卒業論文 2011 年度 (平成 23 年)

同報型アプリケーションのための
オーバーレイを用いたマルチキャスト配送網の構築

慶應義塾大学 環境情報学部
中村 遼

卒業論文要旨 2011 年度 (平成 23 年度)

同報型アプリケーションのための
オーバーレイを用いたマルチキャスト配送網の構築

現在、動画や音楽などのリッチコンテンツのストリーミングを用いた放送型サービスが増加している。こういったコンテンツの多くは、有力なコンテンツホルダによる、多数の視聴者が予想される人気のコンテンツである。一方で、インターネット配信は手軽に不特定多数の受信者にむけて配信できるという特徴から、個人などの小規模なコンテンツホルダによる、よりユーザ指向の強いコンテンツのライブストリーミングも増加している。

こういった放送型ストリーミングサービスを実現する多くの既存システムは、ユーザからのストリームを、専用に潤沢な計算機資源の用意されたライブストリーミングサーバが各受信者へユニキャストで送信することで実現されている。既存システムには、このようなサーバ・クライアントモデルで構築されていることに起因するいくつかの問題がある。まず、受信者数の増加に従って中継サーバの負荷が増大するという問題がある。また、ユーザは事業者の提供する形でしかサービスを提供することができないという問題がある。

本研究では、IP マルチキャストの配送網をリングトポロジのオーバーレイネットワークで構築するシステムの設計、実装、評価を行った。本論文で提案するアーキテクチャを導入することで、ユーザが自由に送信者となることができる 1 対多型の通信基盤を構築することが可能となる。本研究で提案するシステムを用いることで、ユーザは任意のアプリケーションを用いて、自由に不特定多数の受信者に対してライブストリーミングサービスを行うことができるようになる。

キーワード
1.IP マルチキャスト、2. オーバーレイネットワーク、3. リングトポロジ

慶應義塾大学 環境情報学部

中村 遼

Abstract of Bachlor's Thesis

Academic Year 2011

Overlay routing architecture for Multicast applications

Broadcasting services have been increasing thanks to rich contents such as streaming video and music. Many of these contents, which are expected to gather many listeners, are broadcasted by leading content holders. On the other hand, user-oriented contents, which are not expected to gather many listeners, have also been increasing. These contents are broadcasted by small-scale content holders such as end users, because broadcasting on the internet is very easy.

Existing system for broadcasting services are constructed on the server-client model. Relay servers, which are prepared by service providers, relay streams from users to listeners. Relay servers need a large amount of network and computing resources. There are problems with this model. First, the server load increases as the number of listeners increase. Second, users can only provide services in a way prepared by the service providers.

In this study, an overlay routing system which uses a ring topology for delivering IP Multicast is designed, implemented and evaluated. By using the proposed architecture, any source multicast network can be constructed. By using the proposed system, users can provide broadcasting services freely using any data format generated by any application.

Keywords :
1. IP Mutlicast, 2. Overlay Network, 3. Ring Topology

Keio University, Faculty of Enviromental Information

Ryo Nakamura

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

現在、インターネット上における動画や音楽などのリッチコンテンツのストリーミングサービスが増加している。特にその中でも、Ustream などをはじめとする、ユーザが 1 対多型での放送型リアルタイムストリーミングを行うサービスへの需要が高まっている。現在ライブストリーミングされているコンテンツの多くは、多数の視聴者が予想されるコンサートなどである。Ustream によってインターネット配信されたあるコンサートでは、全世界で同時視聴者数が 10 万人を超えた。このようなコンサートの他にも、多くの人気のあるコンテンツがライブストリーミングされている。

一方で、小規模のコンテンツホルダによる、少数のユーザに向けた、よりユーザ指向の強いライブストリーミングも増加している。インターネット配信は手軽に不特定多数の受信者に配信できるという特徴がある。録画機能付きの携帯電話や、3G による屋外でのインターネット接続などを利用して、誰でも、どこからでもライブストリーミングを行うことができる環境が整っている。その結果、漫画の執筆作業やゲームのプレイ画面、屋外でのイベントの様子など、小規模コンテンツホルダや個人ユーザによる様々なコンテンツがライブストリーミングされている。

しかし、既存のライブストリーミングサービスは、サーバ・クライアントモデルで構築されており、ストリーミングアプリケーションと配信基盤が一体となっている。そのため、ユーザが行えるサービスの形態は、事業者が中継サーバを提供しているサービスの形態に制限されている。また、サーバ・クライアントモデルのストリーミング配信では、中継サーバが送信者からのストリームを複数の受信者に向けてコピーして送信する。そのため、受信者数の増加にともなって中継サーバがコピーすべきトラフィックも増加し、規模性に欠けるという問題がある。

そこで本研究では、ライブストリーミングのための配信基盤とアプリケーションを分離し、配信基盤に着目する。本研究では特に、個人ユーザを含む小規模コンテンツホルダによる自由なライブストリーミングを実現するために、アプリケーション透過性を持つ配信基盤として、誰もが自由に送信者となることができる 1 対多型通信のための配送網を構築する。この配送網の実現によって、ユーザは不特定多数の相手と自由に 1 対多型の通信を行うことができるようになる。その結果、ユーザは任意のアプリケーションを用いて、自由に不特定多数の受信者に対してライブストリーミングを行うことができるようになる。

1.2 本研究の目的

本研究では、任意のユーザによる不特定多数の受信者への 1 対多型通信の実現を目的とする。そのために、リング状のオーバーレイネットワークを用いた IP マルチキャスト配送網の構築を行う。IP マルチキャストを用いることで、大抵のホストは変更を加えることなく任意のアプリケーションからこの配送網を利用することができる。また配送網をオーバーレイネットワークで構築することによって、世界規模での網の拡張が可能となる。これによって、世界中のユーザが自由に 1 対多型の通信を用いてサービスを行うことができるようになることを目指す。本論文では、リングトポロジのオーバーレイネットワークを構築するにあたって、まず効率的なノードの並び順について検証を行った。その上で、本研究の提案するシステムの設計、実装、ならびに評価を行った。

1.3 本論文の構成

本論文の構成を以下に示す。第 2 章では、1 対多型通信を用いるアプリケーションについて整理し、本研究の提案するモデルを示す。3 章では、本研究の用いるマルチキャスト通信モデルについて整理し、本研究の目的を実現するためのマルチキャスト通信基盤に必要な要件を示す。第 4 章では、第 2 章、第 3 章の議論をふまえた上で実現すべき機能要件について整理し、本研究で提案する手法について述べる。第 5 章では、システムの実装について述べる。6 章では、実験環境に対する要件を整理し、構築した実験環境と評価結果を示す。7 章に、本研究のまとめと今後の課題を示す。

第2章　1対多型の通信モデル

インターネットにおける通信の大半は1対1型の通信である。エンドホストは1つの対向ホストのIPアドレスを指定して通信を行う。一方で、1つのホストから複数のホストへと、1対多型の通信を行う同報型アプリケーションも多く存在する。1対多型の通信モデルは、複数ホスト間のメッセージングや、ストリーミング等、多くの場面で有効である。しかし、既存の1対多型の通信モデルはサーバ・クライアントモデルで実現されており、規模性やアプリケーション透過性に問題がある。本章では、既存の1対多型通信を実現するための手法の問題点を明確化するため、まず同報型アプリケーションについて整理する。その上で、本研究の提案する1対多型の通信モデルを示す。

2.1　1対多型通信の今と未来

2.1.1　1対多型通信の今

インターネット上における1対多型の通信は、複数ホスト間のメッセージングやストリーミングなどに用いられている。同報通信をメッセージングに用いる理由として、何らかの識別子をもって複数のホストをグルーピングし、そのグループに対して一斉にメッセージを届けることができる点があげられる。ホストはグループの識別子のみを知っていれば、そのグループに対してメッセージを送信することで、メンバの有無や数に関わらず、サービスを開始することができる。そのため、ユーザはアプリケーションを起動する際に何も設定をする必要がない。

一方、1対多型の通信を行う他のアプリケーションとして、音楽やビデオコンテンツなどの放送型ライブストリーミングがあげられる。現在、計算機の発達により、手軽な機材でも動画を撮影しつつリアルタイムでエンコードすることが可能になった。また家庭へのアクセス回線の高速化や屋外無線の高速化によって、ユーザが動画を撮影、エンコードし、インターネット上に配信する環境が整っている。その結果、ユーザがインターネット上の複数の視聴者に対してライブストリーミングを行うサービスが増加している。

このような1対多型の通信は、現在はサーバ・クライアントモデルで実現されている。送信者は、インターネット上に存在する中継サーバに対してパケットを送信し、中継サーバが複数の受信者にむけて、パケットをコピーして送信する。ストリーミングのような帯域や遅延に関する要求の厳しいアプリケーションは、中継サーバに特別に潤沢なネットワーク資源や計算機資源を事前に用意することで実現されている。

2.1.2 1対多型通信の未来

既存の1対多型の通信には、サーバ・クライアントモデルで構築されていることに起因するいくつかの問題がある。ストリーミングのような帯域、遅延などに厳しい要求を持つアプリケーションでは、サーバ・クライアントモデルは規模性に欠けるという問題がある。ユーザの送信したストリームは、事業者が用意した中継サーバによって受信者数分コピーされ送信される。そのため、受信者数の増加に従って中継サーバがコピーしなければならないトラフィック量は増加し、中継サーバの負荷が上がる。事業者が中継サーバに潤沢なネットワーク資源や計算機資源を用意することはできるが、特定のコンテンツに突発的に視聴者が集中した際には、中継サーバの負荷が増加し、コンテンツの品質に影響が出る場合がある。

