2P

2.2.2 ピュア P2P

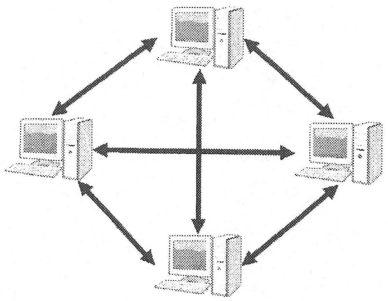
ピュア P2P では，ハイブリッド P2P とは違い，インデックス・サーバを持たずにピアとピアだけで直接通信を行う（図 8)．各ピアはファイルとメタデータを所持し，クライアントとサーバの両方の機能をもっている．ピュア P2P のネットワークを形成しているのはピアだけであり，データの検索も「フラッディング」や「分散ハッシュテーブル (DHT)」などの方法で行う．「分散ハッシュテーブル」については，第三章で説明する．全てのノードを管理するインデックス・サーバが存在しないので，ノードが膨大になったときのスケーラビリティ，インデックス・サーバがダウンしたときの耐故障性がハイブリッド P2P より優れている．しかし，実装が複雑であり，ノード数が増えた場合の探索クエリによる帯域消費が増大するという欠点がある．また，耐故障性が高いが，その逆として，一度システムが機能すると，システムそのものを駆逐することが難しいという欠点も挙げられる．ピュア P2P には，ノードが構成する論理ネットワーク構造に制約がなく任意のグラフ構造を取り得る unstructured 型と，ノードが規則的な論理ネットワーク構造を構築する structured 型がある．

- unstructured 型

  探索に「フラッディング」を使う．ノードが構成する論理ネットワーク構造に制約が無く，任意のグラフ構造を得る．アルゴリズムが簡単であるが，オブジェクトの探索や経路制御の効率が悪い．また，全てのノードに問い合わせメッセージが到達する保証がない．

- structured 型

  分散ハッシュテーブル (DHT) を利用している．ノード群はリストやツリーのような経路表を構築し，規則的な論理ネットワーク構造を構築する．規則的な構造がある上での検索なので，効率的かつ網羅的である．しかし，短所として DHT の実装は難しいということが挙げられる．



←→ データ検索あるいはデータ交換のための直接接続

図 8　ピュア P2P

2.2.3　スーパーノード型ハイブリッド P2P

スーパーノード型ハイブリッド P2P とは，ハイブリッド P2P とピュア P2P の欠点を補い，利点を生かした P2P モデルである．ピュア P2P では，全てのノードがファイルをアップロードやダウンロードなどの同一の機能をもっていたが，スーパーノード型 P2P では特定の選ばれたノード（スーパーノード）がハイブリッド P2P でのインデックス・サーバのような役割を果たす（図 9）．しかし，このスーパーノードは固定的なものではないので，スーパーノードがダウンした場合でも他のノードから再度選び出すことでスケーラビリティを確保できる．このスーパーノードの存在によりピュア P2P の欠点である探索クエリによる帯域圧迫が起こらなくなる．しかし，探索データの分散化など，実装が難しくなること，ノード数がある程度の規模になるまでは安定しない可能性があるということが欠点として挙げられる．

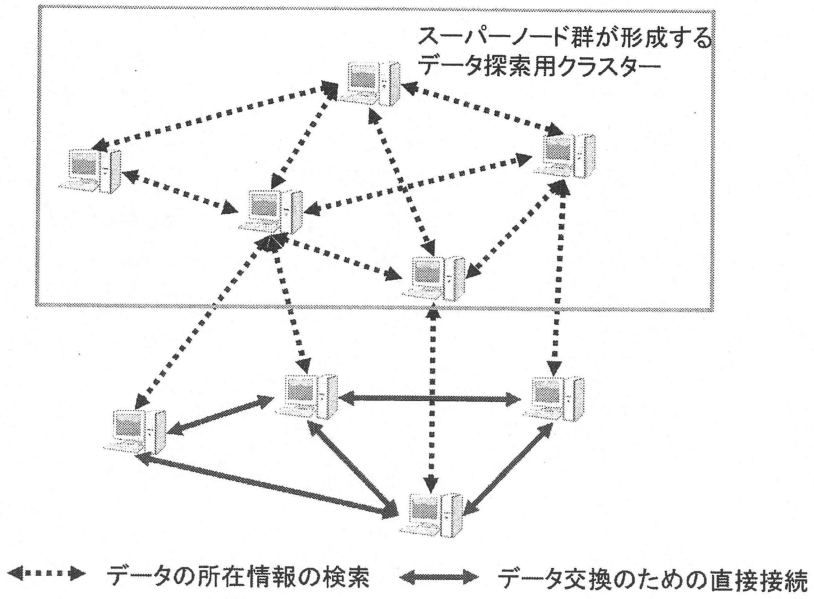


図 9　スーパーノード型ハイブリッド P2P

第 3 章
分散ハッシュテーブル

3.1 分散ハッシュテーブル

分散ハッシュテーブル (DHT:Distributed Hash Table) は，ピュア P2P でのデータ探索技術の一つである．代表的なアルゴリズムに Pastry，Chord[6]，CAN[7]，Tapestry[8] などがある．DHT で用いるハッシュ値，ハッシュテーブル，分散ハッシュテーブルについて以下に述べる．

- ハッシュ値

  与えられたデータから固定長の疑似乱数を生成するハッシュ関数を使うことで生成された値のこと．ハッシュ関数には「SHA-1」,「MD5」などがあり，本研究では「SHA-1」を使用している．ハッシュ関数は不可逆な一方向関数なので，ハッシュ値から元データを得ることはできない．また，ハッシュ値は元データが同じでハッシュ関数も同じであれば，同一のものが生成される．
- ハッシュテーブル

