構造化オーバレイにおける反復探索と再帰探索の比較

首藤 一幸[†]　加藤 大志[‡]　門林 雄基[††]　土井 裕介[‡‡]

DHT に代表される構造化オーバレイでの探索様式を比較する。特に、遅延、効率、耐攻撃性の 3 つの評価軸のそれぞれにおいて、再帰探索と反復探索の得失を議論する。また、一つの探索様式で全ての要求を満たすことは困難であり、個々の探索様式がそれぞれに背反となる特性を持っていることを明らかにする。

A Comparative Study of Iterative and Recursive Lookup Styles on Structured Overlays

Kazuyuki Shudo[†]　Daishi Kato[‡]　Youki Kadobayashi[††]　Yusuke Doi[‡‡]

In this paper, we conduct a comparative study of iterative and recursive lookup in structured overlays, such as Distributed Hashtables. We discuss pros and cons of each lookup style with respect to delay, efficiency, and resiliency to attacks. In addition, we clarify one lookup style cannot satisfy all requirements and the lookup styles have contradictory characteristics.

1　はじめに

大規模なインターネット上分散システムでは、ノード数が数万、数百万と増えてもなお性能や可用性、耐故障性を保つために、自律的・非集中的にアプリケーションレベルのネットワークを構成することが不可欠となる。ここで構成されるネットワークは、下位ネットワークの物理的なトポロジとは独立した独自のトポロジを持つため、オーバレイネットワークと呼ばれ、検索やマルチキャストなどの機能を提供する。

現在すでに、同時参加台数が 100 万を超えるコンテンツ共有ネットワーク（FastTrack, eDonkey2K, Gnutella）が存在する。その検索用ネットワークはオーバレイである。

オーバレイネットワークを構築・保守する方式は、大きく、非構造化オーバレイ（unstructured overlay）と構造化オーバレイ（structured overlay）に分類される。非構造化オーバレイはコンテンツ共有アプリケーション Gnutella や Winny の検索用ネットワークに代表され、どのノードを隣接ノードとするかについて方式上の制約がない。一方で、構造化オーバレイでは、隣接ノードが方式によって決まる。構造化オーバレイの応用としては、検索を行う方式、つまり分散ハッシュ表（DHT）[15, 10, 12, 16] や、マルチキャストを行う方式 [4, 3] が提案されている。本稿は、構造化オーバレイを対象とする。

構造化オーバレイは、ルーティングを基本処理に据えるという抽象化 [6] が可能である。この抽象化に従うと、検索やマルチキャストといった応用はルーティング/転送処理を使って実装できる。そこでは、検索対象やマルチキャストグループ、およびオーバレイ上のノードの双方に ID が振られ、前者の ID を担当するノードの探索が行われる。ID は、例えば 160 bit や 128 bit 長の数値であり、目標 ID を担当するノードが転送の到達先となる。

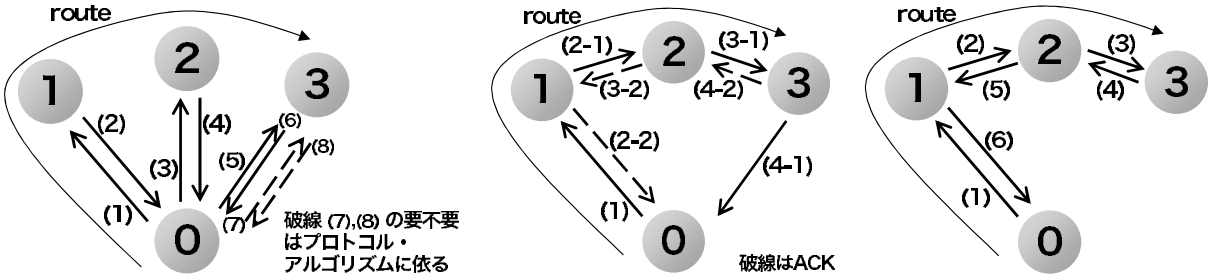
ルーティングのアルゴリズムや経路が同一であっても、経路上の各ノードが相互にどのノードと通信をするかというパターンはいくつか考えられる。このような通信パターンとして、反復探索と再帰探索が知られている。本稿の目的は、この両者を比較し、性質の違いと得失を明らかにすることである。

