第 XXIV 部

Integrated Distributed Environment with Overlay Network

第 24 部
Integrated Distributed Environment with Overlay Network

第 1 章　IDEON ワーキンググループの掲げる目標と現状

IDEON（Integrated Distributed Environment with Overlay Network）ワーキンググループは、オーバーレイネットワーク（ON）という手段を用いて統合分散環境（IDE）の実現にアプローチする有志が集うワーキンググループである。自由で創造的なランデブー、ロケーションおよびルーティングを実現するインフラストラクチャとしてのオーバレイネットワークの研究開発・運用、およびその上での生活（研究者自身の日々の情報のやりとりをオーバーレイ上に移すことによる技術の洗練化）などを目的とする。

具体的には、以下のようなトピックについて議論しつつ、開発を進めている。

- アプリケーション層マルチキャスト
- 運用可能な分散ハッシュテーブル
- 分散システムにおける自律的な信用の扱い

2005 年 10 月現在の開発プロジェクトには次のものがある。

- wija（コミュニケーションの可能性を追求する実験場としての、拡張可能な Jabber/XMPP クライアント）
- libcookai（Pastry を用いた分散ゲーム向けミドルウェア）
- eigentrust 改（トラスト管理／ SNS への実装）
- mchord (Chord の実装)
- PuchiChord（研究プラットフォームとしての Chord の実装）

以下、IDEON メンバが中心となって行った調査および研究開発事例の現状を報告する。また、活動記録として、新規研究計画や論文発表の概要、会議開催記録なども報告する。

第 2 章　オーバーレイ・ネットワーク調査報告

2.1 はじめに

本節ではオーバーレイ・ネットワークについて解説する。一般的に、あるネットワークの上にかぶせる形で構築されるネットワークをオーバーレイ・ネットワーク（Overlay Networks）とよぶ。この呼び名はしばしば、アンダーレイ（Underlay）と対照される。アンダーレイとは、OSI7 階層モデルにおけるネットワーク層あるいはトランスポート層を指すことが多い。これに対し、本章で扱うオーバーレイ・ネットワークはアプリケーション層で実現されているものが多い。

オーバーレイ・ネットワーク研究が大きく盛り上がるきっかけは NSF（米国立科学財団）の Project IRIS であった。IRIS（Infrastructure for Resilient Internet Systems）では同プロジェクトに先駆けて登場した P2P アプリケーションの可能性と、9/11 テロ攻撃によって表面化したインフラストラクチャの脆弱性問題という 2 点から体系的研究としての可能性を見出したと推察できる。

当初、研究の主眼は単一障害点を回避することにあった [8]。インターネットアーキテクチャ全体を俯瞰した場合、データリンク層やネットワーク層においては単一障害点を回避するための機能が豊富に用意されているが、アプリケーション層では依然としてサーバ集中型モデルにもとづいてアプリケーションが構築されているため、サーバ近傍において障害が発生した場合、サービスの可用性が著しく損われてしまう。

このようなアプリケーション層における可用性向上という命題に対しては、サーバを分散して構成するという解が有用である。これを裏付ける理論研究、方式研究、計測研究、アーキテクチャ研究、実証研究などが今日までに幅広く行われている。

なおサービスを分散した計算機群によって構成す

ることにより、可用性のみならず、規模拡張性やスループットを向上させることも可能である。これらの側面からの研究も行われている。

また、マルチホーミング環境においてオーバーレイを用いることでよりよい性能が得られるといった、性能改善に関する論文も数多く発表されているが、これらはアーキテクチャの根幹にかかわる研究ではないため本章では割愛する。

これまでの研究で、オーバーレイ・ネットワークの性質として取り扱われているものは、可用性、規模拡張性のほか、セキュリティ、耐故障性、自己組織化である。このほか公平性、多様性、自発性、相互接続などさまざまな側面からのアプローチが可能であり、今後も研究の余地は数多く残されている。

本章ではトップダウンでオーバーレイ・ネットワークを解説する。まず第 2.2 節ではオーバーレイ・ネットワークの応用研究を紹介する。つぎに第 2.3 節ではオーバーレイ・ネットワークのサービスモデルについて述べる。第 2.4 節ではオーバーレイ・ネットワークの構造として、方式研究を俯瞰するうえで有用な構造と、ソフトウェア実装をすすめるうえで有用な構造について述べる。第 2.5 節ではオーバーレイ・ネットワークに応用されてきた理論を挙げ、このような理論研究・方式研究とシステム研究の融合領域における “cross-fertilization” の重要性について述べる。第 2.6 節ではオーバーレイ研究において利用可能なツールを紹介する。

2.2 オーバーレイの応用研究

前節で述べたように、オーバーレイ・ネットワークはサーバを分散して構成するための手段として用いられる。本稿では、クライアント機能だけでなくサーバ機能が分散されるということのみに焦点をあてる。P2P という文脈ではクライアントがサーバを兼ねるとか、クライアントとサーバの区別がないといった点が強調されることがあるが、そのような視座は他稿にゆずることとし、ここではサーバを分散した応用研究についてみていきたい。

Shark[10] は規模拡張性を有する分散ファイルシステムである。これは SFS（Self-certifying File System）[174] をもとに AFS のような大きなキャッシュを持たせたもので、さらにファイルサーバにかかる負荷を下げるため、オーバーレイの一方式である Coral[90] を用いてクライアント間での協調キャッシュを行っている。コードなどは `http://www.scs.cs.nyu.edu/shark/` から入手できる。

Glacier[101] では、大規模なビザンティン障害が起きた場合でもデータが失われない分散ストレージを提案している。データの複製アルゴリズムとしては erasure code を使ったもので目新しいものではないが、オーバヘッド 9.6 倍で、60%のノードに同時障害が起きた場合でも可用性が 99.9999%だという。ここでオーバヘッド 9.6 倍とは、1 T の実データを分散ストレージ上に保持するために 9.6 T の容量が必要であることを指す。提案システムをオーバーレイの一方式である Pastry[236] を用いて実装し、serverless e-mail system（ePOST）に適用し実際に使っているという。評価では、設定誤りがあるとトラフィックが急増するという問題点も報告されていた。コードなどは `www.epostmail.org` から入手できる。

上記では分散ストレージを中心として応用研究を紹介したが、このほかにもコンテンツ配信網の基盤としてオーバーレイを用いた CoDEEN[296] や DNS サーバの実装基盤としてオーバーレイを用いた CoDoNS[222] などがある。CoDoNS では 6 時間連続稼働させてキャッシュヒット率を上げると従来の DNS サーバより短いレイテンシで検索できるという結果が報告されている。単純に考えると、オーバーレイをつかうとアンダーレイだけの場合と比べて性能が劣るように思えるが、実際にはそうではないのである。

日本発の研究成果としては、RFID におけるタグから物品情報を検索するための ONS（Object Naming Service）の基盤としてオーバーレイを用いた事例 [51] や、ゲームサーバをネットワーク上に分散するためにオーバーレイを用いた事例 [120]、移動透過保証プロトコル LIN6 における移動ノードの位置情報を分散管理するためにオーバーレイを用いた事例 [324] などが報告されている。

2.3 サービスモデル

オーバーレイ・ネットワークのサービスモデルとしては、ストレージ、オブジェクト間のメッセージング、およびエニキャスト、マルチキャストが代表的である [41]。以下ではそれぞれのサービスモデルを紹介する。

2.3.1 DHT

ハッシュ表を分散して保持することにより分散ストレージが構成できる。つまり任意の (key, value) 対の書き込み、読み出しにあたって適当なハッシュ関数 H を用意し、key のハッシュ $H(k)$ に対し value を対応づけるのである。$H(k)$ 近傍のノードに値を保持させることで、ハッシュ表を分散して保持することができる。これが分散ハッシュテーブル（DHT: Distributed Hash Table）として知られているサービスモデルである。

例として、図 2.1 を見ていただきたい。3 台の計算機 M_a、M_b、M_c に対しストレージを分散することを考える。ここでは $[0, H(M_a)]$ の区間を M_a が、$[H(M_a)+1, H(M_b)]$ を M_b が、$[H(M_b)+1, H(M_c)]$ を M_c が担当している。これに対し、ドキュメント D_a は M_a に、ドキュメント D_b は M_b に格納される。DHT ではハッシュ関数 H と格納規則を共有するため、格納先はおのずと決まる。