  ハッシュテーブルとは，検索をする際の手がかりとなる情報であるキーとキーに関連する情報であるハッシュ値 (エントリ) の組を複数格納した表であり，キーに対応する値をすばやく参照するためのデータ構造である．
- 分散ハッシュテーブル

  ハッシュテーブルを複数のノードで分散管理する技術のこと．分散ハッシュテーブルに参加している各ノード (ピア) には，それぞれアドレスのハッシュ値であるノード ID が割り当てられている．また，コンテンツ (キー) に対するハッシュ値を求め，これらを ID 空間に写像し，その空間を複数のノードで分割管理することで，特定ノードに負荷が集中することなく大規模なコンテンツ探索を実現する．ノードが管理するコンテンツのハッシュ値の範囲は各ノードが分散ハッシュテーブルに参加するときに自律的に決定する．このノードが管理する範囲は，ノードの参加，脱退により変化する．分散ハッシュテーブルでどのようにノードの管理範囲が決まるかを図 10 で簡単に説明する．

参加ノードであるノード A，B，C のアドレスをそれぞれハッシュ関数にかけ，それにより求められたハッシュ値を同じ ID 空間上に写像する．コンテンツにもノードに使ったものと同じハッシュ関数を使い，ハッシュ値を求めて同じ ID 空間上に写像する．ID 空間に写像された各ノードは，自分のハッシュ値から別のノードのハッシュ値までの間の範囲のコンテンツを管理する．例えば，ハッシュ値が 0152 であるノード B なら，次に ID 空間上に表れるノードであるノード A のハッシュ値 2781 までのコンテンツを管理することになる．同様に，ノード A は 2781 から次のノードであるノード C のハッシュ値 3306 までのコンテンツを管理する．分散ハッシュテーブルのアルゴリズムが違えば，コンテンツ管理の方法も変わってくる．

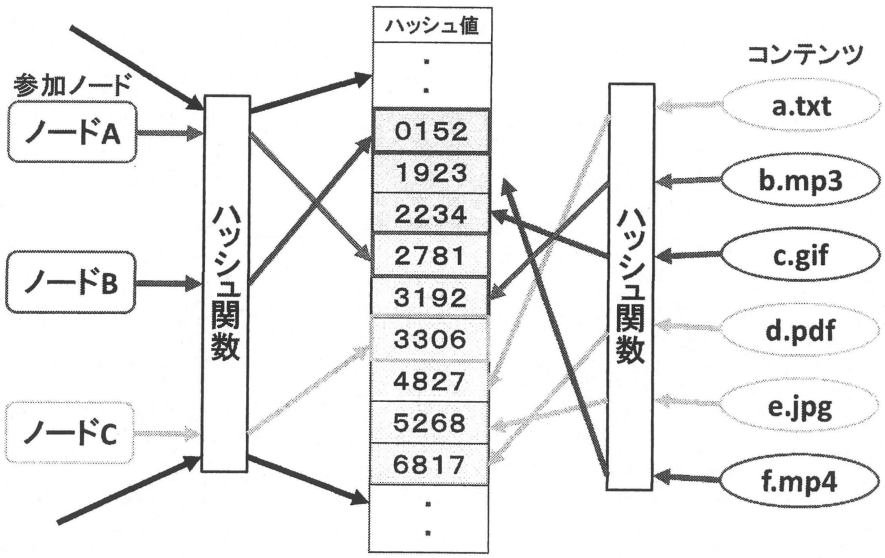


図 10　分散ハッシュテーブル

3.2　Pastry

Pastry では，ノードは保持するコンテンツのポインタ（コンテンツ名と自身の IP アドレスの組）を分散ストレージに登録する．コンテンツ情報はコンテンツのハッシュ値に一番近いノード ID(ハッシュ値) をもつノードが管理する．Pastry での近さ (距離) は共通プレフィックスの長さ (共通接頭語のビット数) で決まる．Pastry の特徴として，プレフィックスに基づいたルーティングを行うことが挙げられる．コンテンツを検索するときは，コンテンツを管理するノードに向けてクエリを出し，コンテンツを発見する．コンテンツ発見にかかる Pastry ネットワーク上でのホップ数は $O\log(n)$ である．また，個々のノードは木構造に似たデータ構造になる．Pastry では，各ノードは以下の情報を持つ．

- 経路表

  各ノードが表 1 のような経路表を持つ（ただし，*には 0~F までのいずれかが当てはまっている）．レベルが上がるにつれて，経路表を所持するノードのノード ID と，経路表のエントリであるノードのノード ID との一致する桁数が多くなる．

- リーフセット

  自分の ID 値に近い前後のノード情報．リーフセットは前後のノードと論理リンクを張るので，全てのノードがリング状に繋がる．

表 1　ノード 8A23 の経路表の例

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
レベル 1	0***	1***	2***	3***	4***	5***	6***	7***	self	9***	A***	B***	C***	D***	E***	F***
レベル 2		81**	82**	83**		85**	86**			89**	self		8C**	8D**	8E**	8F**
レベル 3	8A0*		self			8A5*									8AE*	
レベル 4				self					8A28							

表 2　リーフセットの例

nodeID:1232		
	low_leaf	high_leaf
1	1230	1234
2	1225	1236
3	1222	1239
4	121B	123C

nodeID:1234		
	low_leaf	high_leaf
1	1232	1236
2	1230	1239
3	1225	123C
4	1222	1241

nodeID:1236		
	low_leaf	high_leaf
1	1234	1239
2	1232	123C
3	1230	1241
4	1225	1243