2　反復/再帰探索

構造化オーバレイ上のルーティング、すなわち担当ノードの探索では、宛先として目標 ID が指定され、探索要求は最終的にその ID を担当するノードに到達する。その後、探索要求元への返答が要るような応用であれば、探索結果として得られた経路や、到達先からの何らかの返答（DHT であれば値）が要求元へ返される。こういった探索は、要求元ノードと到達先ノード

[†]ウタゴエ（株）
Utagoe, Ltd.
[‡]日本電気（株）
NEC Corporation
[††]奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[‡‡]（株）東芝
Toshiba Corporation

情報処理学会 研究報告, 2006-OS-103(-2), pp.9-16,
SWoPP高知2006 (並列/分散/協調処理に関するサマー・ワークショップ),
2006年 7月



a. 反復探索　　b1. 再帰探索 (fast)　　b2. 再帰探索 (slow)

図 1: 探索様式（経路長 $n = 3$ の場合）

表 1: 構造化オーバレイ各実装が採用している探索様式

実装	様式
Bamboo [11]	再帰 (fast)
Chord [15]	反復
MS Pastry	再帰
Overlay Weaver [17][1]	反復, 再帰 (fast)

を含む経路上の各ノードが相互に通信を行うことで、達成される。

ここで、インターネットプロトコルのルーティング/転送であれば、要求元から到達先までパケットが順に転送されていき、転送が完結する。しかし、構造化オーバレイ上の担当ノード探索では、ここで行われる通信のパターン、つまり探索様式に大別して 2 通りあり、それぞれ**反復探索** (iterative lookup)、**再帰探索** (recursive lookup)[11] と呼ばれる[1]。

反復探索では、探索の要求元が経路上の各ノードに対して自ら問い合わせを行い、次ホップを決めていく。もう一方の再帰探索では、インターネットのルーティングと同様に、経路上の各ノードが探索要求を転送していく。図 1 は、両様式の通信パターンを図示したものである。

再帰探索には、要求元への返答の返し方によって 2 通りの通信パターンがある。通信回数を少なくするためには、到達先から要求元へ直接返答を返せばよいが、経路に沿って返答を返すことにも意義があり、どちらの方法も用いられている。本稿では、両者を再帰探索 (fast) および (slow) と呼び分ける。

表 1 に、構造化オーバレイの各実装がどの探索様式を採用しているのかを示す。

3 探索様式の比較

本節では、それぞれの探索様式について、いくつかの性能上・機能上の点に対する定性的な比較を行う。その上で、個々の探索様式におけるトレードオフの関係を示す。

[1] 反復/再帰ルーティング (iterative/recursive routing) と呼ばれることも多いが、routing とは厳密には経路情報の構築を指し、適切な語ではないため、本稿では使用を避ける。

表 2: 経路長 n の場合のメッセージ数とホップ数

	反復	再帰 (fast)	再帰 (slow)
要求元に返答（例:DHT）			
ホップ数	$\mathbf{2n}$ or $\mathbf{2(n+1)}$	$\mathbf{n+1}$	$\mathbf{2n}$
メッセージ数	同上	$2n+1$	同上
返答なし（例:メッセージ配送）			
ホップ数	$\mathbf{2n-1}$	$\mathbf{n}$	$\mathbf{n}$ (同左)
メッセージ数	同上	$2n$	$2n$ (同左)

3.1 探索遅延

探索を完遂するために要する時間を、ここでは探索遅延と呼ぶ。構造化オーバレイの応用には、探索の到達先から要求元に対して返答を返す応用と返さない応用があり、前者では返答が返るまで、後者では到達先にメッセージが到達するまでを探索遅延と呼ぶ。前者の例としては DHT が、後者の例としてはメッセージ配送 (DOLR) [6] が挙げられる。

探索遅延は、探索様式、つまり通信パターンに強く影響される。本節では、探索遅延に対する様式の影響を整理する。なお、遅延およびスループットの改善のためのデザイン方針は、既に Dabek ら [5] により詳細に検討されている。

理想的な環境

まず、理想的な環境における探索遅延を考える。つまり、全ノードの能力が均一で、ノード間の通信遅延が一定であるような環境を想定する。近傍ルーティング（後述）は考慮しない。また、ノードの故障もないものとする。

このような理想的な環境における探索遅延は、オーバーレイ上で交わされるメッセージのホップ数に比例する。ホップ数は、経路長 n の関数となり、探索の到達先から要求元に対する返答があるか否かによって異なる値をとる。