なお、オーバーレイ・ネットワークを指して DHT とよぶ文献もあるが、概念構築のすすんだ今日では、DHT はオーバーレイの 1 つのサービスモデルにすぎないことに注意する必要がある。

DHT は key → value の検索を分散環境でおこなうものだとみることもできる。DHT では key が完全一致しなければ値にはアクセスできないが、DHT に対する拡張として、問い合せをブロードキャストすることにより完全一致の制約を取り払ったもの [60] や、範囲を指定した検索を行えるもの [19] などが知られている。

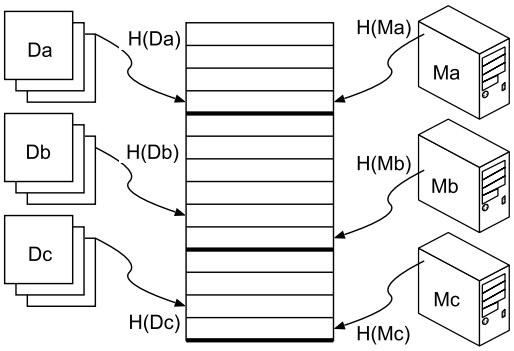


図 2.1. 分散ハッシュテーブル（DHT）の構成例

2.3.2 DOLR

DHT ではアクセスの客体が値であったのに対し、DOLR（Decentralized Object Location and Routing）[236] ではアクセスの客体はオブジェクトである。また、DHT ではアクセスの主体は (key, value) 対へ put、get 操作をおこなうのに対し、DOLR ではオブジェクトの識別子を指定し、任意のメッセージを送る。このため DOLR では、より多様なアプリケーションに対しサービスを提供することができる。

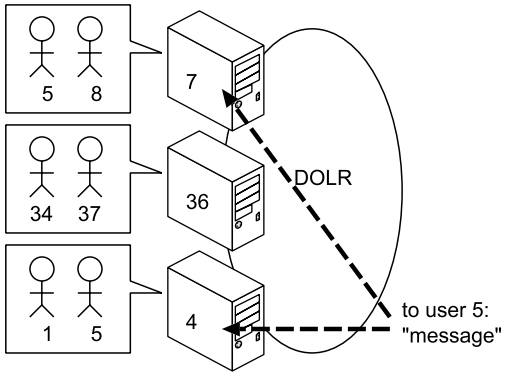


図 2.2. DOLR の利用例

たとえば IM（Instant Messaging）などのユーザ対話型アプリケーションでは利用者の状況変化を相手に知らせる必要があるが、これを DHT サービスモデルで構成した場合、ポーリングが必要となり煩雑であるうえ、通信量が不必要に増えてしまう。あるいは DHT の値としてノードの IP アドレス等を格納することもできるが、この場合、ノード間の通信プロトコルを別途実装しなければならない。DOLR ではそのようなアプリケーションをごく自然に実装することができる。

また DOLR では同じ識別子をもつオブジェクトがオーバーレイ中に複数あってもよい。そのような場合、メッセージは当該識別子の近傍の k 個のノードに配送される。ここで k は冗長度である。たとえば図 2.2 では、ユーザ 5 をあらわすオブジェクトへのメッセージをノード 4 と 7 へ、冗長に配送している。

2.3.3 CAST

CAST はエニキャストとマルチキャストを提供する。CAST では、ノードがグループを指定して参加・離脱をおこなう。グループ内のいずれかのノードにメッセージを届けるのがエニキャストで、グループ内のすべてのノードにメッセージを届けるのがマルチキャストである。同様の機能はこれまでネットワーク層において提供されていたが、これをアプリケーション層で実現するというものである。とくに、ア

プリケーション層における選択的な同報通信は ALM（Application-Layer Multicast）としても知られている。ALM はオーバーレイと並行して発展してきた研究領域であるが、近年、オーバーレイを用いた ALM として Scribe[237]、Overcast[140] などが提案されている。

2.4 オーバーレイの構造

ここでは構造というキーワードをもとにオーバーレイ・ネットワークを俯瞰してみたい。

以下では、オーバーレイ・ネットワークの方式研究を 3 つの側面に分けて考える。ここでは 3 つの側面として、オーバーレイへの加入および近接性の維持方式（Rendezvous）、資源探索方式（Location）、経路制御方式（Routing）を用いる。Rendezvous ではノードがオーバーレイに加入するときに踏むべき手続きと、加入後に隣接ノードとの通信を維持するために必要な手続きを規定する。Location では探索する資源を識別子空間に射影する方式と、アプリケーションに対する資源探索のサービスモデルを規定する。Routing では、ある識別子にあててメッセージを送るアルゴリズムを規定する。これらは直交している場合も、密接に絡み合っている場合もある。また論文や実装において、とくにこれらの側面を区別することなく記述される場合も多い。なお、この 3 分類法は以下の議論をすすめるために導入するもので、ひろくコンセンサスが得られているものではないことをここで断っておく。

オーバーレイ・ネットワークの方式は、大きく構造化オーバーレイ（structured overlay）と構造のないオーバーレイ（unstructured overlay）の二種類に分類することができる。これらは、アプリケーション層におけるトポロジ（以降これをトポロジと記す）の有無による分類であって、下位層のトポロジとは関係がない。

構造のないオーバーレイとは、Gnutella に代表される、ノード間のトポロジを規定しないオーバーレイである。Gnutella ではトポロジに関する前提がないため、目的とする資源を探索する際に隣接ノードに対してブロードキャストを行う。このためノード数が一定以上になると規模拡張性が問題となる。また資源とノードを同じ識別子空間に射影する必要がないため、さまざまなサービスモデルが構成できる。その反面、資源探索や経路制御の効率を損うこととなる。このような理由から、構造のないオーバーレイに関する方式研究はあまりみられない。

一方で、構造のないオーバーレイはアプリケーションとして広範囲に利用されることが多いため、ノードの可用性や寿命などに関する計測研究の題材となる。その成果は、構造化オーバーレイにおけるシミュレーション評価のさいに論拠として用いることができる。たとえば文献 [246] では Gnutella におけるホスト数の経時変化、ボトルネック帯域、遅延、ノードの寿命（uptime）、セッションの持続時間、Gnutella ネットワーク全体の耐故障性について調べている。また文献 [139] では BitTorrent におけるホスト数の経時変化、セッションの持続時間、スループットの分布などを調べている。

構造化オーバーレイはノード間のトポロジとして一定の構造を規定することにより資源探索や経路制御の効率を求めたオーバーレイである。別の言い方をすれば、リングなどのトポロジで規定される隣接関係にしたがってノード間でメッセージを転送することにより、目的とする資源を探索する際にブロードキャストを行う必要がなく、また特定のトポロジを前提としたメッセージ転送の効率化が可能となる。

構造化オーバーレイでは、図 2.1 に示したように資源とノードを同じ識別子空間に射影することで、経路制御だけでなく資源探索の効率化をはかることができる。その代償として、資源探索のサービスモデルが制約をうけることになる。

ほとんどのオーバーレイに関する方式研究は構造化オーバーレイに関するものである。構造化オーバーレイは理論的解析やシミュレーションにもとづいて領域計算量や通信量のオーダーが導出されているため、可用性に加えて規模拡張性にも優れるという点が評価されていると考えられる。また、構造のないオーバーレイではワイルドカードなどを含む複雑な問い合せを実現でき、ノードの多様性（heterogeneity）をより効率的にサポートできるといった根拠のない通説が語られることがあったが、今日、構造化オーバーレイではこれらの機能をより効率的に実現できることが分かっている [32]。

2.4.1 構造化オーバーレイ

構造化オーバーレイとしては、ノード間のトポロジとしてリングを採用したもの（Chord[266]、Accordion[166]）、メッシュ（Pastry[236]）、ツリー