3.2.1　ルーティング

Pastry ではメッセージを送るときにノード ID の数字列が一桁ずつ一致していくようにルーティングしている．例えば，図 11 で E791(本論文では，ノード ID が****であるノードを，「ノード****」や「****」と表記する) から 4377 へとメッセージを送るときは，E791 の経路表から一桁目が 4377 と一致する 4B4F を選び，次に 4B4F が経路表から 4377 と ID 値の先頭から二桁一致する 4361 を選ぶ．このように一桁ずつ一致していくように繰り返すことで，4361 → 437A → 4377 とルーティングを行い，目的のノード 4377 にたどり着く．また，Pastry のルーティングには，リーフセットが参照されることがある．例えば，先の例で 4361 までルーティングした後，4361 のリーフセットに 4377 の情報があれば，4361 から直接 4377 へとルーティングする（図 12）．

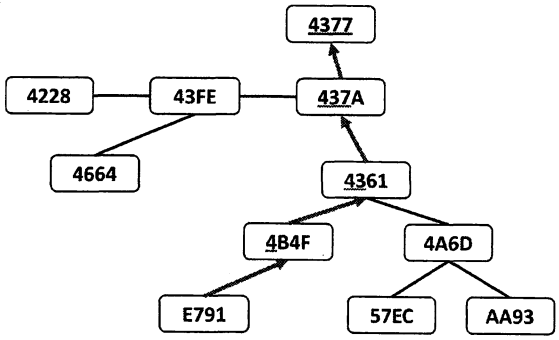


図 11　Pastry のルーティング（1）

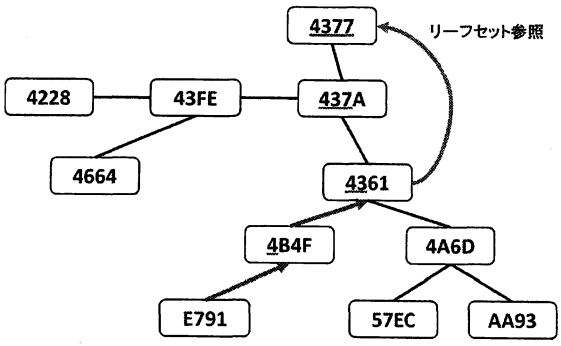


図 12　Pastry のルーティング（2）

第 4 章
MADPastry

4.1 オーバーレイホップと物理ホップ

図 13 のように構築された MANET 上で，オーバーレイネットワークを構築すると，オーバーレイネットワークと物理的なネットワークである MANET との間で問題が出る．例として，ノード S からノード A，B，C，D を経由してノード T へとオーバーレイホップする場合を考える．オーバーレイネットワーク上では 5 ホップとなり，一見何の問題もなく思える．しかし，このオーバーレイホップに従ってルーティングする際の MANET 上の物理ホップを計測すると，13 ホップとなる．ノード S からノード T の MANET での最短となる物理ホップ数は 6 ホップなので，オーバーレイホップすることでより多くのノードを経由して物理ホップしたことになる．第 2 章で述べたように，オーバーレイネットワークでは，ノードの配置を気にすることなく別のネットワークを構築しているので，このような無駄な経路をルーティングしたことで物理ホップが増大した．有線で接続されたネットワークではあまり問題とならないが，MANET では無線を用いているので物理ホップの増大は致命的な問題となる．

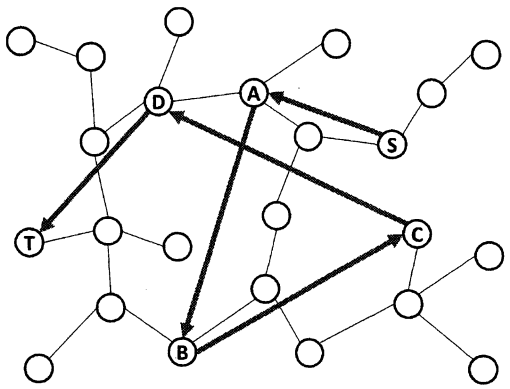


図 13　オーバーレイホップと物理ホップ

4.2 MADPastry

MADPastry は，DHT の手法の一つである Pastry を用いて，MANET 上にオーバーレイネットワークを構築する．しかし，MANET 上でオーバーレイネットワークを構築すると，上記のような問題が出てくる．そこで，MADPastry では物理的に近いノード同士をクラスタリングすることで，無駄な物理ホップを削減している（図 14）．

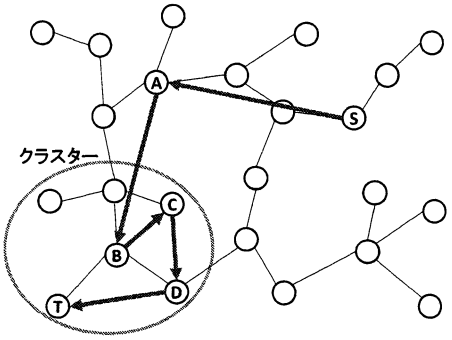


図 14　オーバーレイホップと物理ホップ

4.2.1 クラスタリング

MADPastry に参加するノードは，ランドマークキーと呼ばれる ID 値をあらかじめ知っているものとする．ランドマークキーはクラスター毎に決まっており，各クラスターの代表ノードであるクラスターヘッドを決めるのに用いられる．クラスターヘッドは，ランドマークキーに最も近い ID 値をもつノードが選ばれる．また，クラスターヘッドは定期的にビーコンを出しており，ビーコンはクラスターの境界のノードまで伝搬される．ビーコンを受け取ったノードは，クラスターヘッドまでのホップ数を計測し，最もホップ数が小さいクラスターに入る．

クラスターに属するノードは，各々のノード ID の先頭をクラスター共通の値に変更する．例えば，ハッシュ関数によって求まったノード ID が”2310”というノードが，クラスター 1 に属する場合，ノード ID の一桁目をクラスター 1 で共通の”1”に変更して”1310”とする．このように，クラスター内で共通の数字をノード ID の一桁目に使用することで，物理的に近いノードをオーバーレイネットワーク上でも近くすることができる．

