卒業論文　　2008年度（平成20年度）

# LinkPoint: ネットワーク負荷軽減のために DNSによる近接性を取り入れたP2Pシステム

慶應義塾大学 環境情報学部

氏名：黒宮 佑介

担当教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

高汐 一紀

重近 範行

Rodney D. Van Meter III

植原 啓介

三次 仁

中澤 仁

平成21 年2 月10 日

卒業論文要旨 - 2008 年度 (平成 20 年度)

# LinkPoint: ネットワーク負荷軽減のために DNS による近接性を取り入れた P2P システム

本論文では，既存のインターネットアーキテクチャにおいて Peer-to-Peer（P2P）システムが抱えるトラフィックの問題を解決する手法を述べ，その有効性を検証する．P2P システムは，優れた負荷分散性・耐障害性を備えている反面で，ネットワークの管理・制御が難しいという側面を持つ．特に，下位層トポロジを考慮せずに展開される P2P ネットワークのトポロジは，トラフィックの影響範囲を拡大させており，その集約がインターネット全体の課題となっている．

そのため，本論文では，トラフィックの影響範囲の集約を目的とした，DNS による近接性について研究を行った．本手法では，DNS の管理ドメインについて着目し，管理ドメイン間の距離を**ネットワーク距離**とする．そして，管理ドメインのラベルとして IP アドレスから得られる Fully Qualified Domain Name（FQDN）について，ドメインレベルの最長一致とレーベンシュタイン距離[1]を用いて比較することで，近接性を判定している．

また，FQDN によって近接性を反映できることを示すために，実ネットワークを用いた試験運用と，既存の P2P システムから採取した実測データを用いて実験を行った．実験においては，FQDN 間の相違性をネットワーク距離とした場合を，既存の手法で計測されている Hop Count と比較した．実験の結果より，FQDN 間の相違性をネットワーク距離として用いる手法が，既存の手法と比べて，よりコストが低い手法として有効であることを証明した．

この手法を用いることで，P2P システムの可能性はこれまで以上に広がり，同時にインターネットの発展を促進させると考えられる．さらに，効率的な P2P システムを設計する上で重要な概念である近接性は，限られた資源を融通して使っていくという意味で，インターネットだけではなく，現代の社会のあらゆる部分で必要となる要素である．したがって，このような近接性を考慮する研究を行い，社会に研究成果を還元することは，これからのユビキタス社会やモビリティ社会などの発展の可能性を多角的に分析・模索する上で，重要であると考えている．

キーワード
1．P2P, 2. 近接ノード選択, 3. 局所性, 4．ネットワーク距離, 5. トラフィック制御

慶應義塾大学 環境情報学部

黒宮 佑介

[1] 2 つの文字列の類似度を示す数値．具体的には，ある文字列から目的の文字列を導き出すために，置換・削除・挿入を最低何回行う必要があるかを計算し，文字列の類似度が低いほど大きな値となる．

Abstract of Bachelor's Thesis - Academic Year 2008

# LinkPoint: A P2P System with DNS Proximity for Saving Network Traffic

This thesis explains a method for solving traffic problems on Peer-to-Peer (P2P) systems in the current Internet architecture, and verifies the effectiveness of the method. P2P is difficult to be managed and controlled over a network, while its properties such as load balancing and fault tolerance are considered as their strengths. In particular, P2P network topologies deployed without considering the lower-layer topologies affect a wide range of network traffics. Therefore, aggregation of those traffics is a problem to solve for the entire Internet.

In this thesis, DNS proximity is studied for aggregating the affected areas of those network traffics. In the research, *network distance* is defined as a distance between managed DNS domains. The proximity is determined by observing domain level longest match comparisons and Levenshtein distances[2] on Fully Qualified Domain Names (FQDN) obtained from node IP addresses.

In order to prove that the proximity can be measured by FQDNs, the author experimented on operation on the real network, as well as calculation of network distances on sample data collected from actual P2P networks. In the experiments, a network distance obtained from FQDNs is compared to the Hop-Count that are used for measuring traffic information on existing methods. From the experiments, it was proven that the method of utilizing differences among FQDNs as network distances is more efficient because of its low cost.

With this method, the author hopes, evolution of the Internet would be promoted as well as possibilities of P2P systems. Moreover, proximity is not only a critical concept for designing an efficient P2P system, but also a necessary concept for various aspects of the society in terms of accommodating limited amount of resources. The author envisions that this research on proximity will offer an important support to the society for analyzing and developing possibilities of ubiquitous and mobility societies in the near future.

Keywords :
1. P2P, 2. Neighbor Selection, 3. Locality, 4. Network Distance, 5. Traffic Engineering

Keio University, Faculty of Environment and Information Studies
Yusuke Kuromiya

[2]Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. (Wikipedia)

# 目 次

# 図 目 次

# 表 目 次

# 第1章　序論

## 1.1　はじめに

インターネットの利用形態が多様化が進み，扱われるデータも，従来のような文書や画像だけでなく，音声や動画などのより多くのデータを必要とするものに変わってきている [1]．代表的なものに，ユーザが発信するコンテンツがあり，以前は文章やイラストなどが中心だったが，最近では音楽や映像が中心となり，その規模も拡大してきている [2]．例えば，YouTube[3] やニコニコ動画 [4] などでは，ユーザが自ら作成した動画を気軽に配信できる環境があり，毎日様々な種類の動画がアップロードされている．そして，高画質化や長時間化だけでなく，コンテンツ自体の品質が向上することで，インターネットの利用機会が増加し，インターネットに対する要求はますます高まっている．

また，ネットワークアーキテクチャも要求に対応するため，複雑化している．大容量のデータを転送するために大量のトラフィックを発生させる可能性のあるコンテンツについては，Contents Delivery Network（CDN）が利用されている．CDN では，コンテンツを配信するためのサーバを世界中に分散して設置することで，コンテンツの要求が行われたネットワークにとって，物理的・ネットワーク的に近いサーバを選び，そこからデータを転送する手法を用いている．このような手法を用いることで，コンテンツを配信する際に，インターネットにおけるトラフィックの影響範囲を集約し，ネットワークへの負荷を抑えることができる．

しかし，分散システムとしての CDN が対象とするのは，国やサービスプロバイダであり，ユーザ単位でサービスを構成することは難しい．特に，CDN の運用には大量のコストが必要であり，ユーザ自身が作成したデータやコンテンツなどに代表される，個人単位での情報発信や二次利用を想定した大容量のデータを配布・転送したいと思った場合に，CDN を利用することは困難である．

新たな大容量のコンテンツの配信方法として，Peer-to-Peer（P2P）ネットワークを利用したネットワークシステム（P2P システム）が展開・利用されるようになってきている．P2P ネットワークとは，平面的に接続性を提供しているインターネット上に，自由に仮想網を構築することによって成立するオーバーレイネットワークの 1 つで，下位層トポロジとは別に，独自の論理トポロジによって構成されている．

P2P システムの特徴として，互いの計算機が対等な役割を担っていることが挙げられる．P2P ネットワークでは，サービスにおいて，管理・制御などの中心的な役割を担う計算機は存在しないため，各計算機が互いの役割を自律分散的に決定し，協調し合うことでネットワークが成立する．すべての処理を各計算機が自律的に行うことから，余剰資源が活用できるだけでなく，障害点の分散なども自動的に行われるため，P2P ネットワーク全

体が 1 つのシステムとして機能することが可能である．そのため，クライアント・サーバモデルと比べて高い耐障害性を備えているだけでなく，大容量のデータを高速かつ低コストで転送することが可能である．

一方で，P2P システムには独自に構成された論理トポロジの管理・制御が難しいという問題があり，データ転送のためのトラフィックがインターネットバックボーンに必要以上の負荷をかけることがある [5]．具体的には，P2P ネットワークのトポロジがインターネット上において冗長な経路を含んでいたり，不要な通信によりトラフィックの影響範囲が必要以上に拡大している場合がある．P2P システムが，既存のインターネットアーキテクチャにとって必要以上の負荷をかける構成となってしまうのは，P2P ネットワークが構築する仮想網が下位層トポロジを考慮しないためである．特に，P2P システムはコンテンツ配信などの大量のトラフィックを発生させる用途に利用されることが多いため，インターネットバックボーンへの負荷も大きく，その管理・制御は単一の組織や国だけではなく，インターネット全体の課題となっている．

## 1.2 本研究の目的

本研究では，P2P システムが抱える問題を多角的な視点から分析・追求することで，P2P システムの利点を最大化し，従来のネットワークアーキテクチャでは困難だった新しいサービスを提案できる社会の実現を目指す．

また，本論文では特に，P2P システムが発生させるトラフィックの管理・制御に着目する．以下に 2 つの理由を示す．

1. P2P システム自体のパフォーマンスの向上
   P2P システムが発生させるトラフィックが，P2P システム自体のパフォーマンスに影響を与えている．具体的には，同一組織内に閉じた通信でデータを取得可能な場合にも，P2P ネットワーク上のトポロジによって別の組織を経由した経路が選択されることがある．その際，トラフィックの影響範囲が拡大し，必要以上のネットワーク資源を消費することが少なくない．そのため，別の計算機がデータを要求した場合に，ネットワーク資源の不足によって要求に応えられないなど，P2P システム自体のデータ転送におけるパフォーマンスが低下する可能性がある．

2. インターネット全体のパフォーマンスの向上
   P2P システムが発生させるトラフィックが，インターネット全体のパフォーマンスに影響を与えている．1.1 節で述べたように，P2P システムはコンテンツ配信などの大量のデータを扱う用途に利用されることが多く，余分なトラフィックが発生した場合にも，そのトラフィックは非常に大きなものとなる．このようなトラフィックによりインターネットバックボーンが圧迫されることで，一般のインターネットユーザにも，ネットワークからの要求が返ってこない，応答が遅れるなどのパフォーマンスの低下が起こる可能性がある．

したがって，P2P ネットワークのトポロジを，既存のインターネットアーキテクチャにとって負荷の掛かりにくい構成にし，トラフィックの影響範囲を集約・縮小することは，P2P システムのユーザだけでなく，インターネットユーザにとっても，利便性向上や可能性を発展させるために必要である．

## 1.3 本研究により期待される成果

本研究により，P2P システムは今後も増え続ける大容量コンテンツや新しいサービスなどの様々な要求に応えることが可能となり，負荷や障害，システムの規模に対して，より柔軟性を備えたネットワークアーキテクチャとなることが期待される．また，本論文で対象とする P2P システムが発生させるトラフィックの管理・制御は，P2P システムの利活用を促し，誰もがその恩恵を受けることができるようになるために，非常に重要な課題である．特に，P2P システムという観点では，クライアントサイドだけでなく，インターネットクラウドなどで利用されているサーバサイドでの活用も期待される．具体的には，クライアント・サーバモデルにおけるサーバについて，P2P システムを活用することで，サービスの展開がこれまで以上にスムーズに行えるだけでなく，規模性や耐障害性を兼ね備えることが可能になると考えられる．

さらに，本研究の対象である P2P システムは，近接性の追求という観点でも，今後の社会に貢献できると考えている．インターネットは，コミュニケーションのコストを引き下げることにより，物事を効率化したが，一方で，資源の浪費を助長するような使い方がなされていることも事実である [6, 7]．また，限られた資源を融通して使っていくことは，インターネットなどの情報通信分野だけでなく，現代社会のあらゆる分野・部分で必要となる要素である．P2P システムによる近接性の追求は，それらの問題を解く 1 つの糸口であり，これからのユビキタス社会やモビリティ社会などの発展の可能性を多角的に分析・模索する上で，必要不可欠な要素になると考えている．

## 1.4 本論文の構成

本論文は 8 章から構成される．第 2 章では，既存のアーキテクチャの背景を述べる．第 3 章では，P2P システムの概要と従来の P2P システムにおける課題について述べる．第 4 章では，第 3 章で述べた課題に取り組む関連研究について述べる．第 5 章では，第 3 章で述べた課題の要因を導き出し，その解決手法と設計について述べる．第 6 章では，開発した実装について述べる．第 7 章では，本提案手法と実装を定性的・定量的な側面から評価する．最後に第 8 章で本論文の結論と，今後の方針を述べる．

# 第2章　背景

P2P システムは，既存のインターネットの応用方法とは異なるネットワーク構築手法によって構成されている．本章では，インターネットの応用方法の一般的なモデルであるクライアント・サーバモデルにおける分散システムと，P2P システムの一般的なモデルである P2P モデルにおける分散システムを比較し，その性質や特徴について述べる．

## 2.1　クライアント・サーバモデル

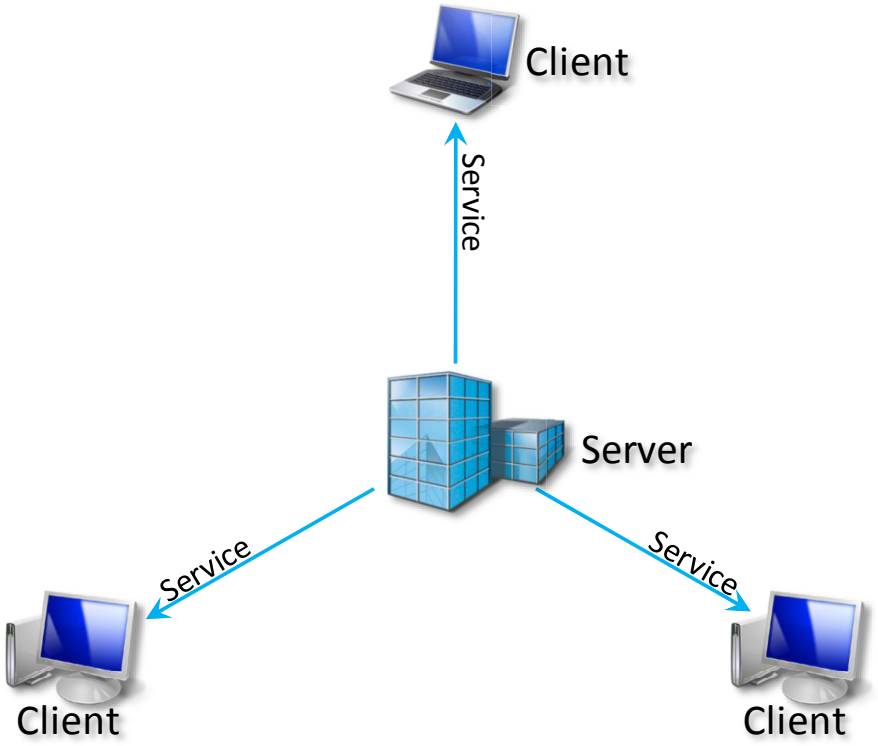



図 2.1: クライアント・サーバモデル

クライアント・サーバモデルを図 2.1 に示す．クライアント・サーバモデルとは，今日のインターネットサービスを支えている通信モデルである．サービスを利用する場合に，クライアント（サービスの利用者）とサーバ（サービスの提供者）が明確にわかれており，クライアントはサーバとのみ通信を行う．そして，複数のクライアントがネットワーク中

に存在する場合でも，サーバがすべての通信を管理できるため，通信の管理性や制御性に優れる．例えば，データ転送において，大量のトラフィックが発生する可能性がある場合には，キューイングなどを行うことでクライアントを待ち状態にするといったことが可能である．

既存のインターネットサービスにおいて，サーバは分散されて複数台設置されることが多い．その理由は大きく分けて 2 つある．

1. 負荷の分散
   1 章で述べた CDN に代表されるサービス時における負荷の分散を目的としている．サーバはすべてのクライアントに対してサービスを提供することから，通信・計算処理を大量に処理しなければならず，定常的に高負荷になりやすい．そのため，単一のサーバにすべての処理を任せるのではなく，複数のサーバを設置し，処理を分散させることで，必要以上の負荷が掛からないようにしている．また，データセンタなどの大規模なサーバ群では，ロードバランサなどの処理の分散に特化した装置を用いる場合もある．

2. サービスの継続
   サーバは定常的に高負荷になりやすいため，ソフトウェアの障害やハードウェアの故障が発生しやすい．そのため，サービスを単一のサーバで運用する場合，サーバの障害や故障によりサービス全体が停止し，データを損失する可能性がある．特に，最近ではクラウドコンピューティング [8, 9, 10, 11] などのように，サービス自体をビジネスにしていたり，サービスに大きく依存したビジネスが登場してきているため，サービスの停止が直接的な損失につながる可能性も高くなってきている．そのような事態を回避・防止するために，サーバを物理的・ネットワーク的に分散させて設置する事例が多く存在する．

## 2.2 P2P モデル

P2P モデルとは，大容量コンテンツの配信などに使われる新しい通信モデルである．サービスを利用する場合，サービスの利用者と提供者が明確にわかれておらず，ネットワーク内の各計算機（ピア・ノード）が自律的に役割を判断し，ネットワークの状況によって振る舞いを変えるため，通信は複数の計算機に対して行われる．そして，複数のクライアント（サービスの利用者）がネットワーク中に存在する場合でも，通信が 1 箇所に集中することがなく，負荷分散性や耐障害性に優れる．

P2P モデルには，基盤となる P2P ネットワークを構成する際の設計思想やポリシによって複数のモデルが存在する．これらは，主に，P2P ネットワークの構造とノードの役割によって分類することが可能である．図 2.2 に P2P モデルの分類を示す．

まず，P2P モデルの構造について，構造化 P2P ネットワークと非構造化 P2P ネットワークに分類することが可能である．通常の P2P システムでは，P2P ネットワークは自律的に形成されるが，自律的に形成された P2P ネットワークにおいて構造化を行い，検索な

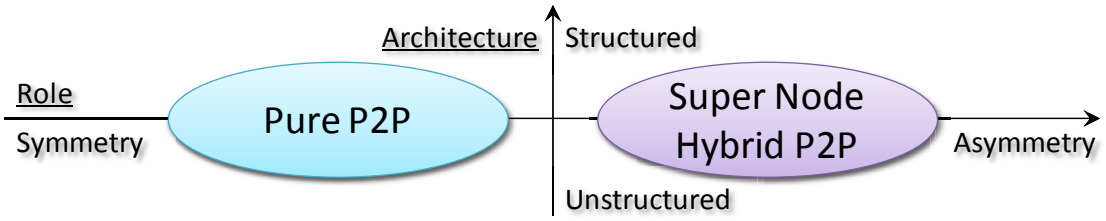



図 2.2: P2P モデルの分類

どの集約ができる方法として，分散ハッシュテーブル（DHT）の研究 [12, 13, 14, 15] が活発に行われている．

次に，P2P ネットワーク内のノードの役割について，対称な役割と非対称な役割に分類することが可能である．P2P ネットワークでは，ノードの役割を漸次的に変化させることが可能であり，サーバなどの中心への依存性が異なるシステムを設計可能である．本節では，P2P ネットワーク内のノードの役割を明確に区別した P2P モデルとして，ハイブリッド P2P モデルとピュア P2P モデルを取り上げる．

### 2.2.1 ハイブリッド P2P モデル

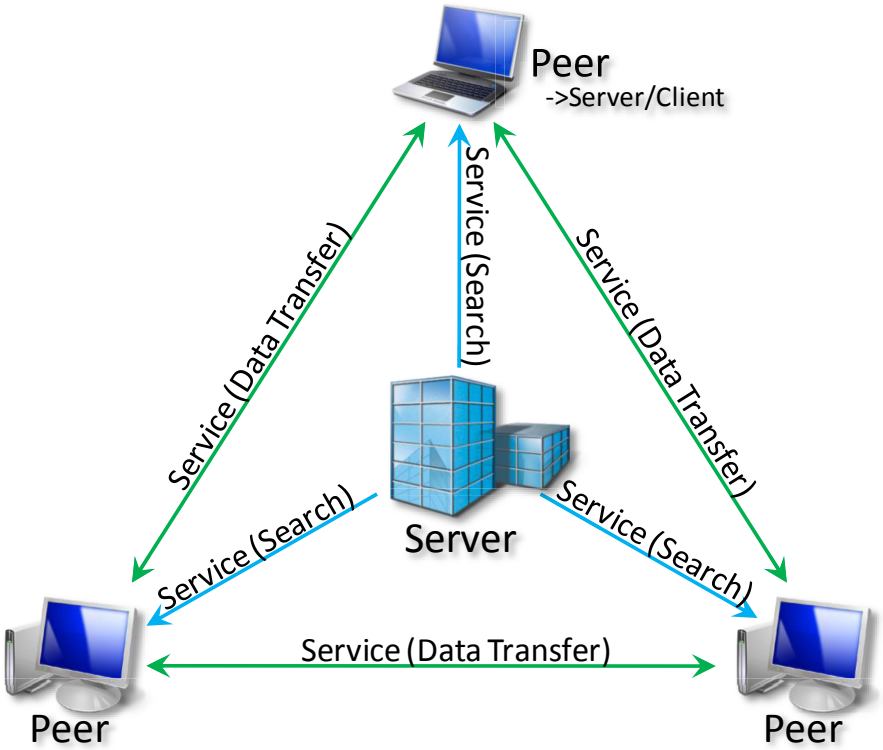



図 2.3: ハイブリッド P2P モデル

ハイブリッド P2P モデルを図 2.3 に示す．ハイブリッド P2P モデルとは，P2P ネットワークの中心にサーバを設置し，ノードの認証やデータの管理を行うモデルである．クライアント・サーバモデルの特徴である管理性や制御性を保証しつつ，P2P モデルの負荷分散性や耐障害性を活かすことができることから，それぞれのモデルのメリットを取り入れたモデルとして活用されている．また，最近では，P2P ネットワーク内のノードからスーパーノードと呼ばれる特別なノードを選択し，それらにサーバの役割の一部を任せることで，ハイブリッド P2P モデルよりもさらにネットワークの負荷分散性や耐障害性を高めた，スーパーノードモデル [16, 17, 18] が登場している．

### 2.2.2 ピュア P2P モデル

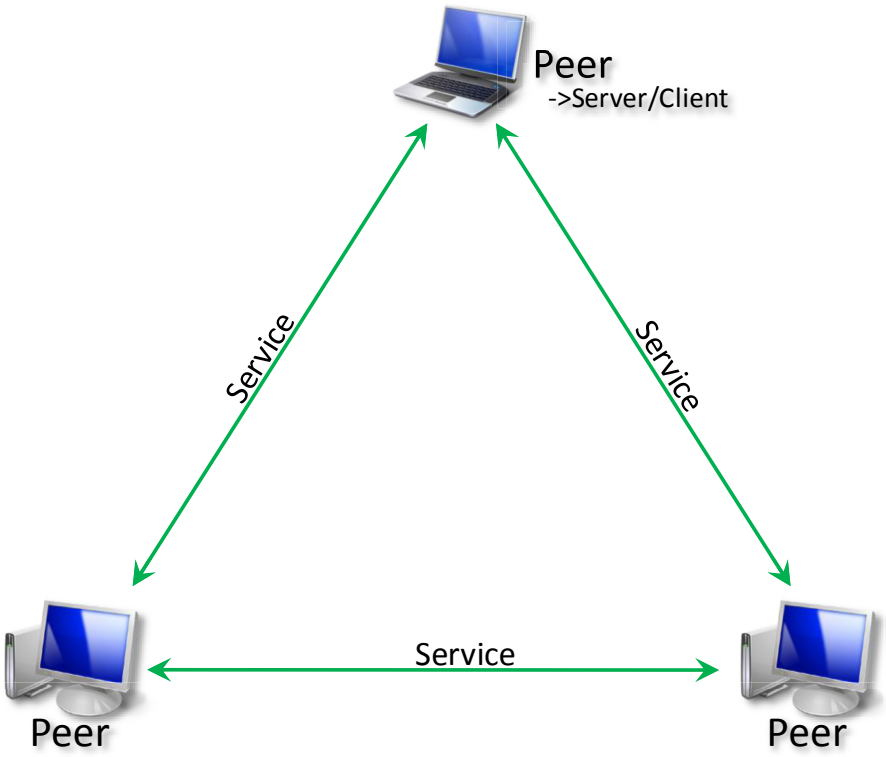



図 2.4: ピュア P2P モデル

ピュア P2P モデルを図 2.4 に示す．ピュア P2P モデルとは，P2P ネットワークのノードをすべて対等に扱い，各ノードの自律性，分散性と協調性によってネットワークを構成するモデルである．P2P モデルの特徴である負荷分散性や耐障害性を追求することで，負荷や障害に対して非常に堅牢な P2P ネットワークを構成することが可能であり，P2P モデルのメリットを最大限に活かすことができる．

### 2.2.3 ハイブリッド P2P モデルとピュア P2P モデルの比較

各 P2P モデルの特徴を表 2.1 にまとめる．

表 2.1: P2P モデルの比較

| | ハイブリッド P2P モデル | ピュア P2P モデル |
|---|---|---|
| 管理性・制御性 | ○ | × |
| 負荷分散性・耐障害性 | ○ | ◎ |

ハイブリッド P2P モデルは，管理性・制御性を保証しつつ，P2P モデルの特徴である負荷分散性・耐障害性を活かしたモデルである．一方で，ピュア P2P モデルは，管理性・制御性を保証しない代わりに，P2P モデルの特徴である負荷分散性・耐障害性に特化したモデルである．

本論文で扱う P2P システムのトラフィック管理・制御は，ハイブリッド P2P モデルとピュア P2P モデルのどちらにも必要な機能であるため，本論文では両者を P2P モデルとして扱う．

## 2.3 各モデルの特徴

クライアント・サーバモデルと P2P モデルの特徴を表 2.2 にまとめる．

表 2.2: クライアント・サーバモデルと P2P モデルの比較

| | クライアント・サーバモデル | P2P モデル |
|---|---|---|
| モデル | 中央集権 | 自律分散 |
| 役割 | わかれている | わかれていない |
| 通信相手 | 単一 | 複数 |
| 計算機 | 専用 | 汎用 |
| 優位点 | 管理性・制御性 | 負荷分散性・耐障害性 |

クライアント・サーバモデルは，中央集権モデルで，役割は明確に分かれている．通信相手は単一で，サービスには専用の計算機を用いることが多く，管理性・制御性に優れる．それに対し，P2P モデルは自律分散モデルで，役割はネットワークの状況によって変わる．通信相手は複数で，サービスには汎用の計算機を用いることが多く，負荷分散性・耐障害性に優れる．

クライアント・サーバモデルには，管理性・制御性に優れるというメリットがある反面，障害や故障による単一障害点を持ち得るというデメリットがある．一方で，P2P モデル

には，負荷分散性・耐障害性がある反面，管理や制御が難しいというデメリットがある．したがって，今後の P2P システムでは管理や制御に関してのデメリットを克服する必要があり，それらが達成されなければ，P2P システムの可能性を大きく発展させることは難しい．

# 第3章　P2Pシステムの問題点

本章では，2章で述べたそれぞれのシステムの性質・特徴を踏まえて，P2Pシステムが抱える問題について述べる．

## 3.1　P2Pモデルの問題点

2.3節でも述べたように，P2Pシステムには負荷分散性や耐障害性に優れる反面，管理や制御が難しいという性質がある．そのため，P2Pシステムが行う通信が，際限なくインターネットバックボーンへ流れることで，他のインターネットサービスの通信に影響を与えている．したがって，ネットワークトポロジについて，現在のインターネットアーキテクチャを考慮し，論理トポロジに下位層トポロジ情報を反映できるようにする必要がある．

### 3.1.1　トポロジ

下位層トポロジを考慮しないで展開した場合の，典型的なP2Pネットワークのつながりを図3.1に示す．

図3.1では，各ノードが自律的にP2Pネットワーク網を構築しているが，大半の通信がインターネット上を通過するために，インターネットバックボーンを圧迫している．

P2Pネットワークにおけるトポロジの管理が難しい要因として，ネットワーク全体を把握できないことが挙げられる．P2Pネットワークには，ハイブリッドP2Pモデルなどの場合を除いて基本的に中心が存在しない．そのため，あるノードがネットワーク内のどのノードと通信するかは各ノードの自律的判断に任せられており，その際に，冗長な経路が構成されたり，無駄な通信が発生してしまう可能性がある．

### 3.1.2　トラフィック

3.1.1項で述べた，P2Pネットワークの論理トポロジが下位層トポロジに対して冗長な構成となっているために，P2Pシステムが発生させるトラフィックが問題となっている．P2Pシステムが発生させるトラフィックの問題は，大きく2つに分類される．

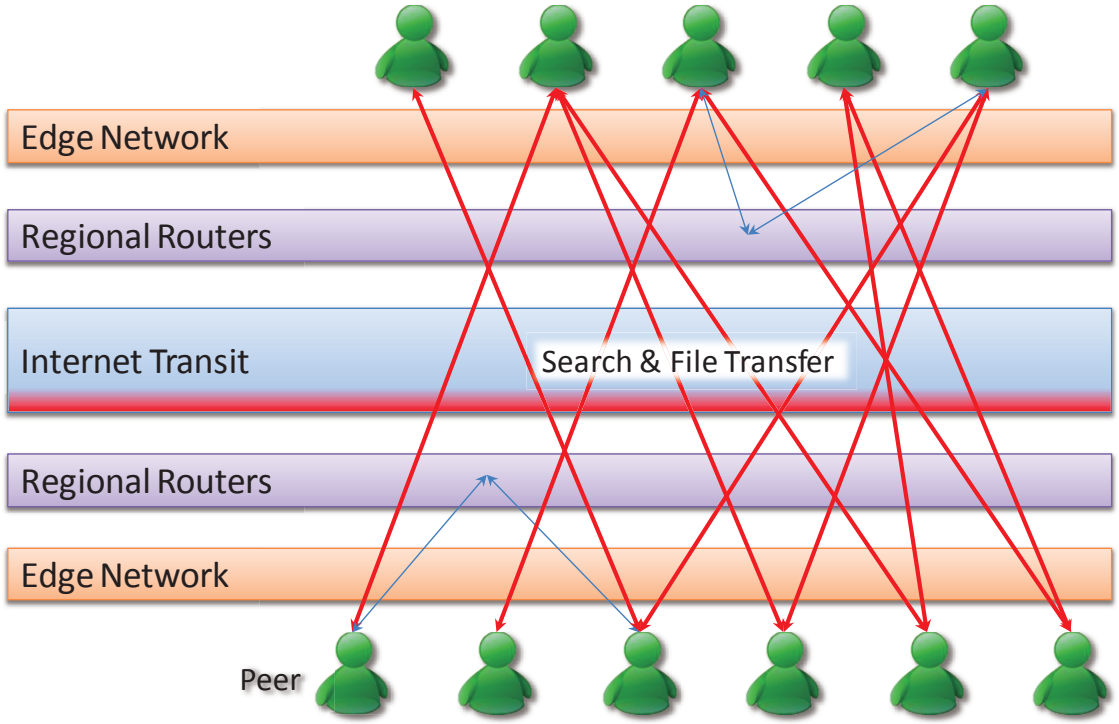



図 3.1: 基本的な P2P ネットワークのトポロジ

#### IX トラフィック

P2P ネットワークが発生させるトラフィックは，インターネットバックボーンの根幹である Internet eXchange（IX）に直接影響を与えている．これは，インターネット上におけるトラフィックの影響範囲が拡大することで，IX を通過するトラフィックが増加してしまうことが原因である．そのため，P2P ネットワークのトポロジにおいて，通信を近接で完結させ，IX を通過するトラフィックをできるだけ削減する必要がある．

#### 地域 IP 網トラフィック

P2P ネットワークが発生させるトラフィックは，IX 以外にも地域 IP 網の帯域を消費している．これは，P2P ネットワーク上において，不必要なノードが通信に関与し，トラフィックの影響範囲が拡大してしまうことが原因である．特に，地域 IP 網では IX に比べて帯域が少ないために，無駄な通信によって帯域が埋め尽くされてしまうケースもあり，トラフィックの影響範囲を最小化する必要がある．

## 3.2 ネットワーク距離

ネットワークトポロジを管理・制御する際に，下位層トポロジを評価する指標として，ネットワーク距離という考え方がある．

インターネットにおける下位層トポロジは非常に複雑であると同時に，全体の構造を把握するためには非常に大きなコストと時間が掛かる．そのため，下位層のネットワークトポロジを評価する際には，直接トポロジを把握するのではなく，ネットワーク距離という指標を用いることで，間接的に下位層トポロジを評価する．

ネットワーク距離とは，例えば，データパケットが送信元と受信先を往復する際に必要とする時間:Round Trip Time（RTT）や，経由するルータ数:Hop Count を指標として表現される距離である．RTT や Hop Count を指標としたネットワーク距離は，すでにネットワークトポロジを管理・制御する際の手法として取り入れられ，データ転送におけるネットワークのパフォーマンスを上げるために使用されている．

## 3.3 P2P システムにおけるネットワーク距離

本節では，近接性の指標として 3.2 節で述べたネットワーク距離を P2P システムに取り入れた場合について述べる．

3.1.1 項で述べたように，P2P ネットワークのトポロジには基本となるモデルや規則性が無いため，その管理・制御は各ノードの自律的な判断に任せられている．そのため，自律的な判断の段階で各ノードが互いの近接性を考慮することで，P2P ネットワークトポロジは管理・制御することが可能であり，それによってトラフィックを集約することができると考えている．