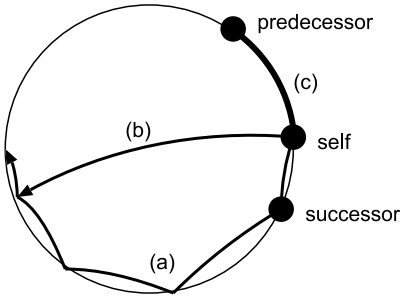


図 2.3. Chord のリングトポロジ

(Kademlia[173])、バタフライ（Viceroy[169]）、de Bruijn グラフを採用したもの（Koorde[150]）などが知られている。IP ネットワークではアドレス割り当てとネットワークトポロジを設定ファイルなどで陽に指定することによって任意のトポロジを構成することができるが、オーバーレイでは識別子そのものには構造がなく、またトポロジにおいても下位層の制約をうけないため、経路制御の方式設計における自由度が高い。

ここでは具体例として、Chord における経路制御の概略について述べる。Chord ではリングトポロジを採用しているため、まず自らの上流と下流の隣接ノード（predecessor、successor）を知ることになる。したがって最も単純には、自分宛てのメッセージ以外はすべて successor ノードに送ればよい。しかしこの方式では、ノード総数 N の場合にメッセージのホップ数が $O(N)$ となってしまう（図 2.3(a)）。改良案として、宛先の識別子と自らの識別子の距離をみてリングの途中まで飛ばせばよい。これによりメッセージのホップ数を $O(\log N)$ に抑えることができる（図 2.3(b)）。

Chord ではリング上の区間 (predecessor, self] を自らが担当する（図 2.3(c)）。このため DHT サービスモデルを用いた場合、$H(k)$ が上記区間に入る場合は value を自らが保持することになる。このような順序関係にもとづく key → value の対応づけをおこなうことにより、どの資源をどのノードが持っているかといったディレクトリ機能を省略することができ、資源探索が効率化される。しかしその反面、資源探索において key を一意に指定する必要があり、ある範囲をもった key や、ある部分文字列に一致する key といったものを表現することはできない。

なお、これらの経路制御方式の詳細については各々の原論文を参照していただきたい。

先に述べたように、オーバーレイの方式研究には 3 つの側面があるため、トポロジ以外の点でも設計上の選択肢が存在する。例えばリング・トポロジを採用したオーバーレイの亜種についてみてみよう。経路制御方式を変更したものとして、近接ノードの選択方式を改良し、ホップ数を抑えつつ耐故障性を高めたもの（Accordion[166]）などが存在する。また、資源探索方式を変更したものとしては、識別子空間として文字列を使うことで資源探索の局所性をもたせたもの [105]、リングを多重化することで局所性をもたせたもの [91]、探索キーとして範囲指定ができるもの [19] などがある。

またオーバーレイへの加入方式と近接性の維持方式について工夫を加えたものもある。Kademlia では、加入したばかりのノードを経由することをなるべく避け、オーバーレイ内での接続時間の長いノードを経由することで、頻繁な加入・離脱が起きる環境における可用性改善を目指している [173]。これは接続時間の短いノードのほうが、接続時間の長いノードとくらべてより離脱しやすいという傾向が構造のないオーバーレイの観測結果 [246] から抽出できたことに由来している。

オーバーレイ方式研究においては、これら 3 側面における選択肢を掛け合わせることで無数に方式ができ上がってしまうという点に注意して研究をすすめたい。たとえば Accordion は Kademlia にみられるような近接ノード選択方式を Chord の経路制御方式にあてはめたものだという見方もできる。

2.4.2 構造化オーバーレイのアーキテクチャ

前項ではオーバーレイ・ネットワーク方式研究における構造についてみてきたが、ここでは、オーバーレイ・ネットワークのソフトウェア実装における構造（コンポーネント・アーキテクチャ）について述べる。

トップダウンでみた場合、アプリケーションはサービスモデルを構成するコンポーネント（DHT、DOLR、CAST の各コンポーネント）との界面を持つことは明らかであるが、各々のサービスモデルコンポーネントと経路制御コンポーネントとの関係は明らかではなかった。

今日、上記サービスモデルは、KBR（Key-Based Routing）という単一の界面のうえに実装でき、また KBR は経路制御アルゴリズムを隠蔽できることが

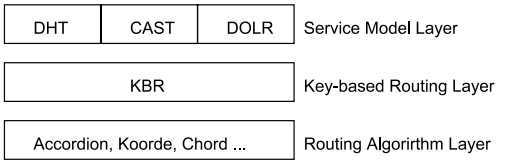


図 2.4.　KBR を収斂層としたコンポーネント・アーキテクチャ

知られている [41]。言い換えれば、KBR は、ネットワーク層における IP のように収斂層（convergence layer）としての役割を果たしているのである（図 2.4）。

KBR では、サービスモデルとの界面として route、forward、deliver という 3 つの基本操作が規定されている。route はメッセージをある識別子にあてて転送する。forward は経由ノードにおける upcall で、宛先、メッセージ、次ホップのノードを変更することができる。deliver はメッセージが宛先に到着したときに呼ばれる upcall である。これらの基本操作を用いて DHT、DOLR、CAST の各サービスモデルを実現することができ、また実現にあたっては経路制御アルゴリズムを自由に選ぶことができる。

2.5 オーバーレイの理論

オーバーレイ・ネットワーク研究では、符号理論や分散アルゴリズム理論だけでなく、さまざまな理論が応用されている。そのうち代表的なものをここで紹介する。

- **グラフ理論**：前節でも述べたとおり、構造化オーバーレイでは特定のトポロジを前提として資源探索や経路制御の効率化をはかる。ここで重要な役割を果たすのがグラフ理論である。たとえばオーバーレイ加入ノードによるメッセージ転送妨害や改ざんといった脅威を考えた場合、単一のノードが無数のノードと隣接関係を維持することは望ましくないとされている [52]。このためトポロジとして、各ノードの in-degree、out-degree を抑制しつつグラフの点連結度（connectivity）や辺連結度（edge-connectivity）を一定以上に保つことが望ましい [167]。このような制約を満たすグラフの一つとして de Bruijn グラフが知られており、Koorde[150] において活用されている。
- **暗号理論**：オーバーレイにおいて、秘匿しておきたいデータを分散記録するといった応用が考えられ、そのような応用では閾値暗号を活用することができる。このほか、認証サービスをオーバーレイ上で分散して構成し、多数ある認証サーバのうち一定数以上の認証サーバによって認証されればアプリケーションからサービスを受けるための認証トークンを閾値暗号によって生成できる、といった方式においても活用されている [148]。
- **線形代数**：オーバーレイは下位層のトポロジの制約を受けないが、その代償として性能が管理できなくなる。結果として、遅延やスループットが heavy tail 分布に従うこととなる。下位層における距離（RTT 等）を考慮してメッセージを転送することで、この問題をある程度軽減できる。このためには距離情報が必要となるが、あらゆる 2 点間の距離を計測することは通信量の指数関数的増大につながり、望ましくない。このため、距離情報の計測と推定を組み合わせることにより通信量を抑える方式が開発されている [34, 216]。たとえば Lighthouse[216] ではインターネットを N 次元空間でモデル化し、$N+1$ ノードとの距離を測り、Gram-Schmidt の直交化法を適用することで任意のノードとの距離を推定することができる。
- **ゲーム理論**：オーバーレイではすべてのノードがサーバの役割を少しずつ分担することになるため、サービスを受けたいが自らは資源を提供する意思のない利用者による「ただ乗り」（free riding）が発生する。この問題に対して、インセンティブ・モデルを構築することにより善良なノードの割合を増やす試みがなされている。Feldman らは囚人のジレンマを一般化した Generalized Prisoner's Dilemma にもとづいてインセンティブ・モデルを構築し、シミュレーションによって有効性を確認している [80]。

以上でみてきたようにオーバーレイでは理論研究の成果をもとに方式研究が行われており、さらにその成果をコンポーネント化したりシステム化するといった実証研究が行われている。このような成果をあげるためには、理論研究・方式研究とシステム研究のそれぞれの専門家が分野を超えて協働することが必要不可欠である。英語では cross-fertilization とよばれる、このような活動様式はオーバーレイのような融合領域では必要不可欠である。