また、既存の1対多型の通信を実現するための中継サーバは、アプリケーションごとに用意される。そのため、ユーザは中継サーバの提供されているサービスしか利用することができない。また、あるアプリケーションが他のアプリケーションのために構築された1対多型の通信基盤を利用することはできない。これは、中継サーバが事業者によって提供されるようなサービスではユーザのサービス選択性を狭める原因となる。また、中継サーバには負荷が集中するため、ユーザが自由にサービスを行うためにユーザ自身で中継サーバを用意することは現実的ではない。

このように、既存の1対多型通信モデルはサーバ・クライアントモデルであり、配信基盤とアプリケーションが結合しているため、規模性とサービス選択性について問題がある。そこで本研究では、配信基盤とアプリケーションの分離を提案する。1対多型の配信基盤を、パケット配送網自体が担うことによって、中継サーバへのトラフィックの一極中集という問題を無くす。また、ユーザが任意のアプリケーションで1対多型の配送網を利用し、自由にサービスを提供できなければならない。そのため、配送網はアプリケーションに対して透過性を持つ必要がある。

本章では、2.2節で1対多型の通信を用いる同報型アプリケーションについて整理を行う。2.3では、同報型アプリケーションの中でもより要求の厳しいストリーミングサービスについて注目し、その要件を整理する。2.4節では、既存のストリーミングサービスの問題点を示し、2.5節において、この問題点を解決するための1対多型の通信モデルを示す。

2.2 同報型アプリケーション

インターネットにおける通信は1対1型の通信を前提としており、配送網は宛先となる1つのホストへとパケットを配送することを目的として構築されている。しかし一方で、1つのホストから複数のホストへと、1対多型の通信を行う同報型アプリケーションも多く存在する。1対多型の通信モデルは、複数ホスト間のメッセージングやストリーミング等、多くの場面で有効である。

メッセージングに同報通信を用いるアプリケーションとして、vxlan[1]やcorosync[2]、bonjour[3]が挙げられる。これらのアプリケーションは、IPマルチキャストを用いた

メッセージングを行う。IP マルチキャストは、特別にアサインされた IP マルチキャストグループアドレスに対して同報通信を提供する。そのため、あるホストグループに参加したいノードは、グループのメンバを知らなくとも、IP マルチキャストグループアドレスのみを設定されれば動作することができる。メッセージングに同報通信を用いるアプリケーションで送信されるデータは、比較的サイズが小さく、帯域や遅延に対して厳しい要求を持たない。

もう 1 つの 1 対多型の通信モデルのアプリケーションとして音楽やビデオコンテンツのリアルタイムストリーミングがある。ストリーミングサービスの中でも、放送型リアルタイムストリーミングは、1 対多型の同報通信を行うアプリケーションである。インターネット上でのビデオストリーミングサービスでは、ユーザはサーバに保存されたコンテンツをストリーミングで受信しつつ、同時に再生する。一方、放送型のライブストリーミングでは、送信者の送信するストリームが同時に複数の受信者に届く。放送型のライブタイムストリーミングは、動画や音楽などのコンテンツをリアルタイムで配信するため、帯域やパケットロス、遅延、ジッタがサービスの品質に大きく影響する。

このように、メッセージングとストリーミングという同報通信を用いる 2 種類のアプリケーションでは、ストリーミングの方がより厳しい要求を持つ。そのため、ストリミーングアプリケーションを実現できれば、同報通信をメッセージングに用いるアプリケーションも実現することができると考えられる。そこで本研究では、帯域や遅延に関する要求がより厳しい放送型のライブストリーミングを想定アプリケーションとして選択する。

2.3 ストリーミングサービス

現在インターネット上でのビデオコンテンツや音楽コンテンツのストリーミング再生がインターネット全体のトラフィックの大きな割合を占めるまでに増加している。Cisco 社の公開しているインターネットにおけるトラフィック分析 [4] では、2012 年にはビデオトラフィックがインターネット上のトラフィックの半分を超えると予測されている。また、ライブストリーミングによるトラフィックは、2015 年まで年平均成長率 48%で増加すると予測されている。

ストリーミングサービスの 1 つの形態として、インターネット上に保存されたビデオコンテンツや音楽コンテンツのストリーミング再生がある。莫大な量のコンテンツを保存し、オンデマンドにユーザへ配信するこのサービスは、インターネットの広帯域化と記憶容量の拡大によって実現した。このような保存された静的コンテンツのストリーミングサービスは、コンテンツの保存されたコンテンツサーバ群と、配信サーバ群、そして高速な回線をそれらのサーバ群に接続することによって構築されている。

一方、ストリーミングサービスのもう 1 つの形態として、ユーザによる放送型のライブストリーミングサービスがある。計算機環境の発達により、手軽な機材でも動画を撮影しながらエンコードすることが可能となった。また、インターネットへのアクセス回線も高速になり、屋外無線技術によって屋外でもインターネットに接続するこ

とが可能となった。その結果、その場で撮影、エンコードし、インターネット上へと配信できる環境が整い、ユーザによるライブストリーミングが増加し始めている。こういったライブストリーミングは、コンサートやイベント等の予め多数の視聴者が予測されるものから、少数の視聴者に向けたユーザ指向の強いものまで、様々なコンテンツが存在する。

2.3.1 ライブストリーミングされるコンテンツ

インターネット上でのライブストリーミングサービスを利用して放送されるコンテンツは大きく 2 つにわけられる。1 つは有力なコンテンツホルダによって提供される、多数の視聴者が予想されるライブストリーミングである。もう 1 つは、小規模コンテンツホルダによる、ユーザ指向の強いライブストリーミングである。これら 2 つの異なるコンテンツについて、以下に特徴と要求をまとめる。

- 有力なコンテンツホルダによる人気のあるコンテンツ

  有力なコンテンツホルダによってライブストリーミングされるコンテンツは、多くの視聴者が予想されるコンテンツである。例えば、大統領の就任演説やコンサート、大学の式典、イベントでの講演などがあげられる。こういった多数の受信者数が予想されるコンテンツのライブストリーミングでは、数多くの受信者数に向けてストリームを転送する配信能力が要求される。こういったコンテンツは、あらかじめ配信時間やストリームのフォーマットなどが決まっていることがほとんどである。また受信者数はコンテンツごとにある程度予想可能である。そのため、事前にストリームの形式を合わせたり、受信者数に合わせて配信能力を増強しておくことが可能である。

- 小規模コンテンツホルダによるユーザ指向のコンテンツ

  個人ユーザを含む小規模コンテンツホルダによってライブストリーミングされるコンテンツは、よりユーザ指向の強い、特定のユーザ層に向けたコンテンツである。例えば漫画の執筆風景や、ゲームのプレイ画面、屋外でのイベントの様子などがあげられる。このようなコンテンツは、ライブストリーミングの手軽さという特徴によって近年増加している。こういったユーザ指向のライブストリーミングでは、配信されるコンテンツは多種多様であり、また動画のフォーマット、配信時間などもユーザによって異なる。そのため、送信者がより自由な形態でストリーミングを行える環境が求められる。また、ユーザ指向のストリーミングは様々な要因によって受信者数が突発的に増加する可能性がある。そのため、受信者数を予測することが難しく、事前に配信能力を増強することは難しい。

2.4 既存のライブストリーミングサービス

既存の放送型のライブストリーミングサービスは、サーバクライアント・モデルで実装されている。既存の放送型ライブストリーミングサービスでは、ユーザからのストリームを、事業者の用意した専用の潤沢なネットワーク資源や計算機資源を持つライブストリーミングサーバが各受信者へユニキャストで中継することで実現されている。このサーバ・クライアントモデルの既存サービスには、アプリケーションと配信基盤がセットになっていることに起因するいくつかの問題がある。

まず 1 つ目として、ユーザのサービス選択性が制限されるという問題がある。ユーザは、ストリームを事業者の用意したアプリケーションレベルの中継サーバを中継することでライブストリーミングを行う。そのため、ユーザは事業者によって中継サーバに用意された形態でしかサービスを提供することができない。ユーザの送信するライブストリーミングは、フォーマット、コンテンツ、タイミング、クオリティ等を、事業者によって提供される形態に制限される。そのため、ユーザが事業者によって提供されていない形態でサービスを行いたい場合は、ユーザ自身で潤沢なネットワーク資源や計算機資源の用意されたストリーミングサーバを用意する必要があるが、現実的ではない。

2 つ目として、規模性に欠けるという問題がある。既存のサーバ・クライアントモデルでは、ユーザの送信したストリームを事業者の用意したライブストリーミングサーバがコピーし、不特定多数の受信者へ向けてユニキャストで送信する。そのため、受信者数の増加に従って中継サーバの負荷が高くなる。例えば、Ustream におけるライブストリーミングの過去最大の同時視聴者数は 137988 人である [5]。このライブストリーミングは、300Kbps と 700Kbps の複数の帯域幅のストリーミングが行われた。この人数を同時に収容するためには、視聴者全てが 300Kbps のストリームを視聴した場合でも約 39.4Gbps もの帯域が必要になる。これだけのネットワーク資源を用意することは現在ではコストが高い。そのため、サービス選択性のためにユーザ自身が資源を用意することは現実的ではない。