ここで，あるクラスターに属するノードが，別のクラスターに移る場合を考える．例えば，すでに図 15 のようなネットワークが構築されているとする（この図ではクラスターヘッドを CH と略記する）．

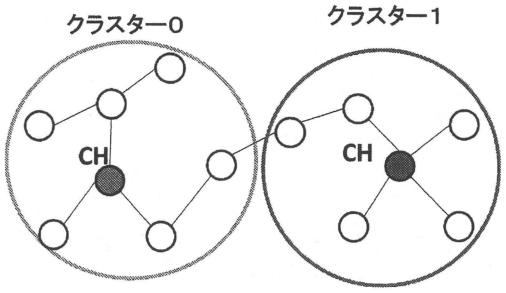


図 15　クラスター

新規ノードがクラスター 1 に入るとする（図 16）．クラスターに入ることでノード ID の一桁目がクラスター共通の数値に変更される．もし，新規ノードの ID 値がクラスター 1 の LM キーに最も近くなるなら，クラスターヘッドの変更が起こる．それに伴い，クラスターの範囲も変化する（図 17）．このようにクラスターヘッドの変更に伴うクラスター変更が何度も繰りかえされる．

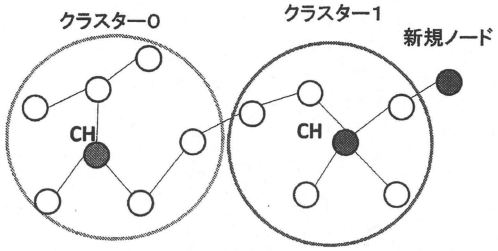


図 16　新規ノード参加

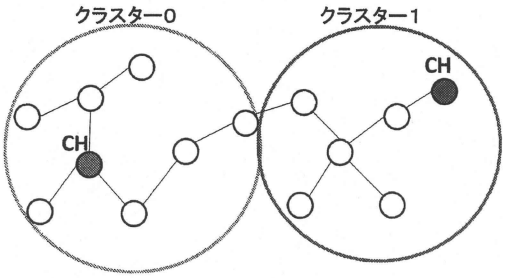


図 17　クラスター変更

4.2.2　ルーティング

MADPastry におけるルーティングの例として，ノード ID 1210 のノード S から，ノード ID 0321 の目標ノード T へのルーティングを示す（ただし，ここでは説明の簡略化のため，ID 値は 4 進数 4 桁とする）．

まず最初に，ノード S はノード T とクラスターが違うので，自身が持つ Pastry 経路表を参照する（表 3）．ノード T のクラスターは”0”なので，経路表の 0 に入っているノード ID 0221 のノード A へルーティングする（図 18）．

表3　ノードSの経路表

0	1	2	3
0221(A)	1210(S)	2300	3223

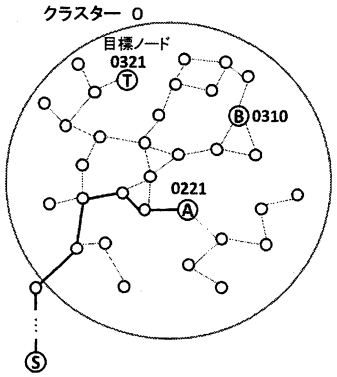


図18　Pastry経路表

次に，ノードTと同じクラスターに属するノードAは，自身のリーフセットを参照する（表4）．リーフセットの中でノードTに最も近いのノードID 0310のノードBへルーティングする（図19）．このように，リーフセットの参照を繰りかえして，目標ノードへ近づいていく．

nodeID:0221		
	low_leaf	high_leaf
1	0212	0223
2	0210	0233
3	0201	0310(B)

表4　ノードAのリーフセット

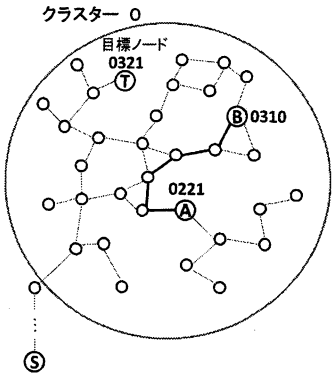


図 19　リーフセット参照

また，MADPastry ではオーバーレイネットワークの経路表以外に AODV 経路表を用いてルーティングを行う．例えば，図 20 でノード A からノード B へリーフセットを用いてルーティングする際，途中で経由するノード C がノード T に関する AODV 経路表を持っていれば，ノード C からノード T へとルーティングする．また，経路表，リーフセットを用いても目的のノードにたどり着かないなら，ブロードキャストして目標ノードを探す（図 21）．