2.6 オーバーレイの研究ツール

研究のためのシミュレーションツールとしては、p2psim が知られている。p2psim は Chord、Accordion、Koorde、Kademlia などのアルゴリズムを実装しており、さまざまなアンダーレイのトポロジモデルのもとで、ノード障害を模擬しながらオーバーレイのホップ数、レイテンシなどを評価することができる。p2psim は `http://pdos.csail.mit.edu/p2psim/` から入手できる。p2psim が対象としているのは経路制御方式のみであり、オーバーレイのその他の側面や、オーバーレイのサービスモデルを取り扱うのは困難だと思われる。この他にもさまざまなシミュレーションツールが公開されているが、トランスポート層以下における ns-2 のようなデファクト標準は決まっていない状態である。

エミュレーションツールとしては、MACEDON、および筆者のうちの 1 人が情報通信研究機構セキュリティ高度化グループにおいて開発している RISSON がある。MACEDON はコードが公開されており、`http://macedon.ucsd.edu/release/` から入手できる。MACEDON の特徴は、オーバーレイネットワークのアルゴリズムを高レベルの言語で簡潔に記述できるという点である [230]。例えば Chord アルゴリズムは 543 行で記述されており、これは p2psim と比べると約 1/9 である。実際にはこれを C++ のコードに変換し、ライブラリとリンクすることで実行ファイルができる。RISSON（Research and Instrumentation Substrate for Secure Overlay Networks）は、セキュアな構造化オーバーレイ [319] の研究・実験基盤として開発がすすめられているフレームワークである。RISSON では、オーバーレイネットワークのアルゴリズムが C++ で簡潔に記述でき、また DHT 以外のサービスモデルを容易に実装できる。また本フレームワークをはじめ、採用している暗号化・署名ライブラリやスレッド・通信ライブラリ、UI フレームワークがプラットフォーム独立であるため、移植性が高いのが特徴である。このほか Bamboo という実装が PlanetLab[18] 上で稼働しており、OpenDHT[226] という名称のもと実験プラットフォームとして利用されている。

構造化オーバーレイに限定せずに研究をすすめるのであれば、開発者向けミドルウェアとして公開されている GNUnet、JXTA が活用できるとおもわれる。これらのミドルウェアは 2.4.2 節で紹介したアーキテクチャとは異なる、独自のアーキテクチャを構築しているようである。

2.7 おわりに

本章では、米国におけるオーバーレイ・ネットワーク研究を中心として応用研究、サービスモデル、アーキテクチャ、理論とのかかわり、ツールなどを紹介した。オーバーレイ研究は理論、方式、システムの各分野で大きな可能性が残されているので、今後もさまざまな専門性をあわせ持つ研究者グループによる大きな研究成果が期待できる。

オーバーレイや P2P というキーワードが目新しく、また融合研究であるためか、我が国ではオーバーレイ関連の研究発表がさまざまな研究会に拡散しており、研究者コミュニティの形成が進んでいないのが残念である。研究者コミュニティが若いということを逆手にとってコミュニティ形成を促進するには、コンテスト、グランドチャレンジなどの触媒を積極的に創っていく必要があると痛感している。もうひとつ痛感したのは研究戦略の重要性である。このような萌芽的な融合領域にプロの研究者を大量動員し、短期間で学問体系を確立するというプログラム・ディレクタの戦略からは多くのものを学ぶことができる。

第 3 章　IDEON におけるオーバーレイ研究開発事例

3.1 WIDE プロジェクトと P2P 技術

WIDE プロジェクト IDEON ワーキンググループは、「統合分散環境（Integrated Distributed Environment）」という WIDE プロジェクトの目標の実現に対し、「オーバレイネットワーク（Overlay Network）」という道具を用いてアプローチする有志が集うワーキンググループである。

IP ネットワークはグローバルな接続性を提供するが、多くのアプリケーションでは、IP ネットワーク上にアプリケーション固有の仮想的なネットワーク（オーバレイネットワーク）を形成し、その上で通信を行うことにより機能を実現している。IRCNet や、SMTP における MTA の網などはその典型的な例である。

IDEON では、このようにアプリケーションごとに独自に形成してきたオーバレイネットワークを整理・系統立てた上で、さまざまなアプリケーションで用いることができる共通の統合分散環境を構築することを最終目的としている。

本章では、その活動の中から分散ゲーム環境構築を目的としたインフラ構築ライブラリである libcookai と P2P 補完貨幣である *i*-WAT について紹介する。

3.2 libcookai：分散ゲーム環境構築を目的としたインフラ構築ライブラリ

ネットワークの広帯域化により、ネットワークを用いたゲームが一般的となった。この中で多人数が同時に同じゲーム世界を共有する Multi-player Online Game（MOG）と呼ばれるオンラインゲームは、ネットワークを用いて遠方の友人と同時に遊ぶことができるなど、ネットワークを十二分に活かしたゲームとして注目を集めている。

MOG のシステムの多くはサーバに中央集権的に処理を集中させるクライアント・サーバ方式で動作している。この方式は、データの管理などのゲーム上の処理を一極集中することができデータの一貫性を取りやすいなどの利点があるが、参加ユーザの数が増えることによってサーバに求められる計算能力やデータ容量、ネットワーク帯域などが増大することにより、クライアント・サーバ型の運用には限界が生じてしまう。そこで、増大するクライアント・ノードへとサーバに一極集中する処理を分散させることができれば、クライアント・ノードの数に応じて増えていたサーバの負荷を、クライアント・ノードの数が増えるにしたがって減らすことができると考えられる。また、サーバにおける処理をクライアント・ノードへと分散させることによって、サーバが存在することによるサーバ停止の問題も同時に解決できると考えられる。

3.2.1 サーバで扱われるデータと Zone

ゲームクライアントはプレイヤーにゲーム世界を演出するためにそのゲーム世界を記述したデータを用いる。したがって、多人数が同時に同じゲーム世界を共有するためには、ゲーム世界を表現するデータを共有する必要がある。また、ある 1 つのクライアント・ノードはゲーム世界の全てを見渡す必要はない。これは、あるクライアント・ノードが必要とするデータは共有されているデータの全てではないことを意味する。以上のことから、クライアント・サーバ型ゲームにおけるサーバはゲーム全体のデータを管理し、個々のクライアント・ノードへとそのクライアント・ノードが必要とするデータを伝える仕事としていると考えることができる。この時、ゲーム全体のデータを Global Status Data（GSD）とよぶ。

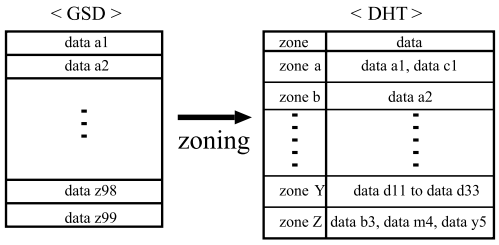


図 3.1.　GSD の zone への分割

GSD は個々のゲームの性質によってさまざまな構造を持つことが考えられる。たとえば、ユーザ・キャラクターの名前、身長などのキャラクターに分類されるもの、ゲームが日本を題材にしたものであれば東京や大阪などの土地に存在する建物のデータなどである。また、クライアント・ノードが必要とするゲーム世界を表現するデータは GSD の一部のみであることから、GSD には構造と局所性があることがわかる。

ここで、GSD を分割し別々のノードが分割された GSD を管理したとしても、クライアント・ノードは自らに必要な GSD の一部が得られることによってゲーム世界を共有したと錯覚することができる。このとき、GSD の構造も考慮して分割することにより、局所性のみの時よりもより柔軟な分割が可能であると考えられる。この分割された GSD を Zone と呼ぶ。

Zone への分割には分散ハッシュテーブル（Distributed Hash Table: DHT）の技術を使用する。DHT は参加しているノードそれぞれがハッシュテーブルを分割して管理し、ハッシュキーに対する検索という形式で目的のハッシュキーのデータを持っているノードを効率的に探し出すことができる手法である。本提案では個々の Zone それぞれに異なるハッシュキーを割り当てることでこれを利用する（図 3.1）。