表 2 に、経路長を n とした場合のホップ数を示す。反復探索のホップ数は、プロトコルやアルゴリズムに依存して決まる。詳細は 3.2 節で述べる。要求元への返答がある場合、再帰探索 (fast) のホップ数が $n+1$ と最も小さい。返答が不要である場合は、再帰 (slow) と再帰 (fast) の違いがなくなり、ホップ数も両者等しく n となる。したがって、以下に述べる現実的状況や最適化手法の影響がない限りは、再帰探索 (fast) の遅延が最も小さいと言える。

図 1 b1 の破線は、探索要求メッセージへの ACK（応答）を表している。再帰 (fast) で下位通信層として信頼性のないもの（例:UDP）を用いる場合、実線のメッセージだけでは送信先ノードの故障を検知できない。反復や再帰 (slow) と同等の信頼性を達成するため、つまり、探索を完遂するためには、送信先ノードの故障を検知して再送などの対応をとる必要がある。そこで、破線で表されている ACK が必要となる。

ACK の送受信は、メッセージ数は増やすが、探索遅延には影響を与えない点に注意したい。ACK の送信よりも次ホップへの転送、例えば図 1 b1 の (2-2) よりも (2-1) を優先して送信することで、探索遅延には影響を与えずに ACK を返すことができるためである。ただし、メッセージ数は増え、返答あり、なし、それぞれ $2n+1$、$2n$ となる。

逆に言うと、探索が完遂されない可能性を許容するなら、ACK を省いてメッセージ数を少なくできるという自由度が再帰 (fast) にはある。

続いて、次の順で、現実環境や応用、最適化の遅延への影響を述べる。

- 返答データのサイズ
- 返答のキャッシュ
- ノード故障
- 近傍ルーティング

返答データのサイズ

DHT などの、要求元に返答を返す応用を想定する。返答メッセージのサイズが探索要求メッセージと比較して大きい場合は、転送遅延の差を無視できない。ここで、探索要求の転送に要する時間を t_{req}、返答データの転送に要する時間を t_{rep} とする。それぞれの探索様式における遅延は次の通り。

反復	$(2n-1)\cdot t_{req} + t_{rep}$
	or $(2n+1)\cdot t_{req} + t_{rep}$
再帰 (fast)	$n\cdot t_{req} + t_{rep}$
再帰 (slow)	$n\cdot t_{req} + n\cdot t_{rep}$

再帰 (slow) の合計ホップ数は、反復のホップ数が小さい方と同じ $2n$ である。しかし、再帰 (slow) では、返答データを経路上の全ノードに渡って転送する必要がある。このため、返答データの転送遅延 t_{rep} の影響が経路長 n に比例して大きくなる。

したがって、t_{rep} が t_{req} より充分に大きい場合は、再帰 (slow) はデメリットが大きい。ただし、次に述べるキャッシュの影響を考慮すると、再帰 (slow) にも利点があり、この転送遅延が軽減される場合もある。

返答のキャッシュ

DHT など、返答がある応用では、問い合わせに対する返答を経路上の各ノードがキャッシュするという最適化が考えられる。キャッシュにヒットした場合には、問い合わせをそのノードから先に転送する必要がなくなるので、ホップ数が減り、探索遅延が減る。

キャッシュの効果は再帰 (slow) とそれ以外とで異なる。再帰 (slow) では返答データを経路上の全ノードが転送するため、それら全ノードがキャッシュし得る。このため、経路を共用するすべてのノードがキャッシュの恩恵を受けられる。一方、他の探索様式では、経路上のノードが返答データを中継しない、つまりキャッシュできないため、キャッシュの恩恵を受けられるのは自ノードのみとなる。

したがって、定性的に言えば、多数のクライアントから少数の固定的な目標 ID 群に対して問い合わせが発生するようなアプリケーションにおいて、再帰探索 (slow) とキャッシュの組み合わせが効果を発揮する。このようなアプリケーションの例として BitTorrent を挙げることができる。BitTorrent クライアントである BitTorrent [2] および Azureus [3] は、DHT を用いることでトラッカーを使わずにファイル片の位置管理を行うことができる。

ノード故障

メッセージ送信先ノードが故障していた場合、別のノード、例えば別の次ホップ候補への送信し直しが必要となり、これは探索遅延を増加させる。故障の検知は、数秒という長い時間のタイムアウトを要する場合があり、探索遅延への影響は小さくない。故障しているノードへのメッセージ送信を防ぐことが、遅延を低く抑えるために重要となる。