また，本論文では，P2P システムにおけるデータの転送パフォーマンスではなく，トラフィックの影響範囲の集約に主眼をおいている．特に，P2P ネットワークのデータ転送パフォーマンスを上げることは，必ずしも余分なトラフィックの削減や影響範囲を集約・縮小させることにはならない．なぜなら，P2P システムに参加するノードの計算機の処理性能や，インターネットへ接続している回線の種類・速度は一定ではないため，3.2 節で述べた RTT や Hop Count などのノード間で計測される指標では，途中のネットワークトポロジを正確に反映できない可能性がある．

したがって，RTT や Hop Count 以外に，ネットワークトポロジを反映するための手法が必要である．5 章以降では，ネットワークトポロジを反映するための新しい手法の提案を行う．

## 3.4 まとめ

本章では，2 章で述べたモデルの違いに着目し，既存のインターネットアーキテクチャにおける P2P システムの問題点を，管理・制御されていないネットワークトポロジとそれが原因で発生するトラフィックとした．本論文では，ネットワーク距離の指標として，既存の RTT や Hop Count とは違う，新しい指標を用いることで，ネットワークトポロジの管理・制御を行い，P2P システムが発生させるトラフィックの問題を解決する．

# 第4章　関連研究

本章では，ネットワーク距離を利用した，P2P システムのトポロジ管理・制御に関する研究について紹介を行い，その特徴について述べる．

## 4.1　P4P

P4P[19] は，Internet Service Provider（ISP）と P2P ネットワーク事業者が協力し，P2P ネットワークにおける接続ノードの選択を管理・制御するプロジェクトである．具体的には，ISP が自社のネットワークトポロジやポリシなどを P2P ネットワーク事業者に提供することで，P2P ネットワークのトポロジの管理・制御を行っている．P2P ネットワーク事業者はネットワーク距離について，実際に運用されているネットワークから直接取得することができるため，データ転送によるトラフィックの影響を最小限に抑えることが可能である．P4P を用いない場合のモデルを図 4.1 に，P4P のモデルを図 4.2 に示す．

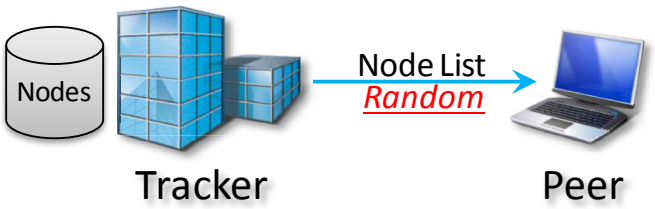



図 4.1: P2P の構成

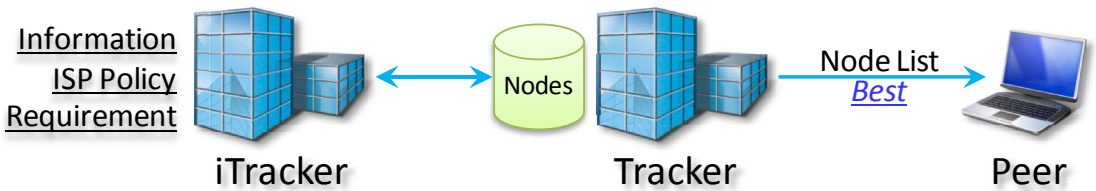



図 4.2: P4P の構成

P4P は，iTracker というデータ転送制御システムを用いている．iTracker とは，ノードがデータ転送を行う際に，データの転送元のノードを問い合わるためのサーバである

Tracker を補助するためのサーバである．通常の P2P システムの場合，図 4.1 に示すように，Tracker はランダムな方法でノードを選択し，リクエストを送ってきたノードにノードリストを返す．一方，iTracker を利用した場合，図 4.2 に示すように，事前に ISP から取得した情報を基に，リクエストを送ってきたノードにとってネットワーク距離が近いと思われるノードを選択し，それらの情報を Tracker に提供することでノードリストを返す．そのため，既存の転送プロトコルに変更を加えることなく P2P システムの管理・制御が可能になり，ノード間のデータ転送におけるネットワーク距離の最小化・影響範囲の集約が可能となっている．特に，ノードの選択方法については，単純なネットワーク距離だけでなく，ISP のポリシや経路情報なども反映することが可能であり，情報を一括して iTracker が管理するため，設定の変更が容易である．

しかし，P4P の強みとなっている ISP のネットワークトポロジやポリシなどの情報は機密性の高い情報であることが多く，開示を望まない ISP などへの対策や，情報を開示することでの ISP へのメリットが明確にならない限り，実際のサービスにおける運用は困難であると考えられる．また，ISP から情報の提供がなされる場合も，その形式や種類の標準が存在しないため，どのように情報をまとめ，ISP へメリットを還元していくのかなど，今後の動向が注目されている．さらに，P4P を運用するには，先述した iTracker が必要となる．iTracker はサーバであり，クライアント・サーバモデルと同様の問題を抱えてしまうため，P2P システムのノード数の変化によって，ネットワーク距離の最小化・影響範囲の集約に支障が出ないよう運用する必要がある．

## 4.2 ONO

ONO[20] は，CDN のサーバを P2P システムのネットワーク距離を測る際の指標とすることで，P2P ネットワーク内のネットワーク距離が近いノードを発見する手法である．1.1 節で述べたように，CDN は世界中に分散して設置されていることから，CDN サーバへのネットワーク距離を各ノードが計測することで，間接的に自分から近いノードを見つけることが可能となる．

ONO のモデルを図 4.3 に示す．ONO のネットワーク距離の指標には，CDN サーバまでの RTT が使われており，広範囲な経路の測定，推測や調査を実行することなく，自分に近いノードを発見することが可能となる．また，Akamai[21] などの商用として運用されている CDN のサーバを指標とすることから，地理的・ネットワーク的な近さにおいて信頼性の高い情報を得ることが可能である．

しかし，ONO には，ポリシ面，物理面と技術面の 3 つの問題がある．

1. ポリシ面
   ONO が取り入れている CDN サーバの利用方法は本来の CDN サーバの利用方法では無いため，CDN 運用事業者のポリシなどの影響によって，CDN サーバから得られる情報が P2P システムのトポロジ管理・制御に有効に働かない可能性がある．

2. 物理面

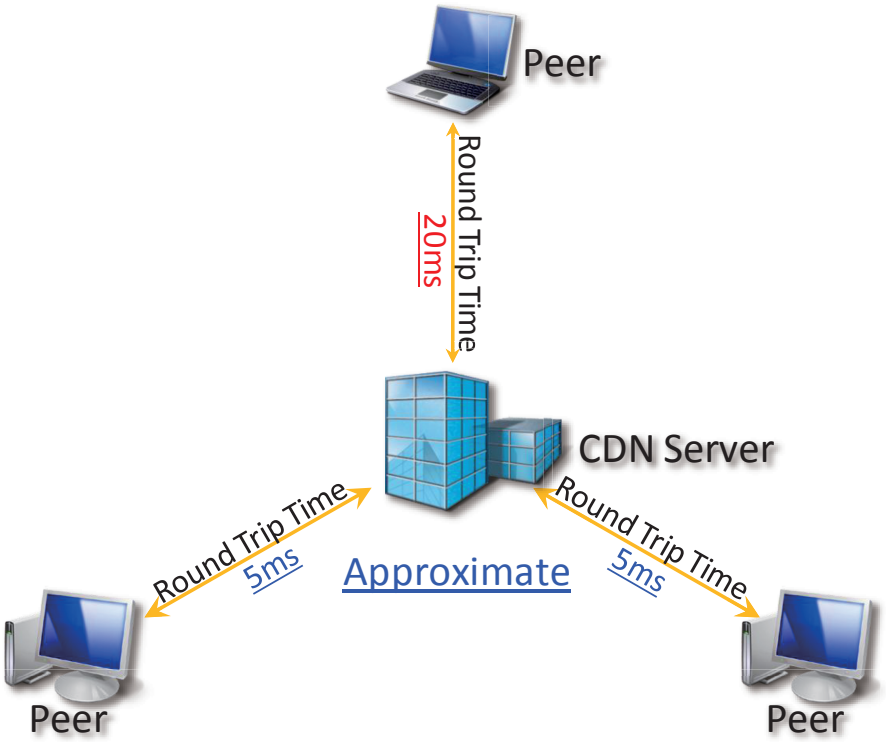



図 4.3: ONO の構成

CDN サーバは世界中に設置されているが，均等に分散して設置されているわけではないため，CDN サーバが分散されて設置されていないエリアがある．

3. 技術面
CDN サーバまでの RTT がネットワーク距離の指標として十分な結果を導き出せない場合には，P2P システムのトポロジ管理・制御が有効に機能しない可能性がある．

## 4.3 Vivaldi

Vivaldi[22] は，あるノードに対する RTT を，他ノードへの実測データを基に推測するアルゴリズムを使うことで，P2P ネットワーク内のネットワーク距離が近いノードを選択する手法である．Vivaldi のモデルを図 4.4 に示す．Vivaldi では，すべてのノードに対して RTT を計測するのではなく，計測結果をユークリッド座標で表現することで，他のノードに対する RTT をユークリッド距離計算によって推定することが可能となっている．また，ユークリッド距離を計算する際に，バネの原理と呼ばれる誤差を修正するアルゴリズムを取り入れることで，RTT について精度の高い結果を得られるようになっており，ネットワーク距離の計算に関するコストを最小化している．

しかし，RTT を用いた指標は必ずしも下位層トポロジを正確に反映できるとは限らないため，場合によってはトラフィックの影響範囲を拡大させるようなトポロジを形成する

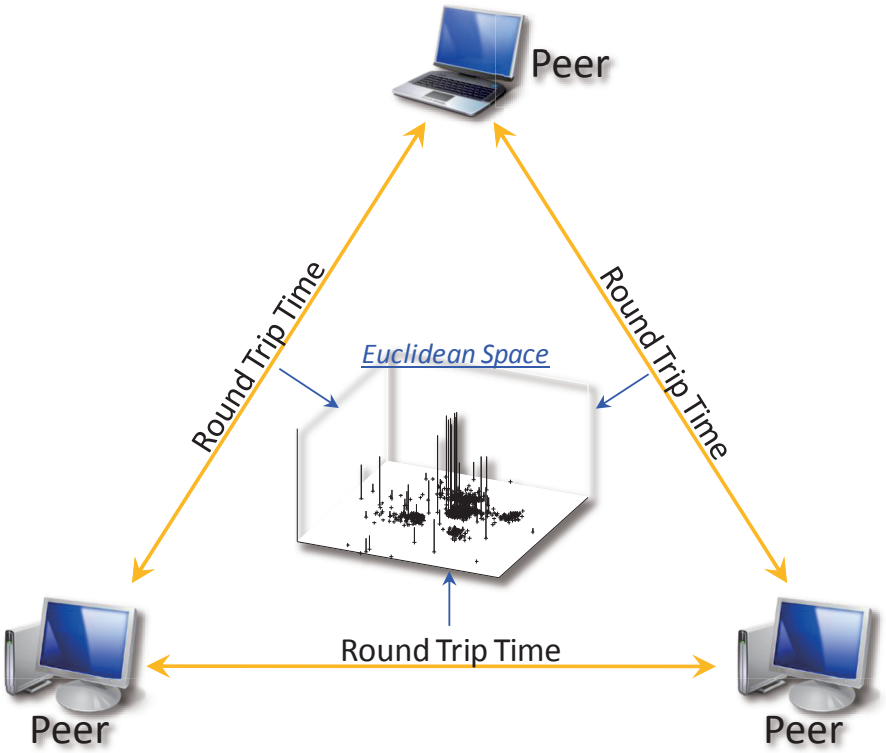



図 4.4: Vivaldi の構成

可能性がある．特に，Vivaldi の提供する情報は，P2P システムにおけるデータ転送のパフォーマンスの向上を優先しており，ネットワークトポロジについての管理・制御の優先順位は必ずしも高くないため，IX や地域 IP 網へのトラフィックの負荷を抑えることができない可能性がある．また，Vivaldi のアルゴリズムを運用する際には，利用する回線の環境が高速であることが求められており，低速な回線を使っているノードや，RTT が大きいノードに関しては，ネットワーク距離の推測が有効に働かない場合がある．

## 4.4 まとめ

本章では，ネットワーク距離を指標とした P2P システムのトポロジ管理・制御に関する研究について紹介を行い，その特徴について述べた．取り上げた研究はどれも BitTorrent[23] に関する研究であり，P2P システムにおけるネットワークトポロジの管理・制御よりも，パフォーマンスの向上を目的としているものが多い．また，既存手法ではネットワーク距離の指標として RTT を用いているが，RTT は計測の度に大きく揺らぐ性質を持つため，ネットワークトポロジの管理・制御には必ずしも有効ではない．

したがって，本論文が対象としている，トラフィックの影響範囲を集約するためのネットワークトポロジの管理・制御を行うための P2P システムには，別のネットワーク距離計算手法が必要である．5 章では，前述したネットワークトポロジを管理・制御するための手法の提案および設計を行う．

# 第5章　アプローチ

本章では，3 章で述べた既存のモデルの問題点をふまえ，P2P システムにおけるネットワークトポロジの管理・制御手法を提案し，設計を行う．

## 5.1　概要

P2P ネットワーク内でトポロジの管理・制御を行う際には，各ノードが自律的に下位層トポロジを判断して，P2P ネットワークのトポロジを管理・制御する必要がある．RTT や Hop Count による下位層トポロジ指標は，3.2 節で述べたように，余分なトラフィックの削減や影響範囲を集約・縮小することにはつながらないため，本論文では**Domain Name System（DNS）[24, 25] を利用したネットワーク距離**を提案する．

現在のインターネットにおいて，DNS は図 5.1 のように管理ドメイン [25] ごとに分割され，階層構造により各レベルが区切られている．そのため，DNS の階層構造において近接性を考慮して，ドメイン間の距離をネットワーク距離とすることで，できるだけ近いドメイン内でトラフィックを局所化することは有効であると考えられる．また，近接性という観点では，DNS におけるドメインは階層構造を持つことから，すべてのレベルでドメインが一致しない場合でも，上位レベルで一致していれば，その一致率により近接性を表現することが可能となると考えられる．

下位層トポロジ指標として DNS を利用することで，RTT や Hop Count では推測することが困難だった，単一組織内のネットワーク距離を求めることが可能となる．そして，データ転送時のノード選択において，単一組織内のより近いノードを選択することで，トラフィックの削減や影響範囲の集約・縮小が可能となると考えられる．

## 5.2　手法

本論文では，下位層トポロジを反映するためのネットワーク距離指標として，DNS を利用する．具体的には，DNS の管理ドメインのラベルとして得られる情報である Fully Qualified Domain Name（FQDN）[26] を用いる．FQDN は図 5.2 のように示される．

FQDN は IP アドレスから取得することが可能であり，ほとんどの場合，FQDN にはノードが所属している組織名（ドメイン名）が含まれている．そこで,「FQDN において，同一のドメイン名を持つノードは単一の組織に属しているため，ネットワーク的に近い」と仮定し，そのようなノードへ優先的に接続することで，トラフィックの削減や影響範囲の集約・縮小が期待できる．

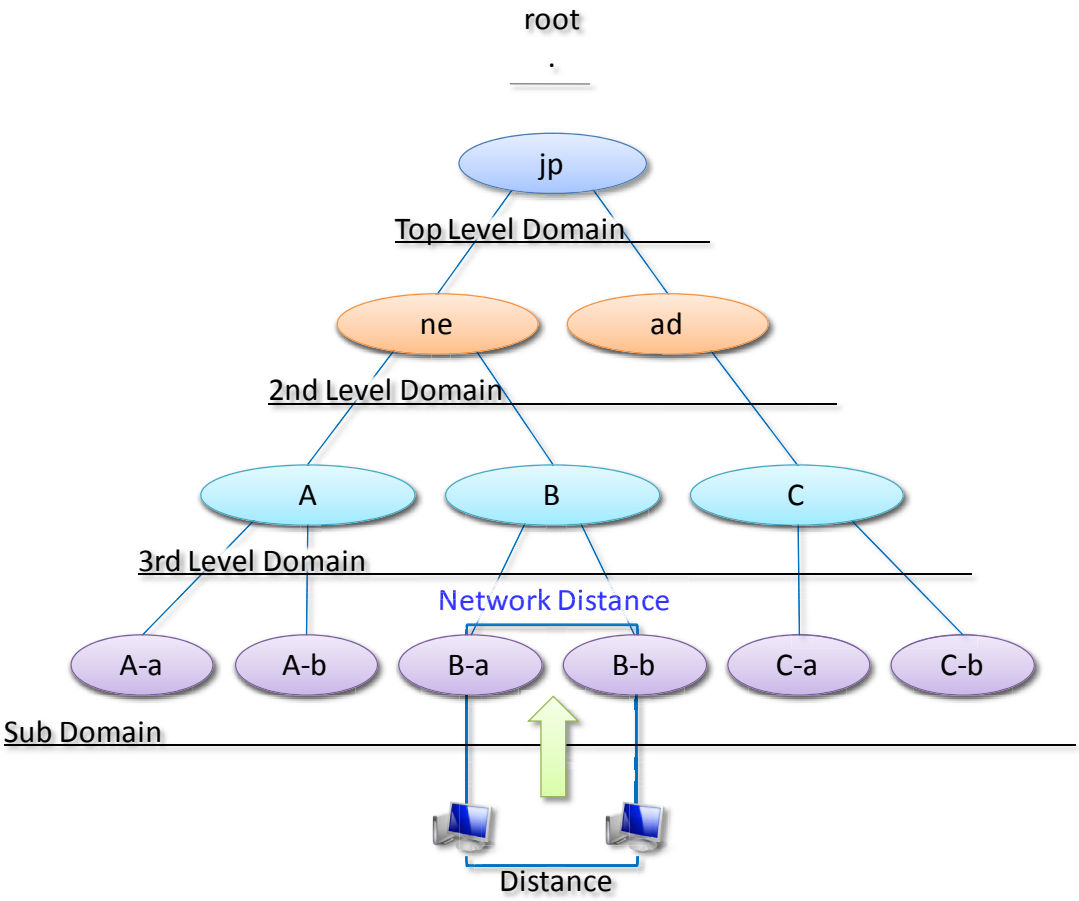



図 5.1: DNS における管理ドメインの階層構造

p1234-ipbf5678marunouchi.tokyo.ocn.ne.jp

図 5.2: FQDN の例

## 5.3 設計

本節では，ネットワーク距離の指標として FQDN を用いた P2P システムの設計を行う．

### 5.3.1 設計概要

本手法では，各ノードが FQDN を認識し，接続先を選ぶ際に FQDN を比較することでノード間のネットワーク距離の表現を可能にする．そして，ノードへ接続を行う際に，ネットワーク的により近いノードに接続を行うことが可能となる．

このような動作を P2P システムの各ノードが自律分散的に繰り返し行うことで，データ転送時のトラフィックがノードの周囲で完結し，トラフィックの影響範囲が集約・縮小される．

### 5.3.2 設計要件

設計における要件と，それらを満たすための機能は以下の 3 点である．

1. 下位層トポロジ情報の取得
   P2P ネットワークの下位層トポロジ情報を取得する．

2. 下位層トポロジ情報の解析
   取得した下位層トポロジ情報を展開，解析する．

3. ネットワーク距離の表現
   下位層トポロジ情報の解析結果を基に，ネットワーク距離を表現する．

### 5.3.3 下位層トポロジ情報の取得

5.2 節で述べたように，ネットワーク距離を表現する際に使用する FQDN を取得する．下位層トポロジ情報である FQDN は IP アドレスから取得することが可能である．具体的には，User Datagram Protocol（UDP）[27] または Transmission Control Protocol（TCP）[28] を使い，DNS サーバの 53 番ポートに向けて問い合わせコマンドを発行すると，IP アドレスに対応したドメイン名が応答として返ってくる．そのため，P2P ネットワーク内のノードとなる計算機が直接インターネットに接続している場合には，ネットワークインタフェースに設定されている IP アドレスを取得することで，FQDN を知ることが可能である．

しかし，この方法では必ずしも FQDN が取得できない場合がある．これは，Network Address Translation（NAT）[29] を利用している場合に，クライアントの計算機のネットワークインタフェースに設定されている IP アドレスがプライベートアドレスとなっているためである．NAT は，近年，家庭や企業などで広く使われており，P2P ネットワーク内のノードとなる計算機は，直接インターネットに接続している訳ではなく，NAT によって間接的にプライベートネットワークからインターネットに接続していることになる．本手法では，NAT を使っている場合にも FQDN を取得できるようにするために，Universal Plug and Play（UPnP）[30] や P2P ネットワーク内のノードとの通信によってグローバルアドレスを取得できるようにする．

### 5.3.4 下位層トポロジ情報の解析

下位層トポロジ情報として利用する FQDN には，地理情報や回線情報など，様々な情報が含まれている場合が多く，それらの地理情報，回線情報の種類も ISP や組織ごとに異なる．そのため，単純に処理しただけでは，下位層トポロジ情報を反映できない可能性がある．本研究で提案するアルゴリズムでは，図 5.3 で示す FQDN のドメインの各レベルについて判定を行い，さらに文字列の類似度を求めるアルゴリズムであるレーベンシュタイン距離 [31] を取り入れることで，含まれている情報の差異を吸収する．レーベンシュタ

イン距離とは，2 つの文字列がどの程度異なっているかを数値で求めるアルゴリズムである．具体的には，ある文字列から目的の文字列を導き出すために，置換・削除・挿入を最低何回行う必要があるかを計算し，文字列の類似度が低いほど大きな値が出るというものである．

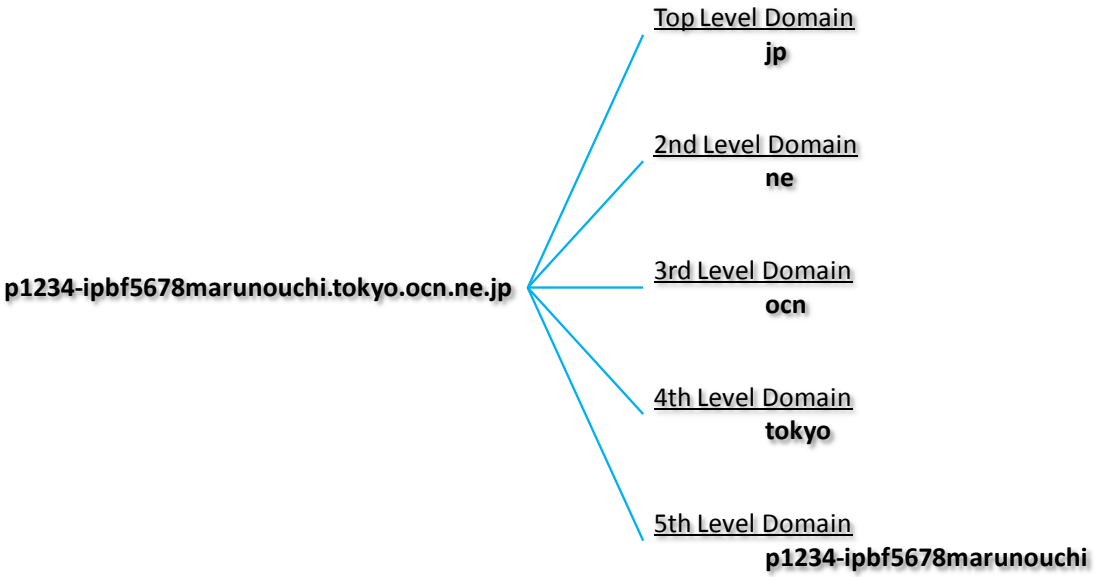



図 5.3: FQDN におけるドメインのレベル

本手法において，ドメインとレーベンシュタイン距離という 2 つのアルゴリズムを用いて FQDN を解析するのは，複数のレベルでトラフィックの影響範囲を集約できるようにするためである．

まず，FQDN のドメインに対し各レベルで判定を行うのは，同一ドメイン内において複数のドメインレベルが存在する場合に，ドメイン間距離の短いノードを判定するためである．例えば図 5.3 の場合，同じドメインである OCN のノードが複数存在した場合に，第 4 レベルドメインが同じ tokyo であるノードを優先的に選択することで，OCN という管理ドメイン内の tokyo というドメインにトラフィックの影響範囲を集約することができる．

次に，レーベンシュタイン距離で判定を行うのは，FQDN のドメインの各レベルの判定によって，絞り込まれたドメイン内で近接ノードを発見するためである．例えば図 5.3 の場合，第 5 レベルのドメインについては，ネットワーク ID と思われる数字や地域名が入っている．そのため，レーベンシュタイン距離を用いることで，同じ tokyo というドメインレベルの中でのネットワーク距離を表現することが可能となり，トラフィックの影響範囲を集約することができる．

### 5.3.5 事前実験

本項では事前実験として，本アルゴリズムと他の文字列処理アルゴリズムとの比較・検討を行う．

- ドメインレベル＋レーベンシュタイン距離（本アルゴリズム）
  FQDN のドメインレベルに着目して各レベルの判定を行い，ホスト名のレベルまで一致した場合には，ホスト名に対してレーベンシュタイン距離を用いる．
- ドメインアルゴリズム
  FQDN のドメインレベルにのみ着目し，各レベルの判定を行う．
- レーベンシュタイン距離アルゴリズム
  FQDN に対してレーベンシュタイン距離のみを用いる．
- 最長一致アルゴリズム
  FQDN を管理ドメインの上位階層（文字列の後方）から最長一致を行う．

### ドメインアルゴリズムとの比較

本アルゴリズムとドメインアルゴリズムを検討するために，図 5.2 で示した FQDN に対して，以下の FQDN とのスコアを計算する．

- p1111-ipbf2222marunouchi.tokyo.ocn.ne.jp
- p1111-adsan22honb3-acca.tokyo.ocn.ne.jp

表 5.1 に本アルゴリズムで FQDN を計算した結果を示す．

表 5.1: 本アルゴリズムによるスコア

| 対象元 | p1234-ipbf5678marunouchi.tokyo.ocn.ne.jp | - |
|---|---|---|
| 対象 1 | p1111-ipbf2222marunouchi.tokyo.ocn.ne.jp | 313 |
| 対象 2 | p1111-adsan22honb3-acca.tokyo.ocn.ne.jp | 301 |

本アルゴリズムを使用した場合，同じ tokyo という第 4 レベルドメインを持つノードにおいて，第 5 レベルドメインのレーベンシュタイン距離が異なるため，2 つの FQDN の優先度において，対象 1:313 と対象 2:301 という差を持たせることが可能となる．
表 5.2 にドメインアルゴリズムで FQDN を比較した計算を示す．

表 5.2: ドメインアルゴリズムによるスコア

| 対象元 | p1234-ipbf5678marunouchi.tokyo.ocn.ne.jp | - |
|---|---|---|
| 対象 1 | p1111-ipbf2222marunouchi.tokyo.ocn.ne.jp | 256 |
| 対象 2 | p1111-adsan22honb3-acca.tokyo.ocn.ne.jp | 256 |

ドメインアルゴリズムの場合，同じ tokyo という第 4 レベルドメインまでを判定し，第 5 レベルドメインについては，異なっているとして計算に含めないため，類似度が低いにもかかわらず，同じ 256 という優先度となってしまう．

以上の事から，本アルゴリズムの方が，自分の FQDN と相手の FQDN の類似度をより正確に求めることができるため，ドメインアルゴリズムに比べて優位である．

### レーベンシュタイン距離アルゴリズムとの比較

本アルゴリズムとレーベンシュタイン距離アルゴリズムを検討する．

レーベンシュタイン距離アルゴリズムの場合，文字列の類似度が低ければ低いほどスコアが高くなる．そして，本アルゴリズムでも一致度が高い場合にはレーベンシュタイン距離を使用するため，類似度の精度という意味では，優位性はない．

しかし，計算コストを考えた場合には，レーベンシュタイン距離の計算がホスト名のみで終わる本アルゴリズムの方が優位である．具体的な数値で計算コストを算出するために，レーベンシュタイン距離の計算コストを計測する実験を行った．表 5.3 に使用した計算機の概要を示す．

表 5.3: 実験で使用した計算機

| 機種 | ThinkPad X40 2371-GDJ |
|---|---|
| CPU | PentiumM 1.4GHz |
| Memory | 1.0GB |
| OS | Windows Vista Ultimate (x86) |

実験には 26 文字のアルファベットと 10 文字の数字の合計 36 文字を使用し，以下の条件で行った．

1. 本アルゴリズム
   （ホスト名最大長）63 文字のランダムな文字列に対して，レーベンシュタイン距離を計算．

2. レーベンシュタイン距離比較
   （FQDN 最大長）255 文字のランダムな文字列に対して，レーベンシュタイン距離を計算．

上記の実験をすべての文字（36 文字）に対して 3 回行った結果を表 5.4 に示す．

表 5.4 より，63 文字の場合には 10msec，255 文字の場合には 160msec の時間がかかっていることから，文字列が 4 倍になった場合には，約 16 倍の時間がかかっていることがわかる．

表 5.4: レーベンシュタイン距離の計算コスト

| | 文字数 | 最高値 |
|---|---|---|
| 1 | 63 文字 | 10msec |
| 2 | 255 文字 | 160msec |

表 5.5: FQDN データに対するアルゴリズム別計算時間

| アルゴリズム | 最高値 |
|---|---|
| 本アルゴリズム | 234msec |
| レーベンシュタイン距離アルゴリズム | 3,504msec |

また，一般的に扱われる FQDN に対して，本アルゴリズムとレーベンシュタイン距離アルゴリズムを適用した場合の結果を調べるため，7.2 節で使用する約 35,000 の FQDN データに対して，計算時間を比較する実験を 3 回行った．表 5.5 に結果を示す．

表 5.5 より，本アルゴリズムでは 234msec で計算が終わっているのに対し，レーベンシュタイン距離アルゴリズムでは，3,504msec と約 15 倍の時間がかかっている．

したがって，本アルゴリズムの方が，計算コストにおいて，レーベンシュタイン距離アルゴリズムに比べて優位である．

### 最長一致アルゴリズムとの比較

本アルゴリズムと最長一致アルゴリズムを検討する．

最長一致アルゴリズムの場合，5.3.5 項で使用した例だと，表 5.6 のようになり，文字列の類似度が高い対象 1 が 26，低い対象 2 が 16 となり，傾向が表 5.1 で示した本アルゴリズムと同様の結果となる．

表 5.6: 最長一致アルゴリズムによるスコア（OCN）

| 対象元 | p1234-ipbf5678marunouchi.tokyo.ocn.ne.jp | - |
|---|---|---|
| 対象 1 | p1111-ipbf2222marunouchi.tokyo.ocn.ne.jp | 26 |
| 対象 2 | p1111-adsan22honb3-acca.tokyo.ocn.ne.jp | 16 |

また，計算コストに関しても，最長一致アルゴリズムは，全アルゴリズムの中で最も計算コストが低いアルゴリズムであるため，図 5.2 で示した FQDN に対しては，最長一致アルゴリズムの方が優位に文字列の類似度を求めることが可能である．

しかし，FQDN の中には，最長一致アルゴリズムで十分に文字列の類似度を求められない場合がある．例えば，自分の FQDN が「softbank123123123123.bbtec.net」である場合に，以下の FQDN とのスコアを計算することを考える．

- softbank123123123124.bbtec.net
- softbank111111111111.bbtec.net

この場合，最長一致アルゴリズムを使って類似度を求めると，表 5.7 のようになる．

表 5.7: 最長一致アルゴリズムによるスコア（BBTEC）

| 対象元 | softbank123123123123.bbtec.net | - |
|---|---|---|
| 対象 1 | softbank123123123124.bbtec.net | 10 |
| 対象 2 | softbank111111111111.bbtec.net | 10 |

このような FQDN の場合，最長一致アルゴリズムでは，対象 1 の方が明らかに文字列の類似度が高いにもかかわらず，対象 2 の文字列の類似度がそれほど高くない FQDN と同様のスコアを算出してしまうため，十分に文字列の類似度を求めることができない．本アルゴリズムを使って類似度を求めると，表 5.8 のようになる．

表 5.8: 本アルゴリズムによるスコア

| 対象元 | softbank123123123123.bbtec.net | - |
|---|---|---|
| 対象 1 | softbank123123123124.bbtec.net | 191 |
| 対象 2 | softbank111111111111.bbtec.net | 184 |

表 5.8 より，本アルゴリズムを使用した場合には，文字列の類似度が高い対象 1 のスコアが 191，文字列の類似度が低い対象 2 のスコアが 184 となる．

したがって，本アルゴリズムの方が，自分の FQDN と相手の FQDN の類似度をより正確に求めることができるため，最長一致アルゴリズムに比べて優位である．

### 5.3.6 ネットワーク距離の表現

本手法では FQDN の構造に着目してネットワーク距離の表現を行う．また，その際，優先度という指標を導入する．

ネットワーク距離の場合，近さが優先順位の基準となるため，文字列の相違が小さければ小さいほど優先順位は高くなる．しかし，このような優先順位の付け方を行った場合，他の指標とネットワーク距離を比べるのが困難になるため，優先度という新しい指標を導