3.2.2 Zoned Federation Model

ここで、Zoned Federation Model（ZFM）を紹介する。ZFM はネットワークゲームにおけるサーバに

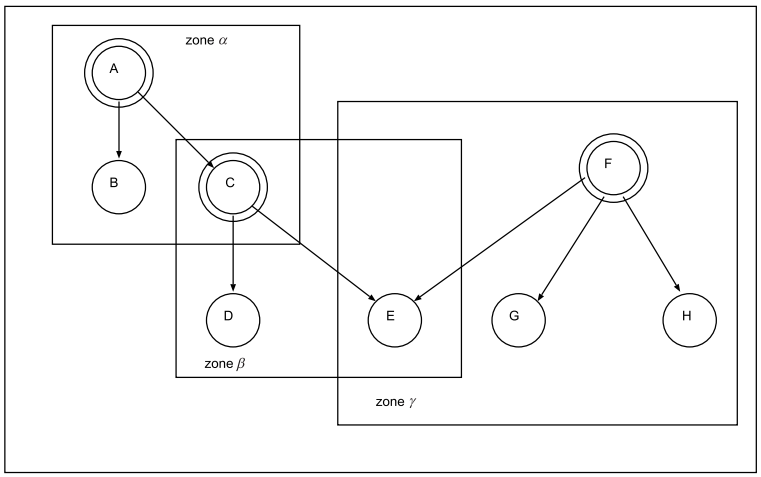


図 3.2. zone の利用形態

要求される機能を満たしたまま、サーバに集中する負荷を Zone というデータの分割単位を用いて効率的にクライアント・ノードに分散させることでサーバに集中する負荷を減少させることを目的とする。

個々の Zone のデータは GSD と比べると十分に少なくできることから、1 つの Zone をクライアント・ノードが処理することが可能となる。したがって、ZFM ではこれら Zone の管理をクライアント・ノードへと分散する。これにより、クライアント・サーバ型でのサーバの役割であるデータ管理をクライアント・ノードへと分散させることが可能となる。

Zone のデータを管理するとき、複数のクライアント・ノードで管理する場合には調停などのオーバヘッドが発生する。これに対して Zone のデータの管理を 1 つのクライアント・ノードで行う方が調停などのオーバヘッドが存在せず、より高速にデータを扱うことができる。ZFM においては応答性に敏感なゲームでの使用を考慮するため、オーバヘッドが少ない方法をとる。したがって Zone のデータを管理するノードは 1 つにすべきである。このデータを管理するノードを Zone owner とよぶ。また、Zone のデータを必要とするノードを Zone member とよぶ。Zone member は Zone owner を経由して Zone のデータを共有することでゲーム世界を共有する。この時、Zone member と Zone owner は Zone のデータを扱うという小規模のクライアント・サーバであると見ることができる。また、応答性に敏感なゲームでの使用を考慮して Zone member と Zone owner の間を直接接続する。これにより、Zone 内部の情報の伝達を高速に保つことができる。

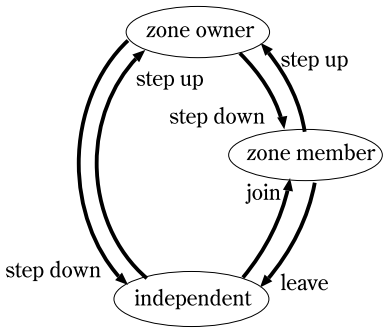


図 3.3. ノードの状態遷移

ZFM の動作を図に表すと図 3.2 のようになる。図 3.2 における四角の枠が Zone をあらわし、それぞれの丸がノードを示している。このとき、二重の丸で示されるノードは各 Zone での Zone owner を示しており、Zone owner から矢印でつながれているノードはその Zone における Zone member である。図 3.2 のように、あるノードは複数の Zone に関与しており、ある Zone では Zone member であるが、別の Zone では Zone owner であるノードというものも存在する。

ZFM においては個々のノードはそれぞれの Zone において Zone owner、Zone member、Zone に関係ないノード（Independent node）、の 3 つの状態をもつ。このそれぞれ 3 つの状態から別の状態へと遷移することを図 3.3 に示すように step_up、step_down、join、leave とよぶ。

上記の状態遷移動作の 4 つに加え、ゲームワールドへの接続（join）、Zone owner によるデータの更新（update）、Zone member によるデータの更新要

表 3.1. ZFM API

関数	概要
initialize()	ゲームワールドへ接続する
step_up(zone)	zone owner へと step up する
join(zone)	zone member となり zone owner からの データ更新情報を受け取る
update (zone, data)	zone のデータを書き換え、 zone member へ更新情報を伝え、 DHT 上のデータを書き換える
commit (zone, data)	zone member から zone owner へ データの変更要求を伝える
release(zone)	zone member から independent node へと 変化する。zone owner との 接続は切られる
step_down (zone)	zone owner から independent node へと 変化する。zone member との 接続は切られる

求（commit）の 3 つを加えた 7 つが、ZFM で規定する関数である（表 3.1）。

個々の Zone における Zone owner や Zone member のリストは DHT 上の Zone 名をハッシュキーとしたノードに格納される。ZFM では DHT 上に記録されるデータにリスト構造を持たせることによってこれを実現している。したがって Zone owner も Zone member もリスト構造を持っており、Zone owner はリストの最初に記録されているものが正式な Zone owner であることになっている。また、Zone におけるゲーム内部データも同じように格納されており、これは最後に記録されているものが正式なデータであることになっている。DHT 上のリストを制御するための動作を読み出し（DHT_get）追加（DHT_add）、削除（DHT_del）の 3 つで表すと、Zone owner となるための step_up の非常に簡単な擬似コードは図 3.4 のようになる。

ZFM において 1 つの Zone に Zone owner や Zone member が 1 つも存在しないということは容易にありうる。この時 Zone のデータを保持するのは DHT 上での Zone 名をハッシュキーとしたノード（データ保持ノード）である。この DHT 上のノードが Zone owner でないかぎりは、そのデータを書き換えることはできない。また、Zone owner はデータの変更

```
function step_up(zone){
  DHT_add(zone, "owner:" + myID);
  foreach data (DHT_get(zone)){
    if(data =~ s/^owner://){
       if(data == myID){
          /* step_up 成功 */
          return true;
       }else{
          DHT_del(zone, "owner:" + myID);
       }
       break;
    }
  }
  /* step_up 失敗 */
  return false;
}
```

図 3.4. step_up() の擬似コード

権限を持っていることから、Zone owner が存在する場合は一番新しい Zone のデータを持っているのは Zone owner であり、DHT 上で示されるデータ保持ノードはそのデータの古いコピーである。そこで、Zone owner は自分がいつ step_down してもよいように DHT 上で示されるデータ保持ノードへとデータの更新を行う。したがって、Zone owner がデータの更新を行う時には Zone member へのデータ更新の通達と、DHT 上で示されるデータ保持ノードへのデータ更新の 2 つを同時に行う。これを行うことで Zone owner や Zone member が存在しなくなってもその Zone のデータは常に最新に保たれる。

3.2.3 libcookai

以上の ZFM を実装したものが libcookai であり、以下の URL から入手できる。

`http://libcookai.sourceforge.net/`

libcookai は C 言語で記述されるライブラリである。表 3.1 をすべて実装し、DHT の実装も含んでいる。

3.3 *i*-WAT：P2P 補完貨幣

3.3.1 P2P と補完貨幣

P2P の考え方は、コンピュータの、普段は活用されていない計算能力、記憶容量、帯域といった資源をネットワーク上に解放し、その力を必要とする他のコンピュータのために使い、有効に活用するという側面を持っている。

P2P システムの参加者は、自らの資源を管理する

経済的な主体であるため、このことが実際に適切に行われるためには、参加者のインセンティブに注目する必要がある。システムの設計者は、たとえば何らかのかたちで保証された価値を代表する交換媒体をデザインしなければならない。そのような媒体がなければ、P2P システムの参加者は、資源を提供するだけ損をしてしまい、誰も資源を提供しなくなってしまう可能性があるためである。そのような媒体のデザインを含め、P2P システムにおいては、インセンティブ整合的な交換のルール、すなわち、利己的な行動が集まった結果、協調的な交換が起こるようなデザインが必要となる。