このように、既存の放送型ストリーミングサービスは、事業者によってサービスと配信基盤が一体となってサーバ・クライアントモデルのアーキテクチャで実現されている。そのため、ユーザのサービス選択性が狭められている。また、受信者数の増加に従って中継サーバを経由するトラフィックが増大し、中継サーバの CPU 負荷やネットワーク負荷が増加するという問題がある。

2.5 これからのライブストリーミングサービス

2.4 節で述べたように、既存のライブストリーミングサービスには、ユーザのサービス選択性と規模性の 2 点において問題がある。この問題は、2.3.1 節で示したユーザ指向なコンテンツのライブストリーミングを行うにあたって、大きな障害となる。ユーザ指向コンテンツでは、様々なユーザが様々な形態でストリーミングを行う。そのため、事業者の提供する形態によってサービスが制限されてしまうのは明らかなデメリット

である。また、ユーザ指向コンテンツでは、突発的に受信者数が増加する可能性がある。そのため、受信者数を予想できる有力なコンテンツホルダによるライブストリーミングと異なり、事前に配信能力を増強するといった対策が難しい。

これまでの議論から、小規模コンテンツホルダによるユーザ指向の強いコンテンツのライブストリーミングに着目した際、以下のような要求を示すことができる。

- ユーザのサービス選択性
- 急激な受信者数、送信者数の増加に耐えうる配信基盤

この要件を実現するため、本研究では、アプリケーションと配信基盤を分離し、パケット配送網自体が 1 対多型の通信を実現するモデルを提案する。ユーザは事業者が用意する中継サーバを利用するのではなく、直接 1 対多型のパケット配送網を利用する。配送網は 1 つの送信者から複数の受信までの配送網を構築し、経路上のルータがパケットを複製し、受信者までパケットを転送する。

この配送網は、任意のアプリケーションから利用できる必要がある。任意のアプリケーションが透過的に配送網を利用できるようにすることで、ユーザのサービス選択性が実現される。サーバ・クライアントモデルでは、ユーザが行えるサービスは中継サーバを用意する事業者によって提供される形態に制限される。しかし、ユーザが配送網に直接データを送信することができれば、ユーザは事業者によって提供されるサービスの形態に制限されることなく、自由にサービスを行うことができる。その結果、ユーザは任意のコンテンツを、任意のフォーマットで、任意のタイミングで、任意のクオリティで配信することができるようになる。

また、配送網自体がパケットを複製して複数の受信者に届けることによって、規模性の問題を解決する。既存のサーバ・クライアントモデルでは、2.4 節で述べたように、中継サーバがボトルネックとなって、突発的な受信者の急増や送信者の増加に対応することが難しく規模性に欠ける。また、一般ユーザが受信者数の増加に耐えられるだけの計算機資源やネットワーク資源を用意することは現実的ではない。そこで、配送網が任意の送信者から複数の受信者への 1 対多型通信用の経路を構築し、経路上のルータ群が適切な箇所でパケットを複製することで、ボトルネックを解消する。これによってユーザは潤沢な資源を用意することなく、自由に送信者となることができる。

本研究では、分離したアプリケーションと配信基盤のうち、配信基盤に着目し、世界規模でユーザが自由に送信者となることができる 1 対多型の通信モデルの実現を目指す。本研究で実現する 1 対多型の通信は、配送網によって実現される。配送網がアプリケーション透過性を持つことによって、ユーザによるサービス選択の自由を実現する。また、配送網中のルータが適切にパケットをコピーし、配送網自体が 1 対多型の配信基盤を担うことで、サーバ・クライアントモデルにおける中継サーバというボトルネックを解消する。これによって、ユーザが自由に送信者となれる環境を実現する。1 対多型の通信を実現するための配送網を構築する手法として、マルチキャストがある。本研究では、ユーザによる自由な 1 対多型の通信を実現するために、マルチキャストを用いることの有用性を検討する。

第3章　マルチキャスト通信モデル

本章では、1 対多型の通信を実現するためのマルチキャスト通信のモデルについて、構築する配送網の形態によって整理し、各モデルについて検討する。

3.1　マルチキャスト通信の実現手法

インターネット上で 1 対多型の通信を実現するための配送網の構築手法として、マルチキャストがある。マルチキャストの通信モデルの概要を図 3.1 に示す。マルチキャストでは、ネットワーク上のルータが任意の送信者から複数の受信者への配送網を構築する。送信者が送信したパケットは、配送網中のルータによってコピーされ、複数の受信者へと転送される。そのため、送信者が送信するストリームは 1 本でよく、受信者数の増加に従って送信者の負荷が上がることは無い。また、ストリームは配送網中のルータによって適宜コピーされるため、同一ストリームが同じリンクを通る可能性を減らし、余計なトラフィックを抑制することができる。

インターネット上でマルチキャストを実現するための既存手法は、大きく分けて IP 層で実現するものとオーバーレイネットワークを用いて実現するものの 2 種類に分類される。さらに、オーバーレイネットワークを用いるものは構築する配送網の形態によって、ハイブリッドマルチキャスト、アプリケーションレイヤマルチキャスト、オーバーレイマルチキャストの 3 種類に分類することができる [6]。各モデルの配送網構築手法について、図 3.2 に示す。

IP 層で実現する手法では、ネットワーク上のルータが IP 層でマルチキャスト配送網を構築する。一方オーバーレイネットワークを用いる手法のうち、ハイブリッドマルチキャストでは、IP 層でマルチキャストが可能なネットワーク同士をプロキシルータがオーバーレイで接続する。アプリケーションレイヤマルチキャストでは、エンドホストがオーバーレイでマルチキャスト配送網を構築する。オーバーレイマルチキャストでは、エンドホストを収容するルータがオーバーレイプロキシとして動作し、プロキシルータ間でオーバーレイのマルチキャスト配送網を構築する。

3.2　IP 層で実現する手法

IP 層で実現する手法では、IP 層での経路上のルータによってマルチキャスト用の配送網が構築される。IP 層で 1 対多型の配送網を構築するため、経路は送信者から複数の受信者への木構造となり、トラフィックは余計な経路を通らず、余剰なトラフィック

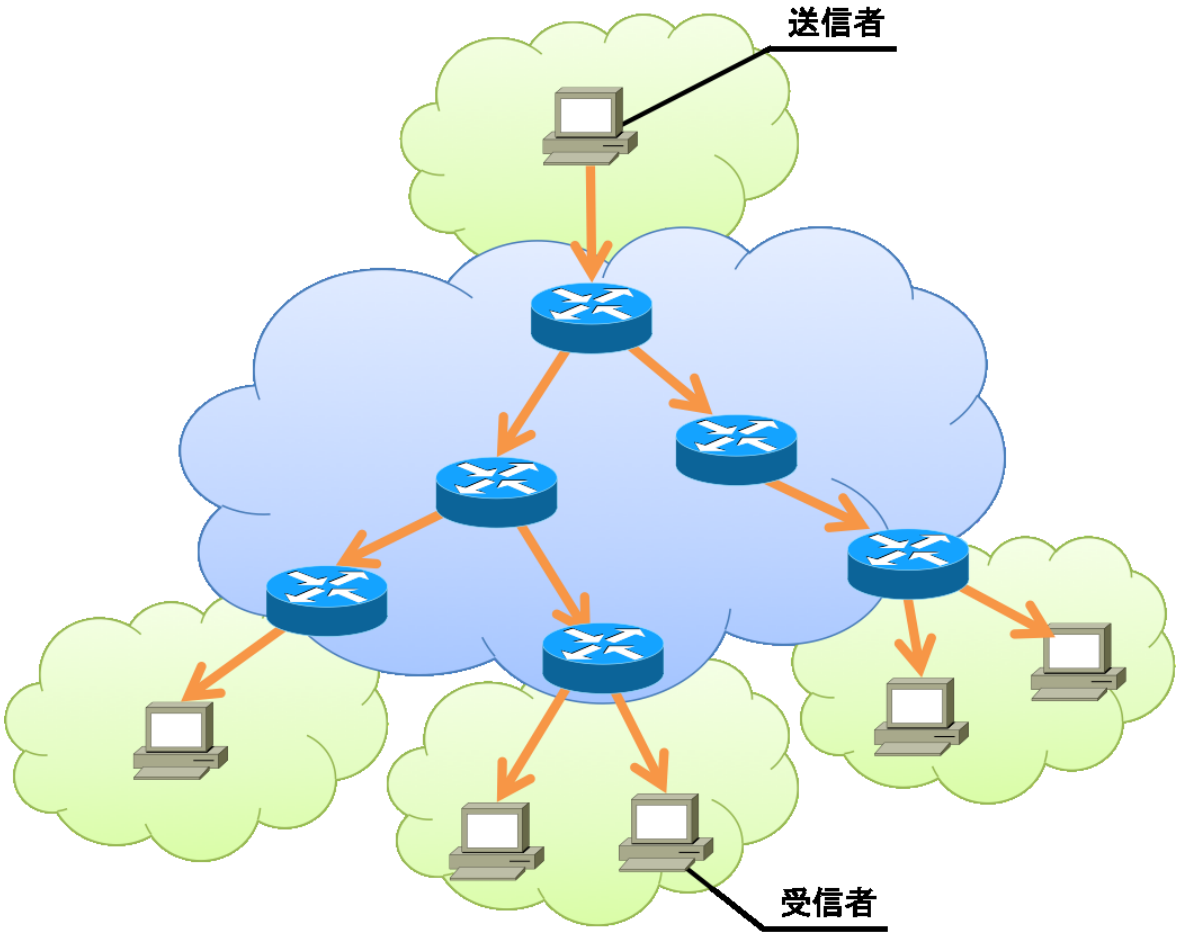

図 3.1: マルチキャスト配送網の概要

を抑制することができる。一方で、送信者から受信までの経路上の全てのルータがこの配送網を構築するプロトコルに対応する必要があるため、配送網の拡張が難しいという問題がある。