入することで，文字列の相違をスコアとして反映できるようにする．図 5.3 を例に優先度の計算を行う．図 5.3 には全部で 5 個のドメインレベルが存在するため，それぞれ，Top Level，2nd Level,… とドメインが不一致するまで比較を行う．そして，不一致が起こったドメインまでをネットワーク距離として設定し，途中のドメインレベルが同じでも既に不一致が起こっていれば，ネットワーク距離として計算しないとする．図 5.4 に例を示す．

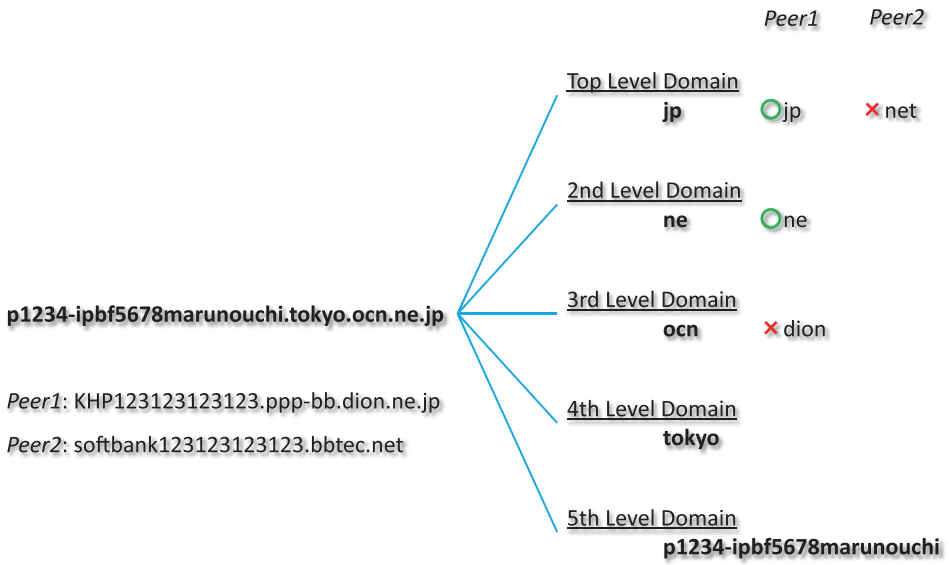



図 5.4: FQDN におけるドメインのレベル比較例（1）

図 5.4 において，「KHP123123123123.ppp-bb.dion.ne.jp」は第 2 レベルドメインまで一致している．

また，「softbank123123123123.bbtec.net」は Top Level で不一致が発生しているため，どのドメインも一致していないということになる．

以上のように，FQDN をドメインレベル別に比較することにより，その一致度からノードの接続先に優先度をつけることが可能となる．本手法では，同一ドメインであるほど優先度が上がるため，ノードはできる限り同一ドメインのノードを選ぶことが可能となる．結果として，トラフィックがノードの所属するドメイン内で集約され，ドメインを跨ぐトラフィックを減らすことができるため，3.1.2 項で述べた IX のトラフィック問題を解決することができると考えられる．

同一ドメイン内でトラフィックが集約する効果については，P2P ネットワーク実験協議会の実験結果 [32] で詳しく報告されている．具体的には，ダミーのノードを用いた配信実験の結果で，実験における同一ドメイン内の通信の割合は最高でも 3 割弱程度であり，同一ドメイン（AS）内にトラフィックを集約することが重要であると指摘している．

別の場合の FQDN のドメイン比較について，図 5.5 に例を示す．

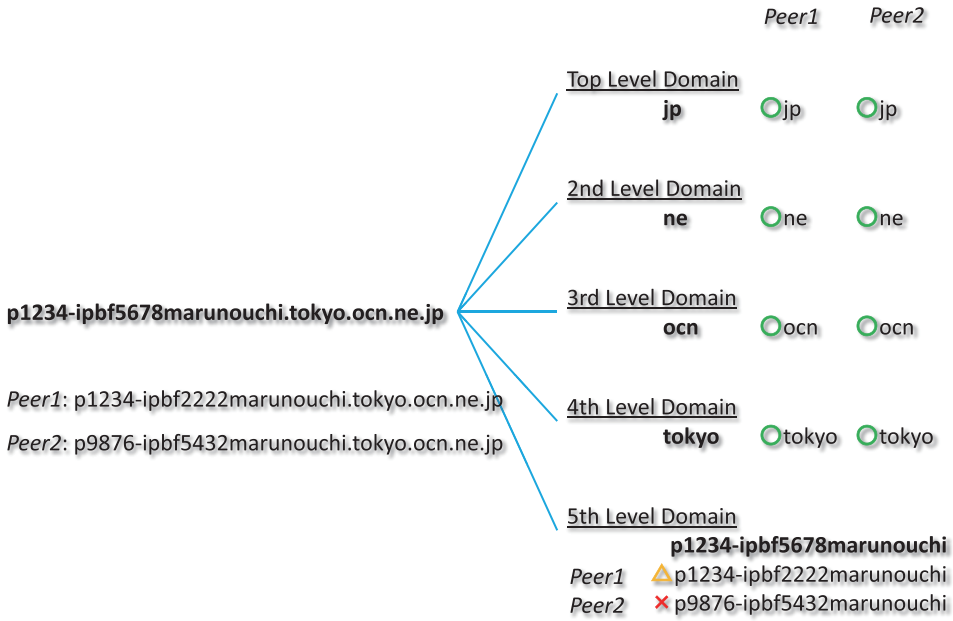



図 5.5: FQDN におけるドメインのレベル比較例（2）

図 5.4 の場合と違い，図 5.5 の場合，先述したアルゴリズムでは，

- p1234-ipbf2222marunouchi.tokyo.ocn.ne.jp
- p9876-ipbf5432marunouchi.tokyo.ocn.ne.jp

は同一の優先度と見なされる．

しかし，ホスト名だけを比較した場合，「p9876-ipbf5432marunouchi」よりも「p1234-ipbf2222marunouchi」の方が「p1234-ipbf5678marunouchi」に対して文字列の類似度が高いため，レーベンシュタイン距離を用いて，優先度を高く設定することで，同じドメインレベル内でのトラフィックの集約を行うことが可能である．

日本の ISP の場合，FQDN の命名には規則性がある．例えば，FQDN に含まれる IP アドレスの上位ビットでネットワーク ID を表している場合などがあるため，レーベンシュタイン距離を用いたネットワーク距離の判定は有効であると考えられる．そのため，本アルゴリズムではドメインレベルの比較に加えて，レーベンシュタイン距離で FQDN の比較を行えるようにし，優先度の粒度を細かく設定することを可能にする．

本アルゴリズムを適用することで，IX を通過するトラフィックを削減できるだけでなく，同一ドメイン内のトラフィックの影響範囲を集約することも可能になるため，3.1.2 項で述べた地域 IP 網に関しても，余分なトラフィックによる帯域の消費を抑えることができると考えられる．

## 5.4 まとめ

本章では，ネットワーク距離を表現する新しい指標として，DNS を利用したネットワーク距離を提案し，設計を行った．本手法を用いることで，従来の方法では難しかった単一組織内のネットワーク距離を求めることが可能となり，同一ドメイン内でトラフィックの影響範囲の集約を行うことが可能となると考えられる．

# 第6章　実現

本章では，5章で述べたアプローチ・設計を基に，ネットワーク距離の指標としてFQDNを用いたP2Pシステムの開発について述べる．

## 6.1　実装概要

本論文ではネットワーク距離の指標としてFQDNを用いたP2Pシステムとして*LinkPoint*を実装した．

LinkPointは，5章で述べた下位層トポロジ情報の取得からネットワーク距離の表現までを実装し，接続するノードの選択だけでなく，P2Pシステム内で交換されるデータにそれらの情報を付加することで，P2Pシステム全体のデータ転送効率を向上，トラフィックの影響範囲の集約を目指している．また，様々な種類のデータをP2Pシステム上で共有することがP2Pシステムの可能性を開拓し，発展を促すと考え，ファイル共有プラットフォームとしての使用方法を想定した．そして，ファイル共有プラットフォームに必要と思われるユーザ・インタフェースの実装や，機能の実装を行った．

### 6.1.1　実装環境

LinkPointの実装環境を表6.1に示す．

表 6.1: LinkPointの実装環境

| 言語 | C# |
|---|---|
| フレームワーク | .NET Framework 3.5 SP1 |
| OS | Windows Vista Ultimate (x86) |
| OSバージョン | Version 6.0.6001 Service Pack 1 Build 6001 |
| 開発環境（IDE) | Visual Studio 2008 Professional Edition |
| IDEバージョン | Version 9.0.30729.1 SP |
| コンパイラ | Visual C# 2008 Compiler Version 3.5.30729.1 |

また，ライブラリとして，SQL Server Compact（Version 3.5 SP1）を利用した．

LinkPoint は先述したライブラリ以外を，すべて独自に実装している．これは，LinkPoint が，既存のパフォーマンスのために構成された P2P システムとは異なったアプローチを採っているためである．特に，ネットワークトポロジの形成は，P2P システムの中核を占めている機能であり，既存の実装を改良するだけでは，規模性や拡張性において限界があると考えられるため，本実装では，P2P のネットワークトポロジの形成部分から設計・実装を行った．

ユーザ・インタフェースの部分では，.NET Framework 3.0 から採用された Windows Presentation Foundation（WPF）を利用した．WPF とは，ソースコードに直接記述していた Graphical User Interface（GUI）のコードを，eXtensible Application Markup Language（XAML）と呼ばれる eXtensible Markup Language（XML）形式の一種で記述出来るようにしたものである．XAML を使用するため，GUI 変更に伴う労力が大幅に削減され，アプリケーションは，用途に応じたユーザ・インタフェースを簡単に採用することが可能となった．P2P システムにおいても，ユーザ・インタフェースは用途によって様々なものが考えられるため，必要に応じて変更が可能な WPF を採用した．

## 6.2 動作概要

LinkPoint の動作について述べる．動作は，大きく 3 つに分けられる．

- P2P システムへの参加
  P2P ネットワークへ加入するために，他のノードへの接続を行い，ネットワーク距離を取得する．
- メタ情報管理
  P2P ネットワーク内へ保持しているデータの情報を配布し，保持していないデータの情報を蓄積・探索する．
- データ転送
  データを保持しているノードへ接続し，データを取得する．データを保持しているノードが複数存在する場合には，ネットワーク距離を考慮し，近いノードからデータを取得する．

### 6.2.1 ネットワークへの参加

LinkPoint が起動する際の動作について述べる．LinkPoint は起動する際に複数のノードに接続する．これは，接続先のノードが P2P システムから離脱した場合に，P2P システムの機能が利用できなくなることを防ぐためである．

また，接続を行う際には，セキュリティの確保や不正な接続を防ぐために，暗号化と認証を行う．暗号化は，公開鍵暗号として 1024-bit RSA[33] を用いた鍵配送を行い，共通鍵を交換した上で，共通鍵暗号である 128-bit RC4[34] により行われる．公開鍵・共通鍵は

接続ごとにランダムに生成されるため，ノード間の通信は傍受されることはない．認証として，LinkPoint プロトコルによる認証処理を行う．バージョン情報や接続の種類などを確認し，通信互換性や接続後の処理を決定する．

暗号化と認証が終わると，LinkPoint は FQDN の取得を試みる．FQDN の取得には，5.3.3 項で述べた方法を使用するが，NAT の有無によって動作を変更しなくてはならないため，FQDN を取得する際に必要となる IP アドレスには優先順位を設定している．優先順位が最も高いのは，ノードの接続時に得られた IP アドレスを基に FQDN を取得するもので，次に UPnP から得られた IP アドレス，最後にネットワークインタフェースから得られた IP アドレスという優先順位になっている．このような優先順位にすることで，グローバルアドレスに変更がある際にも，検知を容易に行うことが可能となる．FQDN の取得が終わると，その情報を接続先のノードへ送信し，ネットワークへの参加処理は終了する．

### 6.2.2 メタ情報管理

LinkPoint の P2P システム内での動作について述べる．LinkPoint が構築する P2P システムでは，各ノードが保持しているデータは，メタ情報として抽象化されて扱われる．そのため，各ノードは保持しているデータのメタ情報を配布し，保持していないデータのメタ情報を蓄積・探索する．メタ情報の配布には，拡散クエリと検索クエリという 2 種類のメッセージを使用する．

- 拡散クエリ
  拡散クエリは，ノード間で定期的に送受信されるメッセージで，各ノードが保持しているメタ情報の一部を相手に送信する．このメッセージは，メタ情報の積極的な拡散を行っており，各ノードがアップロードしているデータのメタ情報の他に，保持しているデータのコピー（キャッシュ）のメタ情報が含まれている．拡散クエリを受信したノードは，含まれているメタ情報を自分の保持しているメタ情報と比較し，必要に応じて追加・変更を行う．

- 検索クエリ
  検索クエリは，ノード間で不定期に送受信されるメッセージである．このメッセージは，LinkPoint 上で検索ボタンが押されたときや，クラスターワードと呼ばれるユーザの嗜好を示すキーワードが設定されている場合に発行される．メッセージの中には，メタ情報ではなく，メタ情報を検索する際に使用される特定のキーワードなどが含まれている．検索クエリを受け取ったノードは，保持しているメタ情報の中からキーワードに該当するデータを探し出し，同じように検索クエリとして他のノードへ転送する．検索クエリの中には経由したノード数が保持されており，一定のメタ情報が集まったとき，または，一定のノード数を経由したときに，発行元のノードへクエリ応答として送信される．

LinkPoint はこの 2 種類のメッセージを送受信することで，ネットワーク中のメタ情報を蓄積・探索する．

### 6.2.3 データ転送

LinkPoint のデータ転送時の動作について述べる．データ転送は大量のトラフィックを発生させるため，LinkPoint では，転送元ノードの選択を 5.3 節で述べたアルゴリズムを基に決定する．

データ転送を行う前に，ユーザはまず取得したいデータを指定する．データは 3 つの方法で指定することが可能である．

1. キーワードなどの条件を設定し，条件に一致するデータをすべて取得する
2. データの内容によって一意に決定されるハッシュ値を指定して取得する
3. 蓄積されたメタ情報を直接指定する

取得するデータが決定されると，データの情報がダウンロードリストに登録される．ダウンロードリストに登録された情報は，一定間隔で検索クエリが投げられる．また，蓄積されたデータ内に該当するメタ情報が発見された場合，含まれているノード情報を基に接続を行うことでデータの取得を開始する．1 つのメタ情報に対して，複数のノード情報が得られた場合は，5.3 節で述べたアルゴリズムを基にネットワーク距離を計算し，接続するノードを選択するが，同じようなネットワーク距離を持つノードが複数存在する場合は，それぞれのノードに接続を行うことで，多重ダウンロードを行う．ネットワーク距離計算の詳細は 6.3.1 項で述べる．

## 6.3 モジュール

図 6.1 に LinkPoint のモジュールを示す．

各モジュールはすべて個別のスレッドとして動作しており，それぞれデータベースと GUI に密接な関わりを持っている．図 6.1 では，各モジュールの主要な動きのみに着目し，データベースと GUI のつながりについては簡略化した．また，図 6.1 には示していないが，LinkPoint の機能や動作を設定するためのモジュールが別に存在する．詳しくは 6.3.2 項で述べる．

本節では，LinkPoint が実装しているネットワーク距離計算と，ユーザ・インタフェースについて説明を行う．

### 6.3.1 ネットワーク距離計算

ネットワーク距離計算は，データベースからノードリストを取り出す，あるいは格納する際に行う．また，ネットワーク距離計算メソッドは Connect，Listen，Download/Upload モジュールにそれぞれ組み込まれている．ネットワーク距離計算部分を単一のモジュールにするのではなく，各モジュールに組み込んでいるのは，ネットワーク距離の優先度が必ずしもすべてのモジュールの要求を満たすわけではないためである．例えば，Connect，

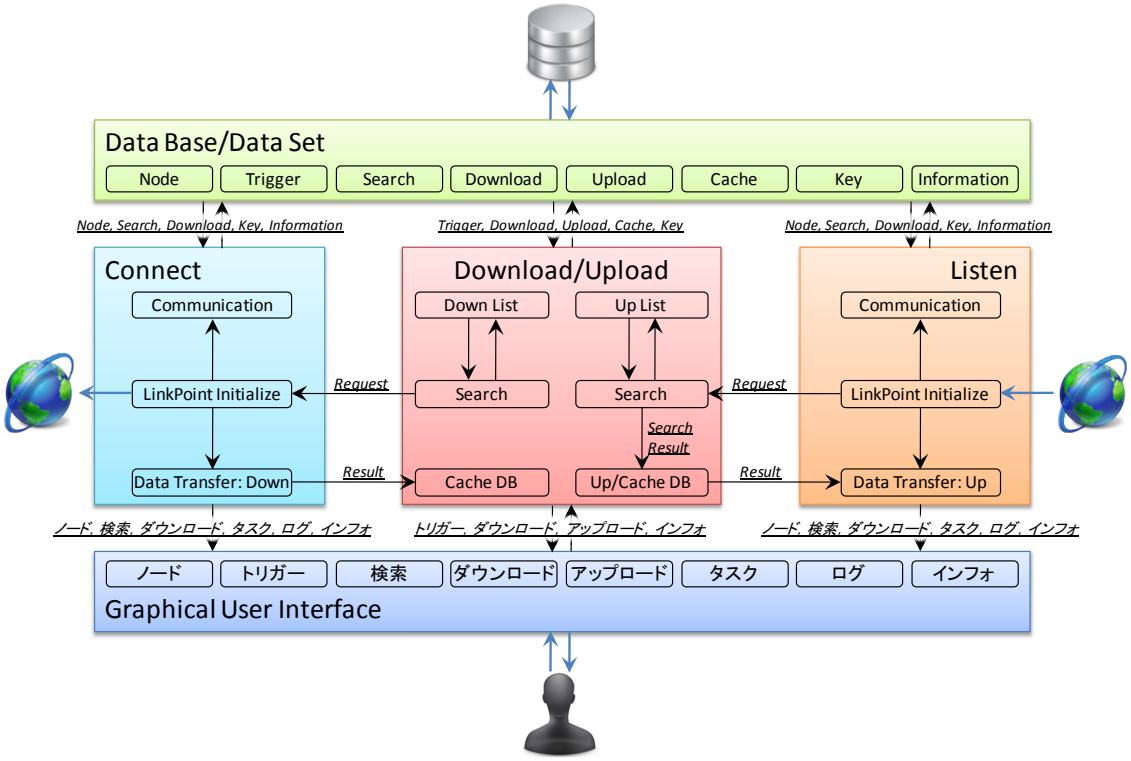



図 6.1: LinkPoint の各モジュール

Listen モジュールの場合には，データ転送ほどの大量のトラフィックを発生させないため，ノード間のネットワーク距離の他に，接続する計算機の性能や回線速度，参加するクラスタなどを考慮することで，ユーザが必要とする情報を持っているノードが近接になるよう P2P ネットワークを構成することができる．一方で，Download/Upload モジュールの場合には，Connect，Listen モジュールと比べて遙かに大きなトラフィックを発生させるために，ネットワーク距離のみを考慮すれば良く，他の要素の計算は不要となる．このように，モジュールの用途に合わせてノードの優先度の要求が変わるため，ネットワーク距離計算は，各モジュールに組み込む形にした．ソースコード 6.1 にネットワーク距離計算を行う関数のソースコードを示す．

ソースコード 6.1: ネットワーク距離計算

```
1  const int DOMAIN_WEIGHT = 64;
2
3  public int HybridFQDNLongestMatch(string sMyFQDN, string sRemoteFQDN)
4  {
5      int iPriority = 0;
6
7      string[] aosSplitMyFQDN = sMyFQDN.Split('.');
8      string[] aosSplitRemoteFQDN = sRemoteFQDN.Split('.');
9
10     // Longest Match
11     int i, j;
12     for (i = aosSplitMyFQDN.Length - 1, j = aosSplitRemoteFQDN.Length - 1; i > 0 ||
           j > 0; i--, j--)
```

```
13      {
14          if (aosSplitMyFQDN[i] != aosSplitRemoteFQDN[j])
15          {
16              break;
17          }
18
19          iPriority += DOMAIN_WEIGHT;
20      }
21
22      // Levenshtein Distance
23      if (i == 0 && j == 0 && aosSplitMyFQDN[0].Length == aosMyFQDN.IndexOf('.')
            && aosSplitRemoteFQDN[0].Length == aosRemoteFQDN.IndexOf('.'))
24      {
25          iPriority += DOMAIN_WEIGHT - ComputeLevenshteinDistance(
                aosSplitMyFQDN[0], aosSplitRemoteFQDN[0]);
26      }
27
28      return iPriority;
29  }
```

ネットワーク計算モジュールは，引数として自ノードの FQDN と接続先ノードの FQDN の文字列を受け取る．そして，それらを各ドメインレベルごとに解析するために，‘ . ’で文字列を分け，レベルが一致するごとに優先度として 64 を加算する．途中のドメインで相違が発生した場合にはそこで関数を抜けて，接続先ノードの接続優先度として iPriority を返す．比較がホスト名まで達した場合には，ドメインレベルの最長一致比較を行うのではなく，5.3.6 項で述べたレーベンシュタイン距離を適用する．レーベンシュタイン距離計算関数のソースコードをソースコード 6.2 に示す．

ソースコード 6.2: レーベンシュタイン距離計算

```
1   public int ComputeLevenshteinDistance(string s, string t)
2   {
3       int n = s.Length;
4       int m = t.Length;
5       int[,] d = new int[n + 1, m + 1];
6
7       if (n == 0)
8       {
9           return m;
10      }
11
12      if (m == 0)
13      {
14          return n;
15      }
16
17      for (int i = 0; i <= n; d[i, 0] = i++)
18      {
19      }
20
21      for (int j = 0; j <= m; d[0, j] = j++)
22      {
23      }
24
25      for (int i = 1; i <= n; i++)
```

```
26      {
27          for (int j = 1; j <= m; j++)
28          {
29              int cost = (t[j − 1] == s[i − 1]) ? 0 : 1;
30
31              // Math.Min は 2 つの引数のうち，小さい方の値を返すメソッドである．
32              d[i, j] = Math.Min(Math.Min(d[i − 1, j] + 1, d[i, j − 1] + 1), d[i − 1, j − 1] +
                    cost);
33          }
34      }
35
36      return d[n, m];
37  }
```

ホスト名については，64 からレーベンシュタイン距離を引いた値を求め，ドメインレベルの最長一致によって求められた優先度に足すことで，最終的な優先度を求める．ドメインレベルの最長一致やレーベンシュタイン距離の優先度に関して，64 という値を用いたのは，FQDN の仕様で，‘ ．’で区切られる文字列の最大長が 63 文字と定められているためであり，レーベンシュタイン距離による優先度がドメインレベルの最長一致で求められた優先度に影響を与えないよう，64 という値を設定した．

ネットワーク距離計算はドメインレベルの最長一致とレーベンシュタイン距離という 2 つのパラメータを用いることで，より細かい粒度で FQDN の比較を可能にしており，特にデータ転送では，接続先のノードを選ぶ際に，自分の近接ノードから接続を行うようになっている．

### 6.3.2 ユーザ・インタフェース

LinkPoint のユーザ・インタフェースモジュールは大きく 2 つに分けることが可能である．1 つは P2P システムにアクセスし，そのサービスを実際に利用するためのインタフェース（メインインタフェース）である．そして，もう 1 つは P2P システムにアクセスする際の設定や，動作を決定するためのインタフェース（サブインタフェース）である．

#### メインインタフェース

メインインタフェースを図 6.2 に示す．

メインインタフェースは，上下 2 つの部位で構成されている．上部は P2P システムの主要な機能である検索に使用されるキーワードボックスが配置されており，P2P システムにより検索されているキーワードと，接続の状況が一目できるようになっている．また，下部は P2P システムのサービスを利用するためのタブが並んでいる．タブの機能の詳細については，付録 A.1 節で述べる．

#### サブインタフェース

サブインタフェースを図 6.3 に示す．

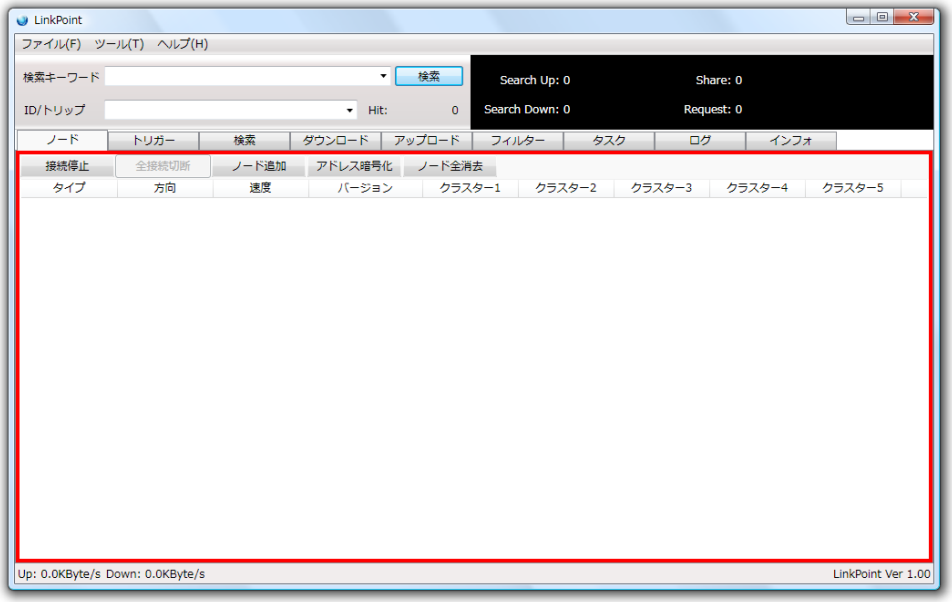



図 6.2: LinkPoint メインインタフェース（ノード）

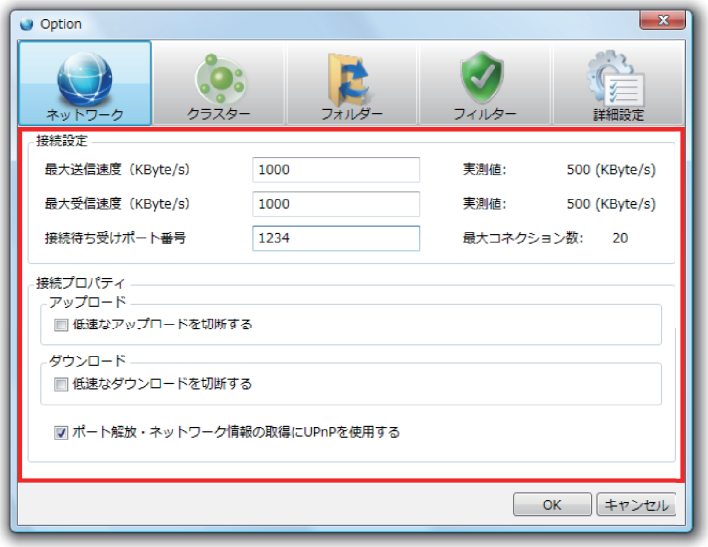



図 6.3: LinkPoint サブインタフェース（ネットワーク）

サブインタフェースでは，主要な 5 つの項目を設定することが可能となっており，上部に並ぶ 5 つのボタンでそれぞれ画面を切り替える．下部には，各ボタンに対応した設定が表示される．サブインタフェースの機能の詳細については，付録 A.2 節で述べる．

## 6.4 まとめ

本章ではP2P システムである LinkPoint の開発について述べた．LinkPoint は WPF を用いた柔軟性の高いユーザ・インタフェースと 5.3.2 項で述べたネットワーク距離計算機構を備え，ネットワーク負荷軽減のためにDNS による近接性を取り入れた P2P システムとなっている．次章で定量的・定性的な評価をすることで，その有効性について考察する．

# 第7章　評価

本章では6章で実装したLinkPointにより推定したネットワーク距離が，Hop Countを指標としたネットワーク距離指標と比べて，同等の機能があるかを検証する．ネットワーク距離の指標としてRTTではなく，Hop Countを用いたのは，本論文の目的であるトラフィックの影響範囲の集約において，Hop Countの方が影響範囲を表現するのに適しているためである．

ネットワーク距離の妥当性を検証するために 実ネットワークを使用した実験とサンプルデータを使用した検証を行った．実ネットワークを使用した実験では，本アルゴリズムが実ネットワーク上で正常に動作し，本アルゴリズムを実装したソフトウェアが，トラフィックの影響範囲を集約することができているか，データ転送量を取得することで検証した．また，サンプルデータを使った検証では，実際に運用されているインターネット上のノードのサンプルを対象に，本アルゴリズムを適用することで，ドメイン内にトラフィックを集約するようなノード選択ができているかを検証した．

## 7.1　実ネットワークを使用した実験

FQDNを用いたネットワーク距離を実ネットワーク上で計測し，検証を行った．6章で実装したLinkPointを用いて，実ネットワーク上にP2Pシステムを展開し，ネットワーク距離が接続先ノードの選択に影響を与えるかどうかを検証した．実験ではP2Pネットワーク内のすべてのノードに対して相互にダウンロードを実行し，以下の2点について観測を行った．

1. 実装のテスト：データ転送時のノード選択
   P2Pシステム上に取得したいデータが十分にある（P2Pシステム上のほぼすべてのノードが取得したいデータを持っている）場合，ネットワーク距離が自分にとって近いノードからデータを取得するかを検証する．また，同じドメイン内で取得したいデータを持っているノードが複数存在する場合に，それらを検出し，多重ダウンロードによって高速化が行われるかを検証する．

2. アルゴリズムの評価：データ拡散時のノード選択
   P2Pシステム上にデータが拡散される場合の振る舞いを検証する．P2Pシステム上に十分データが拡散していない場合，ネットワーク距離の計算を行ったとしても，データを持っているノードが限定されるため，必ずしも近接のノードを選ぶことが

できない場合がある．しかし，データが拡散するにつれ，ネットワーク距離の計算が有効に作用するようになり，データの転送が近接で収束されるかを検証する．

### 7.1.1 実験概要

ノードをドメインが同じネットワークとドメインが違うネットワークに配置し，各ノード上で LinkPoint を動作させた．そして，この P2P ネットワーク上で，7.1 節で述べた 2 つの実験をそれぞれ行った．その際，ネットワーク距離計算の評価指標として，以下の情報を取得した．

- ノードあたりのデータ転送量
  P2P システム上の各ノードにおけるデータ転送量を計測する．
- ネットワーク距離
  P2P ネットワーク上において，FQDN を用いたネットワーク距離と，下位層トポロジの Hop Count を用いたネットワーク距離を比較し，相関を見る．

また，データ転送時のノード選択実験と，データ拡散時のノード選択実験のそれぞれについて，データを 3 種類用意した．

1. 10MB
   楽曲を mp3[35] などで圧縮した場合のデータ量を想定.
2. 100MB
   音楽 CD を mp3 などで圧縮した場合のデータ量を想定.
3. 1000MB（1GB）
   DVD などのドラマ・映画を H.264[36] などで圧縮した場合のデータ量を想定.