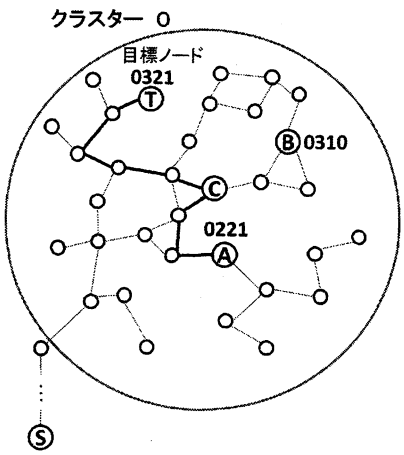


図 20　AODV 経路表

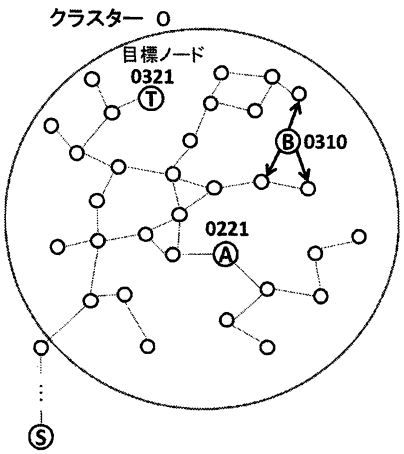


図 21　ブロードキャスト

第 5 章

提案手法について

5.1 MADPastry の問題点

MADPastry では，メッセージが送られる際，まず最初に Pastry の経路表を使い，目標ノードが属するクラスターにたどり着く．MADPastry では，Pastry の経路表は一列分，つまりレベル 1 の部分の情報しか格納されない．よって，経路表を用いたルーティングは，一度しか行われない．参加しているノードが少数のネットワークなら，一つのクラスターの大きさがあまり大きくないので，既存のクラスター形成でも検索範囲が狭いので問題ないが，大規模なネットワークでは検索範囲が広くなってしまう（図 22）．

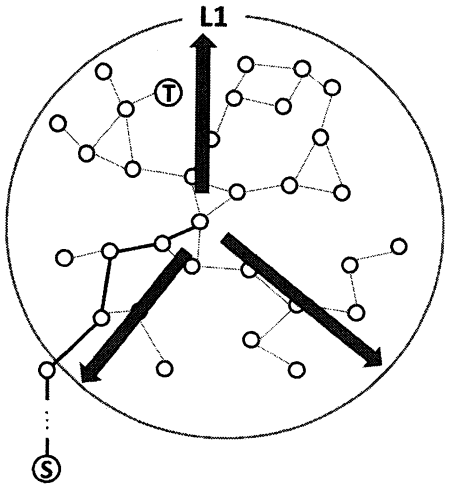


図 22　MADPastry

5.2 二段階クラスタリング

MADPastry では，上記のように大規模なネットワークで検索範囲が広くなるという欠点がある．そこで，クラスターの中で更なるクラスターを形成する二段階クラスタリングを提案する．二段階のクラスターを持つことで，Pastry 経路表を二回使いオーバーレイホップし，小さなクラスターにルーティングする．これにより，図 23 のように検索範囲を狭めることができる．

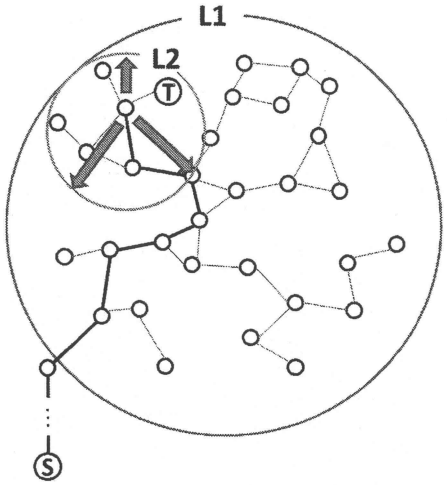


図 23　ブロードキャスト

5.2.1　初期クラスター形成

論文 [4] では，L1 クラスターの初期のクラスター形成について記述されていなかったため，独自にアルゴリズムを考えた．以下では，新たにネットワークに参加するノードを”参加ノード”，既にネットワークに参加しているノードを”参加済みノード”と呼ぶ．また，MADPastry で構築したクラスターを L1 クラスター，L1 クラスター内で構築する更なるクラスターを L2 クラスターと呼ぶ．参加済みノードの通信範囲内にいる参加ノード x は，自身と通信を行えるノードの中から，クラスターヘッドへのホップ数が最も小さいノードを選び，そのノードが属する L1 クラスターへと入る．L1 クラスター数 C，L2 クラスター数 $C2$ の場合のクラスター形成のアルゴリズムを以下に示す．

まず，クラスターヘッドが全て決まるまで，以下のフローチャート（図 24）で L1 クラスターのクラスターヘッドを決めていく．変数 n の初期値は 0 とする．

L1 クラスターのクラスターヘッドが全て決まると，各 L1 クラスター内での L2 クラスター形成は，以下のフローチャート（図 25）に従って行う．また，各クラスターごとに変数 $n2$ があり，初期値は 0 とする．