さらに、P2P システムにおける交換媒体のデザインでは、その媒体そのものが P2P であるように注意しなければならない。さもなければ、P2P であることによる、可塑性や自発性といった、システムを維持する上で有効な性質が失われかねないためである。

日本では地域通貨という呼称が浸透している補完貨幣は、法定通貨を代替したり、補完するメディアとして誕生した、地域で自律的に発行される交換媒体である。我々は、もともとのデザインがとくに自律分散的であり、P2P の性質を備えている補完貨幣である「ワットシステム」に注目し、P2P システムをはじめとするネットワーク上での交換アクティビティに利用すべく電子化した *i*-WAT の開発を進めている。この試みは 2003 年に開始され、2004 年には実用システムとして運用が可能になった。現在は、さまざまな改良を加えると同時に利用の促進を行っている。

ワットシステムは多中心的な貨幣システムで、ワット券とよばれる交換媒体を用いる。ワット券は、手形のようなものであるが、清算のための場所や時間を指定しない。*i*-WAT では、この券を PGP 形式に基づく電子署名を利用することで電子的に表現している。

3.3.2 ワットシステム

i-WAT の詳細を述べる前に、ワットシステムについてもう少し詳しく紹介する。

ワットシステムは、地域通貨の研究会であるゲゼル研究会主宰の森野栄一氏により 2000 年に開発されたシステムである。

ワットシステムでは、以下の手順により取引を行う。

1. 振出取引—ワット券の誕生
   振出人は白紙のワット券に財やサービスの提供元（貸付人）の名前と貸し付けられた負債の額面[1]、日付、および振出人の署名を記載する。これを貸付人に渡すことにより、財やサービスの提供を受ける。
2. 通常取引—ワット券の流通
   ワット券を所持する者は、その券を別の取引に用いることができる。そのためには、券の裏に自分と受取人の名前を書く。受取人はワット券の新たな使用者となることができるが、これを繰り返すことによりワット券は人々の間を巡る。
3. 清算取引—ワット券の帰還
   取引の結果、ワット券が振出人のもとに戻ると清算が起き、そのワット券は無効となる。

ワットシステムには次の利点がある。

自律性　筆記用具、および上記のプロトコルを遵守する意思さえあれば、誰でも自発的にワットシステムに参加できる。

互換性　ワット券は他のどのワット券とも互換である。振出人が信用される範囲において、世界のどこでも利用できる。

拡張性　上記のプロトコルはワットコアとよばれる中核部分を定義したものであり、拡張部として以下を含む条項を追加できる：使用可能な地域、グループ、期間、負債の単位など。

安全性　振出人が清算に失敗すると、貸付人が負債を肩代りする。貸付人が肩代りできない場合は、その次の裏書人が肩代りする。

裏書のチェインが長いほど、その券は信用を獲得していると言える。

3.3.3 *i*-WAT：インターネット・ワット

i-WAT はワットコアをインターネット上のプロトコルとして移植したものである。

i-WAT では、PGP 形式で署名されたメッセージを用いてワット券を電子的に表現しているが、この表現形を *i*-WAT 券とよぶ。*i*-WAT 券のデータは XML で表現され、現在はその転送に XMPP（Extensible Messaging and Presence Protocol）を利用している。

i-WAT 券は振出人によって一意に決められる ID

1　典型的には単位は kWh である。これは自然エネルギーによる発電のコストを代表している。

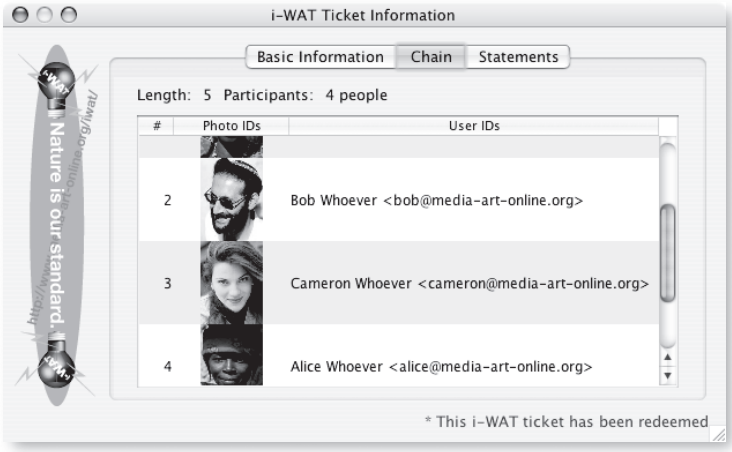
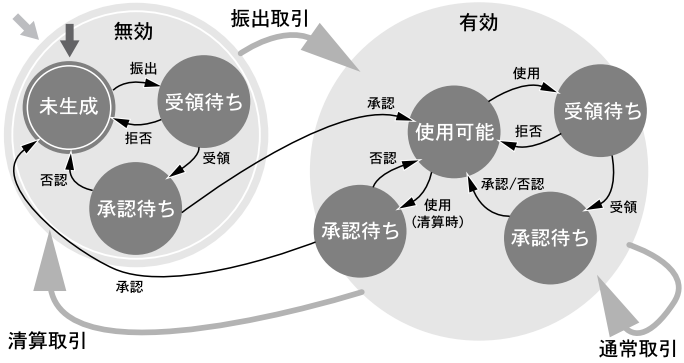


図 3.5.　*i*-WAT 券の視覚表現



* グレイの矢印はワット券の状態遷移を表す
* 黒の矢印は *i*-WAT 券の状態遷移を表す

図 3.6.　ワット／*i*-WAT 券の状態遷移

番号と、負債額、および振出人、使用者、受取人といった参加者の公開鍵ユーザ ID を記載している。裏書は、PGP 署名をネストさせる（署名付きの XML データに更に署名する）ことにより実現している。リファレンス実装では、視覚表現上、PGP のフォト ID を利用している。

図 3.6 はワット券および *i*-WAT 券の状態遷移を表したものである。

ワットコアをデジタルネットワーク上のプロトコルとして実現するにあたり、次の問題を解決する必要があった。

1. 取引は非同期に行われなければならない。
   そのため、受領待ちや承認待ちといった中間状態を用意する必要があった。
2. 二重使用を防止しなければならない。
   振出人が、流通する券が不正に複製されたものではないことを保証する責任を負うという設計にした。*i*-WAT 券が不正に複製されることにより被害を受けるのは振出人であるため、この設計はインセンティブ整合的である。
3. 識別子のコストと可用性の間のトレードオフを考慮しなければならない。
   参加者の識別子として、公開鍵ユーザ ID（現在の PGP の運用ではメールアドレス）を採用した。この識別子は、参加者自身が自律的に付与できるが、信用の輪（web of trust）の形成により、周囲がその正当性を確認していくというしくみを持っている。

3.3.4 *i*-WAT の耐故障性

分散システムの特徴に、メッセージの遅延と欠落を本質的には区別できないという問題がある。そのため、メッセージが再送されて多重に受信側に届く可能性があるが、分散システムのデザインにおいて

は、このことが起きてもシステムとして影響を受けない、冪等性（idempotence）という概念が発達した。

電子的な貨幣システムにおいて問題となる二重使用とその回避も、同一なメッセージが多重に届くことと、そのことへの耐性であることから、本質的には冪等性の問題であると言える。

i-WAT においては、メッセージが欠落するという障害と、貨幣が二重使用されるというセキュリティ上の問題に対して、振出人によるチェックという、統一的な解を与えることができたと考えている。

i-WAT は PGP の利用を基礎としているが、秘密鍵が紛失されるという事故が、実用においても、またテスト中も何度か起きた。*i*-WAT での識別子は鍵の ID（ハッシュ値）ではなく、あくまで公開鍵ユーザ ID であるため、この種の事故に対しては、同一のユーザ ID で鍵ペアを再生成し、信用の輪を再構築することで対処できている。

また、通常取引において、*i*-WAT 券のデータが 3 箇所（振出人、使用者、受取人）に複製されることを利用し、実用において、振出人の持つ *i*-WAT 券のデータベースが紛失した際にも、データを復元できることを確認している。