以下に、IP 層でマルチキャスト通信を実現するための既存研究を挙げる。

3.2.1 IP マルチキャスト

IP マルチキャストは、IP マルチキャストグループアドレスとして特別に割り当てられたアドレスを用いてホストをグループ化し、グループに対するマルチキャスト通信を実現する。IP マルチキャストは既に大抵のホストでサポートされており、ほとんどのホストは変更を加えること無く IP マルチキャスト配送網を任意のアプリケーションで使用可能である。しかし、IP マルチキャストは、IP マルチキャストのルーティングプロトコルとして現在広く用いられている Protocol Independent Multicast (PIM) の技術的な問題によって、ユーザによる自由な 1 対多型の通信を世界規模で実現するには至っていない。

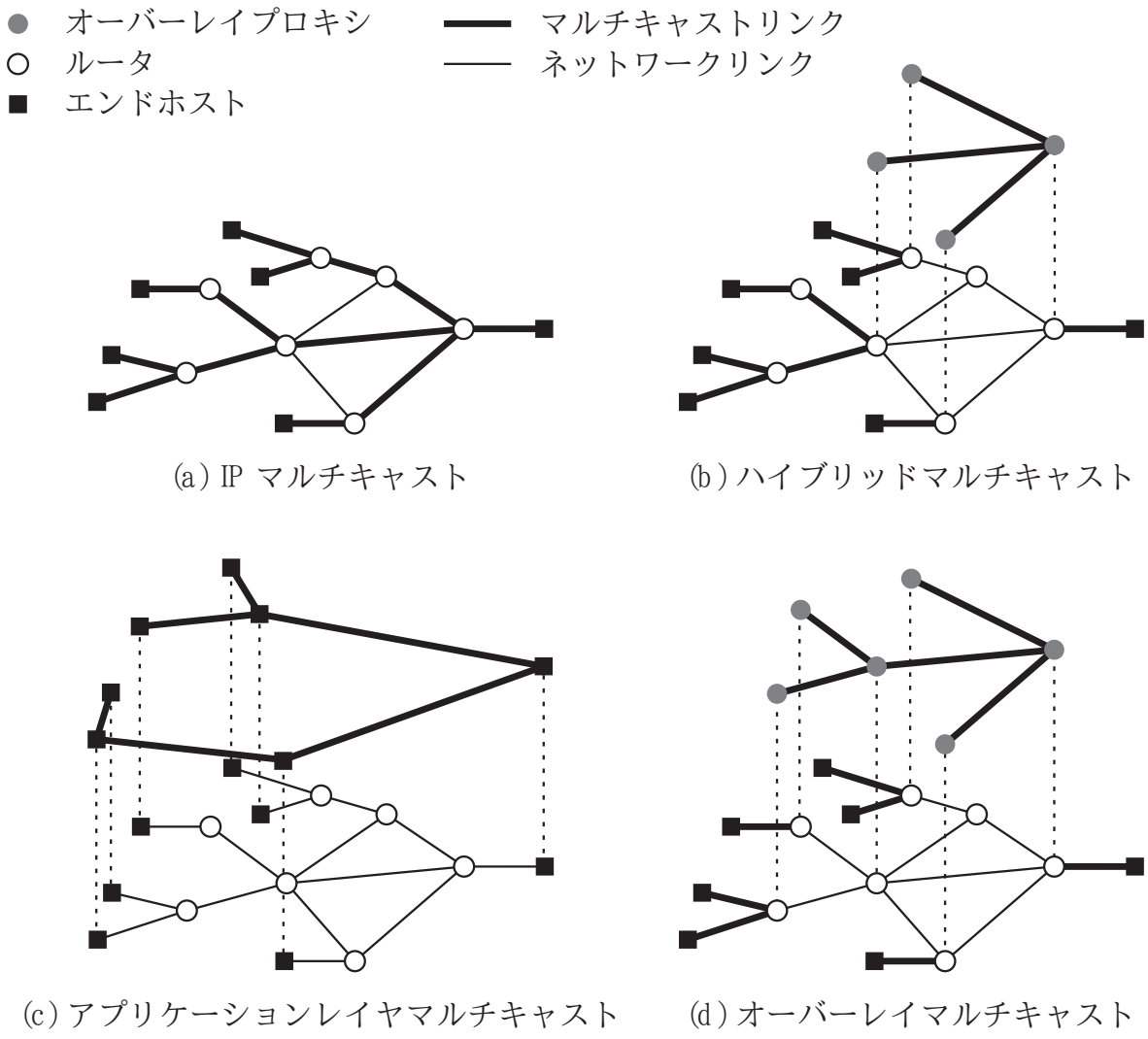

図 3.2: 配送網構築手法によるマルチキャスト通信モデルの分類

PIM は、モードによって構築する配送網の形態が異なる。Dense Mode[7] では、1 つの送信者を起点として、複数の受信者へ向かう木構造の配送網を構築する。そのため、送信者が複数の場合は送信者の数だけ配送網を構築する必要があり、網全体のルータの計算負荷が高いという問題がある。一方 Sparse Mode[8] では、あらかじめ設定されたランデブーポイント (RP) を用いて配送網を構築する。そのため、グループ数の増加に従って RP の負荷が高くなるという問題がある。

また、PIM ルータは送信者とマルチキャストグループアドレスというセットで経路表を保持する。ユーザが新しく受信者としてあるグループアドレスに参加すると、PIM-DM では送信者までユニキャストの経路表を遡ってマルチキャストツリーを作成する。一方 PIM-SM では、同様のマルチキャストツリーを RP まで作成する。このように、PIM では受信者が新しく参加するたびに網全体の再計算が行われるため、ルータの計算負荷が高いという問題がある。また、PIM は IP 層のプロトコルであるため、経路上の全てのルータで PIM を動作させる必要があり、導入コストが高いという問題もある。

3.2.2 Xcast

IP マルチキャスト以外に IP 層で 1 対多型の通信を実現する手法として、Xcast[9] がある。Xcast では、送信者はすべての Xcast データの受信者のアドレスをリストにしてパケットのヘッダに付与し、送信する。Xcast パケットを受信した Xcast ルータは、まず各受信者毎に通常のユニキャストの経路表を検索する。そしてネクストホップが同一である受信者アドレスをリストにして 1 つの Xcast パケットを作成し、ネクストホップへ転送する。このように、Xcast は受信者のアドレスリストをヘッダに付与し、分岐点でコピーすることによって 1 対多型の通信を実現している。

Xcast では、宛先アドレス毎ごとのネクストホップの検索は、通常のユニキャストの経路表によって行われる。そのため、マルチキャスト通信用に新たにトポロジーを構築する必要が無いという利点がある。また、実際の IP ヘッダの宛先として選択されるアドレスは 1 つであるため、Xcast に対応していないルータが経路中にあった場合は、Xcast パケットとは解釈されず、透過することができる。そのため、経路上の全てのルータが Xcast に対応している必要は無い。

しかし、送信者が全ての宛先をヘッダに付与する以上、パケット長は有限であるため、1 つのグループに許容できる受信者数には限界がある。また、送信者は受信者のリストをパケットに付与するため、あらかじめ受信者のリストを知る、ないしは受信者を知るための仕組みが別途必要となる。

3.3 オーバーレイで実現する手法

IP 層で実現する手法に対して、オーバーレイネットワークを用いてマルチキャスト配送網を構築する手法がある。オーバーレイで配送網を構築する場合は、経路上の全てのルータが網を構築するプロトコルに対応する必要がないため、網の拡張が容易であるという利点がある。

本節では、図 3.2 で示した、オーバーレイネットワークを用いてマルチキャスト配送網を構築する手法の各モデルについて特徴を挙げ、各モデルの既存研究について整理する。

- ハイブリッドマルチキャスト

  ハイブリッドマルチキャストは、既存の IP マルチキャストが利用可能なネットワーク間をオーバーレイで接続することによって、IP マルチキャスト到達性の拡大を図る。ハイブリッドマルチキャストには、既存の IP マルチキャスト網を拡大するため、網の拡張が容易であるという利点がある。一方で、オーバーレイで接続される既存の IP マルチキャスト網が運用者によって送信者を静的に設定するように設計されていた場合、ユーザが自由に送信者となることはできないという問題がある。

- アプリケーションレイヤマルチキャスト

アプリケーションレイヤマルチキャスト (ALM) は、ユーザがエンドホストにインストールしたアプリケーションがマルチキャスト通信を実現するための配送網を構築する。ALM には、バックボーンの配送網の構成に影響されることなくユーザがマルチキャスト配送網に参加することができるという利点がある。一方で、そのマルチキャスト配送網はアプリケーションごとに構築され、任意のアプリケーションが他のアプリケーションによって構築された網を利用することができない、という問題がある。また、ユーザが新しい ALM の配送網に参加するためには、その都度新しいアプリケーションをインストールする必要がある。