#### データ転送時のノード選択実験

データ転送時のノード選択実験の詳細な実験概要について述べる．

本実験では，P2P システム上において，実験対象のノードを除くすべてのノードが前述した 3 種類の同じデータを持っている状態で行う．実験対象のノードが持つダウンロードリストには自分が持っていないデータが登録されており，実験終了後には，P2P システム上の全ノードが同じ 3 種類のデータを持っている状態になる．

そして，実験対象のノードがデータを取得する際に，P2P システム上のノードの優先度が高いノードが存在した場合，実際に優先度が高いノードからのダウンロードが行われることを確認する．特に，今回は事前にネットワークの構成を traceroute[37] や ping[37] によって測定しているため，優先度によって得られたネットワーク距離の近さが，実ネットワークにおいて，有効に作用するかを正確に把握することが可能である．

また，LinkPoint には，多重ダウンロード機構が備わっており，P2P システム上に優先度が近いノードが複数存在した場合には，それらに接続を行い，複数のノードからダウンロードを行うことが可能である．そのため，本実験においても，優先度の計算によって得られた結果を基に，多重ダウンロード機構が動作するかを検証する．

#### データ拡散時のノード選択実験

データ拡散時のノード選択実験の詳細な実験概要について述べる．

本実験では，P2P システム上において，すべてのノードが前述した 3 種類のデータを個別に保持している状態で行う．各拠点のノードのダウンロードリストには自分が持っていないすべてのデータがランダムな順で登録されており，実験終了後には，全ノードが P2P システム上のすべてのデータを持っている状態になる．

データの拡散の様子を詳細に見るために，ノードには起動順序を設定し，データの拡散を意図的に遅らせている．これは，7.1.2 項で詳しく説明するが，今回の実験で用意する実ネットワークが十分に広大ではないため，同時にすべてのノードを起動させた場合に，短時間でデータ転送が完了してしまい，データの拡散を十分に観察できない可能性があるためである．

ノードの起動順序は，ドメインが同一のネットワークとドメインが異なるネットワークの各ノードを交互に起動させる順序に設定した．また，起動間隔は今回用意した 3 種類のデータのうち，最も容量が多い 1000MB のファイル転送が，500Kbyte/s の転送スピードでほぼ完了する時間として 30 分とした．実験はデータの拡散傾向を観察するために 3 回行った．

### 7.1.2 実験環境

実験に使用したネットワークを図 7.1 に示す．

図 7.1 における，直線とそれに付随する数字は各ノード間の Hop Count を示し，丸数字はデータ拡散時のノード選択実験時におけるノードの起動順序を示す．

実験ネットワークでは，慶應義塾 [38] の湘南藤沢キャンパス（sfc.keio.ac.jp），三田キャンパス（dmc.keio.ac.jp），矢上キャンパス（ics.keio.ac.jp）にそれぞれノードを設置し，keio.ac.jp ドメインのグループを構成した．さらに，keio.ac.jp 以外のドメインとして，それぞれ bbexcite.jp[39]，bbtec.net[40]，ocn.ne.jp[41] を用意した．

実験は，すべての拠点ノードに対して，データ転送時のノード選択実験とデータ拡散時のノード選択実験をそれぞれを行った．また，すべての実験において，各ノードはネットワーク中の他のノード情報を予め保持しており，P2P ネットワーク全体の構成が収束した状態から実験を開始した．

表 7.1 に実験環境における FQDN の優先度を計算した結果を示す．

本実験では，表 7.1 において，同じドメイン（右下方）間でのデータ転送に着目する．

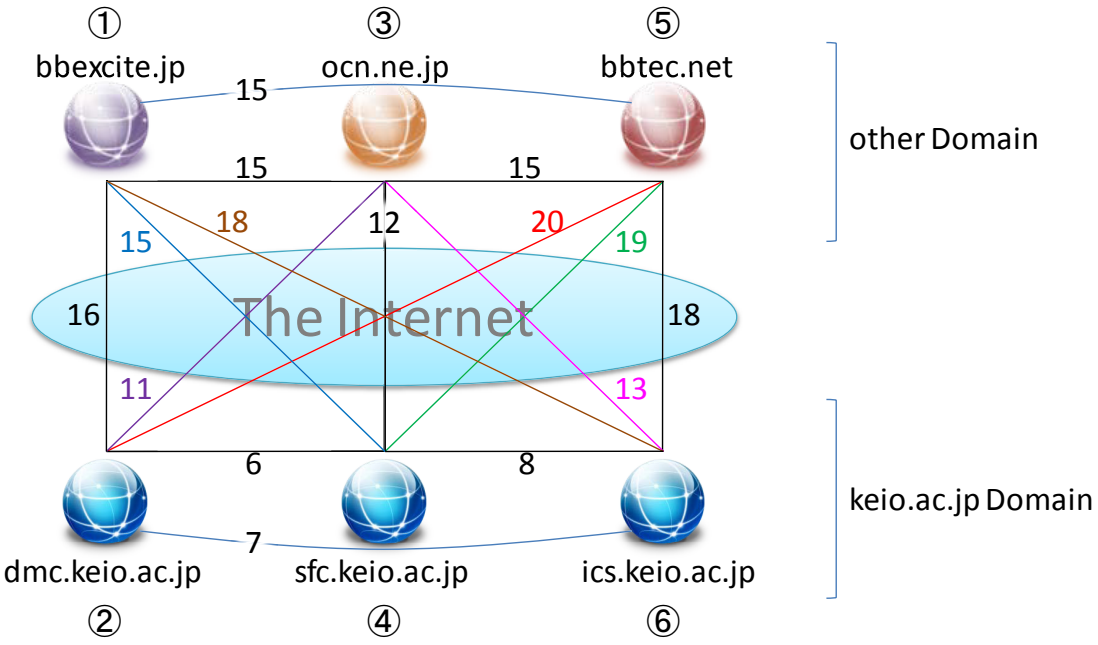



図 7.1: 実験ネットワーク構成

表 7.1: 各拠点間の FQDN による優先度

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 64 | 0 | 64 | 64 | 64 |
| ocn.ne.jp | 64 | - | 0 | 64 | 64 | 64 |
| bbtec.net | 0 | 0 | - | 0 | 0 | 0 |
| dmc.keio.ac.jp | 64 | 64 | 0 | - | 192 | 192 |
| sfc.keio.ac.jp | 64 | 64 | 0 | 192 | - | 192 |
| ics.keio.ac.jp | 64 | 64 | 0 | 192 | 192 | - |

### 7.1.3 実験結果

#### データ転送時のノード選択実験

各拠点のノードが受信したデータ量を図 7.2，図 7.3，図 7.4，図 7.5，図 7.6 と図 7.7 に示す．

また，各データ量における各拠点ノードの受信データ量を表 7.2，表 7.3 と表 7.4 に示す．

図 7.2，図 7.3 と図 7.4 では，ドメインが異なる拠点間の通信が行われており，ほぼすべての拠点からデータを取得しているため，Hop Count の大小を考慮しないデータの転送が発生している．しかし，同一ドメインの拠点間の通信を行う場合には，図 7.5，図 7.6 と図 7.7 に示されているように，Hop Count の小さな拠点（同一ドメイン内の他の拠点）からデータを積極的に転送することで，Hop Count の大きな拠点（異なるドメインの拠点）からのデータ転送量は極めて小さくなっている．

これらの結果から，通常のデータを転送する際に，ネットワーク距離として FQDN に

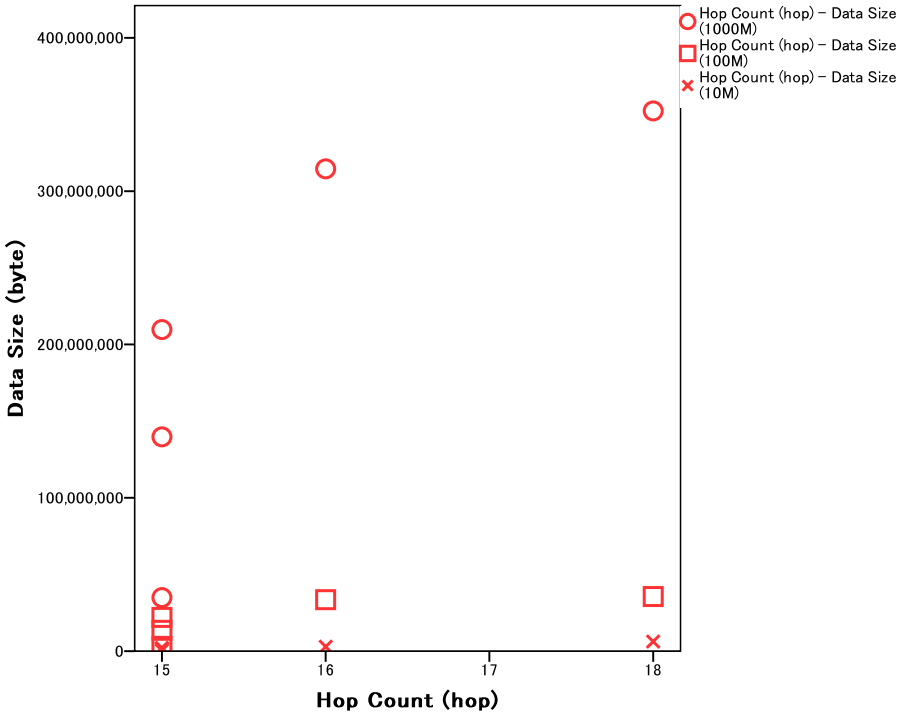



図 7.2: データ転送時に bbexcite.jp ドメインに所属するノードが受信したデータ量

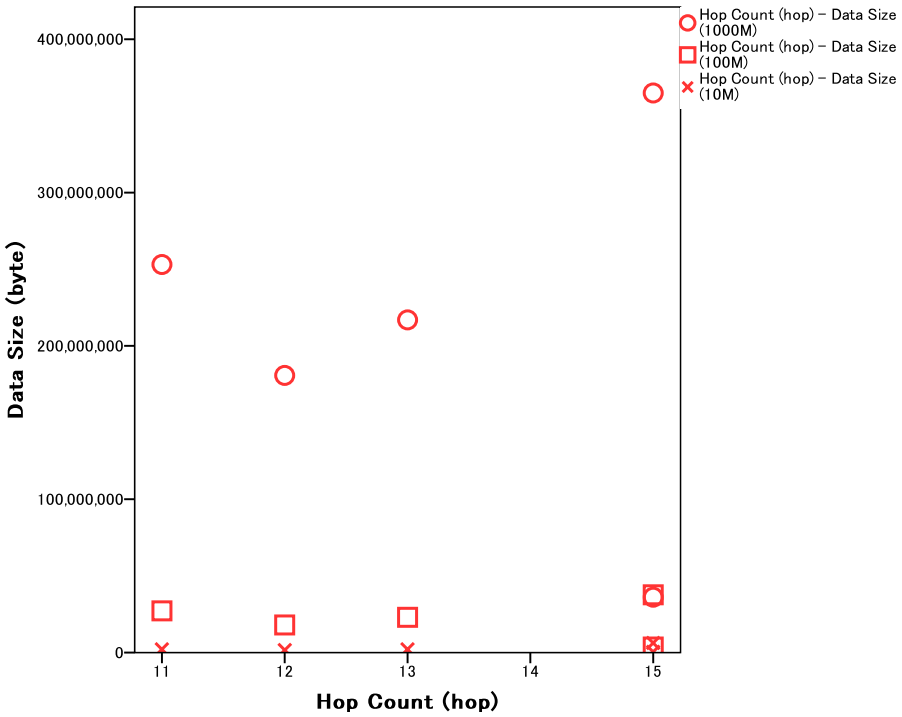



図 7.3: データ転送時に ocn.ne.jp ドメインに所属するノードが受信したデータ量

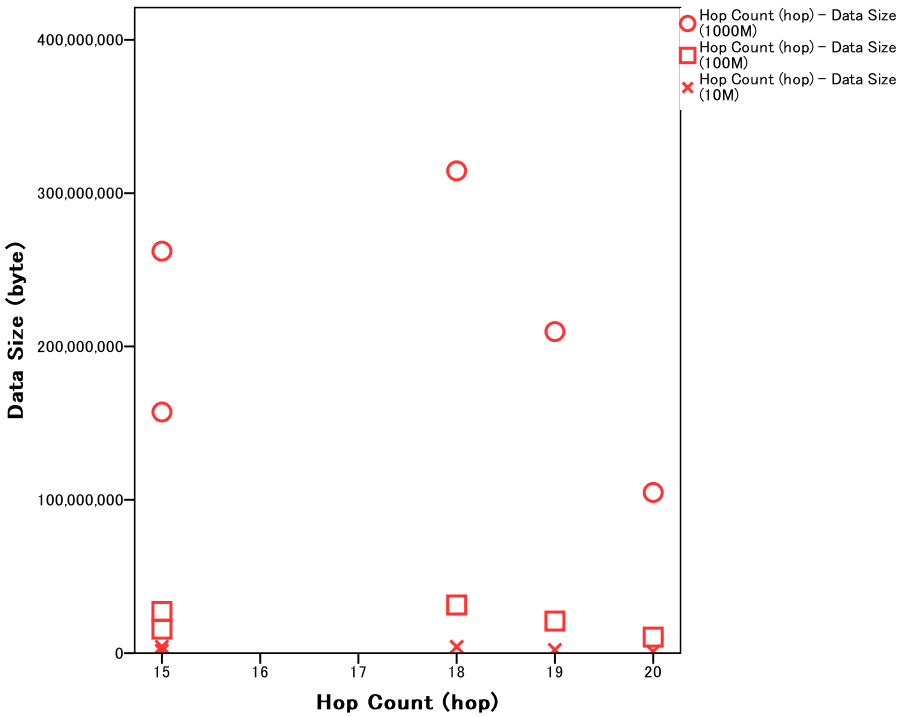



図 7.4: データ転送時に bbtec.net ドメインに所属するノードが受信したデータ量

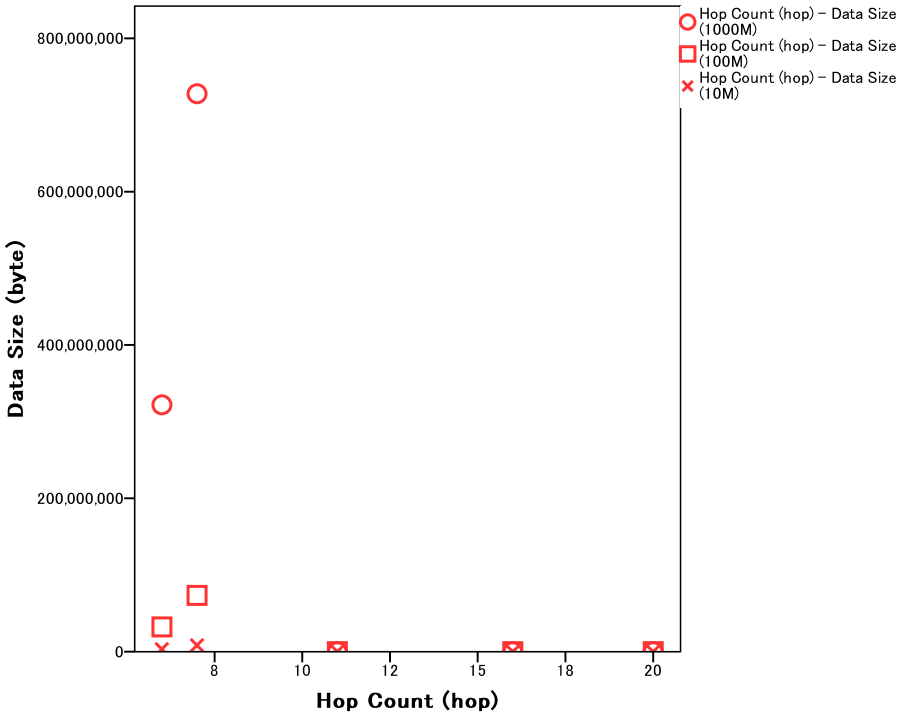



図 7.5: データ転送時に dmc.keio.ac.jp ドメインに所属するノードが受信したデータ量

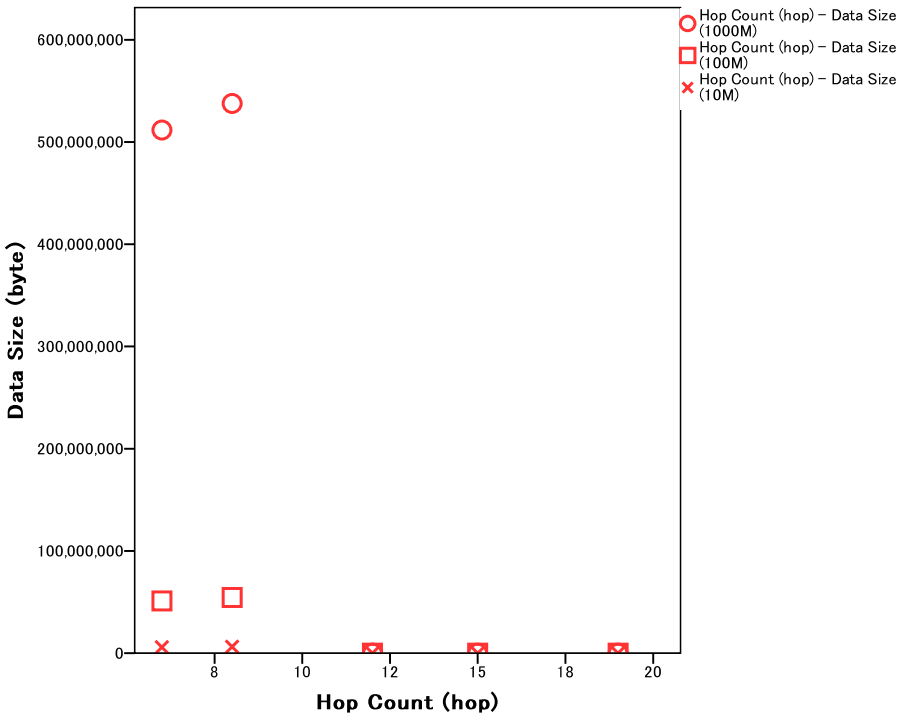



図 7.6: データ転送時に sfc.keio.ac.jp ドメインに所属するノードが受信したデータ量

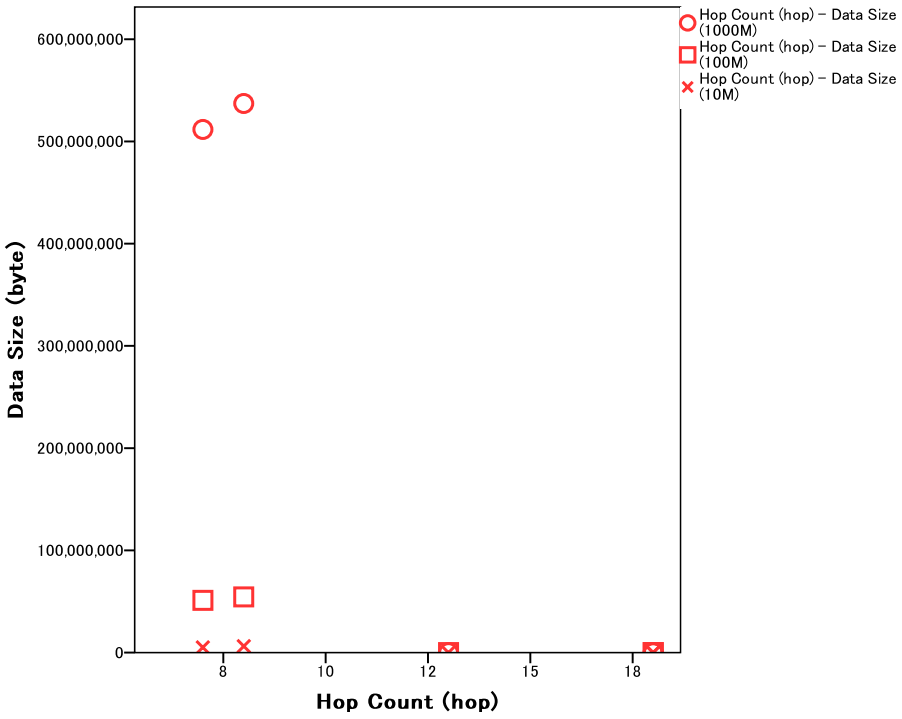



図 7.7: データ転送時に ics.keio.ac.jp ドメインに所属するノードが受信したデータ量

表 7.2: 各ノードの受信データ量（10MB）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 1,310,720 | 491,520 | 2,949,120 | 2,097,152 | 6,291,456 |
| ocn.ne.jp | 6,291,456 | - | 502,734 | 2,097,152 | 1,675,780 | 2,097,152 |
| bbtec.net | 4,194,304 | 1,572,864 | - | 1,048,576 | 2,097,152 | 4,194,304 |
| dmc.keio.ac.jp | 0 | 0 | 0 | - | 3,226,384 | 8,388,608 |
| sfc.keio.ac.jp | 0 | 0 | 0 | 5,872,025 | - | 6,291,456 |
| ics.keio.ac.jp | 0 | 0 | 0 | 5,115,000 | 6,291,456 | - |

表 7.3: 各ノードの受信データ量（100MB）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 13,631,488 | 3,145,728 | 33,554,432 | 22,020,096 | 35,651,584 |
| ocn.ne.jp | 37,748,736 | - | 3,615,780 | 27,262,976 | 18,078,900 | 23,068,672 |
| bbtec.net | 27,262,976 | 15,728,640 | - | 10,485,760 | 20,971,520 | 31,457,280 |
| dmc.keio.ac.jp | 0 | 0 | 0 | - | 32,263,876 | 73,400,320 |
| sfc.keio.ac.jp | 0 | 0 | 0 | 51,084,464 | - | 54,525,952 |
| ics.keio.ac.jp | 0 | 0 | 0 | 51,150,040 | 54,525,952 | - |

よる優先度を取り入れることで，データ転送のトラフィックを同一ドメイン内に閉じ込め，影響範囲を集約できることが確認できた．

### データ拡散時のノード選択実験

各拠点のノードが受信したデータ量を図 7.8，図 7.9，図 7.10，図 7.11，図 7.12 と図 7.13 に示す．

また，各実験における各拠点ノードの受信データ量を表 7.5，表 7.6 と表 7.7 に示す．

図 7.8，図 7.9 と図 7.10 では，ドメインが異なる拠点間の通信が行われており，起動順序はそれぞれ bbexcite.jp:1，ocn.ne.jp:3，bbtec.net:5 となっているが，データ転送時のノード選択実験と同様に，ほぼすべての拠点からデータを取得し，Hop Count の大小を考慮しないデータの転送が発生している．

表 7.4: 各ノードの受信データ量（1000MB）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 139,770,136 | 34,942,534 | 314,572,800 | 209,715,200 | 352,321,536 |
| ocn.ne.jp | 364,904,448 | - | 36,157,793 | 253,104,551 | 180,788,965 | 216,946,758 |
| bbtec.net | 262,144,000 | 157,286,400 | - | 104,857,600 | 209,715,200 | 314,572,800 |
| dmc.keio.ac.jp | 0 | 0 | 0 | - | 321,912,832 | 727,711,744 |
| sfc.keio.ac.jp | 0 | 0 | 0 | 511,705,088 | - | 537,731,280 |
| ics.keio.ac.jp | 0 | 0 | 0 | 511,705,088 | 537,075,504 | - |

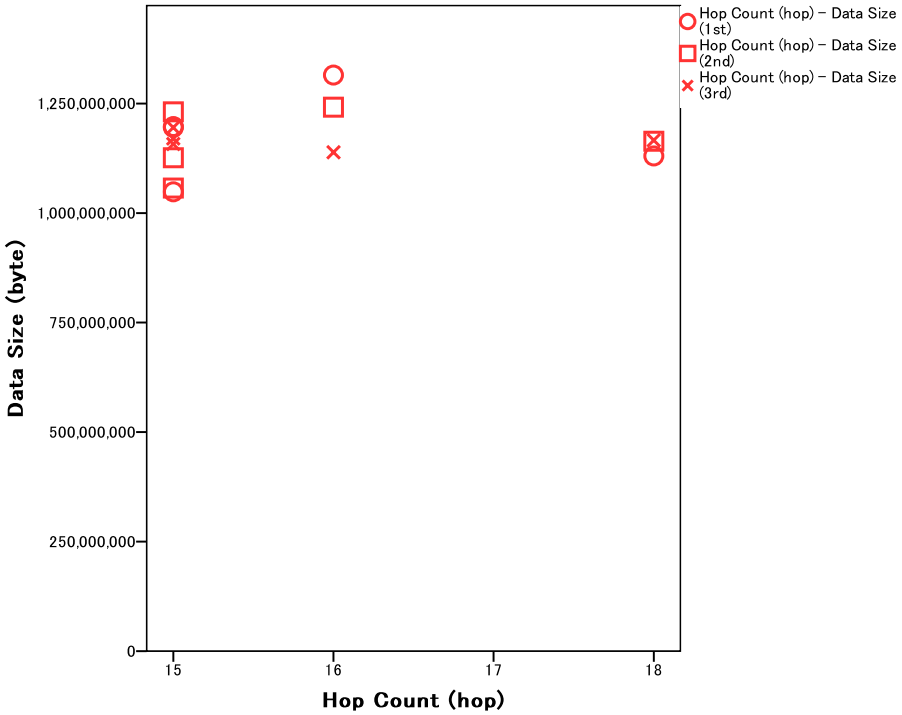



図 7.8: データ拡散時に bbexcite.jp ドメインに所属するノードが受信したデータ量

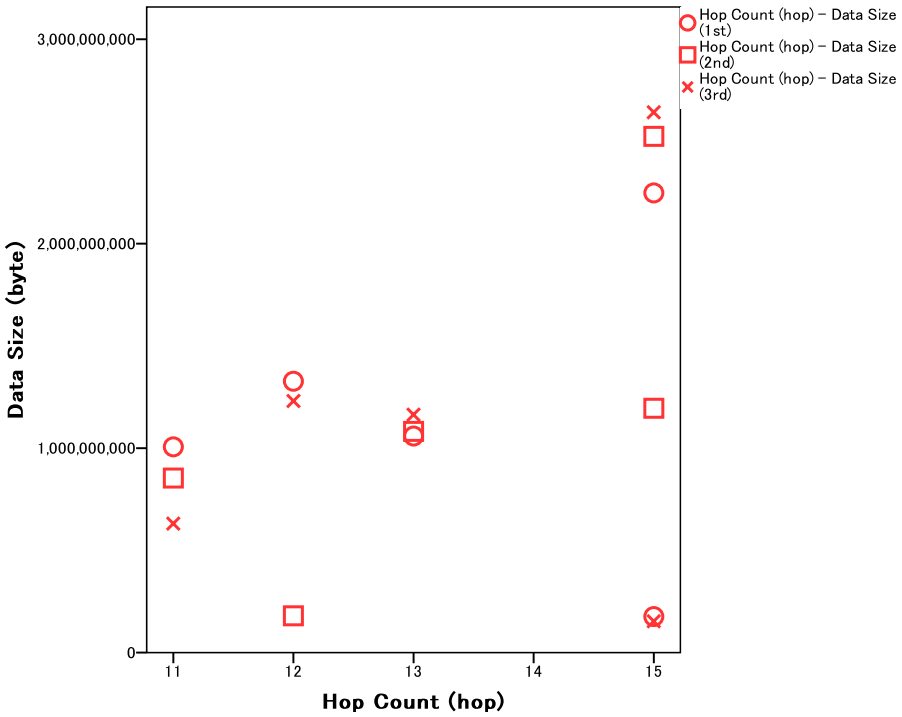



図 7.9: データ拡散時に ocn.ne.jp ドメインに所属するノードが受信したデータ量

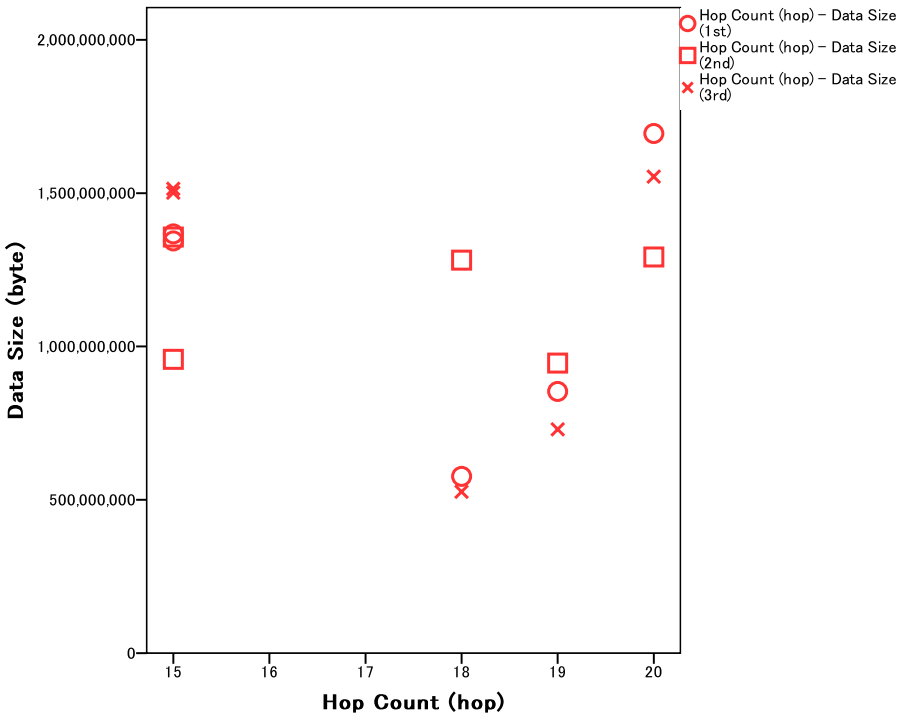



図 7.10: データ拡散時に bbtec.net ドメインに所属するノードが受信したデータ量

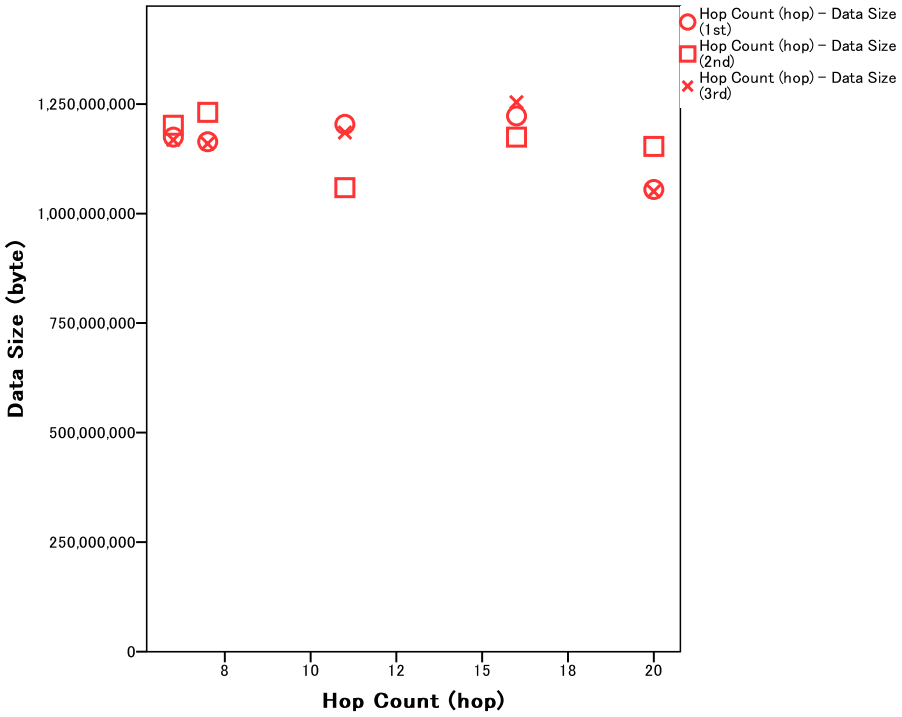



図 7.11: データ拡散時に dmc.keio.ac.jp ドメインに所属するノードが受信したデータ量

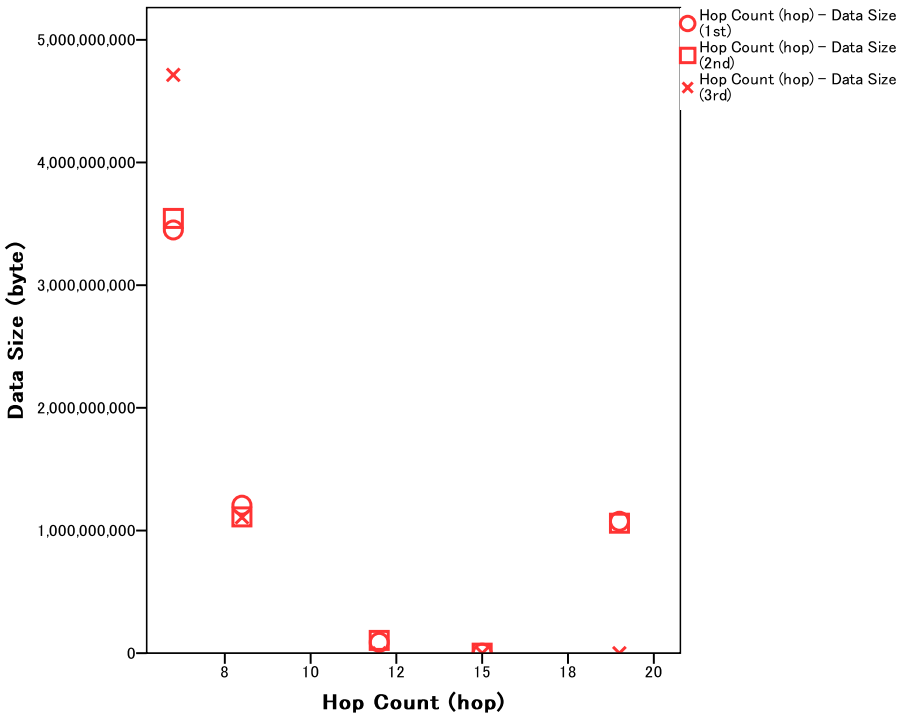



図 7.12: データ拡散時に sfc.keio.ac.jp ドメインに所属するノードが受信したデータ量

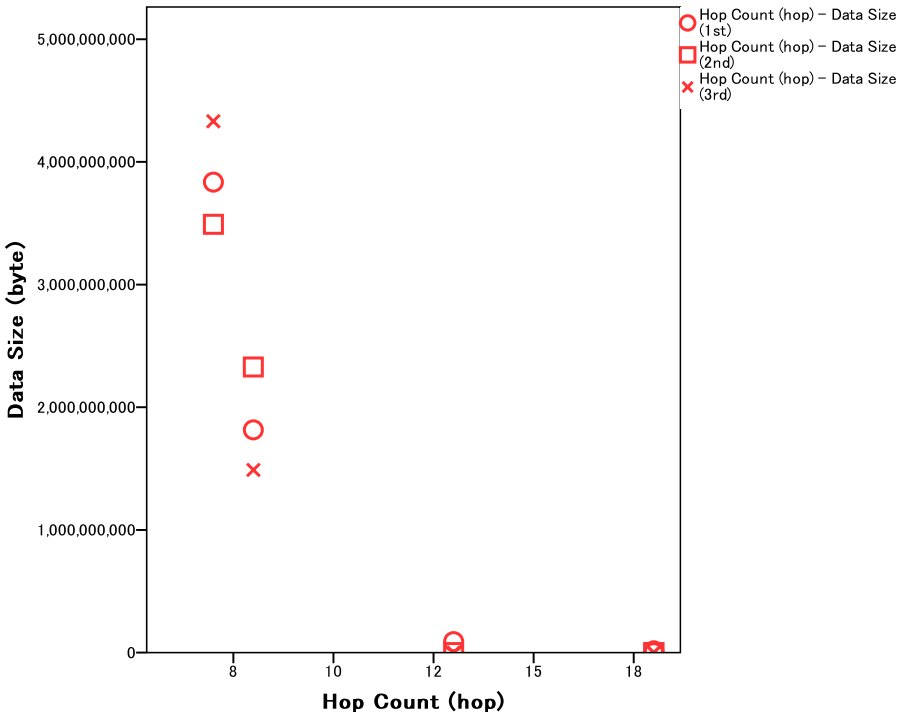



図 7.13: データ拡散時に ics.keio.ac.jp ドメインに所属するノードが受信したデータ量

表 7.5: 各ノードのデータ受信量（1 回目）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 1,195,376,640 | 1,048,576,000 | 1,314,914,304 | 1,197,473,792 | 1,130,364,928 |
| ocn.ne.jp | 2,248,146,944 | - | 176,160,768 | 1,006,632,960 | 1,327,497,216 | 1,059,061,760 |
| bbtec.net | 1,344,274,432 | 1,367,343,104 | - | 1,694,498,816 | 853,540,864 | 576,716,800 |
| dmc.keio.ac.jp | 1,222,639,616 | 1,203,765,248 | 1,054,867,456 | - | 1,174,405,120 | 1,163,919,360 |
| sfc.keio.ac.jp | 0 | 88,080,384 | 1,075,838,976 | 3,449,815,040 | - | 1,205,862,400 |
| ics.keio.ac.jp | 0 | 90,177,536 | 14,680,064 | 3,898,605,568 | 1,816,133,632 | - |