3.3.5 *i*-WAT の入手方法

i-WAT は、同じく WIDE プロジェクト IDEON ワーキンググループにより開発されている、*wija* と呼ばれる XMPP クライアントにプラグインとして同梱されている。

wija は以下の URL より入手できる。

- `http://www.media-art-online.org/wija/`

3.4 広域統合分散環境の実現に向けて

本稿では、WIDE プロジェクト IDEON ワーキンググループの活動の中から、分散ゲーム環境構築を目的としたインフラ構築ライブラリである libcookai と P2P 補完貨幣である *i*-WAT について紹介した。

これらの技術は、広域統合分散環境を実現するための要素技術として、今後さらに発展させていくことができると期待している。

第 4 章　研究計画：IPv6 のアドレス近傍性を考慮した Chord

4.1 概要

P2P 技術は、たとえば BitTorrent や Skype など、さまざまなサービスに用いられている。その P2P 技術の基盤の 1 つに DHT がある。この DHT にはノード同士の近傍性を考慮していないものが多い。また考慮されているものでも、近傍性を知るために高いコストを強いられている。たとえば Kademlia[173] においてはルーティングの際、複数のノードにメッセージを送信することによって近傍性を考慮する。しかしこの方式ではノード数が増加すると、その分ルーティングに必要なメッセージ数が増加してしまうという欠点がある。さらに join の際にルーティングに必要なノードに ping を送信して RTT を測る方式もあるが、これもルーティング自体には必要ない無駄なメッセージとなる。

そこで本章では、DHT のルーティングの近傍性に着目し、近傍性を測る際も無駄なメッセージのやりとりをできる限り排除した DHT ルーティングを提案する。近傍性を測る基準としてノード間の RTT と IPv6 アドレスのネットワーク部の両方を併用することにより、これを実現する。

4.2 関連研究

ノード間の近傍性を考慮した研究として Kademlia と Global Network Positioning を紹介する。最初に取り上げる Kademlia は hash 値同士の XOR を取ってノードや key 間の距離を求める XOR メトリックを用いた DHT である。次に取り上げる Global Network Positioning はネットワークの距離を推定するために、インターネット空間に座標の考えを取り込む手法である。

4.2.1 Kademlia

概要　Kademlia は hash 関数を用いてすべてのノードと key に 160 bit の ID をつける。ノードはつけられた ID にしたがって二分木を構成し、仮想的なネットワークを形成する。各ノードは二分木の葉として扱われる。

160 bit の ID 同士の距離は ID 同士の XOR の値で捉えられ、ある key に対応する value はその key から距離が最も近い k 個のノードに保存される。この k はシステム全体で定義された固有の値である。

また Kademlia の特徴の 1 つとしてルーティングの際、複数のメッセージを同時に送信し、それを非同期に処理することがあげられる。各ノードは K-bucket とよばれるルーティングテーブルを持ち、最大で $160 \times k$ 個のノード情報を持つ。ルーティングの際は探したい value の hash 値である key に距離が最も近い α 個のノードにメッセージを送信し、最も早いレスポンスを処理する。この方法により α 個のノードのうち、いくつかのノードが自分と離れた場所に存在したとしても、その中の 1 つのノードが自分の近くに存在すれば、素早いレスポンスが期待できる。

問題点　Kademlia はこのように、複数のノードにメッセージを送信することにより近傍性を考慮する。しかしこの方法だとルーティングする度に α 個のメッセージが送信され、α 個のレスポンスが返ってくるが、そのうち実際に利用されるのは 1 つだけである。ノード数が増えてくると、この冗長なメッセージ送信によりネットワークの負荷が増加し、スケーラビリティを欠いてしまう。これらのことから、冗長なメッセージ送信を控え、確実に自分からの距離が近いノードを選択してメッセージ送信するような手法を提案する必要がある。

4.2.2 Global Network Positioning

概要　Global Network Positioning[202]（以下 GNP）は前述の通り、インターネット全体を座標系に見立て、各々のノードの座標を調べることにより、ノード同士が直接通信しなくても、ノード同士の座標がわかればその距離がわかるという手法である。GNP では、各ノードはインターネット上での位置を表す coordinate を持つ。この coordinate からノード間のネットワーク距離が推定される。

また Landmark という複数のノードが存在する。これらは座標空間における基準点のようなもので、あるノードが新しく join するとき、自分の位置情報を取得するために用いられる。新しいノードは join 時に Landmark と通信し、それらの位置情報から自分の位置を特定し、自分の coordinate を作る。

問題点　この手法を DHT に適用すると、いくつかの問題点が考えられる。まず join 時に Landmark と通信をしなければならないが、これは本来ルーティングする際には必要のないメッセージである。Landmark ノードはある種サーバのような役割を果たすので、これが動作しなくなると新たなノードが join する際に、自分の coordinate を作成できないといった問題点が考えられる。さらにこの手法は実際に相手ノードと通信して RTT を測るといったことはせず、お互いの位置情報から相手の位置を推定するだけなので正確な RTT を知ることはできない。これらのことから、Landmark のような特定のノードに頼らず、join 時の無駄なメッセージ送信をせずに済むような手法を提案する必要がある。

4.3 提案方式

本節ではこれらの問題を解決する、IPv6 環境における近傍性を考慮した DHT ルーティングを提案する。加えて本提案方式では、近傍性を考慮していない Chord への実装を前提として話を進める。

4.3.1 提案方式の概要

本提案方式ではノードのネットワーク上での位置を大まかに知るために、IPv6 アドレスのネットワーク部を使用する。IPv6 アドレスはアドレス集約を行っているため、IPv6 アドレスのネットワーク部を見るだけで、そのノードが地理的にどこに存在するのかがある程度判断できる。実際には自分のネットワーク部と相手ノードのネットワーク部を比較して、相手ノードとの距離を推定するのに用いられる。

この性質を利用して本提案方式では、まず ID に変更を加える。オリジナルの Chord は IP アドレスや Port 番号など、各ノードで一意な情報を hash 関数にかけ、ランダム ID を生成する。一方提案方式では、ID 長を $idlen$ とすると IP アドレスや Port 番号からのランダム ID $(idlen/2 - a)$ bit と、IPv6 アドレスのネットワーク部の上位 $(idlen/2 + a)$ bit を繋げた ID を用いる。a はシステム全体で定義された固定の値である。これにより、ID を知るだけでその相手との大まかな距離が判断でき、join の際の余分なメッセージのやりとりを防止できる。

またノード間の距離推定を、IPv6 アドレス情報を用いた方法のみで行うと、結果が大まかになることが予想される。そこで本提案方式ではルーティング

を行う際、送信メッセージに RTT 情報を載せることにより、余計なパケットを送信せずに RTT を計測する。こうすることにより直接メッセージをやりとりしなくても相手ノードとの距離はある程度わかり、実際にルーティングメッセージを送信すると、より正確な距離がわかる。

次にルーティングに関しての変更を述べる。まずオリジナルの Chord は successor、predecessor として 1 つずつ、Finger Table として *idlen* 個のノード情報を保持している。本提案方式ではこれを拡張し、システム全体で定数 k を定義する。そして各ノードは successor として k 個、Finger Table として $idlen \times k$ 個のノード情報を保持するよう拡張する。従来の Chord はルーティングの際、key に対して最適な Finger を計算し、そのノードにメッセージを送信する。しかし本提案方式は定数 k の導入により最適な Finger が k 個に増えるので、この中から近傍性を考慮して最近傍ノードにメッセージを送信することにより、ルーティングにおけるノード間の近傍性を実現する。

4.3.2 ルーティングポリシー

ここではルーティングする際、Finger Table から 1 つの Finger を選択した後、k 個のメッセージ送信ノード候補の中からの最近傍ノードの選択方法を説明する。

k 個のノードから選択をする際は、以下の 3 通りのいずれかの状況である。

- k 個のすべてのノードの RTT を知らない場合
- いくつかのノードの RTT を知っている場合
- すべてのノードの RTT を知っている場合

それぞれの場合についての動作を説明する。

k 個すべてのノードの RTT を知らない場合、それぞれのノードの ID からネットワーク的に一番近いと思われるものを選択する。

いくつかのノードの RTT を知っている場合、確率 p で RTT が一番短いノード、確率 $(1-p)$ で RTT がないノードの中から ID を調べ、ネットワーク的に一番近いと思われるものを選択する。