- オーバーレイマルチキャスト

  オーバーレイマルチキャストは、ネットワーク上に設置されたプロキシルータが、オーバーレイのマルチキャスト配送網を構築する。エンドホストは、プロキシルータからユニキャストまたは IP マルチキャストでトラフィックを受信する。オーバーレイマルチキャストは、ネットワーク資源や計算機資源が乏しいことが予想されるエンドホストを配送網から除外することで、高いパフォーマンスの実現を図る。

以下に、アプリケーションレイヤマルチキャストでマルチキャスト通信を実現するための既存研究を挙げる。

3.3.1 Auto IP Multicast Without Explicit Tunneling

Auto IP Multicast Without Explicit Tunneling(AMT)[10] は、ハイブリッドマルチキャストモデルの既存研究である。AMT では、既存の IP マルチキャスト網に設置された AMT-Relay と、非 IP マルチキャスト網内のネットワークに設置された AMT-Gateway の間で、IP マルチキャストパケットをカプセル化し転送することで IP マルチキャスト到達範囲を拡大する。

3.3.2 Universal IP Multicast Delivery

Univeral IP Multicast Delivery[11] は、ハイブリッドマルチキャストモデルの既存研究である。Universal IP Multicast Delivery では、既存の IP マルチキャストを利用可能なネットワーク同士をオーバーレイの配送網を用いて相互に接続する。また、マルチキャスト利用可能でないネットワークに存在するノードも含めて、木構造の配送網をオーバーレイで構築する。

3.3.3 Multicast Midleware

Multicast Midleware[12] は、アプリケーションレイヤマルチキャストモデルの既存研究である。Multicast Midleware では、エンドホストにインストールされたアプリケー

ションが、Scribe[13] による木構造のマルチキャスト配送網を構築する。また、Multicast Midleware では、仮想インターフェースを用いて実際に IP マルチキャストトラフィックを扱うことによって、アプリケーション透過性を実現している。

3.3.4 Split Stream

Split Stream[14] は、アプリケーションレイヤマルチキャストモデルの既存研究である。Split Stream では、エンドホストにインストールされたアプリケーションが Scribe[13] による木構造の配送網を構築する。送信者は送信するストリームをいくつかのストリームに分割し、それぞれを個別に構築した木構造の配送網を用いて送信することで、中継ノードのトラフィック負荷を分散する。

3.3.5 ScatterCast

ScatterCast[15] は、オーバーレイマルチキャストモデルの既存研究である。ScatterCast では、ネットワーク上に設置された SCX と呼ばれるプロキシノード間でオーバーレイの木構造のマルチキャスト配送網を構築する。受信ノードは、自身の近くにある SCX から IP マルチキャストまたはユニキャストでトラフィックを受信する。ScatterCast ではプロキシルータである SCX にモジュールを追加可能にすることで、トランスポートに拡張性を持たせている。

3.4 ユーザによる自由な 1 対多型通信実現のためのモデルの選択

本節では、3.2 節、3.3 節で述べた各モデルについて、ユーザによる自由な 1 対多型通信の実現という目的に向けた観点から比較、検討を行う。ユーザによる 1 対多型通信を実現するための配送網の要件は以下の通りである。

- アプリケーション透過性
- 配送網の拡張性
- 任意のユーザが送信者となれること.

ユーザによる自由な 1 対多型通信を実現するための各要件について、前節で挙げた既存研究がどの程度満たしているかを表 3.1 にまとめる。

2.5 節で述べたとおり、ユーザによる自由なサービス選択のため、配信基盤とストリーミングアプリケーションは分離され、アプリケーション非依存な配信基盤が配送網によって実現される必要がある。そのため、オーバーレイモデルで述べたアプリケーションレイヤマルチキャストモデルの関連研究 [14] のような、アプリケーション依存な配

表 3.1: 関連研究の比較

既存研究	透過性	拡張性	複数送信者の許容
IP マルチキャスト [7][8]	◯	×	×
XCast[9]	◯	×	◯
AMT[10]	◯	×	×
Universal IP Multicast Delivery[11]	◯	◯	×
Multicast Midleware[12]	◯	◯	×
Split Stream[14]	×	◯	×
ScatterCast[15]	◯	◯	×

送網を構築する手法ではアプリケーション透過性を実現することはできない。そこで本研究では、アプリケーション透過性を実現するために、IP 層の機能であり、既に大抵のホストでサポートされている IP マルチキャストを用いる。

一方、IP 層で実現する手法では、ユーザが自由に送信者となることはできないという問題がある。IP 層で実現する手法 [7][8] では、送信者から複数の受信者への木構造の配送網を構築するため、送信者数分の配送網を構築する必要がある。そのため、IP マルチキャストを用いた既存サービスでは、運用者が予め送信者を静的に設定するものがほとんどである。さらに、IP 層でマルチキャスト配送網を実現するにはネットワーク上の全てのルータがその配送網を構築するプロトコルに対応する必要があるため、導入コストが高く、網を拡大するのが難しいという問題がある。

また、複数のユーザが自由に送信者となれる配送網を構築するためには、ルータ間で構築する配送網のトポロジについても考慮する必要がある。IP 層で実現する手法、オーバーレイを用いる手法を問わず、既存研究の多くは、1 つの送信者を起点とした複数の受信者へ向かう木構造のトポロジを構築するため、送信者数分の配送網を構築する必要がある。そのため、送信者の増加に従って配送網中のルータの負荷が増加するという問題がある。

そこで本研究では、配送網の構築モデルにオーバーレイマルチキャストモデルを用いる。エンドホストがプロキシルータからトラフィックを受け取る際に IP マルチキャストを用いることで、配送網のアプリケーション透過性を実現する。そして、プロキシルータ間の配送網をオーバーレイネットワークで構築することによって、バックボーンネットワークの構成に影響されることなく配送網を拡大することができる。また、既存研究の多くは木構造の配送網を構築するため、複数の送信者を許容することが難しい。そこで本研究では、木構造ではなく、リングトポロジでプロキシルータ間のオーバーレイネットワークを構築することで複数の受信者を許容する配送網を構築する。

第4章　アプローチ

本章では、2 章の議論を受けて想定環境を整理した上で、3 章を踏まえて本研究で提案するシステムの概要について述べる。

4.1　想定環境

本研究の目的は、世界規模でのユーザによる自由な 1 対多型通信の実現である。そのため、想定環境の規模は、世界規模を想定する。既存のインターネットにおいて、ユーザが自由に 1 対多型通信を利用することはできない。そこで、本研究の提案手法を用いることによって、世界中のユーザが 1 対多型通信を用いたサービスを自由に行えるようになることを目指す。この目的の実現のためには、世界中の全ユーザを収容するだけの膨大な数のルータが必要となる。そこで、提案手法には、膨大な数のルータで配送網を構築しても動作するだけの規模性が求められる。そして、ユーザはその配送網を利用し、任意のアプリケーションを用いて世界中のユーザに向けてサービスを展開できる必要がある。

本研究では想定アプリケーションとしてストリーミングアプリケーションを用いる。2.2 節で述べた通り、既存の 1 対多型の通信を用いるアプリケーションとして、メッセージングを行うアプリケーションと、ストリーミングを行うアプリケーションがあげられる。メッセージングに 1 対多型の通信を用いるアプリケーションによってやり取りされるデータは、シグナリング用のパケットであり、比較的サイズが小さく、帯域や遅延に対して厳しい要求を持たない。一方ストリーミングアプリケーションは、帯域幅やパケットロス、遅延やジッタがサービスの品質に大きく影響する。遅延に関しては、国際電気通信連合の勧告 G.114[16] の中で、一方向通信のアプリケーションにおいて、許容できるネットワークによる遅延は 400ms 以内とされている。本研究では、このように配送網への要求のより厳しいストリーミングアプリケーションを想定アプリケーションとする。

4.2　機能要件

世界規模でのユーザによる自由な 1 対多型の通信を実現するためのマルチキャスト配送網に対する要件は下記の通りである。

- マルチキャスト配送網の構築

- 送信者が増加しても負荷が上昇しない配送網
- 送信者が送信する単一ストリームを適切に配送網中のルータがコピーすること
- 任意の送信者から複数の受信者へのデータ配送
- 配送網のアプリケーション透過性

ユーザによる自由な1対多型の通信を実現するためには、全てのホストが送信者となれる必要がある。そのため、PIM[7][8] やオーバーレイを用いる多くの既存手法 [12][14][15] のように、送信者毎に木構造の配送網を構築することは、ルータ処理負荷の観点から現実的ではない。また、既存のサーバ・クライアントモデルは中継サーバから複数受信者への通信がユニキャストであるために規模性が無く、ユーザが直接送信者となることが現実的ではなかった。そこで、送信者の送信するパケットを配送網中のルータが適切にコピーし、複数の受信者へと配送する必要がある。また、構築された配送網は任意のアプリケーションから使用可能である必要がある。

4.3 システム概要

本研究では、前節で示した機能要件を実現するマルチキャスト配送網構築手法として、Overlay Any Source Multiast (OASM) を提案する。図 3.2 に対して、OASM の構築する配送網のモデルを図 4.1 に示す。OASM はオーバーレイマルチキャストモデルのアーキテクチャである。OASM ルータはオーバーレイプロキシルータとして実装され、エンドホストを収容するネットワークに接続される。そして、エンドホストに対して IP マルチキャストによる接続性を提供する。OASM ルータは、IP マルチキャストによるグループ管理を行うため、エッジネットワークに対して IGMP Queryer として動作する。また、IP マルチキャストパケットの配送網として、IP マルチキャストグループアドレス毎にリングトポロジのオーバーレイネットワークによる配送網を構築する。