[2] http://www.bittorrent.com/trackerless.html
[3] http://www.azureuswiki.com/index.php/DistributedTrackerAndDatabase

故障検知の方針には次の 2 通りがあり、探索様式との組み合わせによって遅延への影響が異なる。

1. 通信の失敗だけをもって、故障を検知する。
2. 能動的に故障検知のための通信を行う。

前者の場合、再帰探索が不利となる要因がある。次ホップへのメッセージ転送時に、転送先ノードの故障を検知したという状況を考える。故障を検知したノードは、次善の次ホップに転送を行う。ここで、転送を受けた次善のノードも故障ノードに対して転送を試みてしまう可能性が高い。この傾向は、探索の終盤に近付くにつれ顕著となると考えられる。ある目標 ID に向かう経路群は収束していくためである。

これが起こると、故障に伴うオーバヘッド（例:タイムアウト）を 1 度の探索が複数回被ることになる。複数回の被害を防ぐためには、例えば故障ノードの一覧表をメッセージとともに転送していくこと[4]が必要となり、これはメッセージサイズの増大、つまり転送コストの増大を招く。

反復探索であれば、故障ノードの一覧表は探索要求元（図 1 のノード 0）だけが保持・管理すればよく、メッセージサイズも増大しない。この表を参照して故障ノードへの送信を避けることも容易である。

逆に、能動的に他ノードの故障検知を行う場合は、反復探索が不利となる要因がある。再帰では、ノードは、経路表に載っている次ホップに対して自らメッセージ転送を行うため、送信先は多くの場合[5] 経路表に載っているノードに限られる。このため、経路表に載っているノードに限って能動的に生存確認を行っておくことで、探索時に故障ノードに対する送信を防げる可能性が高い。

一方、反復探索では、経路表に載っているノード群と通信相手との間に特に相関はない。探索時の通信相手は目標 ID によって様々となるため、通信相手となりそうなノードを予測することも一般にはできない。このため、故障ノードに対する通信を能動的検知で防ぐことが困難である。

近傍ルーティング

ノード間の実ネットワーク上の近接性を考慮したルーティングを近傍ルーティング (proximity routing) [8] という。メッセージの送信先として自らに近いノードを選択することで、通信遅延を小さくするのである。

近傍ルーティングの 3 つの方法のうち、次ホップを選択する際に近接性を考慮する 2 つの方法 PNS と PRS は、反復探索よりも再帰探索で効果を得やすい。再帰ではメッセージの送信先が経路表に載っているノードであることが多いため、経路表に載っているノードに限って近接性を計測・推測・保持しておくことで、効果を得られる。

一方、反復探索でも近傍ルーティングを行うことは可能ではあり、次の 2 つの方法が考えられる。

1. 各ホップは要求元に対して複数の次ホップ候補を返し、要求元が自律判断する
2. 各ホップが、要求元との近接性を考慮して次ホップを決め、要求元に返す

しかし、反復探索では経路表と直接の通信相手の間に相関がなく、探索の目標 ID によって様々な多くの相手と通信を行うことになる。送信・受信ノードの組み合わせの数が多いため、近接性の計測や保持が再帰探索よりも困難となる。

以上述べてきたとおり、一様な理想環境では再帰探索 (fast) の探索遅延が最小となる。しかし、各探索様式の優劣は、異なる返答データのサイズ（応用依存）や、キャッシュ（最適化）の有無、また、ノードの故障（現実環境）を検知する方法などの影響を受ける。近傍ルーティングの効果も探索様式によって異なる。

表 3: メッセージ数

	要求		返答		ACK※
再帰 (fast)	n	+	1	+	n
再帰 (slow)	n	+	n		
反復	n	+	n		
	or $(n+1)$	+	$(n+1)$		

※: 下位通信層が信頼性のない場合（例:UDP）に必要。ACK の送受は遅延には影響を与えない（3.1 節）。

3.2 プロトコル効率

本節では、探索様式のプロトコル効率への影響を整理する。具体的には、メッセージ数およびメッセージサイズを対象とし、それらが少ないか小さければ効率がよいと考える。

特に断らない限り、下位通信層としては信頼性のないもの（例:UDP）を想定する。信頼性のある通信層（例:TCP）は一般に、オーバレイのプロトコルとは別に独自に ACK や NACK を送受信しており、そこまで含めた考察は下位通信層のプロトコルに依存するため、ここでは行わない。