すべてのノードの RTT を知っている場合、確率 p で RTT が一番短いノード、確率 q で次に RTT が短いノード、確率 r でその次に短いノード、...、そして確率 $(1-p-q-r-\cdots)$ で一番遅いノードを選択する。

こうして選択したノードにメッセージを送信すると再び RTT が得られる。この RTT を元に現在の RTT を調整する。ここでは、$1 > p > q > r > \cdots > 0$ かつ $p+q+r+\cdots \leq 1$ で、それぞれの文字は各ノードが保持する動的な変数である。

4.4 実装、評価

現在この Chord における提案方式を Java で実装中である。評価は 9 台のマシンを使い、3 台をルータ、残りの 6 台で各 20 ノードずつ、計 120 ノードを動かし、オリジナルの Chord とルーティングの早さに関しての評価を行う予定である。

第 5 章　その他活動報告

5.1 研究発表

WOT for WAT: Spinning the Web of Trust for Peer-to-Peer Barter Relationships[240]: P2P 補完通貨は、インターネット上の利己的な参加者同士の間での協働を支援する強力なツールとなり得る。*i*-WAT は、そのような通貨となるべく、現実に補完通貨として利用されているワットシステムに基づいて設計・運用を進めているもので、OpenPGP により署名されたメッセージの交換により、電子的に表現された手形のやりとりを実現している。この論文では、*i*-WAT の信用モデルと、PGP のそれとの関係を明らかにする。そして、*i*-WAT の利用に必要な WOT（Web of Trust）を動的に構築するために、参加者がとるべき行動を提案する。

Reduction Over Time: Easing the Burden of Peer-to-Peer Barter Relationships to Facilitate Mutual Help[243]: この論文では、*i*-WAT の設計の拡張として、ROT（Reduction Over Time; 時間経過に伴う減価）を提案する。減価する *i*-WAT 券の清算を遅延することで、参加者は、振出人の負債を軽減することに貢献でき、また、そのための各々の負担が大きくなるのを避けるために、*i*-WAT 券が高速に流通することが期待できる。このような、減価型の券は、例えば被災地の復興支援など、支援される側の理由が明らかになっているケースで利用できる。支

援される側の負債が貨幣化されることで、支援する側は交換の益を得られるため、これは相互支援のシステムとなっている。

Incentive-Compatibility in a Distributed Autonomous Currency System[241]: この論文では、*i*-WAT の設計が、想定されるモラルハザードに対して、インセンティヴ整合的な耐性を持つという主張を検証する。モラルハザードとは、リスクに対する保護機構が存在することにより、人々が注意義務を怠り、かえってより大きな危険を冒す可能性が高まることを指す。*i*-WAT では、より大きな危険を冒すような行為に対して、他の参加者が各々の利益を追求することによる明確なリスクが存在し、それにより、モラルハザードは抑制されると考えられる。

Multiplication Over Time to Facilitate Peer-to-Peer Barter Relationships[242]: この論文では、*i*-WAT の設計の拡張として、MOT（Multiplication Over Time; 時間経過に伴う増価）を提案する。増価する *i*-WAT 券の清算を遅延することで、参加者は、振出人の負債が増加することによるプレミアムを得ることができる。これを利用することで、振出人は清算の時期をある程度コントロールすることが可能になる。この論文では、ゲーム理論的な分析により、負債が増加するにもかかわらず、振出人には戦略的に債務不履行を選択する理由がないことを明らかにする。

DNS meets DHT: Treating Massive ID Resolution Using DNS Over DHT[51]: この論文では、Distributed Hash Table（DHT）へ DNS リゾルバから透過的に、かつボトルネックを可能なかぎり軽減した形で問い合わせを行う機構を提案する。DHT と DNS のプロトコル変換を各 DHT ノードで行う一方、DNS リゾルバからの問い合わせを各 DHT ノードに負荷分散する方法として動的な名前権限委譲のメカニズムを利用する。

Ecosystem of Naming Systems: Discussions on a Framework to Induce Smart Space Naming Systems Development: この論文（投稿中）では、持続可能なスマートスペースの条件として透過的かつ重層的な名前空間の存在を仮定する。その上で、名前空間を各スマートスペース上のアプリケーション設計者が自由に追加できることを可能にする名前解決システムの枠組みを提案する。提案する枠組みは、単機能なクライアントから多様な Peer-to-Peer 型の名前サーバへ問い合わせを中継するための機構 [51] と、利用者に強く関連づけられた高機能なクライアントが多様な名前サーバに並列に問い合わせを送信するモデルの 2 つからなる。

Advantage of anonymity in mutual-aid on the web: Overcome prejudice against non-conforming marriage style in Japan[209]: この論文では、ウェブコミュニティの運用が実名でなされているもの、筆名で判別可能なもの、全くの匿名でなされているものそれぞれにおけるユーザ間の相互扶助について調査し、情報の信頼性と投稿の障壁という 2 点から評価した。その結果、個人的な情報の提供が必要とされるトピックに関しては匿名コミュニティが効果的であることが明らかになった。

Designing mutual-aid model for RAQ (Rarely Asked Question) in e-government: Practical use of anonymity（投稿中）: この論文では、電子政府の dependability を少数・例外事例の適切な処理をもって評価する視点から、Rarely Asked Question（RAQ）を提案する。政府、市民団体、ウェブコミュニティ、Q&A サイトを調査した結果、RAQ はユーザ同士の相互扶助によって解決され、とくにセンシティブな事例に関しては匿名性の確保が正答率および回答の早さに影響していることが明らかになった。

5.1.1 会議・BOF 等の開催

次のとおり、IDEON の会議（多くの場合 Polycom、Jabber Conference などを利用した多地点遠隔会議）、および合宿 BOF を開催した。

1/24　試験用クラスタ、スマトラ沖地震のような大規模災害における地域通貨

2/25　合宿 BOF 内容、進捗報告

3/23：合宿　セキュアオーバーレイ概要報告、EigenTrust のソーシャルネットワーク上への実装、増減する地域通貨、オンラインゲームに向けたオーバーレイの開発、チェアの変更の提案

3/24：合宿　wija およびスポットライト（802.11b の BSSID から位置を取るミドルウェア）のインストール紹介

4/9　リチャーター、マイルストーン確認

5/12 進捗報告
6/13 リリース予定確認
8/31 学会参加報告、ワークショップ企画検討、Chord シミュレーション
9/7：合宿 進捗報告、オーバーレイ攻撃耐性、オーバーレイとコミュニティと ISP、IPv6 アドレス近傍性とオーバーレイ、サーバレス環境
9/29 ワークショップ企画検討、オーバーレイに「暮らす」ことについて
10/13 ワークショップ企画検討、P2P-CA、研究内容進捗報告
11/15 ワークショップ準備、wija 普及戦略
12/20 進捗報告、Koorde の不明点、データフローオーバーレイ、情報の売買と地域通貨

5.2 まとめと今後の展望

以上のように IDEON の研究活動は学術的な意味でも広範囲にわたる。WIDE プロジェクト内部でもインターネット基盤運用のような領域に比べると比較的自由度が高く多様な研究が存在し、またそのためには外部の研究成果の調査が重要であり、また十分な量の調査を行ってきた。

一方、多様な研究成果をどのように還元するかについては、今後の課題である。現段階では研究成果としてできたソフトウェア・環境を試験的に利用したり、研究内容にともない改良したり、という段階である。今後は、実際にそのソフトウェア・環境を研究者自身が日々利用し、また広めるための検討を行う必要があると感じている。

同時に、さらなる研究の深耕のために、2006 年度は Peer-to-Peer およびオーバーレイネットワークの信頼性および持続可能性を検討する国際ワークショップ（DAS-P2P Workshop[2]）、計算科学とオーバーレイの融合領域などをテーマとした集中形式の会議の開催などを計画している。

2 http://das-p2p.wide.ad.jp/