OASM のシステム概要を図 4.2 に示す。まず、IGMP プロキシがエンドホストからの IGMP Report を受信し、また IGMP Query を送信することで IP マルチキャストアドレスグループの管理を行う。IGMP プロキシによる IP マルチキャストグループへ参加と離脱をトリガーとして、トポロジ構築機構がグループアドレス毎にオーバーレイで構築された配送網への参加や離脱を行う。また、トポロジ管理機構が参加や離脱などを含む配送網のトポロジの変更を受けて、経路表を更新する。経路表には、IP マルチキャストグループアドレス毎にネクストホップである OASM ルータのアドレスが保存される。パケット転送機構は、エンドホストの送信する IP マルチキャストパケットを受信し、経路表に従ってパケットを IP ヘッダからカプセル化し送信する。また、カプセル化されたパケットを受信し、カプセル化を解いてエッジネットワークへと送信する。

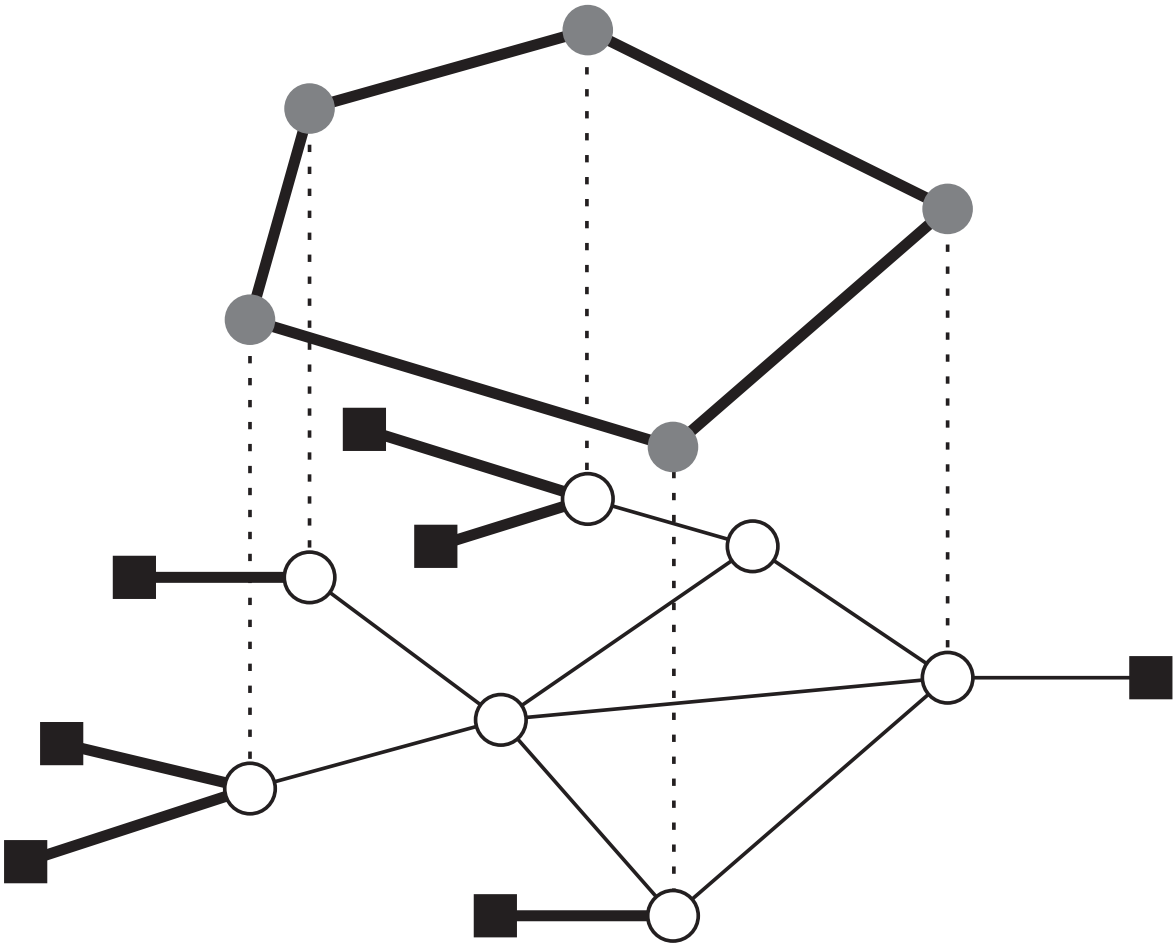

図 4.1: OASM の構築するマルチキャスト配送網

OASM では、IP マルチキャストを用いることでアプリケーション透過性を実現する。IP マルチキャストは既に大抵のホストでサポートされている。そのため、エッジネットワークに対して IP マルチキャストを用いることによって、多くのホストが変更を加えることなく OASM の構築するマルチキャスト配送網を利用することができる。また、IP マルチキャストは IP 層の機能であるため、上位層のアプリケーションに対して透過性を持つ。これにより、ユーザは任意のアプリケーションで OASM の配送網を用いて様々なサービスを展開することができる。

また OASM では、複数受信者を許容する配送網を実現するために、オーバーレイプロキシルータである OASM ルータ間の配送網を、リングトポロジのオーバーレイネットワークで構築する。配送網をリングトポロジで構築する利点として、以下の点が挙げられる。

- 複数送信者で共通の配送網

  リングトポロジでは、送信者毎のトポロジを構築する必要が無い。PIM のような送信者を基点とした木構造の場合、各ルータは送信者毎に配送網を構築する必要がある。そのため、送信者が増加するとルータが保持すべき経路の数も増加し、

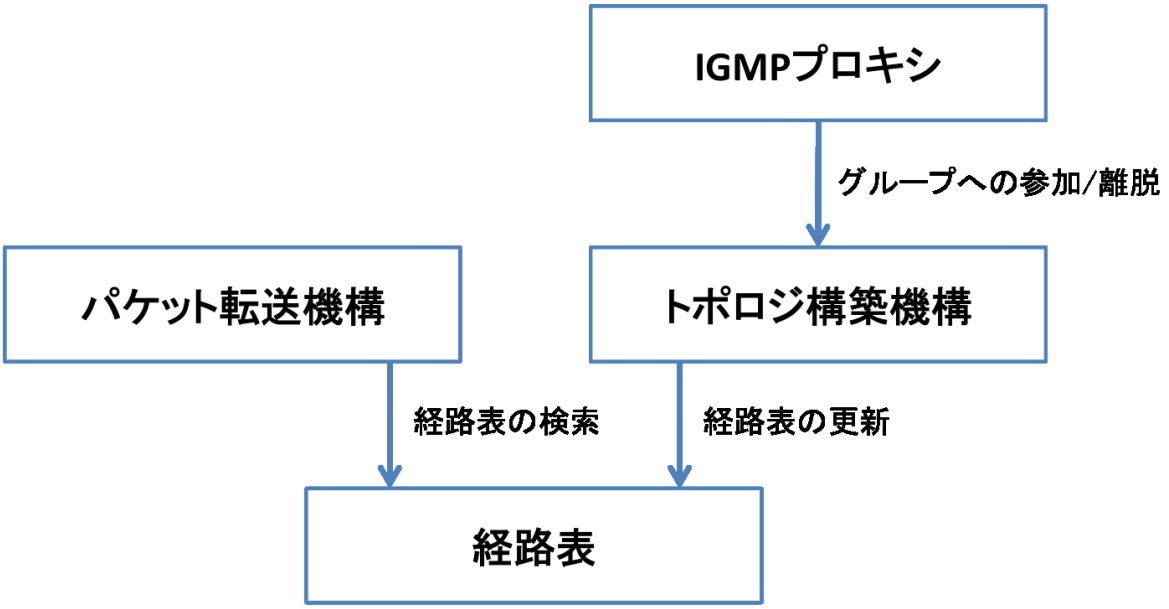

図 4.2: OASM のシステム概要

網全体のルータの負荷が増加するという問題がある。しかし、リングトポロジであれば、送信者に関係無く、パケットは配送網を 1 周することで、そのパケットを受信すべき全てのルータへ配送することができる。このように、リングトポロジであれば複数の送信者で同一の配送網を利用することができる。

- ある宛先に対するネクストホップが常に 1 つ

  リングトポロジでは、ある宛先アドレスに対して保持すべきネクストホップアドレスは常に 1 つである。木構造では、あるルータは 1 つの宛先アドレスに対して複数のネクストホップを持ち、ネクストホップの数だけパケットをコピーする必要がある。そのため、構築されるトポロジによっては、1 つのルータが何度もパケットをコピーする必要が生じ、ボトルネックとなる可能性がある。しかし、リングトポロジではある宛先に対するネクストホップは常に 1 つであり、パケットのコピーも 1 度でよい。そのためパケットコピーによるボトルネックが発生しない。

- 網構築の容易さ

  リングトポロジでは、ルータが参加、離脱した際の網の再構築が容易である。木構造では、あるルータがその網から離脱した際、網全体で経路を再計算する必要がある。しかし、リングでは、ルータが離脱して切れた辺のみを修復すればよく、網全体で経路を再計算する必要が無い。また、ルータが新規に網に参加する際も、参加ルータの入るべき辺が決まれば、その前後のルータが経路を変更するだけでよい。そのため、経路変更の影響範囲を局所化することができる。