表 7.6: 各ノードのデータ受信量（2 回目）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 1,126,170,624 | 1,056,964,608 | 1,241,513,984 | 1,231,028,224 | 1,163,919,360 |
| ocn.ne.jp | 2,524,971,008 | - | 1,195,376,640 | 853,540,864 | 180,355,072 | 1,082,130,432 |
| bbtec.net | 1,356,857,344 | 958,398,464 | - | 1,291,845,632 | 945,815,552 | 1,281,359,872 |
| dmc.keio.ac.jp | 1,174,405,120 | 1,059,061,760 | 1,153,433,600 | - | 1,201,668,096 | 1,231,028,224 |
| sfc.keio.ac.jp | 0 | 104,857,600 | 1,059,061,760 | 3,544,186,880 | - | 1,111,490,560 |
| ics.keio.ac.jp | 0 | 0 | 0 | 3,491,758,080 | 2,327,838,720 | - |

次に，図 7.11，図 7.12 と図 7.13 では，図 7.11 に関しては，起動順序が早く，P2P ネットワーク内のほぼすべての拠点のノードと通信を行っているため，Hop Count の大小に関してドメインが違う拠点間の通信と同じ傾向となっている．しかし，図 7.12 と図 7.13 では，同一ドメイン内の Hop Count が小さい拠点から積極的にデータを転送し，Hop Count の大きな拠点からのデータの転送量は極めて小さくなっている．

これらの結果から，データが拡散していく場合においても，ネットワーク距離として FQDN による優先度を取り入れることで，データ転送のトラフィックを同一ドメイン内に閉じ込め，影響範囲を集約できることが確認できた．

表 7.7: 各ノードのデータ受信量（3 回目）

| | bbexcite.jp | ocn.ne.jp | bbtec.net | dmc.keio.ac.jp | sfc.keio.ac.jp | ics.keio.ac.jp |
|---|---|---|---|---|---|---|
| bbexcite.jp | - | 1,195,376,640 | 1,157,627,904 | 1,138,753,536 | 1,170,210,816 | 1,166,016,512 |
| ocn.ne.jp | 2,642,411,520 | - | 153,406,668 | 631,242,752 | 1,230,482,964 | 1,163,919,360 |
| bbtec.net | 1,501,560,832 | 1,514,143,744 | - | 1,553,989,632 | 729,808,896 | 526,385,152 |
| dmc.keio.ac.jp | 1,254,096,896 | 1,184,890,880 | 1,050,673,152 | - | 1,159,725,056 | 1,168,113,664 |
| sfc.keio.ac.jp | 0 | 0 | 0 | 4,714,397,696 | - | 1,107,296,256 |
| ics.keio.ac.jp | 0 | 0 | 0 | 4,330,618,880 | 1,488,977,920 | - |

## 7.2 サンプルデータを使用した検証

FQDN を用いたネットワーク距離を実 P2P ネットワーク上のノードに対して計算し，提案アルゴリズムの検証を行った．5.3 節で設計・実装を行ったネットワーク距離計算について，実際に展開されている P2P システム上でノード間の近接性に関わるデータを取得し，それらのデータをサンプルとした検証を行った．対象とした P2P システムは，Winny[42] と Share[43] である．これらの P2P ファイル共有ソフトウェアは，日本で最も利用されており，また，データのアップロード，検索とダウンロードまでを単一のプラットフォーム上で実行できるということから，この 2 つを取り上げた．Winny，Share のノード数・ファイル数は，ACCS の 2008 年 9 月の調査 [44] で，表 7.8 のようになっている．

表 7.8: Winny，Share のノード数・データ数

| | 稼働ノード数 | 予想ノード全数 | 収集ファイル数 | 予想ファイル全数 |
|---|---|---|---|---|
| Winny | 約 18 万台 | 約 19 万台 | 約 530 万件 | 約 600 万件 |
| Share | 約 20 万台 | 約 21〜22 万台 | 約 71 万件 | 約 75〜80 万件 |

### 7.2.1 実験概要

P2P システムである Winny と Share をそれぞれ起動し，各 P2P ネットワーク内のノードとの通信を観察する．また，通信が行われたノードに対して，traceroute と ping を行い，Hop Count を取得する．以下に取得した情報を示す．

- FQDN
  通信相手の IP アドレスから FQDN を取得し，6 章で実装したネットワーク距離計算を行う．
- 実測 Hop Count
  traceroute と ping を用いて，通信相手までの Hop Count を取得する．

### 7.2.2 評価実装

サンプルデータを使った実験には，専用のソフトウェア（*CalculateNetworkDistance*）を実装した．実装環境は，6.1.1 節で述べた環境と同様である．また，CalculateNetworkDistance の実装にはライブラリとして WinPcap[45] ライブラリ（wpcap.dll - libcap 0.9.5 Version 4.0.0.1040）を使用した．

### 動作

CalculateNetworkDistance の動作について述べる．CalculateNetworkDistance の動作は，大きく 3 つに分けられる．

1. 通信先情報の取得
   WinPcap ライブラリを用いて，特定のネットワークインタフェースについて，通信を観察し，通信先の情報（IP アドレス・ポート番号・FQDN など）を取得する．

2. 通信先情報の分析
   本論文で提案したアルゴリズム（ドメインレベル＋レーベンシュタイン距離）を用いて優先度を算出する．また，同時に traceroute と ping を行い，その結果を解析する．さらに，WinPcap ライブラリから出力された，IP Header[46] の TTL 値から予測 Hop Count 値を，TCP の 3-way Handshake に掛かった時間から予測 RTT 値を算出する．

3. 通信先情報の出力
   分析によって得られた通信先の情報を出力する．出力するデータフォーマットは後述する．

### モジュール

図 7.14 に CalculateNetworkDistance のモジュールを示す．

CalculateNetworkDistance のモジュールは主に 3 つにわかれており，優先度の算出，計測と予測である．観測した通信ごとにスレッドを作成して，その中で優先度の算出からデータ出力までを逐次行う．そのため，動作中はノード情報の取得・優先度計算スレッド及び traceroute と ping のアプリケーションがバックグラウンドで走る．また，出力はすべてのデータが一覧できる結果と，デバッグ用に traceroute と ping の解析前の結果がそれぞれ出力される．

### データフォーマット

CalculateNetworkDistance は接続が行われた対向について，以下のデータを csv 形式で出力する．

- IP Address
- Port
- FQDN
- Longest Match Priority
- Levenshtein Distance Priority

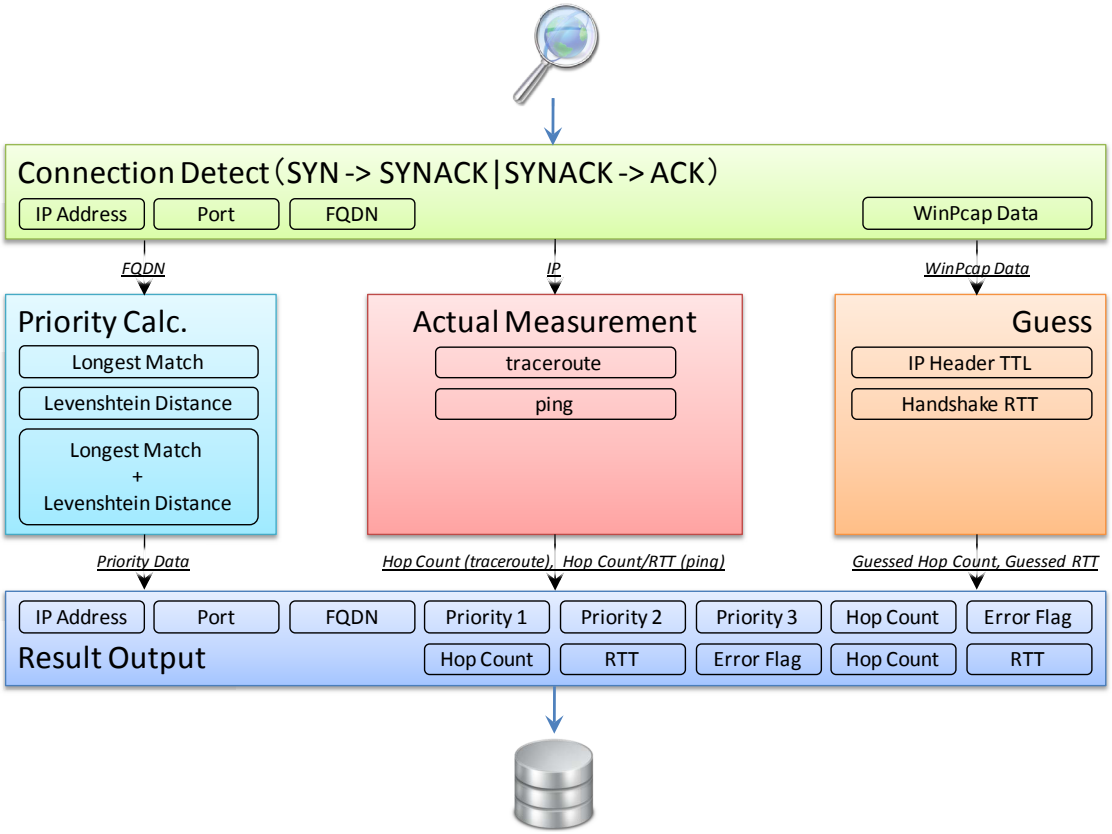



図 7.14: CalculateNetworkDistance の各モジュール

- Longest Match + Levenshtein Distance Priority
- Traceroute Result
- Traceroute Error Flag
- Ping Hop Count
- Ping RTT
- Ping Error Flag
- Guessed RTT (sec/microsec)
- Guessed Hop Count

今回の実験において，当初は実測値が取得できるノードが少量となると予測したため，実測値とは独立して予測値の Hop Count と RTT の取得を試みた．しかし，実際には実測値を取得できるノードが十分に多かったため，予測値については使用しなかった．実測値のデータ数は 7.2.3 項で述べる．

### 7.2.3 OCN 観測実験環境

実験は 1 台の計算機上に，2 つの Virtual Machine を搭載して行った．各 Virtual Machine には NAT によってプライベートアドレス（192.168.1.0/24）を配り，それぞれ Winny, Share を起動させた．表 7.9 に実験 PC の構成を示す．

表 7.9: 実験 PC の構成

| | OS | IP アドレス | アプリケーション |
|---|---|---|---|
| ホスト | Windows Vista Ultimate | 192.168.1.2 | Virtual PC 2007 |
| ゲスト 1 | Windows XP Professional | 192.168.1.3 | Winny v2.0b7.1 |
| ゲスト 2 | Windows XP Professional | 192.168.1.4 | Share Ver1.0 EX2 |

実験は神奈川県の OCN ネットワーク上のノードから行い，期間は，2008 年 12 月 11 日から 12 月 17 日まで 168 時間連続して観測を行った．

Winny と Share をあわせて約 110,000 ノードに接続が行われたが，データとして用いたのは，traceroute と ping が正常に計測できた約 35,000 ノードである．

### 7.2.4 OCN 観測実験結果

すべてのノードについて Hop Count と FQDN 優先度を比較した結果を図 7.15 に，レーベンシュタイン距離が適用されたノード（優先度 256 以上）について Hop Count と優先度を比較した結果を図 7.16 に示す．

図 7.15 と図 7.16 の各要素について説明する．$x$ 軸は，本アルゴリズムによって求めた FQDN による優先度である．範囲は 0〜318 で，優先度は図の右に行くほど高くなる．本アルゴリズムの場合には 6.3.1 項で述べたように，優先度の値は単調増加しないため，$x$ 軸には間欠が発生する．また，$y$ 軸は，実測した Hop Count である．HopCount の分布は，4〜29 で，Hop Count は図の下に行くほど短い．

図 7.15 と図 7.16 より，ネットワーク距離である Hop Count が近い場合に，高い頻度で優先度が高くなっているため，FQDN による優先度は有効であると考えられる．

また，ドメインを解析して計算する本アルゴリズムの構造上，優先度には間欠が発生するが，図 7.15 の優先度が高い部分では，間欠の粒度が細かくなり，優先度の低い部分では見られなかった間欠が観測されている．これは，ドメインレベルによる最長一致の計算を終えて，レーベンシュタイン距離による文字列の類似度が計算された結果によって生じた間欠である．ただ，この間欠は，5.3.5 項で述べた通り，最長一致アルゴリズムの場合でも，判断が可能である．そのため，レーベンシュタイン距離による効果を確認するために，7.2.5 項では，別の FQDN サンプルデータを用いた結果を示す．

次に，実測した Hop Count の分布を図 7.17 に，優先度の分布を図 7.18 に示す．

図 7.17 と図 7.18 の各要素について説明する．ヒストグラムとなっており，$x$ 軸がそれぞれ Hop Count，FQDN による優先度となっている．また，$y$ 軸はヒストグラムの度数と

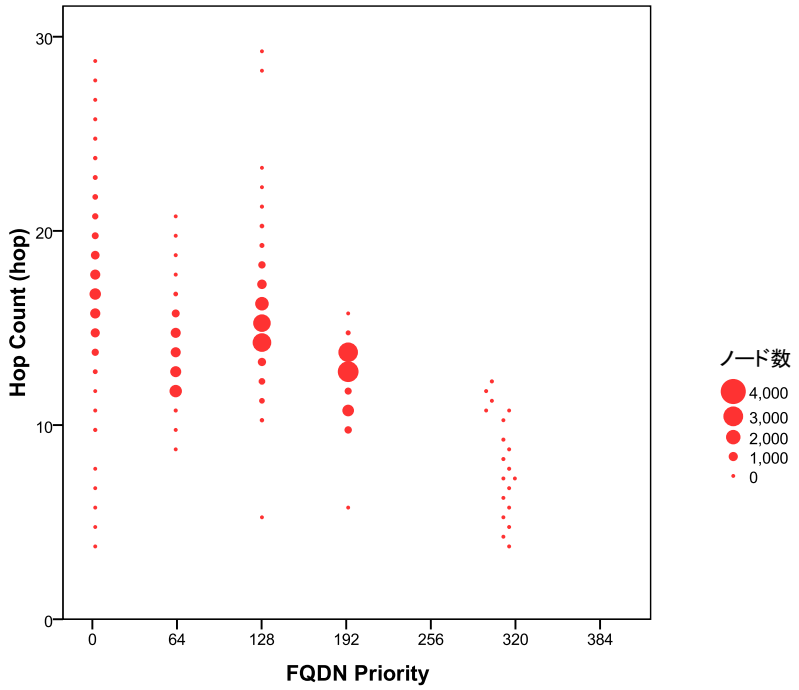



図 7.15: OCN における Hop Count と FQDN による優先度の比較

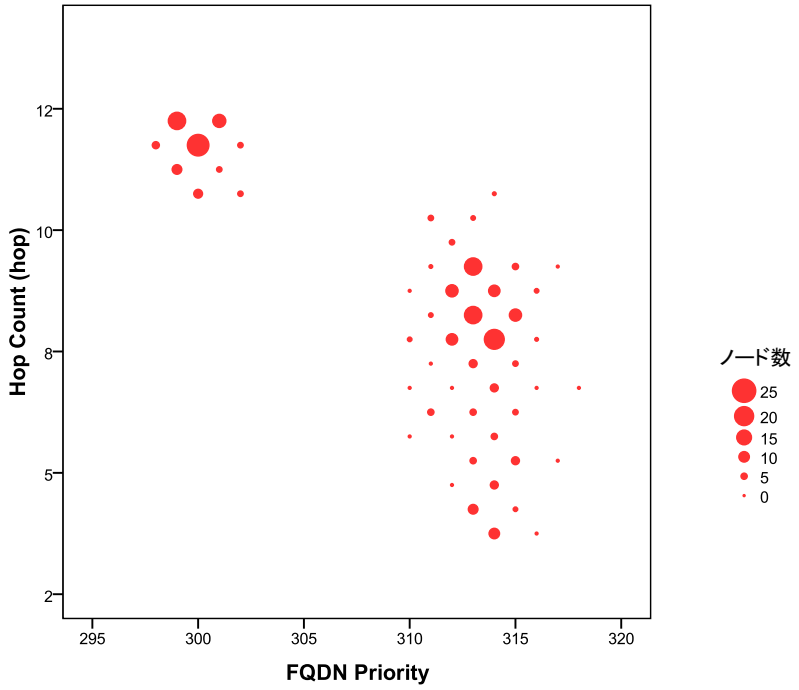



図 7.16: OCN における Hop Count と FQDN による優先度 256 以上のノードの比較

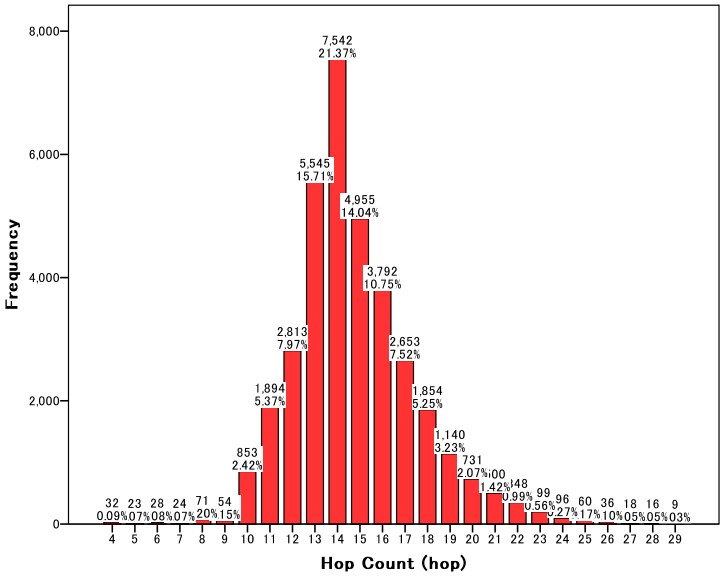



図 7.17: OCN から接続したノードの Hop Count の分布

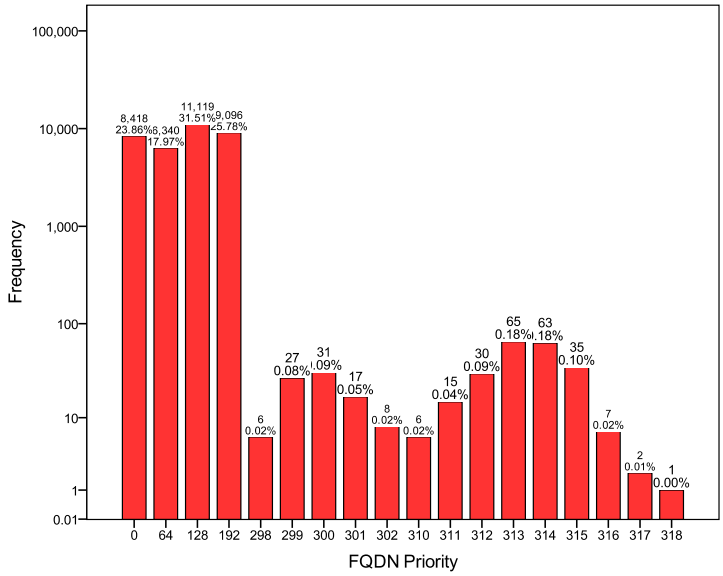



図 7.18: OCN から接続したノードの優先度の分布

なっているが，図 7.18 については，優先度の高いノード数を表現するため，対数の軸を使用している．

図 7.17 より，OCN では Hop Count が 14 のノードが最も多いことが確認できる．そして，Hop Count が小さくなる場合に，ノード数が大幅に減少する傾向にあり，Hop Count が大きくなる場合には，ゆるやかにノード数が減少する傾向にある．これは，Hop Count が小さくなるにつれて，管理ドメインの階層が深くなり，ノード数が限られてくるためである．

さらに，図 7.18 より，OCN では，レーベンシュタイン距離を用いた優先度のノードが 300 個前後存在していることが確認できる．そして，同じ管理ドメインを示す第 3 レベルドメインで一致している優先度 192 のノードを合わせると，OCN のノードが全ノードの約 1/4 を占めていることが確認できる．

#### OCN における Hop Count と FQDN による優先度の一致割合

優先度が高い場合に，ネットワーク距離が近いかを計算するために，約 35,000 あるエントリから任意の 2 つのエントリを抜き出し，優先度が高いと判断したエントリの Hop Count が小さかった場合の割合を全エントリに対して計算した．その際，同じ優先度を持ったエントリがあった場合には，計算に入れず，別途割合を求めた．図 7.19 に結果を示す．

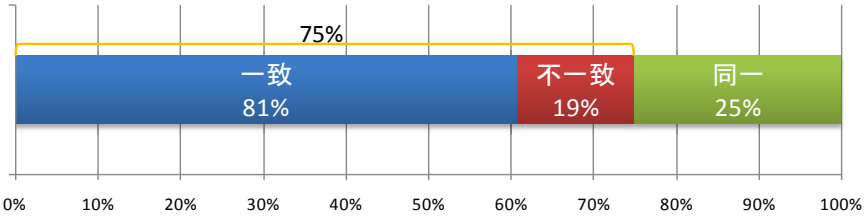



図 7.19: OCN における Hop Count と FQDN による優先度の一致割合

図 7.19 より，FQDN による優先度は Hop Count と比較して，8 割以上同じ結果を導き出すことが可能である．

### 7.2.5 BBTEC 観測実験環境

実験は 1 台の計算機上で，Winny と Share を起動して観測を行った．使用した計算機には，NAT によって 192.168.3.8/24 が配られており，Winny と Share の両方を起動させた．

実験は愛知県の Yahoo!BB（BBTEC）ネットワーク上のノードから行い，期間は，2009 年 1 月 16 日から 1 月 17 日まで 48 時間連続して観測を行った．

データとして用いたのは，traceroute と ping が正常に計測できた 13,750 ノードである．

### 7.2.6 BBTEC 観測実験結果

すべてのノードについて Hop Count と FQDN による優先度を比較した結果を図 7.20 に，レーベンシュタイン距離が適用されたノード（優先度 128 以上）について Hop Count と優先度を比較した結果を図 7.21 に示す．

図 7.20 と図 7.21 の各要素は 7.2.4 項と同様であるが，$x$ 軸の範囲が 0〜188，$y$ 軸の範囲が 3〜29 となっている．

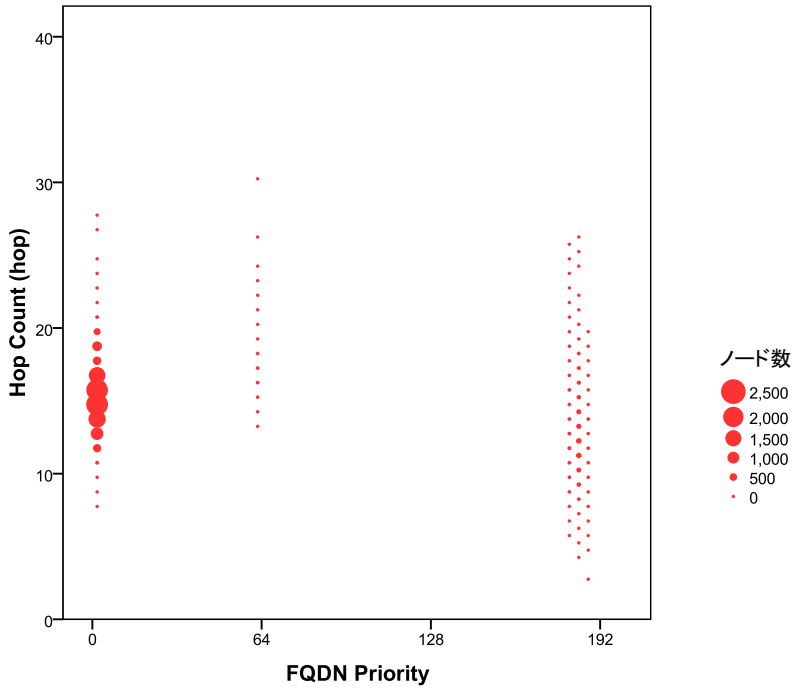



図 7.20: BBTEC における Hop Count と FQDN による優先度の比較

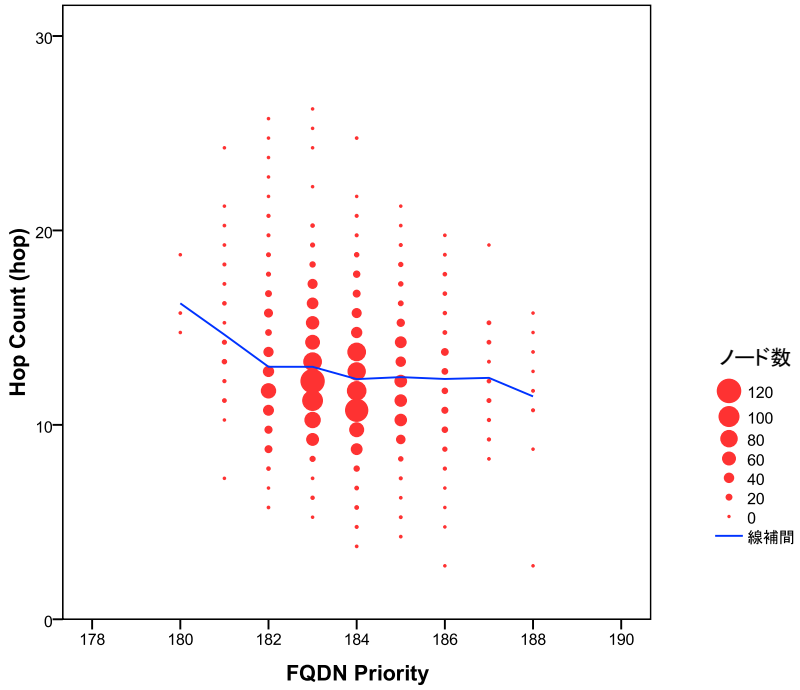



図 7.21: BBTEC における Hop Count と FQDN による優先度 128 以上のノードの比較

図 7.20 と図 7.21 より，ネットワーク的に近い距離の場合に，高い頻度で優先度が高くなっているため，FQDN による優先度は BBTEC においても有効であると言える．

また，7.2.4 項では観測することのできなかったレーベンシュタイン距離の効果を観測することができた．具体的には，図 7.22 において，180 以上の優先度を持つものが，レーベンシュタイン距離計算によって求められたサンプルであるが，優先度が高くなるにつれて Hop Count が減少しているのが観測できる．したがって，レーベンシュタイン距離による文字列の類似度の計算においても，有効性を証明することができた．

次に，実測した Hop Count の分布を図 7.22 に，優先度の分布を図 7.23 に示す．

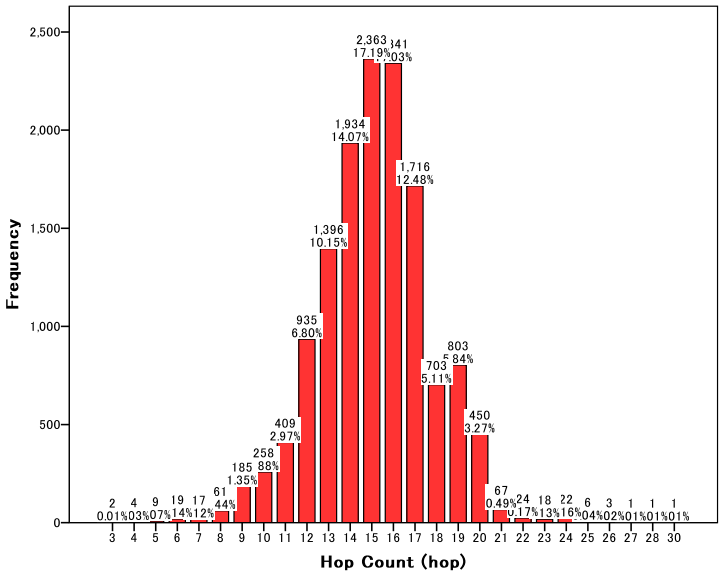



図 7.22: BBTEC から接続したノードの Hop Count の分布

図 7.22 と図 7.23 の各要素は 7.2.4 項と同様である．

図 7.22 より，BBTEC では，Hop Count が 15 と 16 のノードがほぼ同様に多いことが確認できる．そして，OCN の場合と異なり，Hop Count が大きくなる場合に，ノード数が大きく減少しており，Hop Count が小さくなる場合には緩やかにノード数が減少している．これは，BBTEC 内のノードの Hop Count の分布が図 7.21 に示されているように，OCN ほど偏った分布となっていないためである．

さらに，図 7.23 より，BBTEC では，レーベンシュタイン距離を用いた優先度が約 2,400 個存在している．優先度 128 のエントリが存在しないことから．第 3 レベルドメインが一致したエントリにはすべてレーベンシュタイン距離が適用されていることが確認できる．

### BBTEC における Hop Count と FQDN による優先度の一致割合

FQDN による優先度が高い場合に，ネットワーク距離が近いかを計算するために，7.2.4 項と同様の実験を行った．図 7.24 に結果を示す．

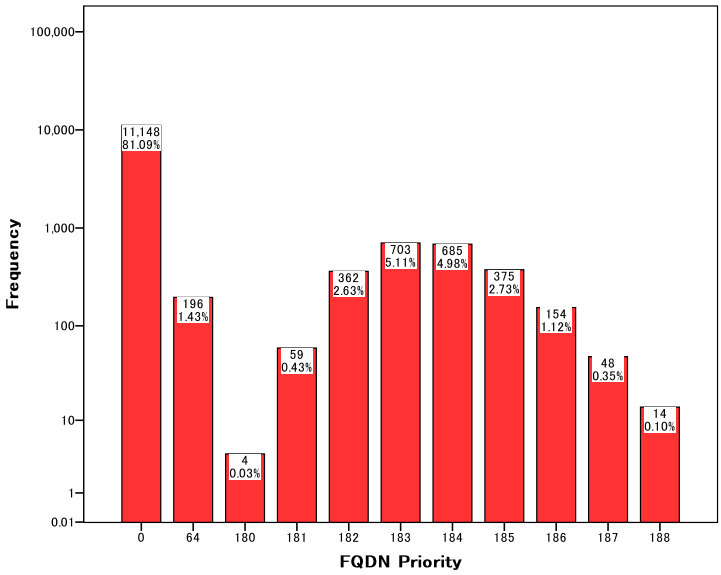



図 7.23: BBTEC から接続したノードの優先度の分布

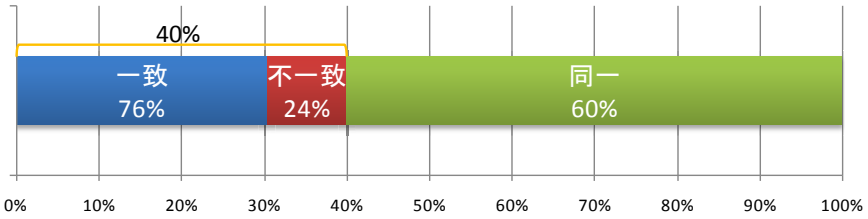



図 7.24: BBTEC における Hop Count と FQDN による優先度の一致割合

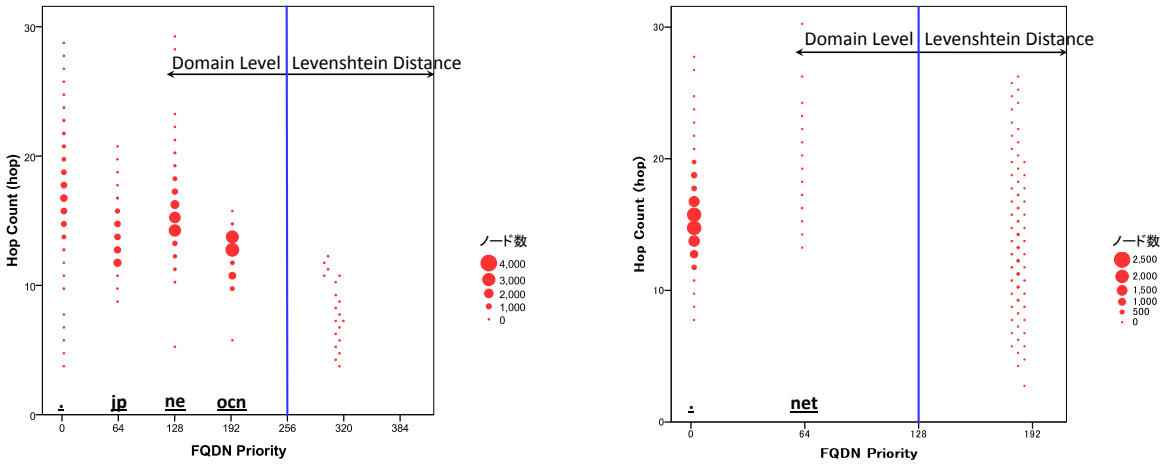
図 7.24 に示すように，FQDN による優先度は Hop Count と比較して，8 割近く同じ結果を導き出すことが可能である．OCN の例より一致率が下がった理由としては，BBTEC の FQDN が階層構造を持っておらず，同じ優先度を持った FQDN が全体の半分以上（60%）存在していることが考えられる．