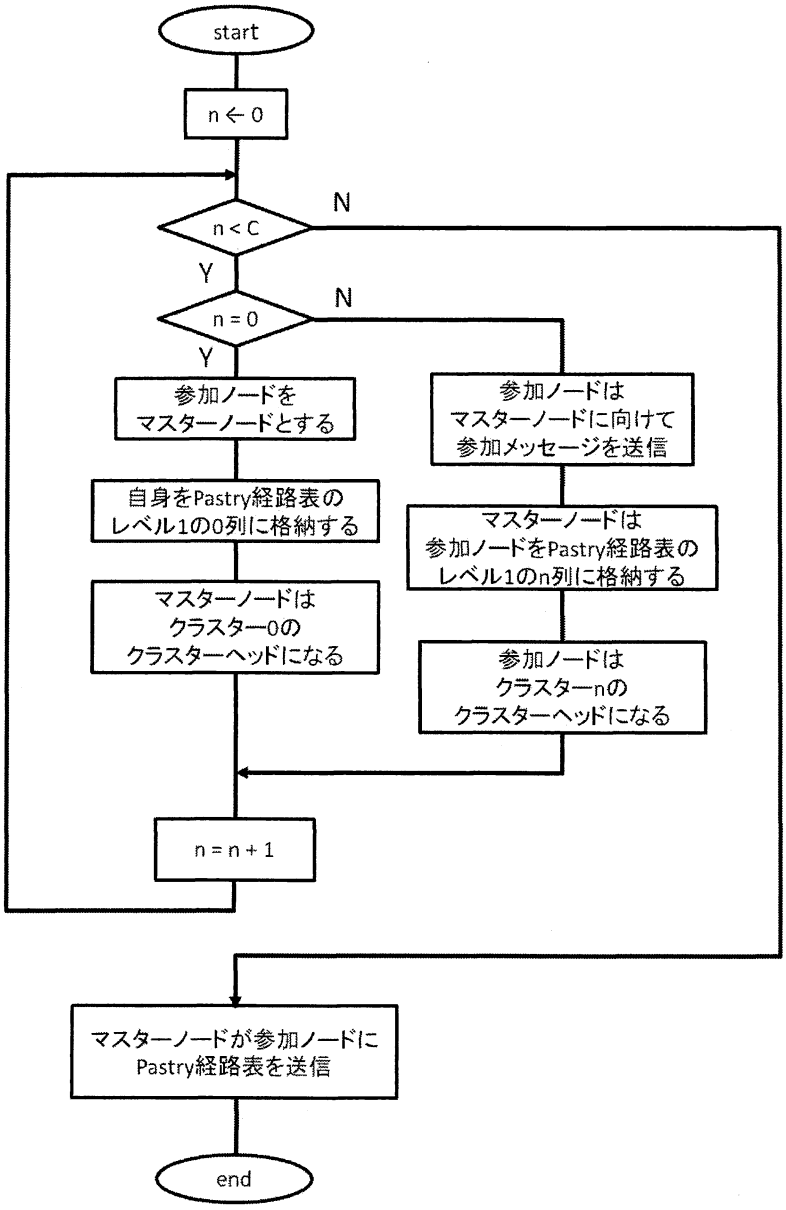


図 24　L1 クラスター形成

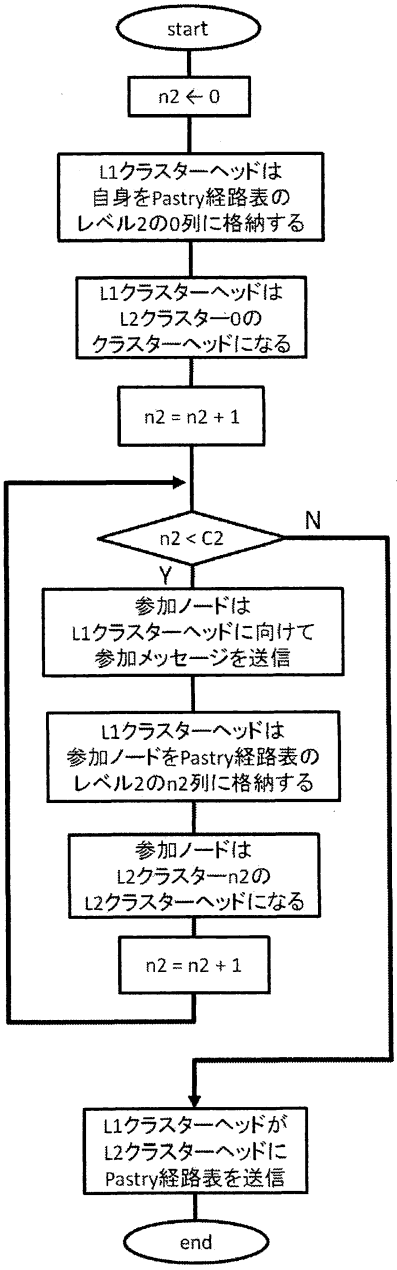


図 25　L2 クラスター形成

クラスター形成について，図で簡単に説明する．ただし，L1 クラスター数 4，L2 クラスター数 4 とするので，クラスター形成に必要な L1 クラスターヘッド数は 4，L2 クラスターヘッド数は 4 となる．まず図 26 のように L1 クラスター形成が完了しており，ノード A，B，C，D を各クラスターのクラスターヘッドとする．

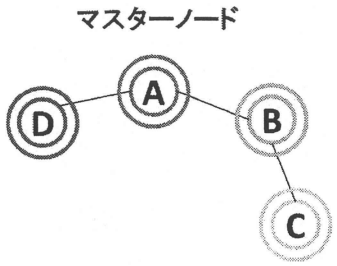


図 26　L1 クラスター形成後

図 27 のようにノード E がノード A の通信範囲内に入り，ネットワークへ参加する際，ノード E の通信範囲にはノード A しかいないので，ノード A が属するクラスターに入る．新規ノードがクラスターに入った際，そのクラスター内で L2 クラスターヘッドの数が決められた数を満たしてなければ，新規ノードを L2 クラスターヘッドとする．よって，ノード E は L2 クラスターヘッドとなる．

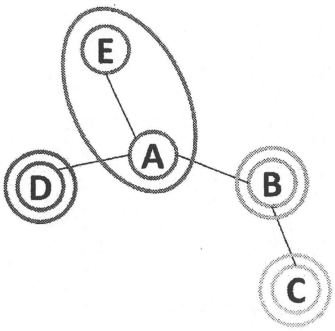


図 27　新規ノード E が参加

ここで，新たにノード F がネットワークに参加する．ノード F の通信範囲内には，ノード D とノード E が存在する．このとき，ノード D，E それぞれが属するクラスターのクラスターヘッドまでの距離を計測する．このとき，ノード E が属するクラスターのクラスターヘッドはノード A なので，ノード F からノード A へのホップ数を計測する．また，ノード D が属するクラスターのクラスターヘッドはノード D なので，ノード F からノード D へのホップ数を計測する．図 28 のように，ノード A には 2 ホップ，ノード D には 1 ホップで到達するので，ノード F はノード D が