OASM では、このリングトポロジの配送網を IP マルチキャストアドレスグループ毎に構築する。配送網をグループアドレス毎に構築することによって、あるグループア

ドレスのトラフィックを、そのグループアドレスのパケットを受け取るべきルータだけに限定できる。これによって、グループに参加していないルータへの余剰なトラフィックを削減する。しかし、リングトポロジでは、木構造に比べてパケットが通過すべきルータ数が増加するため、帯域や遅延に関してボトルネックが予想される。そこで本研究ではまず、より効率的なリングトポロジを構築するための予備実験を行った。

4.4 予備実験

リングトポロジにおいてパケットを全ての受信者に配送するためには、パケットはリング状の配送網を 1 周する必要がある。そのため、パケットは経路上の全てのルータを通る。その際、経路上のあるルータが極端に狭い帯域を持つ場合、そのルータがボトルネックとなって網全体の転送性能が低下することが考えられる。また、リングトポロジをオーバーレイで構築するため、配送網中にネットワーク的に離れたノード同士を繋ぐようなオーバーレイのリンクが多い場合、配送網全体の遅延が増加することが考えられる。

このように、リンクに性能差があるノードを用いてオーバーレイネットワークでリングトポロジを構築する場合、ノードの並び順が配送網全体の性能に影響することが予想される。そこで本研究では、OASM の具体的なトポロジ構築手法を提案する前に、オーバーレイネットワークでリングトポロジを構築した際の配送網全体の帯域と遅延に関する予備実験を行った。計測項目として、まず、5 台のノードでリングトポロジを構築し、各ノードのリンク帯域を変化させながら一定のトラフィックを流し、配送網全体の帯域性能を計測した。また、同様に 5 台のノードでリングトポロジを構築し、各ノードのリンクの遅延を変化させながら、一定のトラフィックを流しつつパケットが配送網を 1 周するのにかかる時間 (RTT) を計測した。

4.4.1 実験環境

実験環境として、1 つのルータの配下に 5 つのセグメントを設置し、各セグメントに OASM ルータとして動作するノードを設置した。この実験環境のトポロジ図を図 4.3 に示す。各ノードのアップリンクとなる NIC(eth0)、エッジネットワークとなる NIC(eth1) の両方をスイッチに接続し、スイッチからルータへ接続した。そして、このスイッチにて各ノードの送信した IP マルチキャストパケットの数を計測した。オーバーレイのリングトポロジは、ノード 1 から順番にノード 1、ノード 2、ノード 3、ノード 4、ノード 5、となるように構築した。リングの始点となるノード 1 のエッジネットワークと、リングの終点となるノード 5 のエッジネットワークを 1 台の計測ノードに接続し、この計測ノードを用いてトラフィックの生成、パケットが配送網を 1 周するまでの時間の計測を行った。各ノードのリンクの性能差を実現するために、ルータの各ポートでトラフィックシェーピングを行うことで帯域幅の差を生み出した。また遅延については、

表 4.1: 予備実験機材

	CPU	メモリ	NIC
ノード 1	Intel Core 2 Duo E6750 2.66GHz	2G	Intel 82546GB
ノード 2	Intel Core 2 Duo E6750 2.66GHz	2G	Intel 82546GB
ノード 3	Intel Xeon 2.40GHz	512M	Intel 8254EB
ノード 4	Intel Xeon 2.40GHz	512M	Intel 8254EB
ノード 5	Intel Xeon 1.80GHz	256M	Intel 8254EB
計測ノード	Intel Core 2 Duo 1.66GHz	1G	Intel PRO/1000

各ノードのアップリンクとなる NIC に tc(Linux Traffic Control) を用いて遅延を発生させた。実験に用いた機材の詳細を下記の表 4.1 に示す。

全ての PC の OS は Debian Linux 6.0.3 squeeze、Linux Kernel 2.6.32-5 を用いた。また、トラフィックシェーピングを行うルータには Juniper SRX240h を、パケット数を計測するスイッチには Cisco WS-C6509 を用いた。

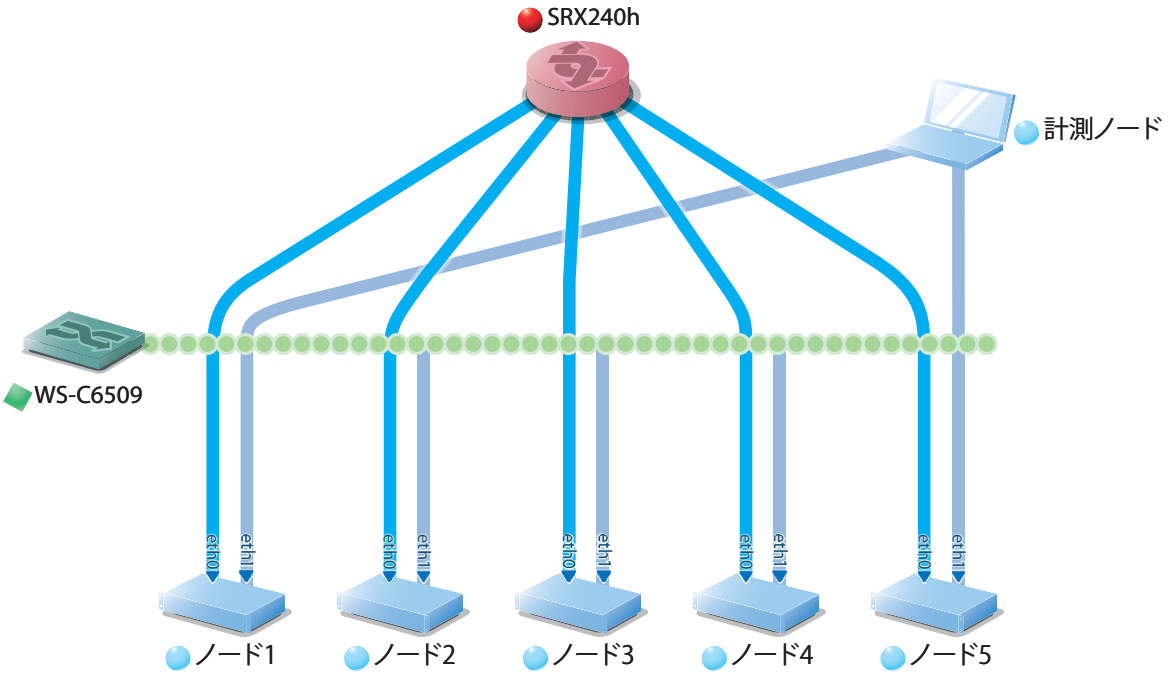

図 4.3: 予備実験のトポロジ

4.4.2 帯域に関する予備実験

帯域性能についての実験では、OASM の設置されるエンドユーザのアクセス回線を想定し、ノード毎にリンク帯域が 100Mbps と 50Mbps になるよう SRX240h を用いて帯域制限をかけた。帯域制限をかけたノード 1 からノード 5 までを用いて構築したリン

グトポロジは下記の表 4.2 の通りである。各トポロジについて、ノード 1 のエッジネットワークから計測ノードが iperf を用いて 25Mbps から 300Mbps までのトラフィックを 25Mbps ずつ増加させながら流した。そして、各帯域について、WS-C6509 において各ノードが送信した IP マルチキャストパケット数を計測した。

表 4.2: ノード毎に帯域を制限した実験トポロジ

	ノード 1	ノード 2	ノード 3	ノード 4	ノード 5
トポロジ 1 (100-100-50-50-50)	100Mbps	100Mbps	50Mbps	50Mbps	50Mbps
トポロジ 2 (100-50-50-100-50)	100Mbps	50Mbps	50Mbps	100Mbps	50Mbps
トポロジ 3 (50-50-100-100-50)	50Mbps	50Mbps	100Mbps	100Mbps	50Mbps

各トポロジについて、全てのパケットを転送できた場合の各エッジネットワークに到達する IP マルチキャストパケット数の合計に対して、各ノードが実際に転送できたパケット数の合計が、どの程度パケットを消失していたかを図 4.4 に示す。全てのパケットを転送できた場合のパケット数は、計測ノードが生成した IP マルチキャストパケットの数に、送信者となるノード 1 以外のエッジネットワークの数 (4) を掛けた値である。図 4.4 から、ボトルネックとなる 50Mbps のリンクを持つノードがリングの後方にいる方が配送網全体のパケットロス率が少ないことがわかる。これは、ボトルネックとなるリンクがリングの前方にあると、そこでトラフィックが制限され、後方の太い帯域を持つノードのリンクを活かすことができないためである。このことから、太いリンクを持つノードから段階的に細いリンクを持つノードへと並べていけば、配送網全体のパケットロスを小さく押さえてスループットを上げることができるといえる。

4.4.3 遅延に関する予備実験

遅延についての実験では、tc を用いて各ノードに 0 または 10ms の遅延をかけた。遅延をかけたノード 1 からノード 5 までで構築したリングトポロジは下記の表 4.3 の通りである。各トポロジについて、ノード 1 のエッジネットワークから計測ノードが iperf を用いて 25Mbps から 300Mbps までのトラフィックを 25Mbps ずつ増加させながら流した。そして、各帯域について、計測ノードがノード 1 のエッジネットワークから計測用の IP マルチキャストパケットを送信し、ノード 5 のエッジネットワークから計測用の IP マルチキャストパケットを受信するまでの時間を計測した。