[4] Overlay Weaver [17][1] はこの方法を採っている。

[5] 2 ノード間で経路表が対称とならない探索アルゴリズムでは、返答（図 1 b1 の (4-1)、b2 の (4)〜(6)）の送信先は、送信元ノードの経路表には載っているとは限らない。しかし、返答の送信先ノードは探索要求メッセージを転送してきたノード、つまり直前まで確実に生存していたノードであるため、返答の送信時も生存している可能性が高い。

表 4: メッセージの構成

a. 要求メッセージの構成

	メッセージ識別子	目標 ID	要求元	TTL	問い合わせ内容
サイズ (byte)	～16	～20	※	1～2	応用依存
再帰 (fast)	○‡2	○	○‡1	○	○
再帰 (slow)	‡3	○	‡3	○	○
反復 (途中)	‡3	○	‡3		‡4
反復 (最後)	‡3				○

b. 返答メッセージの構成

	メッセージ識別子	次ホップ候補（複数）	問い合わせ結果
サイズ (byte)	～16	※（複数）	応用依存
再帰 (fast)	○		○
再帰 (slow)	○		○
反復 (途中)	‡3	○‡5	‡4
反復 (最後)	‡3		○

※: 下位通信層でのエンドポイントの識別情報。例えば IP アドレスとポート番号。

メッセージ数

表 3 に、それぞれの探索様式におけるメッセージ数を示す。

反復探索では、メッセージの構成によってホップ数とメッセージ数が変わってくる。要求メッセージに問い合わせ内容（例:DHT のキー）を含めない場合、一般には、探索の到達先が判明した後でその到達先への改めての問い合わせ（図 1 a の (7)）が必要となる。ただし、要求メッセージに常に問い合わせ内容を含めておけば、改めての問い合わせを不要にできる。つまり、メッセージサイズとメッセージ数の間にトレードオフがある。メッセージ数は、前者の場合 $2(n+1)$、後者の場合 $2n$ となる。

ただし Chord の場合、アルゴリズムの性質上、到達先に対して問い合わせる前に当該ノードが到達先であることが判明する。そのため、(途中) の要求メッセージに問い合わせ内容を含めずとも、到達先への 2 度の問い合わせは不要であり、メッセージ数は $2n$ とできる。

要求メッセージの送信先ノードが故障していた場合は、他のノードへの送信し直しが発生する。失敗率を f とすると、要求メッセージの数が $(1+f)$ 倍となる。

いずれの探索様式においても、複数の経路に渡って並行探索を行う場合は、およそ並行数 α 倍のメッセージが必要になるが、経路の重複を検知できるとき (反復では容易) は α 倍より少ないメッセージ数になることもある。

メッセージサイズ

次に、メッセージサイズに関して考察する。メッセージには、探索自体のためのものに加えて、経路表の構築・保守のためのものがある。後者の内容や通信パターンは、反復か再帰かにはよらず探索アルゴリズムによって決まるため、ここでは扱わない。前者について、各探索様式のメッセージの構成を表 4 に示す。

再帰 (fast) では、探索の到達先から要求元に対して直接返答メッセージを返すために、要求メッセージに要求元のエンドポイント識別情報（例:IP アドレスとポート番号）を含めておく必要がある（‡1)。また、返答を受信した要求元が、どの要求に対する返答なのかを判断するために、メッセージに識別子を含めておく必要もある（‡2)。

反復や再帰 (slow) では、こういった識別子や要求元情報をメッセージに含める必要はない（‡3)。ただしこれは下位通信層がメッセージ送信元に返答を返す機能を持つ場合のことであり、そうでなければ含めることが必要である。TCP と UDP では、データの内容に頼ることなく、送信元に対して返答を返すことが可能である。

反復の要求メッセージには 2 種類、探索の到達先を見つけるまでの (途中) タイプと、到達先に問い合わせを送る (最後) タイプがある。一般に、問い合わせは到達先に対して行えば充分であるため、(途中) タイプのメッセージには問い合わせ内容や問い合わせ結果を含める必要はない（‡4)。しかし、含めることに意義がある場合もあり、例えば DHT での複製の作成・発見に活用できる。