### 7.2.7 アルゴリズム評価

5.3.5 項で述べた，他の文字列処理アルゴリズムについて，OCN と BBTEC のデータから評価を行う．

#### 本アルゴリズム

本アルゴリズムによって，全ノードの FQDN の優先度を求めた結果を図 7.25(a) と図 7.25(b) に示す．




(a) OCN (b) BBTEC

図 7.25: 本アルゴリズムによる優先度

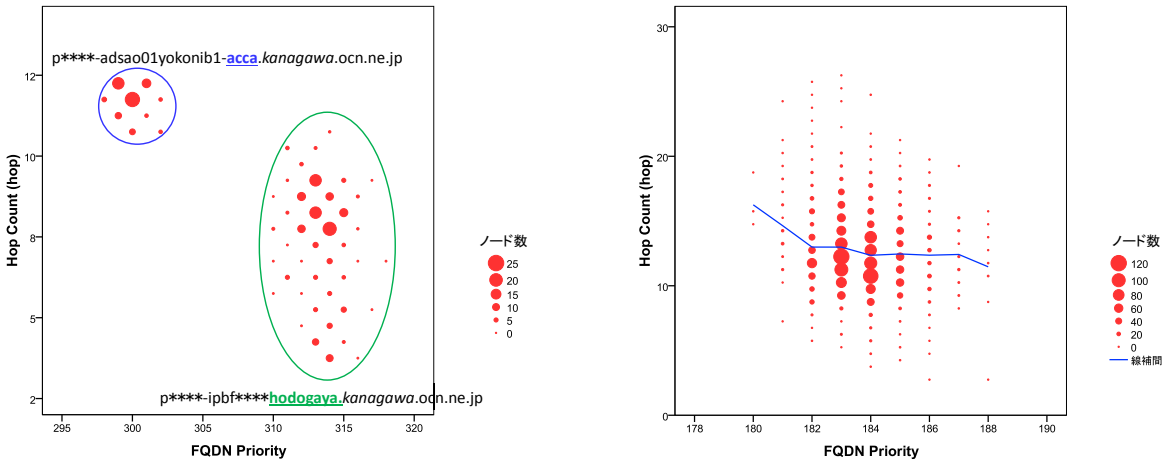
図の各要素について説明する．$x$ 軸は，本アルゴリズムによって求めた優先度である．OCN の範囲が 0〜318，BBTEC の範囲が 0〜188 で，優先度は図の右に行くほど高くなる．本アルゴリズムの場合には 6.3.1 項で述べたように，優先度の値は単調増加しないため，$x$ 軸には間欠が発生する．また，$y$ 軸は，実測値の Hop Count である．範囲は，OCN が 4〜29，BBTEC が 3〜29 で，Hop Count は図の下に行くほど短い．

図 7.25(a) より，OCN では大きな間欠が優先度 192 と優先度 298 以上の間に確認できる．これは，ドメインレベルの判定で 256 となった優先度に，本アルゴリズムの第 2 段階であるレーベンシュタイン距離の優先度が加えられたことによる間欠である．優先度が 256 より小さい値は，本アルゴリズムの優先度計算においてドメインレベルのみを適用したノード，大きい値は，レーベンシュタイン距離を適用したノードとなっている．

また，図 7.25(b) より，BBTEC においても大きな間欠が優先度 64 と優先度 180 の間に確認できる．これは，ドメインレベルの判定で，128 となった優先度に，本アルゴリズムの第 2 段階である，レーベンシュタイン距離の優先度が加えられたことによる間欠である．優先度が 128 より小さい値は，本アルゴリズムの優先度計算においてドメインレベルのみを適用したノード，大きい値は，レーベンシュタイン距離を適用したノードとなっている．

レーベンシュタイン距離を適用したノードに関しては，ドメインレベルだけを適用したノードに比べて分布が細かくなっている．そのため，実際にレーベンシュタイン距離が有効に作用しているか検証するために，レーベンシュタイン距離計算が行われたノードのみに着目して分析を行う．レーベンシュタイン距離を適用したノードのみを対象とした本アルゴリズムの優先度の結果を図 7.26(a) と図 7.26(b) に示す．

本アルゴリズムを採用した場合，OCN では図 7.26(a) より，最上位の分布が 2 つの群にわかれていることが確認できる．これらは，同じ，kanagawa.ocn.ne.jp というドメインを持つノードのうち，保土ヶ谷局収容のノード（優先度最上位）と，そうでないノード（優先度第 2 位）によって発生した分布である．保土ヶ谷局収容のノードの方が優先度が高い




(a) OCN (b) BBTEC

図 7.26: レーベンシュタイン距離を適用したノード

のは，今回 OCN のデータを取得するために用いたノードが保土ヶ谷局収容だったためである．2 つの分布のうち，優先度の低いノード群の方が Hop Count が大きくなっているため，本アルゴリズムはネットワーク距離を正確に検出できていることがわかる．

同様に，BBTEC でも図 7.26(b) より，レーベンシュタイン距離が適用されたノードでは，優先度が高くなるに連れて，Hop Count が小さくなっているため，本アルゴリズムはネットワーク距離を正確に検出できていることがわかる．特に，BBTEC の場合には，FQDN に含まれる IP アドレスの上位ビットがネットワーク ID となっており，ネットワーク的に近いノードは互いに同じ（類似度の高い）ネットワーク ID を持つため，レーベンシュタイン距離を求めることで，近いノードを高い優先度に設定することが可能である．

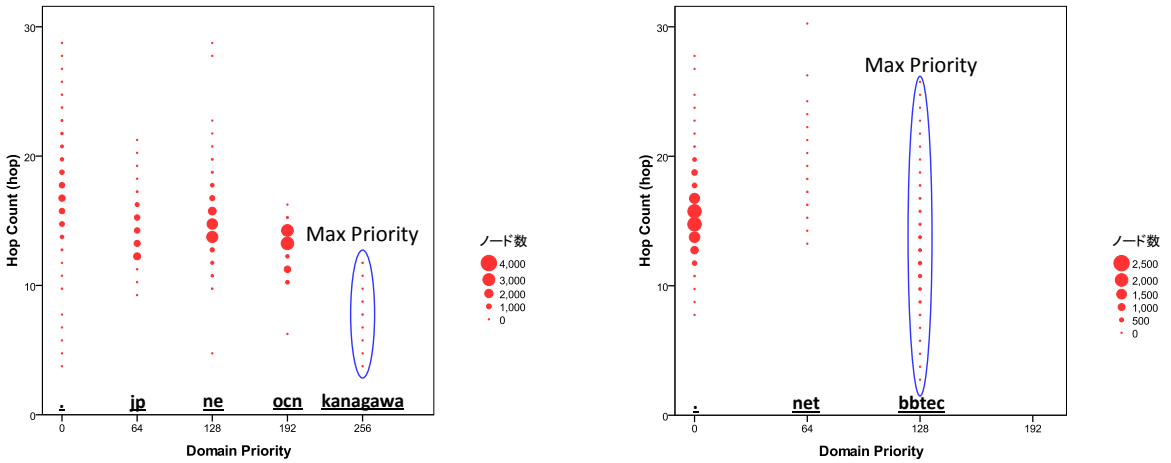
### ドメインアルゴリズム

ドメインアルゴリズムのみによって，すべてのノードの優先度を求めた結果を図 7.27(a) と図 7.27(b) に示す．

図の各要素について説明する．$x$ 軸は，ドメインアルゴリズムのみによって求めた優先度である．OCN の範囲は 0〜256，BBTEC の範囲は 0〜128 で，優先度は図の右に行くほど高くなる．ドメインアルゴリズムの場合にも 6.3.1 項で述べたように，優先度の値は単調増加しないため，$x$ 軸には間欠が発生する．$y$ 軸は，本アルゴリズムの場合と同様である．

図 7.27(a) において，OCN では優先度が高くなるにつれて Hop Count が小さくなり，近接のノード間で通信が完結できることが確認できるが，優先度 192 と優先度 256 の間には Hop Count において大きな差ができている．これは，優先度 192 が第 3 レベルドメインまで一致しているノード群であるのに対して，優先度 256 は第 4 レベルドメインまで一致しているノード群であるためである．

また，図 7.27(b) より，BBTEC においても 64 と 128 の間では優先度が高くなるにつ

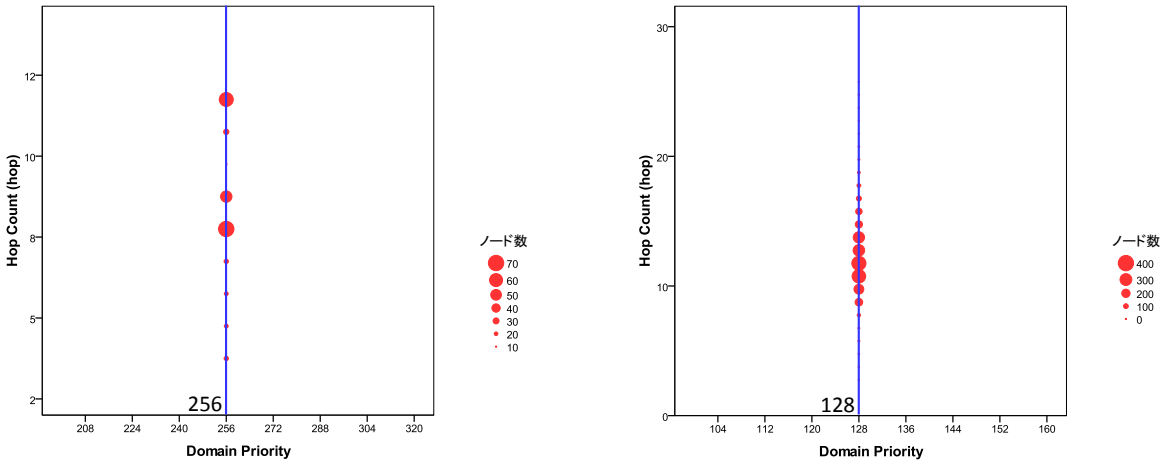



(a) OCN (b) BBTEC

図 7.27: ドメインアルゴリズムによる優先度

れて Hop Count が小さくなり，近接のノード間で通信が完結できることがわかる．しかし，0 と 64 の間には明確な差がなく，Hop Count の分布にも目立った傾向はない．これは，BBTEC が持つ FQDN の上位のドメインが，.bbtec.net となっており，階層構造を持っていないため，.net に該当しないノードがすべて優先度 0 になってしまったためである．

OCN の優先度 256 のノード群，BBTEC の優先度 128 のノード群において，ノードがどのように分布しているかを調べるため，ドメインアルゴリズムの優先度計算において，最上位のスコアを持つノードのみを図 7.28(a) と図 7.28(b) に示す．




(a) OCN（最上位:256） (b) BBTEC（最上位:128）

図 7.28: ドメインアルゴリズムによる優先度が最上位のノード

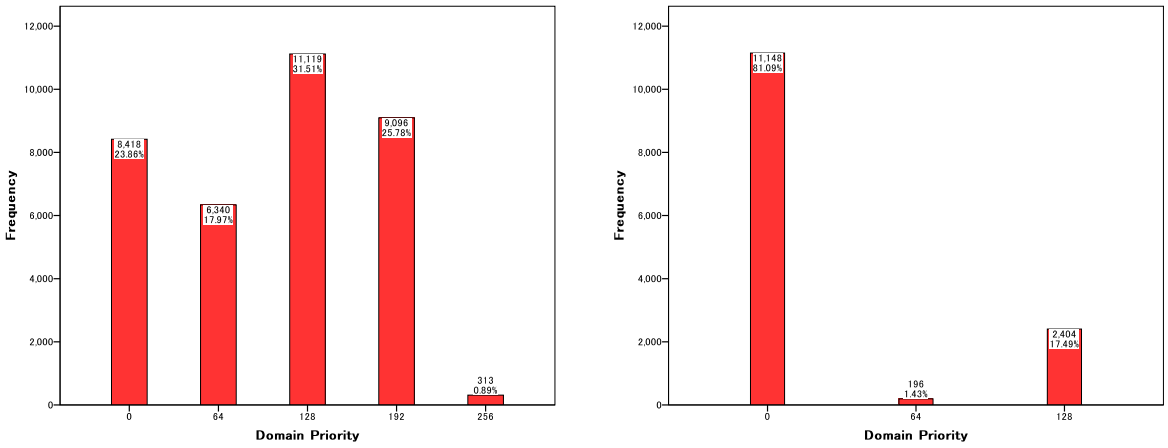
図 7.28(a) より，OCN ではドメインアルゴリズムによって，ドメインのみが判定され，すべてのノードが 256 という値に集約されている．そのため，本アルゴリズムを採用した場合に確認された，それぞれ大小の Hop Count を持つノード群が同一の優先度として判

定されており，ノードの Hop Count の分布により，2 つのノード群が一緒になった結果であることが読み取れる．したがって，優先度という意味では，図 7.26(a) で確認された，2 つのノード群が 1 つの優先度として扱われてしまうため，Hop Count が大きく，ネットワーク距離的に遠いノードに接続する可能性がある．

また，図 7.28(b) より，BBTEC においてもドメインアルゴリズムによって，ドメインのみが判定され，すべてのノードが 128 という値に集約されている．そのため，本アルゴリズムを採用した場合に確認された，複数の Hop Count の異なる分布が観測できなくなっている．特に，BBTEC の場合には，優先度 128 のノード群の中の Hop Count の範囲が 3〜26 であり，小さな Hop Count から大きな Hop Count まで，かなり広い範囲を含んでいるため，Hop Count が大きく，ネットワーク距離的に遠いノードに接続する可能性は OCN よりも高くなっている．

したがって，ネットワーク距離測定の精度という点で，本アルゴリズムが優位である．

図 7.29(a) と図 7.29(b) にドメインアルゴリズムによる優先度の分布を示す．




(a) OCN (b) BBTEC

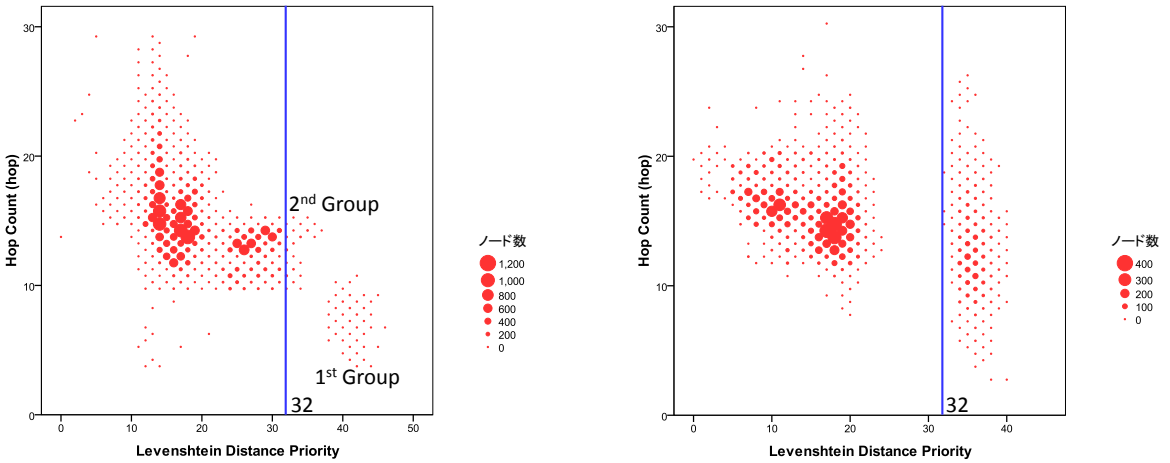
図 7.29: ドメインアルゴリズムによる優先度の分布

図 7.29(a) と図 7.29(b) より，優先度の分布においても，すべてのノードが一定の値に集約されているため，図 7.18 と図 7.23 で確認された，それぞれの優先度の分布が見られなくなっていることが確認できる．

### レーベンシュタイン距離アルゴリズム

レーベンシュタイン距離アルゴリズムのみによって，すべてのノードの優先度を求めた結果を図 7.30(a) と図 7.30(b) に示す．

図の各要素について説明する．$x$ 軸は，レーベンシュタイン距離アルゴリズムのみによって求めた優先度である．OCN の範囲は 0〜46，BBTEC の範囲は 0〜40 で，優先度は図の右に行くほど高くなる．$y$ 軸は，本アルゴリズムの場合と同様である．




(a) OCN (b) BBTEC

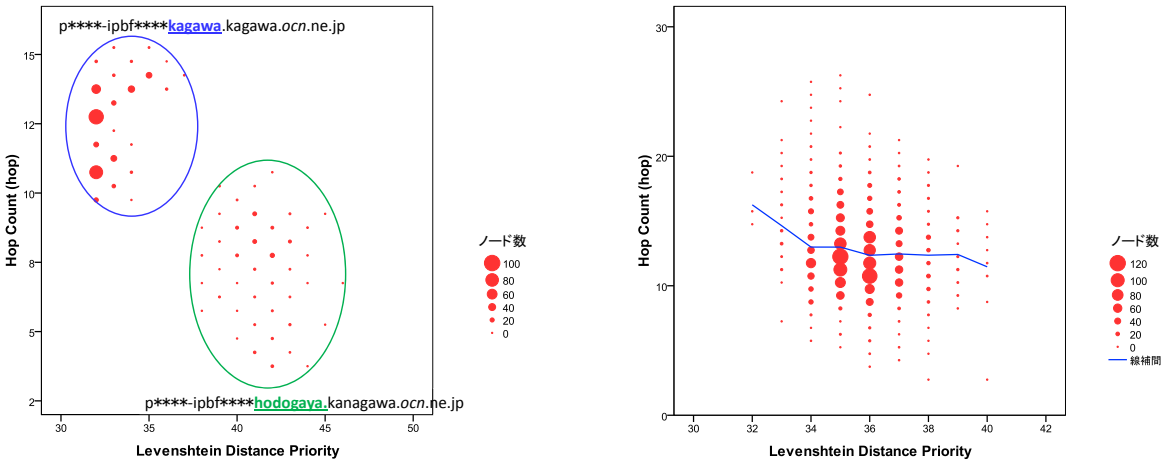
図 7.30: レーベンシュタイン距離アルゴリズムによる優先度

図 7.30(a) より，OCN において，レーベンシュタイン距離アルゴリズムで優先度を計算した場合，40 前後の優先度を持つノード群が，他のノードと比べて突出していることが確認できる．これらのノード群は，本アルゴリズムにおいて，FQDN が第 4 レベルドメインまで一致しており，文字列の類似度が高いノード群であると考えられる．

また，図 7.30(b) より，BBTEC において，レーベンシュタイン距離アルゴリズムで優先度を計算した場合には，30 から 40 までの優先度を持つノード群が，他のノードと比べて突出していることが確認できる．これらのノード群は，本アルゴリズムにおいて，FQDN が第 2 レベルドメインまで一致しており，文字列の類似度が高いノード群であると考えられる．

FQDN 文字列の全てを対象としたレーベンシュタイン距離アルゴリズムにおいて得られたノード群と，本アルゴリズムにおいてレーベンシュタイン距離を使用しているノード群で違いがあるかを検証するために，OCN においてレーベンシュタイン距離が 16 以下（図において 32 以上）のノードのみを図 7.31(a) に，BBTEC においてレーベンシュタイン距離が 12 以下（図において 32 以上）のノードのみを図 7.31(b) に示す．

図 7.31(a) において，OCN では，ノードの分布が 2 つの群にわかれているが，これらは，kanagawa.ocn.ne.jp というドメインの保土ヶ谷局収容のノード群と，そうでないノード群によって発生した分布である．本アルゴリズムを採用した場合にも，同様の分布を観測できたが，第 2 位に位置するノード群の構成が本アルゴリズムの場合と異なっている．本アルゴリズムでは，第 2 位に位置するノード群は，同じ kanagawa という第 4 レベルドメインを持っているノード群の中で，保土ヶ谷局収容ではないノード群だった．しかし，レーベンシュタイン距離アルゴリズムの場合，kanagawa という第 4 レベルドメインではなく，ocn という第 3 レベルドメインを持っている中で，kanagawa ではないノード群（kagawa という第 4 レベルドメインを持つノード群）が第 2 位に位置するノード群となっている．そのため，レーベンシュタイン距離アルゴリズムを採用して，最上位のノード群に接続できなかった場合，トラフィックの影響範囲は ocn.ne.jp 内で集約することはできるが，第 4




(a) OCN (b) BBTEC

図 7.31: レーベンシュタイン距離アルゴリズムによる優先度が 32 以上のノード

レベルドメインの粒度での集約はできない.

また，図 7.31(b) より，BBTEC では，優先度上位のノード群は本アルゴリズムによって優先度を求めた場合と同様の結果となる．これは，先述したように，BBTEC が持つ FQDN の上位ドメインが.bbtec.net のみで，第 3 レベルドメインがホスト名（レーベンシュタイン距離計算対象）となるためである.

また，文字列の長さを n とした場合に，本アルゴリズムを採用した場合の平均計算コストは,

$$O(\frac{n}{2})$$

となるが，レーベンシュタイン距離アルゴリズムを採用した場合の平均計算コストは,

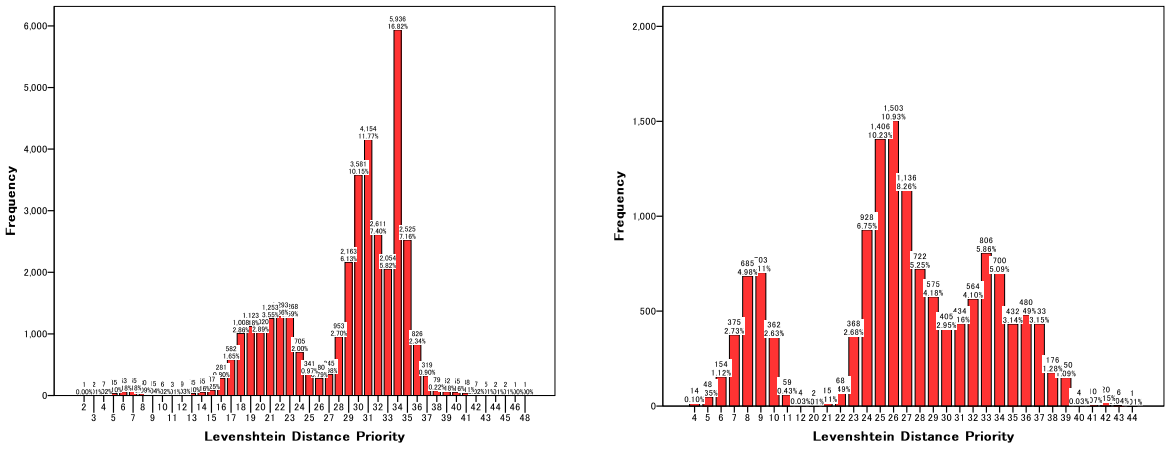
$$O(n^2)$$

となるため，レーベンシュタイン距離アルゴリズムを採用した場合の方が，計算コストは高くなる.

したがって，OCN においてはネットワーク距離の第 2 位の精度，及び，計算コストという点で，BBTEC においてにおいては，計算コストという点で，本アルゴリズムの方が優位である.

図 7.32(a) と図 7.32(b) にレーベンシュタイン距離アルゴリズムによる優先度の分布を示す.

図 7.32(a) と図 7.32(b) より，レーベンシュタイン距離アルゴリズムを用いた場合，優先度の粒度が他のどのアルゴリズムよりも細かくなることが確認できる．しかし，このような分布において，それぞれのノードが同一管理ドメイン内かどうかを判断することは困難であり，同時に，優先度が十分に高い場合でも，先述したように必ずしもネットワーク的に近くないノードが選ばれる可能性があるため，本アルゴリズムの方が精度において優位であることが確認できる.

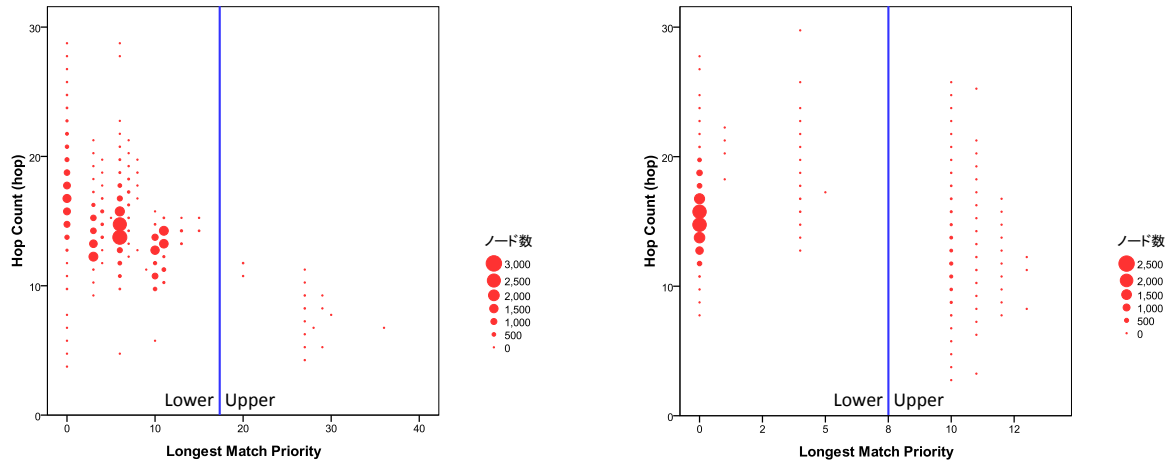



(a) OCN (b) BBTEC

図 7.32: レーベンシュタイン距離アルゴリズムによる優先度の分布

### 最長一致アルゴリズム

最長一致アルゴリズムによって，すべてのノードの優先度を求めた結果を図 7.33(a) と図 7.33(b) に示す．




(a) OCN (b) BBTEC

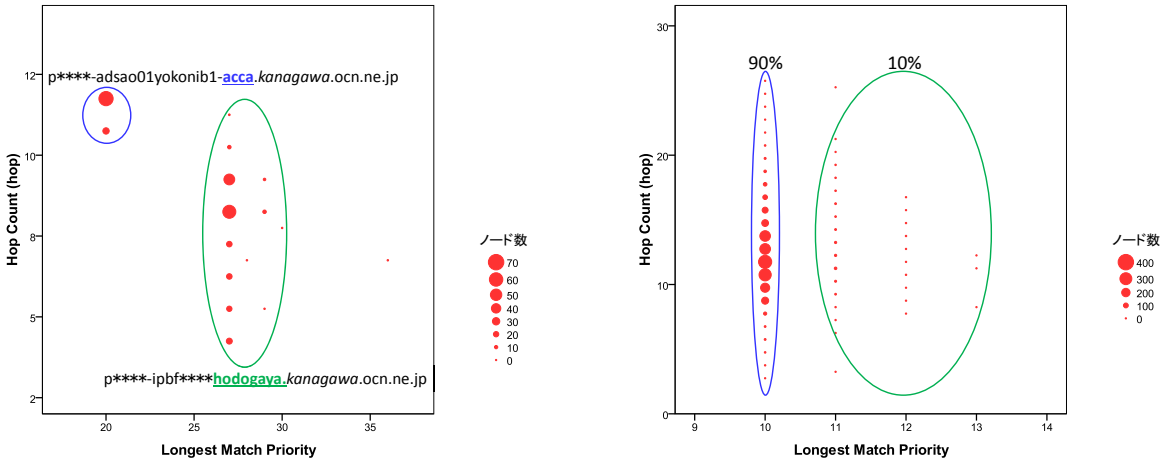
図 7.33: 最長一致アルゴリズムによる優先度

図の各要素について説明する．$x$ 軸は，最長一致アルゴリズムによって求めた優先度である．OCN の範囲は 0〜36，BBTEC の範囲は 0〜13 で，優先度は図の右に行くほど高くなる．$y$ 軸は，本アルゴリズムの場合と同様である．

図 7.33(a) より，OCN では最長一致アルゴリズムによる優先度が，20 より小さい場合に Hop Count が大きいノードが多く，20 より大きい場合に Hop Count が小さいノードが多いという傾向になっていることがわかる．これは，最長一致アルゴリズムが，.ocn.ne.jp 以降のノードの近接性を表現できるドメインレベルまで行われているためである．

また，図 7.33(b) より，BBTEC では最長一致アルゴリズムによる優先度が 8 より小さい場合に Hop Coun が大きいノードが多く，8 より大きい場合に Hop Count が小さいノードが多いという傾向になっていることがわかる．これは，最長一致アルゴリズムが，第 2 レベルドメインまで行われているためである．

優先度の大きい部分について，どこまで近接性が表現できるようになっているかを分析するために，OCN において，最長一致アルゴリズムの優先度が 20 以上のノードのみを図 7.34(a) に，BBTEC において最長一致アルゴリズムの優先度が 10 以上のノードのみを図 7.34(b) に示す．




(a) OCN (b) BBTEC

図 7.34: 最長一致アルゴリズムによる優先度が 20 以上のノード

最長一致アルゴリズムを採用した場合，図 7.34(a) より，OCN では最上位ノードの分布が 2 つの群にわかれていることが確認された．これらは，同じ，.kanagawa.ocn.ne.jp というドメインを持つノード群のうち，保土ヶ谷局収容のノード群（優先度最上位）と，そうでないノード群によって発生した分布である．優先度の低いノードの方が Hop Count が大きくなっているため，最長一致アルゴリズムを採用した場合にもネットワーク距離を正確に検出できていることがわかる．

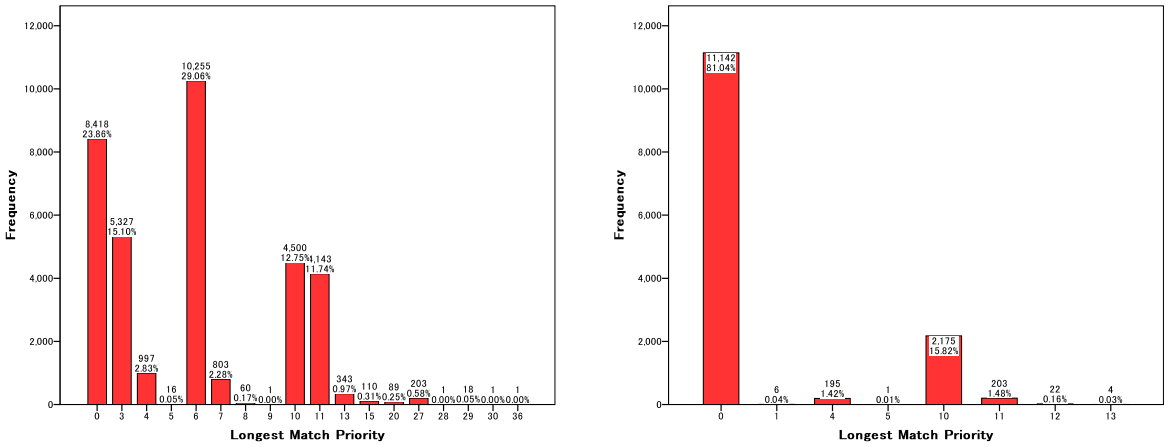
そのため，OCN の FQDN においては，最長一致アルゴリズムを採用した場合でも，本アルゴリズムと同等の結果を導き出すことができることが確認された．このような場合，本アルゴリズムは，最長一致アルゴリズムよりも低い計算コストでネットワーク距離を計算できないため，計算コストという意味で最長一致アルゴリズムの方が優位という結果になる．しかし，必ずしもすべての FQDN でこのような結果になるとは限らない．次に，BBTEC から観測した結果について考察する．