属するクラスターに入る．ノードFが入ったクラスターは，L2クラスターヘッドが足りていないので，ノードFをL2クラスターヘッドとする．

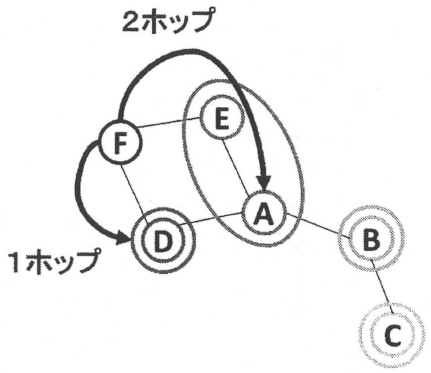


図28　新規ノードFが参加

この動作を繰り返すことで，図29のノードD，F，G，Hが属するL1クラスターのように，L1クラスター内でL2クラスターヘッドが充足したら，そのクラスター内でのL2クラスター形成が完了する．これ以降そのクラスターに入るノードは通常参加を行う．

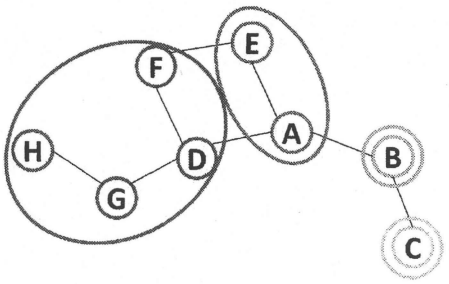


図29　L2クラスター形成完了

第 6 章
評価実験

提案手法の有効性を示すために，コンテンツ検索に関するシミュレーション実験を行った．

6.1 シミュレーション実験

シミュレーション実験を行うために，Java 言語を用いてシミュレータを作成した．各実験で共通するシミュレーション環境を以下に示す．

- シミュレーション領域
  ノード密度が $100nodes/km^2$ となるように設定する．
- 通信範囲
  250m に設定する．
- クラスター数
  L1 クラスター数 16，L2 クラスター数 4 に設定する．
- ノード配置
  シミュレーション領域にランダムに配置する．ただし，マスターノードはシミュレーション領域の中心に配置する．

各実験で異なる条件を表 5 に示す．

表 5　シミュレーション条件

	コンテンツ検索	
	条件 1	条件 2
ノード数	200，500，1000	500
リーフセット数	16	16，8，6，4

6.2 コンテンツ検索に関する実験

条件を変えることで，どのようにオーバーレイホップ数，物理ホップ数が変化するかを調べた．

6.2.1 条件 1

ノード数の変化によるホップ数の変化を調べるために，ノード数 200，500，1000，リーフセット数 16 で実験を行った．実験結果の値は，ノードの配置に用いる乱数の種を変え，10 回の試行の平均を用いた．図 30 にオーバーレイホップ数，図 31 に物理ホップ数の結果を示す（図中の表記で，L1 は通常の MADPastry を，L2 は二段階クラスタリングを表す）．

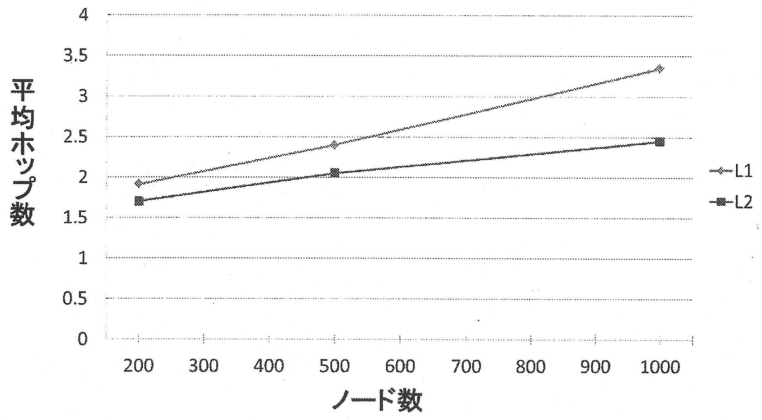


図 30　オーバーレイホップ数

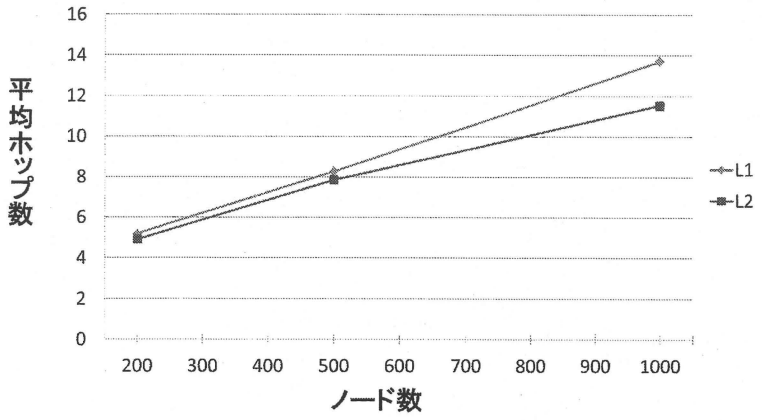


図 31　物理ホップ数

オーバーレイホップ数，物理ホップ数共に，ノード数 200 では L1 と L2 の差があまり見られない．ノード数 500 以上で L1 と L2 の差が出始め，ノード数 1000 ではオーバーレイホップ数で約 1 ホップ，物理ホップ数で 2 ホップ以上の差が表れた．

6.2.2 条件 2

リーフセット数の変化によるホップ数の変化を調べるために，ノード数 500，リーフセット数 16，8，6，4 で実験を行った．実験結果の値は，ノードの配置に用いる乱数の種を変え，10 回の試行の平均を用いた．図 32 にオーバーレイホップ数，図 33 に物理ホップ数の結果を示す（図中の表記で，L1 は通常の MADPastry を，L2 は二段階クラスタリングを表す）．