反復の大きな特徴は、(途中) タイプの返答として、次ホップ候補を返す必要があることである（‡5)。返された次ホップ候補が故障していた場合に備え、複数の次ホップ候補を返すことが望ましい。

まとめ

メッセージの数とサイズについての各探索様式の性質を以下にまとめる。

再帰 (fast)

× 要求メッセージ（$\times n$）が識別子と要求元を含む。

○ ACK（$\times n$）は小さい。

○ 探索が完遂されない可能性を許容することでメッセージ数を n だけ少なくできる、という自由度がある（3.1 節)。

再帰 (slow)

○ 下位通信層によっては、要求メッセージ（$\times n$）に識別子と要求元を含めずに済む。

× 問い合わせ結果を含む返答メッセージが n 回転送される。

反復

○ 下位通信層によっては、(途中) タイプのメッセージ（$\times 2(n-1)$ or $2n$）に識別子、要求元を含めずに済む。

○ (途中) タイプのメッセージは問い合わせ内容/結果を含まない。

× 返答メッセージ（$\times n$）が次ホップ候補（複数）を含む。

再帰 (fast) では、探索の信頼性を犠牲にすることでメッセージ数を約半分に減らすことができる。再帰 (slow) では、問い合わせ結果が大きい場合に、経路長 n に比例して通信量が増える。反復では問い合わせ内容の送信を 1 度で済ませることが可能であり、問い合わせ内容が非常に大きい場合にメリットがある。

3.3 攻撃への耐性

一般的に、探索様式を工夫するだけでオーバーレイの安全性を確保できるわけではない。オーバーレイで安全性を確保するためには、加入方式、識別子の付与方式、経路の交換方式などにおける改善が必要である。したがって本稿での焦点は、特定の探索様式により安全性が劣化するか否か、という点に絞られる。

探索様式の安全性について議論をすすめるために、まず脅威モデルについて論じる。あらゆる攻撃に対して頑健な探索様式は、これまでに知られていないため、想定する攻撃を数種類に限定し、脅威モデルを想定して段階的に議論をすすめていく必要がある。また一般論として、想定する脅威モデルによって方式の優劣は大きく変わる。

オーバーレイにおける脅威モデルとして、文献 [14, 7, 13] が知られている。またアドホック・ネットワークにおける脅威モデルとしては、文献 [9] が代表的である。本稿だけで脅威モデルを網羅するのは無理なので、ここでは脅威モデルに以下の制約を設ける。

1. **敵対者は少数である**: 敵対者が多数であれば、Sybil Attack[7], Eclipse Attack[13] などの攻撃が可能となり、探索様式にかかわらず脆弱である。つまり、そもそも安全性がないため方式の優劣を論じる余地がない。ID の付与方式や加入方式において何らかの対策が講じられており、少数の敵対者が多数の ID を取得することはないものとする。
2. **敵対者はオーバーレイそのものの破壊を目的としない**: オーバーレイそのものの破壊が目的であれば、少数の敵対者であってもオーバーレイ全体に対して blackhole, routing loop, detour, partition, wormhole などのルーティング攻撃 [9] を無差別的に行うことが可能である。これらの攻撃が無差別的に行われた場合、探索様式にかかわらず脆弱である。
3. **敵対者は選択的な攻撃を行う**: 前項より、オーバーレイそのものの破壊が目的ではないので、必然的に、敵対者は特定組織や地域などを対象とした選択的な攻撃を行うと想定される。

つぎに、敵対者となりうるのは誰で、どのような攻撃が可能か、という観点から整理を試みる。なお文献 [14] では敵対者のモデルとして自分宛てのパケットのみを操作できるとしており、盗聴・改竄は含まれていないが、以下では盗聴・改竄も考慮している。

1. リクエストの送信ノード
   (a) **トラフィック増幅攻撃**: 他ノードを経由して攻撃トラフィックを増幅する。
   (b) **他ノードに対する DoS***: 他ノードのサービス妨害を目的として不要なリクエストを多数発生させる。
2. リクエストの中継ノード
   (a) **トラフィック増幅攻撃**: 複数の隣接ノードに同一リクエストを送信するなどして、トラフィックを増幅する。
   (b) **転送妨害**: リクエストの転送を行わないことにより、サービスを妨害する。
   (c) **トラフィック解析**: リクエストを解析することにより、誰と誰が通信しているかを解析する。
3. リクエストの宛先ノード
   (a) **トラフィック増幅攻撃***: リクエストに対し長大な応答を返すことにより、サービスを妨害する。
   (b) **応答妨害***: データの問い合わせに対するサービスを拒否する。