最長一致アルゴリズムを採用した場合，図 7.34(b) より，BBTEC では上位の約 90%のノードの優先度が 10 となり，11 以上のノードは 10%未満となっている．これは，最長一致アルゴリズムが右からの前方一致であるため，IP アドレスの下一桁（0〜9）の相違によって，ほとんどのノードが優先度計算を終えているためで，最上位のノードでも IP アドレスの下三桁（IPv4 アドレスにおける第 4 オクテット）までしか一致していない．

本項の冒頭でも述べたとおり，BBTEC の場合には，FQDN に含まれる IP アドレスの上位ビットがネットワーク ID となっており，最長一致アルゴリズムで評価される下位のビットに関しては，重要ではない．そのため，最長一致アルゴリズムでは，IP アドレスの上位ビットが同じでネットワーク的に近いノードを適切に判断することができない可能性が高く，図 7.34(b) においても，優先度が高くなっている場合に，Hop Count が小さくなる傾向は確認することができない．

したがって，BBTEC の FQDN においては，OCN の場合と異なり，ネットワーク距離の精度という点で，本アルゴリズムの方が優位である．

図 7.35(a) と図 7.35(b) に最長一致アルゴリズムによる優先度の分布を示す．




(a) OCN (b) BBTEC

図 7.35: 最長一致アルゴリズムによる優先度の分布

図 7.35(a) と図 7.35(b) より，最長一致アルゴリズムにおいては，ヒストグラムにおいて複数の値の増減が確認できる．これは，最長一致の段階において，ドメインレベルの一致度が現れたものである．そして，OCN ではドメインレベルが階層化されているため，複数の増減が確認でき，BBTEC では OCN ほどドメインレベルが階層化されていないため，値の増減が 1 つだけ確認できる．

### まとめ

OCN と BBTEC における本アルゴリズムと他のアルゴリズムの比較を表 7.10 に示す．

本項では，FQDN を評価するアルゴリズムについて，どのような FQDN の場合に，どのようなアルゴリズムを使うのが効率的なのかを検証するために，OCN と BBTEC から取得したサンプルデータについて，実測値と各アルゴリズムの比較を行った．

そして，OCN のように，ドメインレベルが階層化されており，階層の中で都道府県名や地域名を FQDN に用いている場合には，本アルゴリズムと最長一致アルゴリズムが，ほぼ同等の精度を持つことがわかった．

表 7.10: 本アルゴリズムと他のアルゴリズムの比較

| | OCN | BBTEC |
|---|---|---|
| ドメイン | ○（精度） | ○（精度） |
| レーベンシュタイン距離 | ◎（計算コスト・第 2 位精度） | ○（計算コスト） |
| 最長一致 | △（同等） | ○（精度） |

また，BBTEC のように，ドメインレベルがさほど階層化されておらず，IP アドレスを FQDN の一部として使用するような命名規則の場合には，本アルゴリズムが，他のアルゴリズムと比べて，最も FQDN の判定に適しているアルゴリズムであることがわかった．

### 7.2.8 他の管理ドメインについてのアルゴリズム評価

本項では，7.2.7 項の実験結果から得られた，FQDN の命名規則とそれぞれの場合に有効な FQDN の優先度計算アルゴリズムの組み合わせについて，他の管理ドメインでどのような結果が出るかを考察する．他の管理ドメインの例として，OCN のアルゴリズム評価に使用したデータから，OCN と BBTEC を除く上位 10 個の管理ドメインを対象とした．表 7.11 に今回の考察の対象としたドメインを挙げる．

表 7.11: 考察の対象としたドメイン名

| FQDN | ホスト名 | サブドメイン | 第 3 レベル | 第 2 レベル | トップレベル |
|---|---|---|---|---|---|
| 1 | FL1-123-45-67-89 | .tky | .mesh | .ad | .jp |
| 2 | p123a45 | .tokynt01.ap | .so-net | .ne | .jp |
| 3 | nttkyo123456 | .tkyo.nt.ngn.ppp | .infoweb | .ne | .jp |
| 4 | i123-45-67-89 | .s05.a015.ap | .plala | .or | .jp |
| 5 | 123-45-67-89 | N/A | .eonet | .ne | .jp |
| 6 | 89.67.45.123 | .dy | N/A | .bbexcite | .jp |
| 7 | pl123 | .nas955.p-tokyo | .nttpc | .ne | .jp |
| 8 | q6789 | .dynamic.ppp | .asahi-net | .or | .jp |
| 9 | 123x45x67x89 | .ap123 | .gyao | .ne | .jp |
| 10 | OFSfb-123p123-123 | .ppp11 | .odn | .ad | .jp |

これらの FQDN はいくつかのパターンに分けることが可能である．

- 地域名・ID のみが入るもの
  p123a45.tokynt01.ap.so-net.ne.jp
  nttkyo123456.tkyo.nt.ngn.ppp.infoweb.ne.jp
  pl123.nas955.p-tokyo.nttpc.ne.jp
  OFSfb-123p123-123.ppp11.odn.ad.jp

- IP アドレスのみが入るもの
  123-45-67-89.eonet.ne.jp
  89.67.45.123.dy.bbexcite.jp
- IP アドレスの一部が入るもの
  q6789.dynamic.ppp.asahi-net.or.jp
- 地域名・ID と IP アドレスが入るもの
  FL1-123-45-67-89.tky.mesh.ad.jp
  i123-45-67-89.s05.a015.ap.plala.or.jp
  123x45x67x89.ap123.gyao.ne.jp

地域名・ID のみが入るものでは，FQDN のドメインレベルが階層的になっているものが多く見受けられる．ドメインレベルの深いものでは，第 8 レベルドメインまであり，ドメインレベルが浅いものでも，第 5 レベルドメインまでが存在する．これらの FQDN は，本アルゴリズムの第 1 段階であるドメインレベルを用いることで，同一管理ドメイン内での近接性を考慮し，第 2 段階でレーベンシュタイン距離を求めることで，同じドメインレベルでの近接性を考慮することができると考えられる．また，多くの FQDN が OCN の場合と同様に，最長一致アルゴリズムを用いることで，本アルゴリズムと同様の結果を出すことが可能であると考えられる．

ホスト名において IP アドレスのみが入るものでは，FQDN のドメインレベルがさほど階層化されていないものが見受けられる．特に，eonet[47] の FQDN は，BBTEC の例のように，最長一致アルゴリズムではさほど優先度に差が出ないが，本アルゴリズムによってレーベンシュタイン距離を求めることで，同じドメイン内での近接性を考慮することができる．しかし，ドメイン名に IP アドレスのみが入る場合でも，bbexcite の例では，本アルゴリズムはうまく機能しない可能性がある．bbexcite の FQDN は IP アドレスが‘．’区切りで入っているため，IP アドレスの部分に関してもドメインレベルでの評価が行われ，各オクテットが完全一致せず，IP アドレスの類似度が高いにもかかわらず，低いスコアとなってしまう可能性がある．

ホスト名において IP アドレスの一部が入るものでは，FQDN は階層化されているものの，各ドメインレベルの示す内容がどのようなノードでも同じであるため，本アルゴリズムの第 2 段階におけるレーベンシュタイン距離による優先度が同一ドメイン内での優先度となる可能性が高い．しかし，レーベンシュタイン距離の適用範囲は比較的狭いため，計算コストはレーベンシュタイン距離アルゴリズムのみを使った場合よりもコストが低くなり，他のどのようなアルゴリズムよりも優位であると考えられる．

FQDN に地域名・ID と IP アドレスの双方が入るものでは，ドメインレベルが階層化され，かつ，ホスト名で IP アドレスを表現しているものが多く見受けられる．これらの FQDN は，本アルゴリズムによって，第 1 段階としてドメインレベルの比較を行い，第 2 段階でレーベンシュタイン距離を求めることで，同じドメイン内での近接性を，他のどのようなアルゴリズムよりも正確に表現できると考えられる．

#### まとめ

本項では，FQDN の命名規則とそれぞれの場合に有効な FQDN の優先度計算アルゴリズムの組み合わせについて，他の管理ドメインにおける結果を検証した．

OCN，BBTEC 以外のドメインについて，FQDN は 4 つのパターンに分けることが可能である．そして，地域名・ID のみが入るものについては，計算コストという意味で最長一致アルゴリズムの方が優位であるが，精度という意味では，4 パターンすべてにおいて，本アルゴリズムの優先度が最も良い結果となることが考えられる．また，IP アドレスの一部が入るものや，地域名・ID と IP アドレスが入るものに関しては，本アルゴリズム以外で，計算コストと精度の両方を担保できるアルゴリズムは存在しない．

したがって，どのような FQDN を持つ管理ドメインに所属していても，精度の高い優先度を計算できる（汎用性が高い）という意味で，本アルゴリズムが優位である．

## 7.3 考察

本章では，本論文で提案した FQDN を用いた優先度について，その有効性を実ネットワークを使用した実験と，サンプルデータを使用した検証により評価を行った．

実ネットワークを使用した実験では，2 つの実験を行った．データ転送時のノード選択実験では，自分と同じドメインのノードがデータを持っていた場合，そこからデータを取得することが確認できた．また，データ拡散時のノード選択実験においては，最初は同じドメインにデータを持っているノードが存在しないために，ドメイン（ネットワーク距離）に関係なくデータを取得するが，データがある程度拡散した場合には，自分と同じドメインなどのネットワーク的に近いノードからデータを取得することが確認できた．したがって，本論文の目的であるトラフィックの影響範囲の集約が行われることを定性的に評価できた．

次に，サンプルデータを使用した実験では，実際に動作している P2P システムである，Winny と Share を対象に，それぞれのネットワークのノードの FQDN と Hop Count を取得し，本アルゴリズムを適用した場合の効果について評価を行った．OCN と BBTEC の 2 つのドメインからそれぞれ観測を行うことで，どちらも優先度が高い場合に，ネットワーク距離である Hop Count の値が小さかったため，本アルゴリズムの有効性を確認することができた．また，5.3.5 項で述べた，他のアルゴリズムを使用した場合と本アルゴリズムを使用した場合の結果を比べることで，ドメインレベルの最長一致とレーベンシュタイン距離を組み合わせて使うことの有効性についても確認することができた．したがって，本論文の目的であるトラフィックの影響範囲の集約が行われることを定量的に評価することができた．

## 7.4 まとめ

本章では，FQDN を用いたネットワーク距離について，Hop Count を用いたネットワーク距離と比較・検討を行った．具体的に，実ネットワークを使った実験と，サンプルデー

タを使用した検証を行うことで，FQDN を用いたネットワーク距離が，既存の Hop Count を用いたネットワーク距離と比較して，同等の指標としての機能があることが検証された．そして，FQDN の優先度計算において，本アルゴリズムがコスト的・精度的に優位であることを証明した．

# 第8章　結論

本章では，本論文の成果をまとめ，今後の展望を述べる．

## 8.1　まとめ

本論文では，既存の P2P システムが抱えるトラフィックの問題について，DNS の管理ドメインというアプローチから解決を図った．そして，DNS の管理ドメインのラベルとして得られる FQDN を基にネットワーク距離を表現し，P2P ネットワークのトポロジを管理・制御する機構を設計，実装した．本手法を用いることで，P2P システムは，自身のネットワークトポロジによるトラフィックの影響範囲を集約し，既存のインターネットアーキテクチャに適応することが可能となる．

P2P システムが抱える問題は，大きく分けて 2 つある．1 つは，P2P ネットワークのトポロジが自律分散的に構成されることである．P2P システムの性質上，各ノードが P2P ネットワーク全体のトポロジを把握することは難しく，また，サーバのような全体を管理・制御するノードもいないことから，トポロジは各ノードが自律的に判断を行うことで形成，運用される．このようなネットワークアーキテクチャの場合，各ノードは自分の通信相手以外の情報を知ることができないため，冗長な経路や無駄な通信が発生しやすい．しかし，P2P ネットワーク上において，それらを検知してネットワークの再構成を行うことは困難であるばかりでなく，オーバーヘッドも大きく，システム全体のパフォーマンスを低下させる可能性がある．

次に，P2P システムが発生させるトラフィックである．P2P システムは，大容量のコンテンツ配信などに使われることが多く，それらが発生させるトラフィックは非常に大きなものになるため，その影響範囲はできるだけ集約されることが望ましい．しかし，先述したネットワークトポロジの問題により，トラフィックの影響範囲が必要以上に拡大し，P2P システムだけでなく，インターネット全体のパフォーマンスを低下させている可能性がある．特に，インターネットバックボーンの根幹である IX においては，P2P システムが発生させるトラフィックにより，貴重なネットワークリソースが消費され，深刻な問題となっている．

本論文では，このような背景に着目し，トラフィックの影響範囲が集約されるようにネットワークトポロジを構築することが可能な P2P システムである LinkPoint を提案，実装した．具体的には，P2P システム自体に P2P ネットワークを構成する際のアルゴリズムを組み込むことで，各ノードが自律的にトラフィックの影響範囲が集約されるネットワークトポロジを構成する．このような仕組みを取り入れることで，P2P システムのパフォー

マンスの低下を最小限にすると同時に，トラフィックの影響範囲を集約することが可能となる．

また，既存の P2P システムに関する研究では，トラフィックの影響範囲の集約よりも，P2P システム自体のデータ転送パフォーマンスを向上させる傾向があり，直接アルゴリズムを組み込むのではなく，プラグインとして情報を提供する役割を担うものがある．本論文でそのようなアプローチを採らなかったのは，トラフィックの影響範囲の集約において，P2P システム本体のアルゴリズムの設計が必要であると考えたためである．特に，データ転送パフォーマンスの向上に関しては，トラフィックの影響範囲の集約とは相反する部分もあり，既存の P2P システムを改良するだけでは大きな効果は得られないと考えている．

したがって，本論文では，前述したアルゴリズムを実現するために，DNS の管理ドメインに着目したネットワーク距離の表現手法を提案した．DNS はドメインごとに階層構造を持っているため，ドメインレベルの相違をネットワーク距離として表現することで，同一ドメインに所属するノードと積極的に通信を行うことが可能となる．そして，同一ドメインに所属するノードとの通信は，同一ドメイン内で完結することができるため，結果としてトラフィックの影響範囲を集約することが可能となる．そのため，本論文では，ドメインの近さを表現するために，管理ドメインのラベルである FQDN に着目し，文字列処理によって管理ドメインの近さを判定できる手法を設計・実装した．

FQDN によるネットワーク距離の有効性を示すために，実ネットワークにおける実験とサンプルデータを用いた検証を通して評価を実施した．実ネットワークにおける実験では，実際に本アルゴリズムを動作させることで，ドメインの近さを判定し，トラフィックの影響範囲が集約できることが確認された．特に，データが P2P システム上に十分に拡散していない場合においても，本アルゴリズムによって同じドメインからデータを取得する様子が確認され，FQDN によるネットワーク距離が有効であることを証明した．

サンプルデータを用いた検証では，本アルゴリズムによって近いと判定したノードが，約 80%の確率で実測した Hop Count においても近接であり，本アルゴリズムによって表現されるネットワーク距離の有効性を示した．特に，本論文で採用している FQDN を解析する文字列処理アルゴリズムに関しては，既存の他の文字列処理アルゴリズムに比べて優位であることが証明され，安価な方法で実測値の予測をすることに成功した．

P2P システムにおけるネットワークトポロジの管理・制御は，困難であると同時に，多大なコストが掛かる作業である．しかし，本論文で提案，実現したネットワーク距離推定手法を用いることで，トラフィックの影響範囲の集約を安価かつ確実に行うことができるだけでなく，ユーザのパフォーマンスの低下を最小限に抑えることができる．そして，P2P システムは既存のインターネットアーキテクチャに適応することが可能となり，P2P システムだけでなく，インターネット全体の利便性向上や可能性の発展に寄与できる．

## 8.2 今後の課題と展望

本論文の今後の課題と展望について述べる．

### 8.2.1 様々な P2P システムへの対応

本論文では，P2P システム自体にアルゴリズムを組み込むことで，トラフィックの影響範囲の集約を行ったが，既存の P2P システムについても，本手法は適用可能である．特に，P2P システムが発生させるトラフィックはインターネット全体の課題となっているため，本手法によるトラフィックの影響範囲の集約は，そのような問題を解決する糸口になると考えている．

また，P2P システムは応用範囲が広いため，構成される P2P ネットワークも，設計方針や運用ポリシにより様々な構造が考えられる．そのため，P2P ネットワーク自体のアーキテクチャも，構造化されたモデルや非対称性のネットワークへシフトしていく可能性がある．そのような場合に，本手法がどのように適応され，ネットワーク・トラフィックの集約が可能となるかを検証していく必要がある．

### 8.2.2 他のパラメータの導入

本論文では，ネットワーク距離の指標として FQDN を用いたが，既存の P2P システムでは，RTT や Hop Count などの実測データが用いられることが多い．これらの指標は計算コストが高いが，その分，ネットワーク距離を正確に反映することができるため，精度は FQDN よりも良い場合がある．したがって，今後は，FQDN だけでなく，RTT や Hop Count などのデータと組み合わせることで，本アルゴリズムの精度を向上できるのではないかと考えている．

特に，RTT を用いた研究は 4 章でも述べたように，様々な研究が行われている．パフォーマンスの向上を目的とした研究とは相容れない部分もあるが，既存研究の成果を活かせる部分もあると考えられる．そのため，新たな要素を取り入れることで，より一層アルゴリズムの汎用性が高められ，今後も出てくる様々な要求に柔軟に応えることが可能となると考えられる．

### 8.2.3 管理ドメイン判断手法の検討

本手法において管理ドメインを判断する際には，管理ドメインのラベルである FQDN を用いたが，これはノードの IP アドレスを DNS へ問い合わせることで取得を行っている．しかし，このような方法では DNS サーバに負担が掛かる可能性があるだけでなく，FQDN が DNS へ登録されていない場合や，IP アドレスから FQDN を引いてこれない場合にアルゴリズムが正常に機能しなくなる可能性がある．そのため，事前に IP アドレスの各組織・ドメインへの割り当てを把握しておき，DNS などへ問い合わせをすることなく，ノードの所属している管理ドメインを判断できることが望ましい．

また，事前に IP アドレスの割り当てを把握しておくことは，DNS の負担を減らし，P2P システムのパフォーマンスを上げるだけでなく，各管理ドメインのつながりやポリシを考慮できるようになると考えられる．今回の手法では，管理ドメイン内のつながりについては，FQDN を評価することで再現することができたが，同じ FQDN を持つノードが P2P

システム内に存在しなかったり，その数が極端に少なかった場合には，FQDN の高いレベルでの一致度によって接続するノードが選択される．しかし，このような方法では，自分にとって必ずしも近いノードが選ばれるとは限らないため，各管理ドメインのつながりなどを考慮することで，管理ドメインが違う場合でもネットワーク的に近いノードに接続することが可能になると考えられる．

### 8.2.4 ネットワーク負荷の計測方法について

本論文の評価において，ネットワーク負荷は各ノードが他のノードから受信したデータ量を計測することによって表現した．しかし，このような表現方法では，途中のリンクのネットワーク負荷までは表現することができないため，P2P システムの経路全体について，ネットワーク負荷を軽減できているかはわからない．そのため，より詳しくネットワーク負荷の分析を行うために，途中のリンクのセッション数やスループットについて，P2P システムが発生させるトラフィックの割合を分析する必要がある．

### 8.2.5 ネットワーク距離の基準

本手法では，FQDN をネットワーク距離の基準としたが，このような論理的な距離だけでなく，物理的な距離を考慮することも必要であると考えている．特に，世界規模に展開する P2P ネットワークの場合には，AS[48] の組織が全世界で 30,000 を超えている [49] 現状を鑑みた場合に，FQDN だけではデータが不十分である可能性がある．そのため，GeoIP[50] などの地理情報サービスを活用することで，論理的な距離だけでなく，物理的な距離も考慮して，ネットワークトポロジを構成できるようになることが望ましい．

### 8.2.6 ISP への提言

本論文では，FQDN を用いることで，ネットワーク的な近接性を判定できることを示したが，現在の DNS の管理ドメインにおいては，7.2.8 項で述べたように，管理組織ごとに様々な FQDN が設定されており，含まれている情報の粒度も違っている．そのため，現在の FQDN の付け方では，必ずしもトラフィックを集約することができない管理ドメインが存在するのも事実である．そこで，本論文の研究成果を基に，近接性の判定が有用になるようなドメイン名をつけることを，ISP に対して提言することが必要である．

# 謝辞

本論文の作成にあたり，ご指導頂いた慶應義塾大学環境情報学部教授村井純博士，同学部教授徳田英幸博士，同学部教授中村修博士，同学部准教授楠本博之博士，同学部准教授高汐一紀博士，同学部准教授三次仁博士，同学部准教授植原啓介博士，同学部専任講師重近範行博士，同学部専任講師中澤仁博士，同学部専任講師 Rodney D.Van Meter III 博士に感謝致します．

また，絶えず御指導と御助言を頂きました慶應義塾大学 DMC 機構専任講師，斉藤賢爾博士に感謝致します．

そして本研究を進めていく上で，様々な励ましと助言，お手伝いをいただきました，慶應義塾大学 SFC 研究所上席所員中村友一氏，慶應義塾大学政策・メディア研究科後期博士課程岡田耕司氏，石原知洋氏，海崎良氏，堀場勝広氏，田崎創氏，工藤紀篤氏，久松剛氏，松園和久氏，水谷正慶氏，松谷健史氏，同研究科修士課程金井瑛氏，空閑洋平氏，奥村祐介氏，遠峰隆史氏，六田佳祐氏に感謝致します．同大学総合政策学部上原雄貴氏，永山翔太氏，飯塚裕貴氏，同大学環境情報学部佐藤貴彦氏，波多野敏明氏，三部剛義氏と徳田・村井・楠本・中村・高汐・重近・バンミーター・植原・三次・中澤合同研究プロジェクトの皆様に感謝致します．

共に卒論を執筆した，慶應義塾大学総合政策学部中里恵氏，同大学環境情報学部峯木厳氏，立石幹人氏に感謝致します．

最後に，大学入学から 4 年間に渡る研究生活で，私を支え続けてくれた家族に心から感謝致します．

以上を持って，謝辞といたします．

# 参考文献


[1] Approaching the Zettabyte Era. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481374.pdf.

[2] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki and Akira Kato. Observing Slow Crustal Movement in Residential User Traffic. *ACM CoNEXT2008*, December 2008.

[3] YouTube. http://www.youtube.com/.

[4] ニコニコ動画. http://www.nicovideo.jp/.

[5] 日本インターネットプロバイダー協会. ISP を取り巻く状況と提案. http://www.soumu.go.jp/joho_tsusin/policyreports/chousa/internet_policy/pdf/080627_2_si5-2.pdf.

[6] Revealed: the environmental impact of Google searches. http://technology.timesonline.co.uk/tol/news/tech_and_web/article5489134.ece.

[7] Revealed: How The Times Got Confused About Google and The Tea Kettle. http://www.techcrunch.com/2009/01/12/revealed-the-times-made-up-that-stuff-about-google-and-the-tea-kettles/.

[8] salesforce.com. http://www.salesforce.com/.

[9] Amazon Web Services. http://aws.amazon.com/.

[10] Microsoft Windows Azure. http://www.microsoft.com/azure/.

[11] MobileMe. http://www.apple.com/mobileme/.

[12] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.

[13] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31(4):161–172, 2001.

[14] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. pages 53–65, 2002.

[15] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, Berkeley, CA, USA, 2001.

[16] FastTrack. http://www.kazaa.com/.

[17] Skype. http://www.skype.com/.

[18] Gnutella Protocol Specification. http://wiki.limewire.org/index.php?title=GDF.

[19] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4p: provider portal for applications. *SIGCOMM Comput. Commun. Rev.*, 38(4):351–362, 2008.

[20] David R. Choffnes and Fabian E. Bustamante. Taming the Torrent: A practical approach to reducing cross-ISP traffic in P2P systems. In *Proceedings of the ACM SIGCOMM '08 Conference*, August 2008.

[21] Akamai Technologies, Inc. http://www.akamai.com/.

[22] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of the ACM SIGCOMM '04 Conference*, Portland, Oregon, August 2004.

[23] BitTorrent. http://www.bittorrent.com/.

[24] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592.

[25] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343.

[26] A. Marine, J. Reynolds, and G. Malkin. FYI on Questions and Answers - Answers to Commonly asked "New Internet User" Questions. RFC 1594 (Informational), March 1994. Obsoleted by RFC 2664.

[27] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[28] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[29] K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994. Obsoleted by RFC 3022.

[30] UPnP Forum. http://www.upnp.org.

[31] Levenshtein V.I. Split orthogonal arrays and maximum independent resilient systems of functions. *Designs, Codes and Cryptography*, 12:131–160(30), October 1997.

[32] P2P ネットワーク実験協議会. http://www.fmmc.or.jp/p2p_web/news/20081015.pdf.

[33] B. Kaliski. PKCS #1: RSA Encryption Version 1.5. RFC 2313 (Informational), March 1998. Obsoleted by RFC 2437.

[34] Kalle Kaukonen and Rodney Thayer. A stream cipher encryption algorithm "arcfour". 1999.

[35] D. LeGall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[36] Schwarz T. Schafer R., Wiegan T. *The emerging h.264/avc standard*, 2003.

[37] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. Updated by RFCs 950, 4884.

[38] Keio University. https://www.keio.ac.jp/.

[39] BB.excite. http://bb.excite.co.jp/.

[40] Yahoo! BB. https://ybb.softbank.jp/.

[41] Open Computer Network. http://www.ocn.ne.jp/.

[42] Isamu Kaneko. *The Technology of Winny.* ASCII, October 2005.

[43] Share. Freenet:http://127.0.0.1:8888/SSK@szGkEWGmpXKoQHqsu9ubZze-kOgPAgM/CAT/17//.

[44] ACCS ファイル共有ソフト利用実態調査. http://www2.accsjp.or.jp/news/release081212.html.

[45] WinPcap: The Windows Packet Capture Library. http://www.winpcap.org/.

[46] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[47] ケイ・オプティコム. http://eonet.jp/.

[48] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930 (Best Current Practice), March 1996.

[49] Autonomous System (AS) Numbers. `http://www.iana.org/assignments/as-numbers/`.

[50] MaxMind, Inc. `http://www.maxmind.com/`.

[51] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174 (Informational), September 2001. Updated by RFC 4634.

# 付 録A　LinkPointの機能

本章では，LinkPoint のインタフェースにおいて，表示する内容と，その機能について述べる．

## A.1　メインインタフェース

- ノード（図 A.1）
  接続しているノードのステータスを表示する．接続タイプや方向などの他に，クラスタや回線速度などの情報が表示され，上部のステータスよりも詳細な情報がわかるようになっている．

- トリガ（図 A.2）
  P2P システム内のデータを探す際に，検索以外の方法として提供されているのがトリガである．トリガは，キーワードやハッシュ値を指定して検索するデータを設定できるようになっており，ユーザが検索画面で明示的にダウンロードを選択しなくても，バックグラウンドで条件に一致するデータを探し，データが発見された時点で自動的にダウンロードリストへ登録することができる．

- 検索（図 A.3）
  検索を行うタブである．上部のキーワードに指定された条件に一致するデータを表示する．表示されるデータには 3 種類あり，データそのものを持っている場合（Complete），データの一部を持っている場合（Local），データを持っていない場合（Remote）のそれぞれについて，表示・非表示を切り替えることができる．また，過去に検索を行ったキーワードがボタンとして列挙され，必要に応じてキーワードを変更，検索結果をリストに表示させることが可能となっている．

- ダウンロード（図 A.4）
  トリガに登録され，条件に一致したデータや，検索画面でダブルクリック，又はダウンロードを指定されたデータが登録され，データサイズや保持しているデータの詳細が表示されている．LinkPoint では，バックグラウンドで検索が行われており，データが発見された場合には適宜ダウンロードタスクが起動され，データのダウンロードを始める．データのダウンロードがすべて終わった場合には，キャッシュの変換が行われ，ダウンロードフォルダーにデータが展開され，該当するデータはリストの登録を解除される．

- アップロード（図 A.5）
  自ノードがアップロードしているデータを表示する．アップロードフォルダーに登録されたデータと完全なキャッシュとして保持しているデータが表示され，データサイズや保持しているデータの詳細が表示される．また，アップロードを停止したいデータや削除したいデータがある場合は，この画面で設定することができる．

- フィルタ（図 A.6）
  トリガや検索でキーワード検索を行う際に，対象から除外するキーワードやハッシュ値などを指定する．捏造されたデータや不正なデータを登録することで，不必要な検索結果を表示しないようにできる．フィルタは複数作成することができるため，より多くの条件を設定するほど，トリガや検索の精度を上げることができる．

- タスク（図 A.7）
  P2P システムのタスクの状況が表示される．表示されるのはダウンロードとキャッシュ変換の内容で実行中のタスク，または実行が終わったタスクが列挙される．ダウンロード中のタスクに関しては，ダウンロード速度や終了予定時間などが一覧できるようになっており，進行状況を数値として見ることができる．

- ログ（図 A.8）
  P2P システムのログが表示される．主な内容は起動の際のデバッグメッセージと，ネットワークに接続する際に取得できる IP アドレスやバージョンに関する情報である．

- インフォ（図 A.9）
  P2P システムの情報が表示される．自ノードのデータ転送量や管理しているメタ情報数，ノード数などが表示され，詳しい P2P システムのステータスを知ることができる．

## A.2 サブインタフェース

- ネットワーク（図 A.10）
  P2P システムのネットワーク設定を変更することができる．接続設定では送信速度，受信速度を変更したり，着信接続を受け付けるポート番号を変更することが可能である．また，接続プロパティでは，低速な接続の切断機能や UPnP 機能の有無を設定することができる．

- クラスタ（図 A.11）
  P2P システムにおいて，所属するクラスタを設定することができる．クラスタに使用する単語は最初から用意されているわけではなく，自分で設定することが可能である．クラスタに使用する単語は 1 単語あたり 255 文字（全角の場合は 127 文字）までで，最大 5 つまで設定することが可能である．