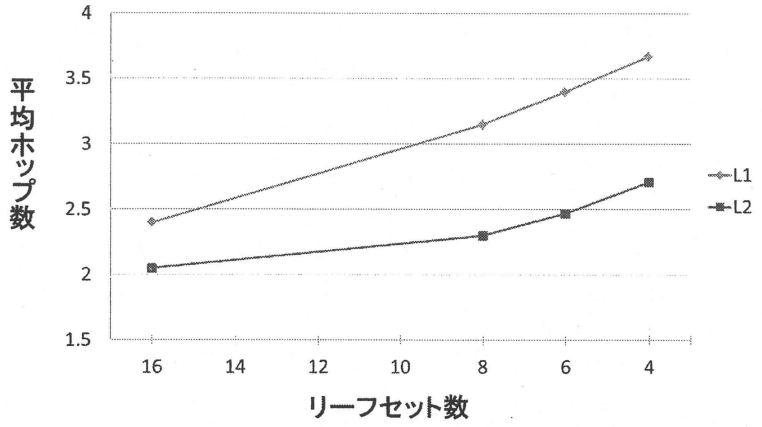


図 32　オーバーレイホップ数

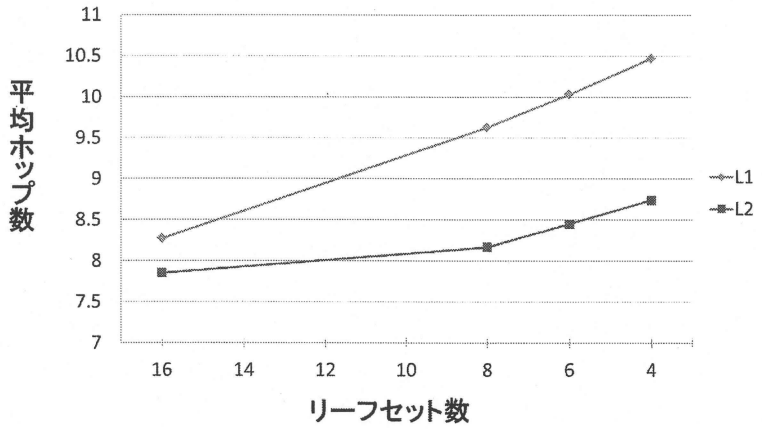


図 33　物理ホップ数

リーフセット数 16 では，L1 と L2 の差は約 0.5 ホップであった．リーフセット数を 8 に減らすと L2 の効果が良く表れ，L1 と L2 の差が約 1.5 ホップとなった．

6.3 考察

参加ノード数が 200 程度のネットワークでは，L1 と L2 の差があまり見られなかった．この理由として，L1 クラスターに属するノード数は平均 13 ノードなので，Pastry 経路表を用いて目標ノードのクラスター内に入ると，リーフセットを用いてすぐに目標ノードが見つかることが原因と考えられる．ノード数が少数のときは L2 の効果が表れなかったが，ノード数 500 を越えると徐々に表れ始め，ノード数 1000 では顕著に表れた．よって，L2 は参加ノード数 1000 以上の大規模なネットワークにおいて有効であると考えられる．ノード数が増加するにつれ，L2 の効果が顕著に表れたので，更にノード数が増えれば，効果が良く表れると考えられる．また，リーフセット数 16 での実験では，L1 と L2 との差はあまり出なかった．この理由として，リーフセット数が大きすぎたためだと考えられる．しかし，リーフセット数を減少させた際の実験では，L2 の効果によってホップ数を削減することができた．MADPastry では，リーフセット全ての更新を保証していないので，リーフセットの更新が少ない状況でホップ数軽減できた L2 は，有用であると考えられる．

おわりに

本論文では，MADPastry を基にした二段階クラスタリングの提案を行った．二段階クラスタリングでは大規模なネットワークを想定し，クラスター内で更なるクラスターを構築することで，クラスター内での検索範囲を狭めた．また，シミュレーション実験による MADPastry との比較を行った結果，大規模なネットワークでの検索ホップ数を削減でき，リーフセットが少ない状況での検索ホップ数の増加を軽減できた．

今後の課題として，ノード移動に対応した実験を行うこと，クラスター構築やコンテンツ検索時のトラフィック量を計測することが挙げられる．

謝辞

日ごろから多くの御指導を頂きました太田義勝教授，鈴木秀智准教授に深く感謝いたします．そして，日頃何かとお世話になりました落合美子事務員に感謝いたします．また，本論文作成にあたって特にお世話になりました太田義勝教授に深く感謝いたします．最後に，日頃から熱心に討論して頂いた研究室の諸氏に感謝いたします．

参考文献

[1] R.Winter,T.Zahn, and J.Schiller, "Random Landmarking in Mobile, Topology-Aware Peer-to-Peer Networks", In Proc. of FTDCS, May 2004.

[2] Matthew Caesar, Miguel Castro, Edmund B.Nightingale, Greg O'Shea, Antony Rowstron, "Virtual Ring Routeing: Network Routing Inspired by DHTs", 2006, SIGCOMM'06, September 11-15.

[3] 金 玲, 亀山 渉, 「DHT を用いたモバイルアドホックネットワークにおける情報発見手法」, 信学技報, vol. 105, no. 624, MoMuC2005-84, pp. 1-6, 2006 年 3 月.

[4] T. Zahn and J. Schiller,"MADPastry: A DHT Substrate for Practicably Sized MANETs", In Proc. of ASWN, 2005.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in Proceedings of Middleware, Heidelberg, Germany, Nov 2001, pp.329350.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan,"Chord: A scalable peer-to-peer lookup service for internet applications" , in Proceedings of SIGCOMM, San Diego, CA, Aug 2001, pp.149160.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network" , in Proceedings of SIGCOMM, San Diego, CA, Aug 2001, pp.161172.

[8] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing" , Tech. Rep. CSD-01-1141, U. C. Berkeley, Apr 2001.