表 5: 攻撃に対する脆弱性の度合い

	反復	再帰 (fast)	再帰 (slow)
トラフィック増幅攻撃	○ 困難	△ 脆弱	△ 脆弱
転送妨害の発見	△ 容易	× 困難	× 困難
トラフィック解析	× 脆弱	× 脆弱	○ 困難

4. オーバーレイ外部のノード

   (a) **トラフィック解析**: トラフィックを傍受することにより、誰と誰が通信しているかを解析する。

   (b) **中間者攻撃***: ARP spoofing 等により他ノードになりすまし、中間者攻撃を行う。

なお、上記のうち * を記したものは探索様式で解決するものではない。したがたって以下の議論ではトラフィック増幅攻撃、転送妨害、トラフィック解析の３種類の攻撃だけを取り扱う。

反復探索、再帰探索のそれぞれにおける３種の攻撃に対する脆弱性の度合いをまとめると表 5 のようになる。以下ではその理由について述べる。まずトラフィック増幅攻撃であるが、反復探索では始点ノード自身が探索パケットをすべて送ることからトラフィック増幅効果はない。再帰探索では、始点ノードあるいは中継ノードが複数の隣接ノードに対して探索パケットを送り、かつそれぞれの隣接ノードが並行して探索を行った場合に高い増幅効果が得られる。しかしこの攻撃は宛先ノードにおいて検出することができ、また経由ノードにおいても、シーケンス番号などの簡単な仕組みで増幅効果を抑制することができる。

転送妨害は、反復探索の場合には簡単に検知できる。なおここで転送妨害として想定しているのは grayhole のみである。他の妨害攻撃 (routing loop, detour, partition, wormhole) については反復探索であっても始点ノードだけで検知するのは難しい。なお、再帰探索の場合には grayhole を検知することも難しい。

つぎにトラフィック解析について述べる。反復探索ならびに再帰探索 (fast) の場合、すべてのノードとの直接通信が発生するため、ノード ID から IP アドレスへの対応づけが容易である。IP アドレスから組織や地域を特定するのは容易であるので、特定組織や地域を対象とした攻撃が可能となってしまう。再帰探索 (slow) の場合、O(1) の隣接ノードのみと通信すれば良いため、ノード ID から IP アドレスへの対応づけが困難である。つまりアドレスの漏洩が起きにくい。このため特定組織や地域を対象とした攻撃は困難だと考えられる。もちろん、隣接ノード数が O(1) でないアルゴリズム (Accordion 等) についてはこの限りではない。また、オーバーレイへの加入・離脱を繰り返せば、特定組織や地域に属する隣接ノードを探索することも可能となるが、この問題は加入・離脱に対し何らかのペナルティを与えることによって軽減できる。

少数の敵対者でかつ、最もやっかいなのは blackhole, routing loop, detour, partition, wormhole などのルーティング攻撃を特定目標に対して行う場合である。オーバーレイに加入している敵対者による選択的攻撃については、複数の Overlay Routing アルゴリズムを並行して利用する Secure routing[2] によって無力化することができる。また、オーバーレイに加入していない敵対者による選択的攻撃は、反復探索の場合には容易であるが、再帰探索の場合には困難であるということに注目する必要がある。

以上の議論から、オーバーレイ上に Secure routing 方式を採用することを前提とした場合、オーバーレイでは再帰探索 (slow) を用いるべきだと言える。トラフィック解析が容易であれば、特定組織や地域に属するノードを対象とした、さまざまな選択的攻撃が可能となるからである。また、そのような選択的攻撃はオーバーレイに加入できない敵対者であっても実行できる。いっぽう、再帰探索ではトラフィック増幅攻撃が問題となるが、これはシーケンス番号などの簡単な仕組みで検知し、抑制することができる。

3.4 トレードオフの関係

これまで見てきたように探索様式の違いによって優劣があるが、それは着目点によって異なる。以下では、探索様式毎にその優劣をまとめる。

再帰探索 (fast) は応答性に優れる。難点は、セキュリティ、すなわち、アドレス漏洩にあり、それらに対処することは不可能ではないものの簡単ではない。よって、実験環境などの閉じた環境、もしくは、別の方法によってセキュリティが既に保証された環境において、再帰探索 (fast) を採用することは妥当である。