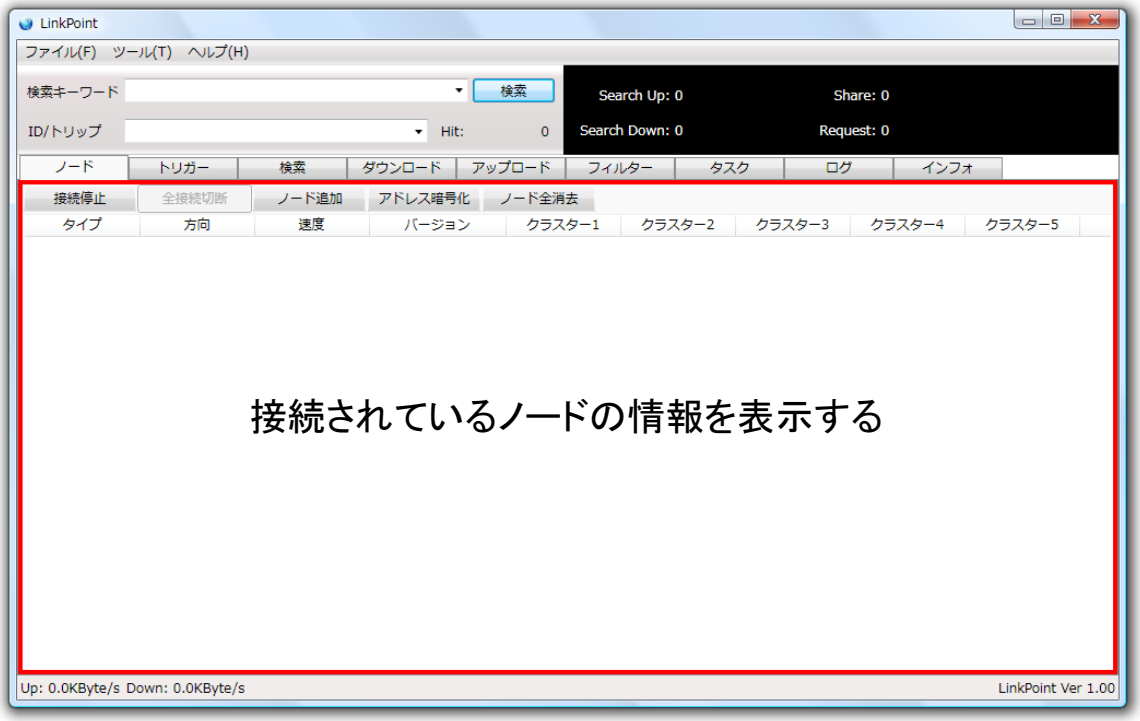



図 A.1: LinkPoint メインインタフェース（ノード）

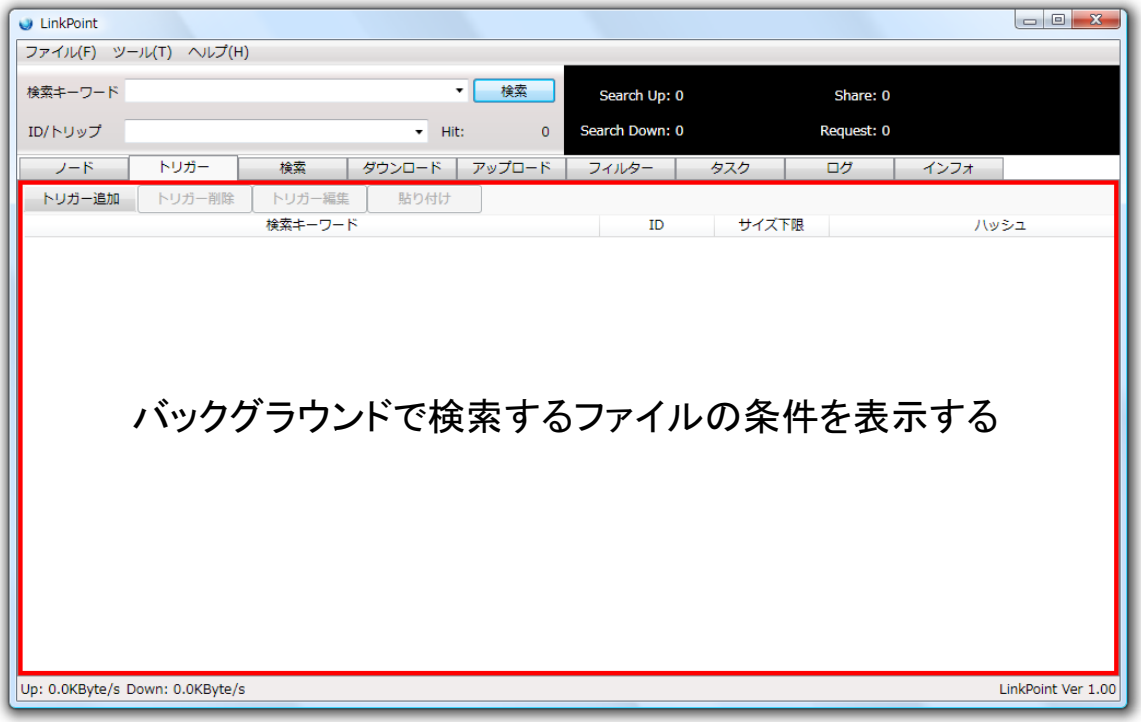



図 A.2: LinkPoint メインインタフェース（トリガ）

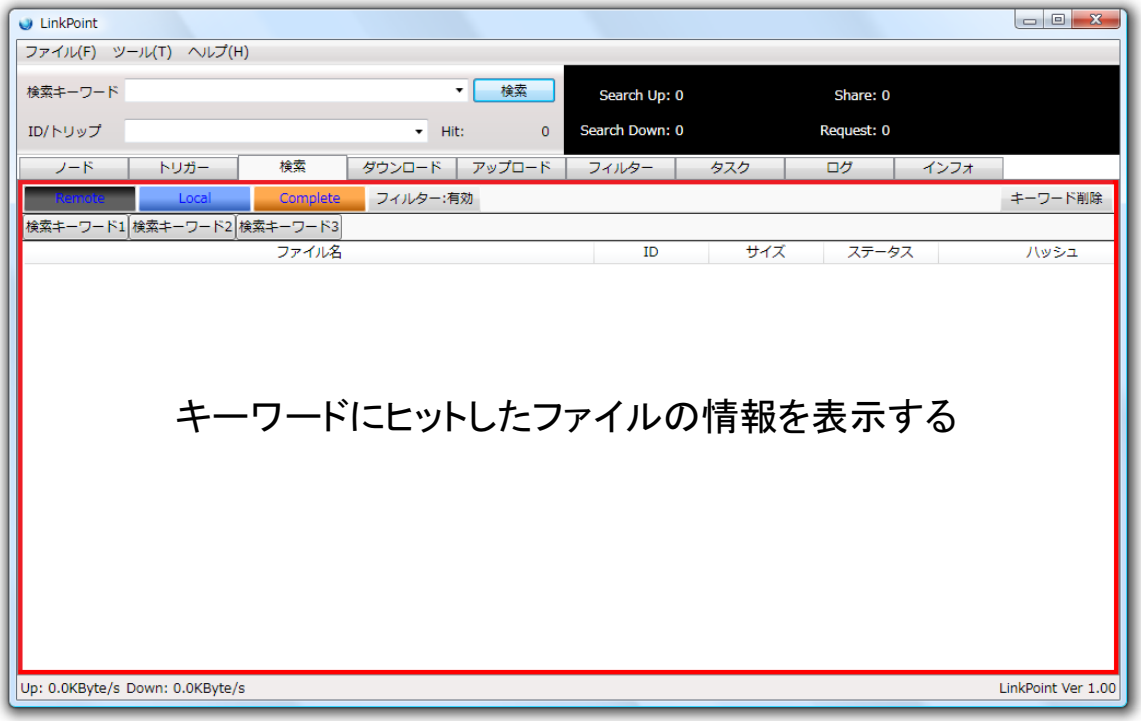



図 A.3: LinkPoint メインインタフェース（検索）

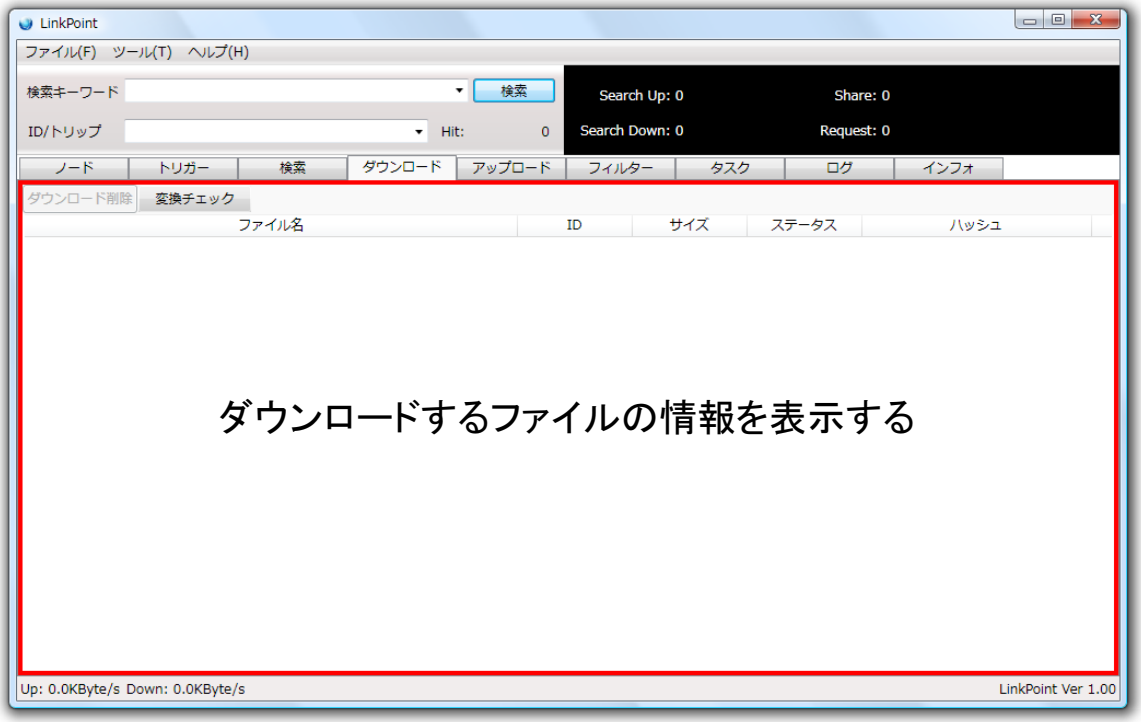



図 A.4: LinkPoint メインインタフェース（ダウンロード）

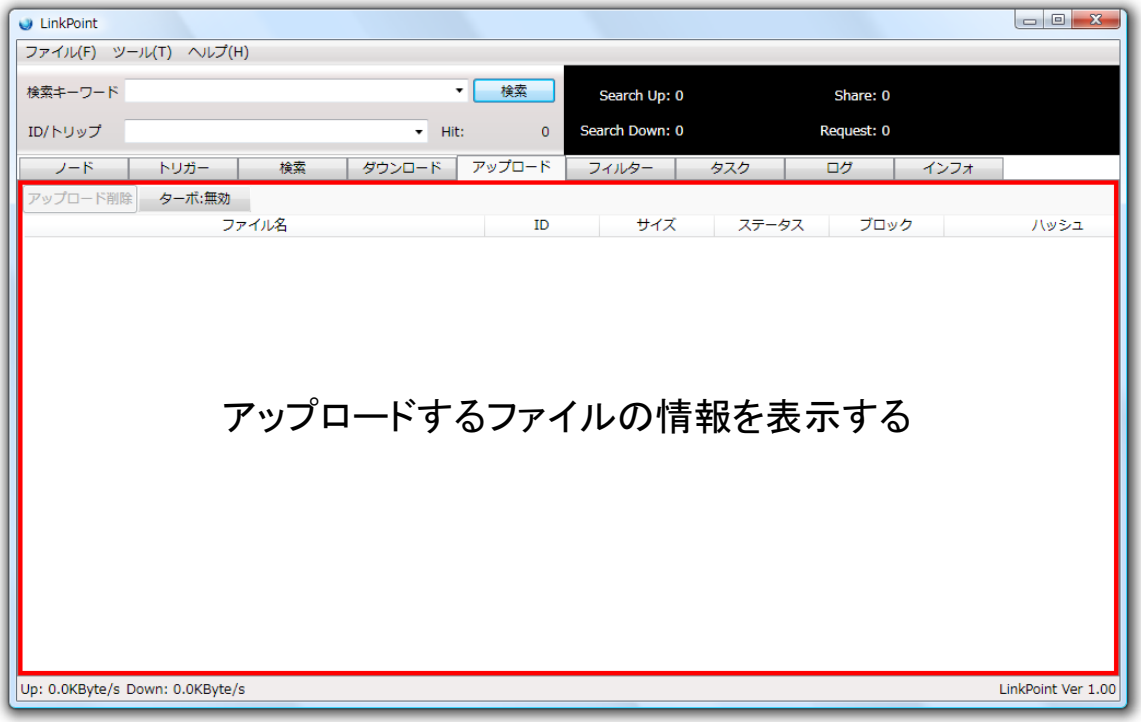



図 A.5: LinkPoint メインインタフェース（アップロード）

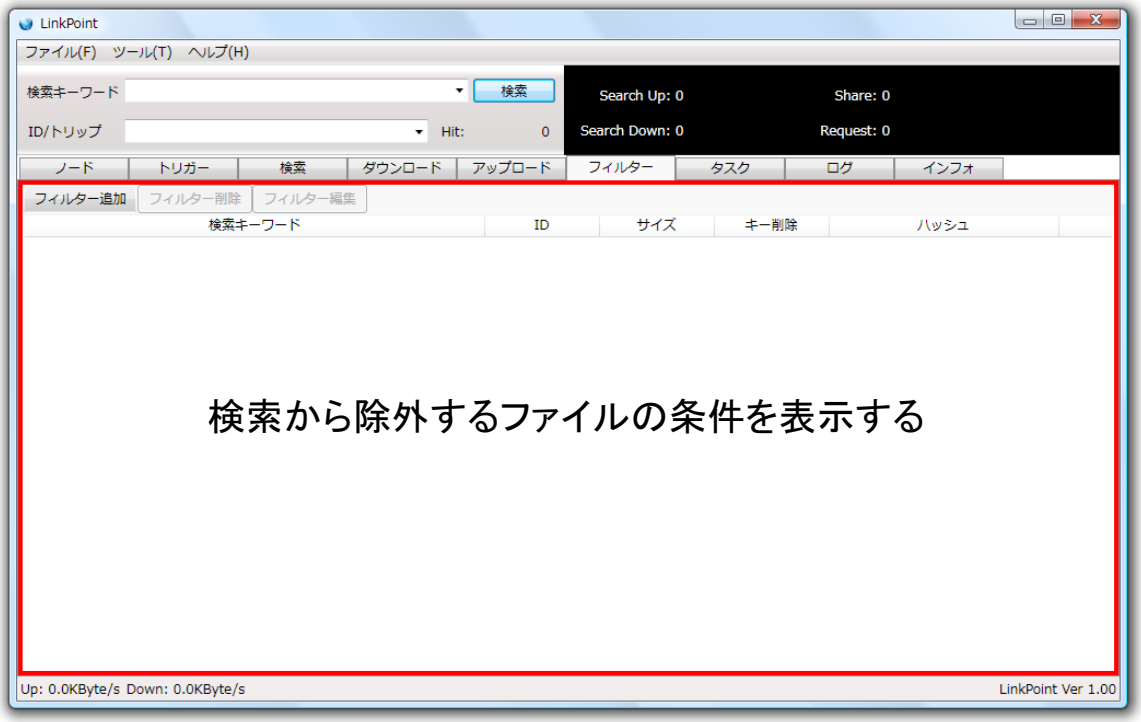



図 A.6: LinkPoint メインインタフェース（フィルタ）

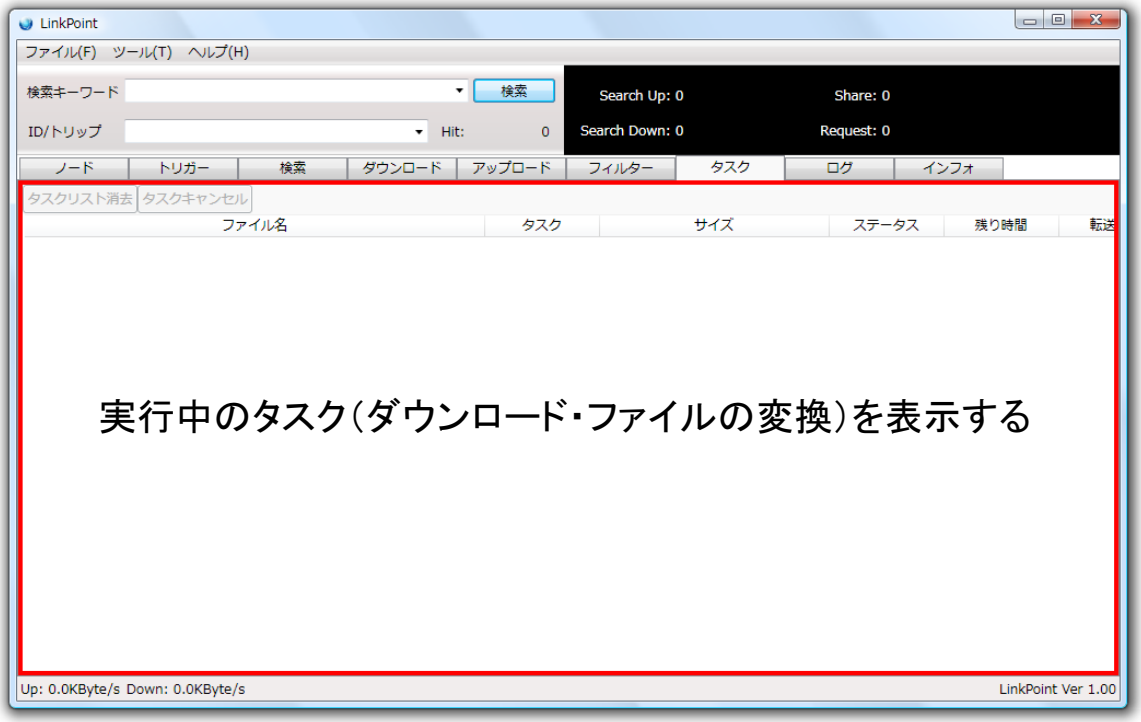



図 A.7: LinkPoint メインインタフェース（タスク）

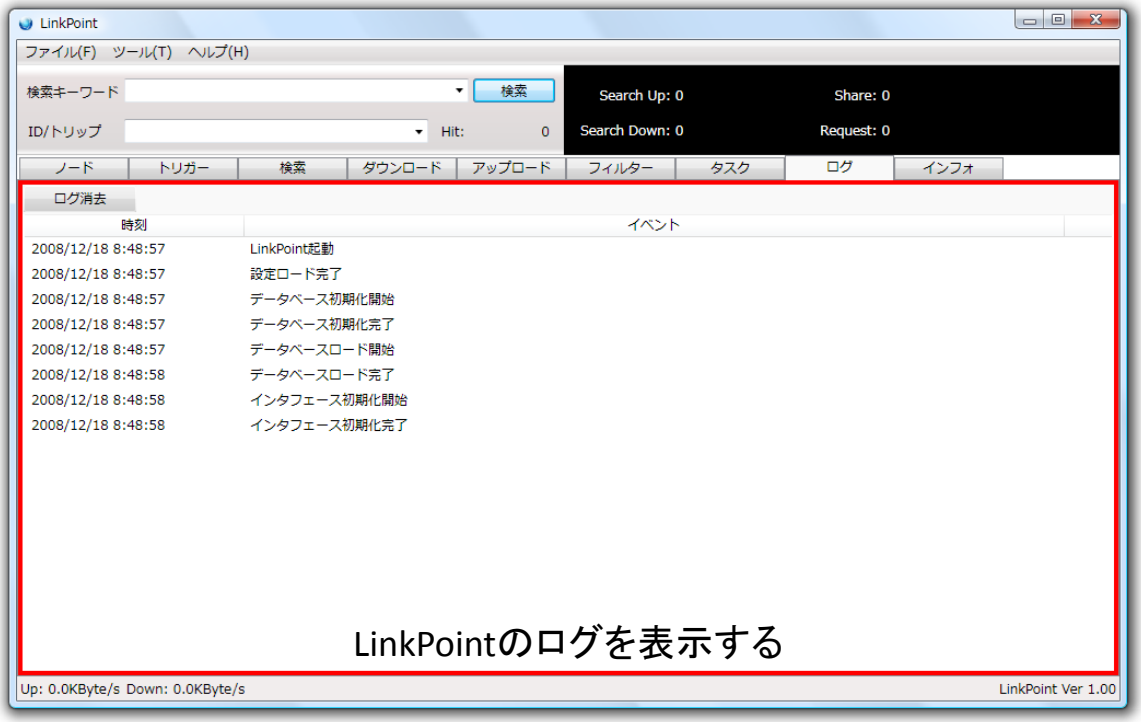



図 A.8: LinkPoint メインインタフェース（ログ）

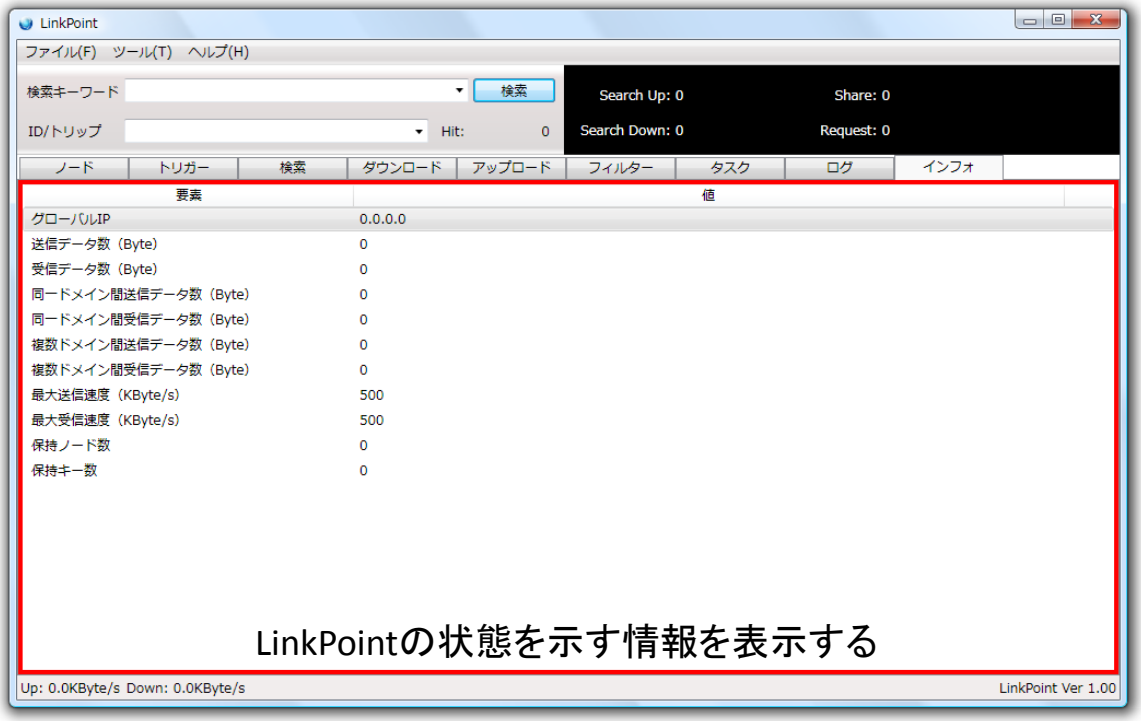



図 A.9: LinkPoint メインインタフェース（インフォ）

- フォルダ（図 A.12）
  P2P システムで利用するフォルダを設定することができる．デフォルトではプログラムファイルが存在するフォルダの下に設定されるが，ダウンロード・キャッシュ・アップロードをそれぞれ別の場所に設定することが可能である．また，アップロードフォルダに関しては，サブフォルダをアップロードファイルに含めるかどうかの設定が可能である．

- フィルタ（図 A.13）
  P2P システムで適用するフィルタを設定することが可能である．図 A.6 でキーワードやハッシュ値を指定する際に，別のフィルタを作成したい場合にはこの画面で新しくフィルタを作成することができる．作成するフィルタに文字数や個数の制限はない．

- 詳細設定（図 A.14）
  P2P システムの詳細な設定を行うことが可能である．デフォルト ID では，データを P2P システムに登録する際の ID を設定することが可能である．また，アクション・パフォーマンスでは検索機能やダウンロードの設定を行うことができる．図 A.15 に ID 設定を示す．ID は，自分のニックネームと識別子で構成され，表示は「ニックネーム@識別子」となっている．ニックネームは自分の好きな文字列，識別子は入力文字列から一意に決定されるランダムな文字列である．識別子の文字列生成には Secure Hash Algorithm 1（SHA-1）[51] を使用した．

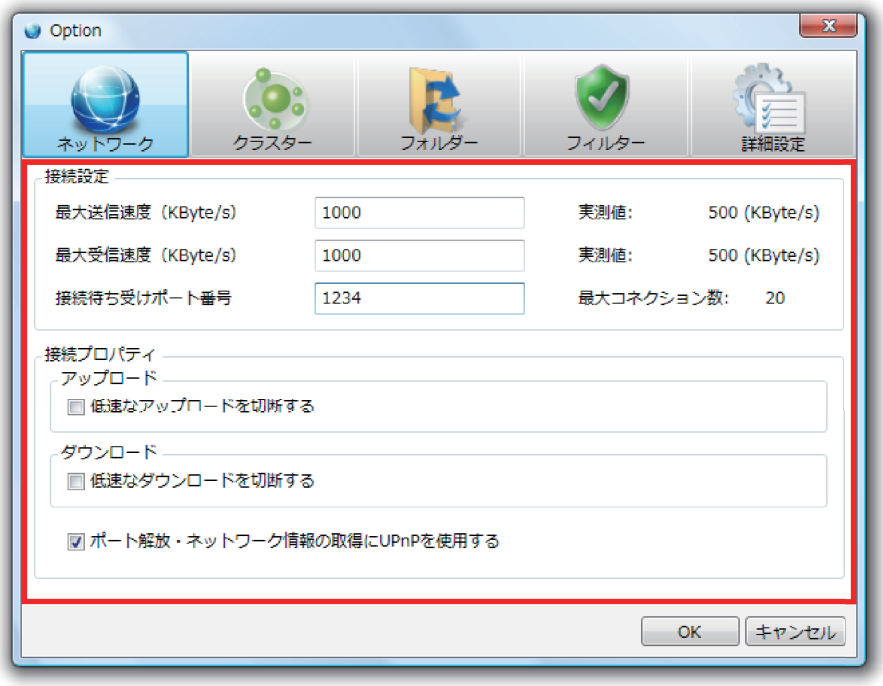



図 A.10: LinkPoint サブインタフェース（ネットワーク）

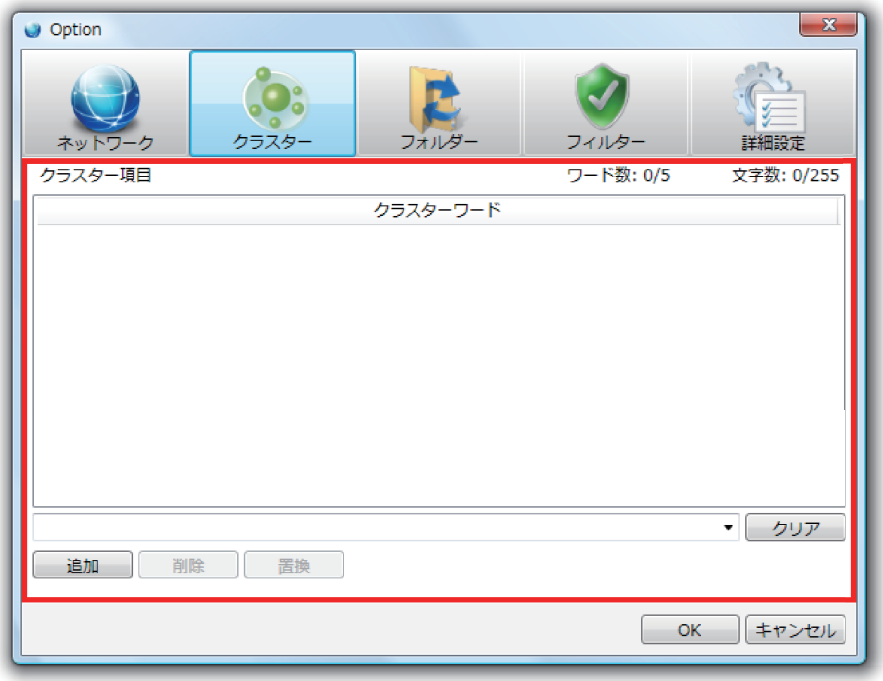



図 A.11: LinkPoint サブインタフェース（クラスタ）

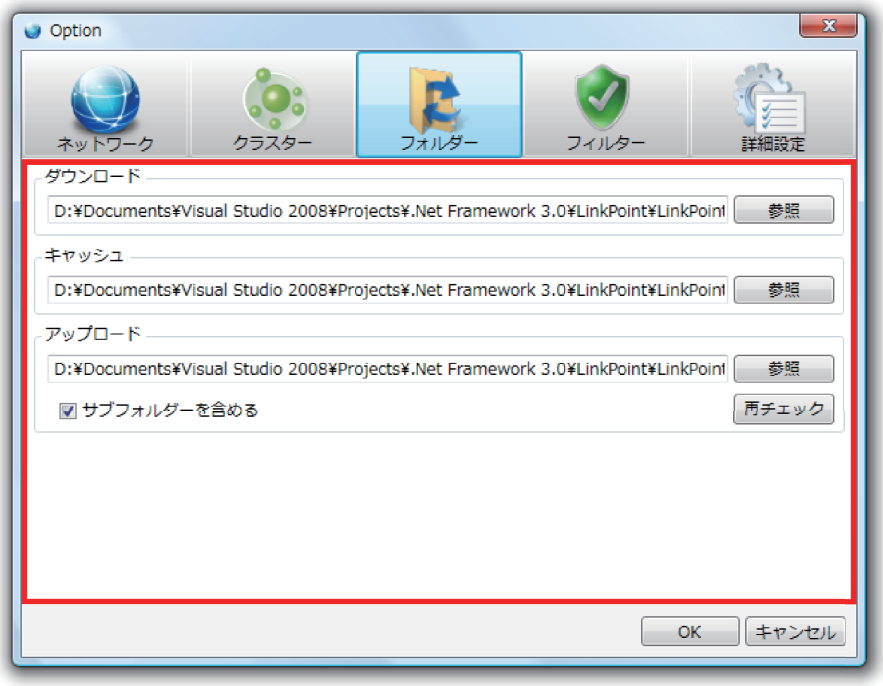



図 A.12: LinkPoint サブインタフェース（フォルダ）

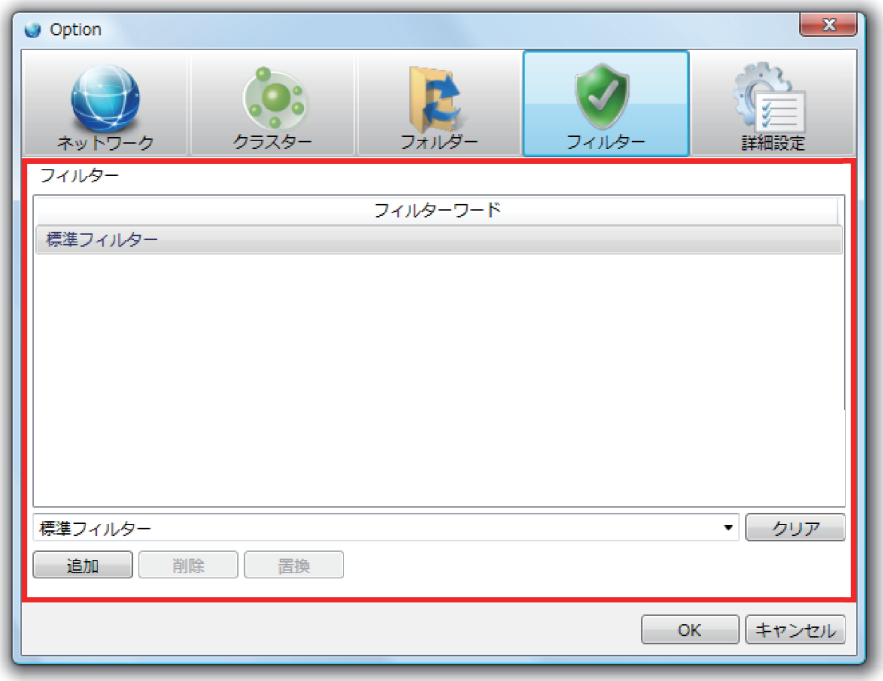



図 A.13: LinkPoint サブインタフェース（フィルタ）

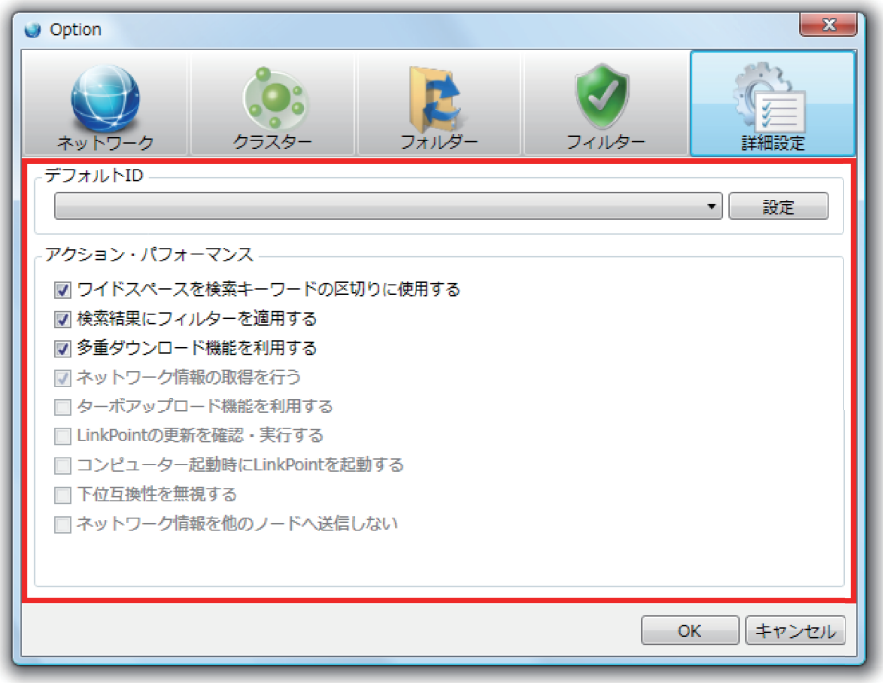



図 A.14: LinkPoint サブインタフェース（詳細設定）

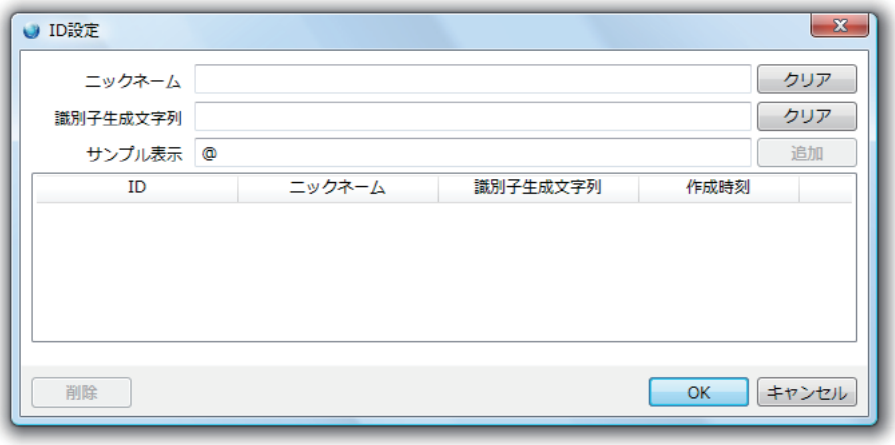



図 A.15: LinkPoint サブインタフェース（ID 設定）