再帰探索 (slow) は、アドレス漏洩が起きづらく、他の様式と大きく区別できる。しかし、応答性には劣る。そのため、DHT にはキャッシュなどの最適化手法の導入が望まれる。悪意のあるノードが排除できないネットワークで運用するなど、アドレス漏洩が致命的となる場合は、再帰探索 (slow) の採用が必須となる。

反復探索は、要求元が探索の手順をすべて制御するため、ルーティング攻撃への耐性が高い。また、並行探索の実装が比較的容易であると考えられる。しかし、

表 6: 様式毎の得失の関係

	遅延	効率	安全
再帰 (fast)	○	○	×
再帰 (slow)	×	△	○
反復	×	△	△

応答性には劣る。そのため、DHT にはキャッシュの導入が望まれるが、その効果は再帰 (slow) より低いものとなる。応答性を向上させるためには、近傍ルーティングという手法もあるが、その効果は再帰 (slow) および再帰 (fast) より低いものとなる。

以上を表 6 にまとめたように、求める性質によって探索様式の選択は分かれるものと考えられる。

4 まとめ

本稿では、構造化オーバーレイにおける探索様式を反復探索・再帰探索 (fast)・再帰探索 (slow) の 3 つに分類した上で、いくつかの技術的な評価軸からそれぞれの探索様式の得失を論じた。評価軸には代表的なものとして、探索遅延、プロトコル効率、実装容易性、耐攻撃性を選択した。

議論の結果、次のことが明らかになった。それぞれの探索様式に優れた点と劣る点があり、単一の探索様式ですべての要件を満たすことは困難である。例えば、性能面を優先するのであれば、一般的には再帰探索 (fast) が最適である。ただし、DHT のデータ分布や問い合わせ分布などによっては、キャッシュの効果などから再帰探索 (slow) が最適となる可能性も存在する。また、遅延の低減に効果のある並行探索は、反復探索に対する実装が最も容易であると考えられる。一方、セキュリティ面では再帰探索 (fast) や反復探索では、特定の ID に対応するノードが一般に知られてしまい、特定の ID に対する攻撃が容易になる、という問題がある。ただし、再帰探索では悪意を持つノードによる経路に対する操作が検出困難となる。そして、ルーティングにおける透明性確保と、特定の ID に対応するノードの隠蔽は背反する問題である。

アプリケーション開発者は、このような背反する特性を把握しつつ、アプリケーションの要件に応じて適切な探索様式を選択すべきである。本稿がそのためのガイドラインの一つとなり、構造化オーバーレイ関連ソフトウェア開発の一助となれば幸いである。

謝辞

本研究は WIDE プロジェクトの IDEON ワーキンググループで行われました。共同研究の場を提供してくださった WIDE プロジェクト、議論に参加してくださった IDEON 諸氏に感謝致します。

参考文献

[1] Overlay Weaver: An overlay construction toolkit. http://overlayweaver.sourceforge.net/.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proc. OSDI*, 2002.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in a cooperative environment. In *Proc. SOSP'03*, 2003.

[4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE JSAC*, Vol. 20, No. 8, 2002.

[5] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris. Designing a DHT for low latency and high throughput. In *Proc. NSDI*, 2004.

[6] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *Proc. IPTPS'03*, 2003.

[7] J. R. Douceur. The sybil attack. In *Proc. IPTPS*, 2002.

[8] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proc. ACM SIGCOMM'03*, 2003,.

[9] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proc. MobiCom 2002*, 2002.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. ACM SIGCOMM 2001*, pp. 161–172, 2001.

[11] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proc. USENIX '04*, 2004.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. Middleware 2001*, 2001.

[13] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *Proc. ACM SIGOPS European Workshop*, 2004.

[14] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. IPTPS*, 2002.

[15] I Stoica, R Morris, D Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *Proc. ACM SIGCOMM 2001*, pp. 149–160, 2001.

[16] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *Journal on selected area in communications*, Vol. 22, No. 1, pp. 41–53, 2004.

[17] 首藤一幸, 田中良夫, 関口智嗣. オーバレイ構築ツールキット Overlay Weaver. 情報処理学会論文誌: コンピューティングシステム, Vol. 47, No. ACS 15, 2006（予定）.