表 4.3: ノード毎に遅延をかけた実験トポロジ

	ノード 1	ノード 2	ノード 3	ノード 4	ノード 5
トポロジ 1 (10-10-0-0-0)	10ms	10ms	0ms	0ms	0ms
トポロジ 2 (0-0-10-10-0)	0ms	0ms	10ms	10ms	0ms

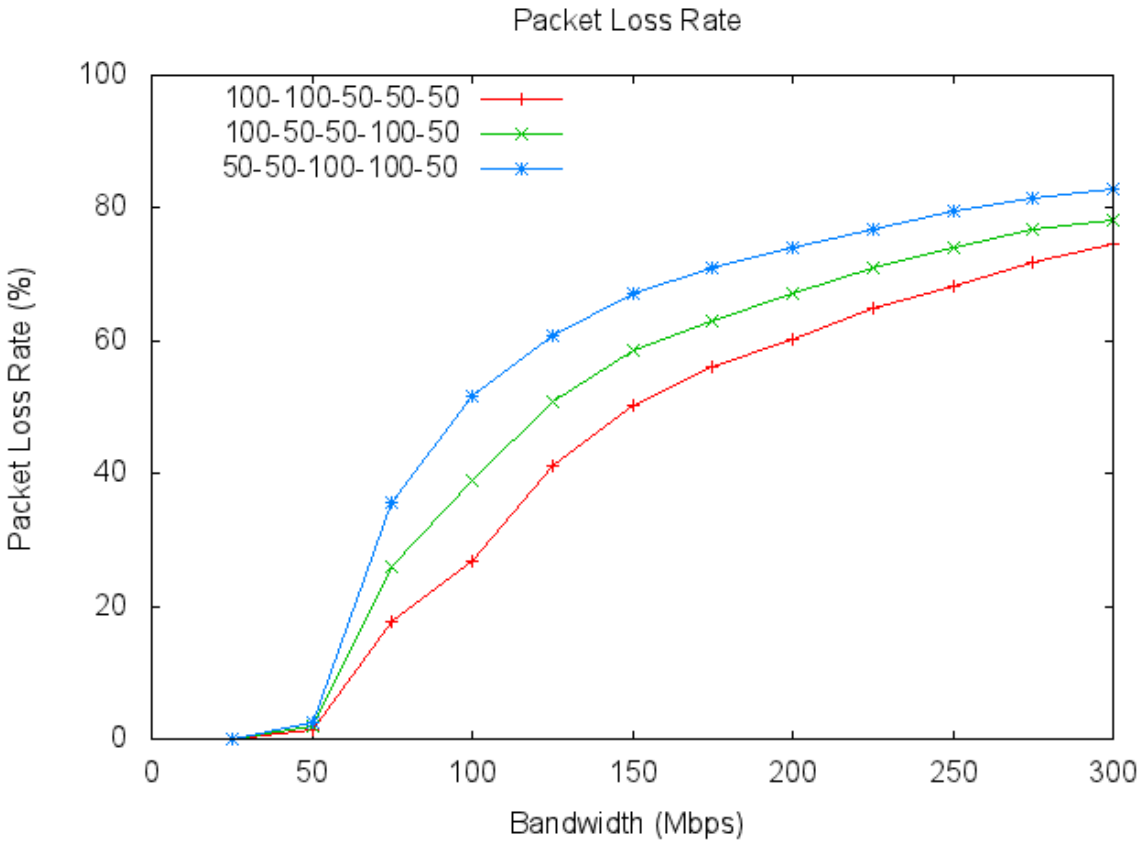

図 4.4: 帯域についての予備実験:トポロジ毎のパケットロス率

トポロジ 1(10-10-0-0-0) で計測した RTT のグラフを図 4.5 に示す。また、トポロジ 2(0-0-10-10-0) のトポロジで計測した RTT のグラフを図 4.6 に示す。図 4.5 と図 4.6 から、異なる遅延のあるリンクを持つノードの並び順は、ジッタの傾向には影響が無いことがわかる。両トポロジともに遅延は 10ms の箇所が 2 箇所であり、パケットがリングを 1 周するのに要する時間は同程度である。また、RTT のばらつきの傾向にも変化がなく、RTT の増加傾向も同程度である。一方、両方のトポロジにおいて、計測ノードからかけたトラフィックが 200Mbps 以上になると、RTT が増加している。これは、トラフィック量が 200Mbps を超えると OASM のキューにパケットが溜まり始め、1 台のノードがパケットを転送するのにかかる時間が増加するためである。

全てのノードのリンクに遅延をかけ、その遅延を 10ms から 300ms まで、10ms 毎に増加させた際の各遅延での RTT の分散を図 4.7 に示す。図 4.5、4.6 から、RTT は流したトラフィック量が 200Mbps を境に大きく変化することがわかっている。そのため、流したトラフィック量が 200Mbps 未満と以上で分けて分散を計算した。図 4.7 から、遅延が 50ms を超えると、200Mbps 以上のトラフィックをかけた際にジッタが増大することがわかる。これは、遅延を発生させるために用いた tc のキューが溢れたことが原因だと考えられる。実ネットワークにおいてジッタが発生する要因もルータのキューが埋まるためであることから、トラフィック量が多い場合、網全体の遅延が短い方がジッ

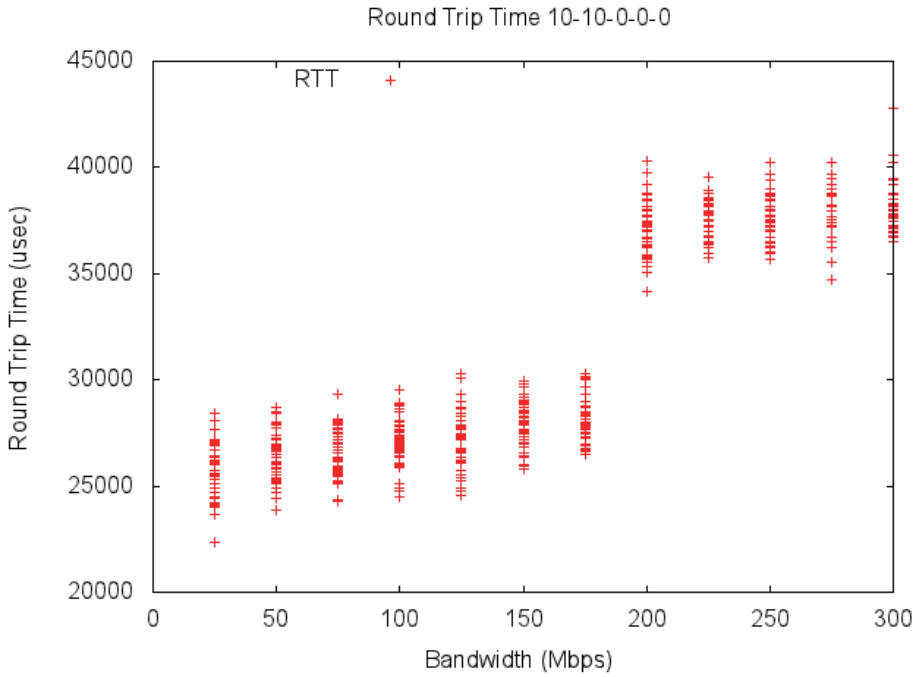

図 4.5: 遅延についての予備実験:トポロジ 1

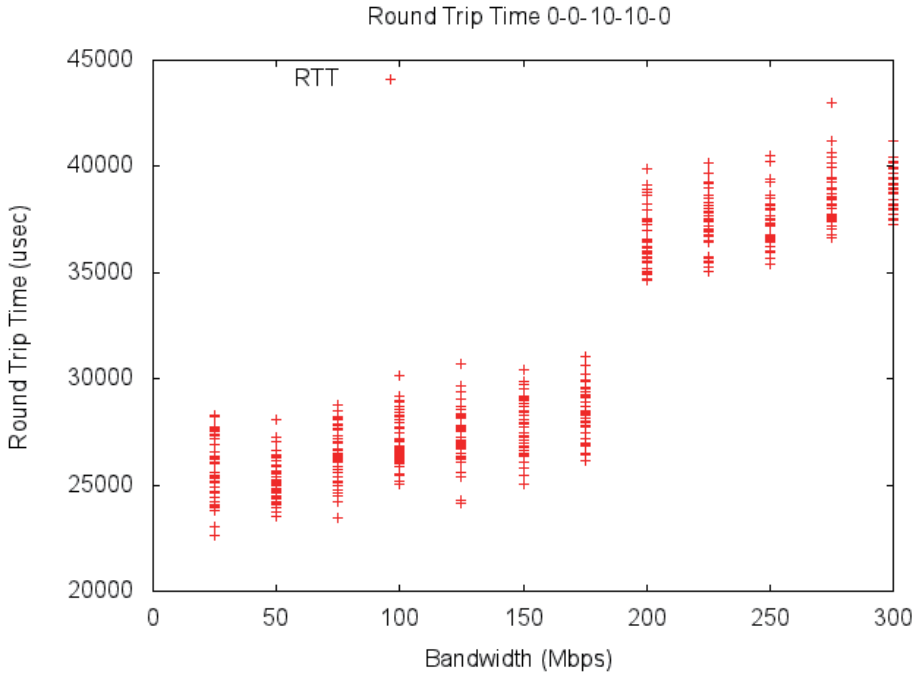

図 4.6: 遅延についての予備実験:トポロジ 2

タを抑制できると言える。

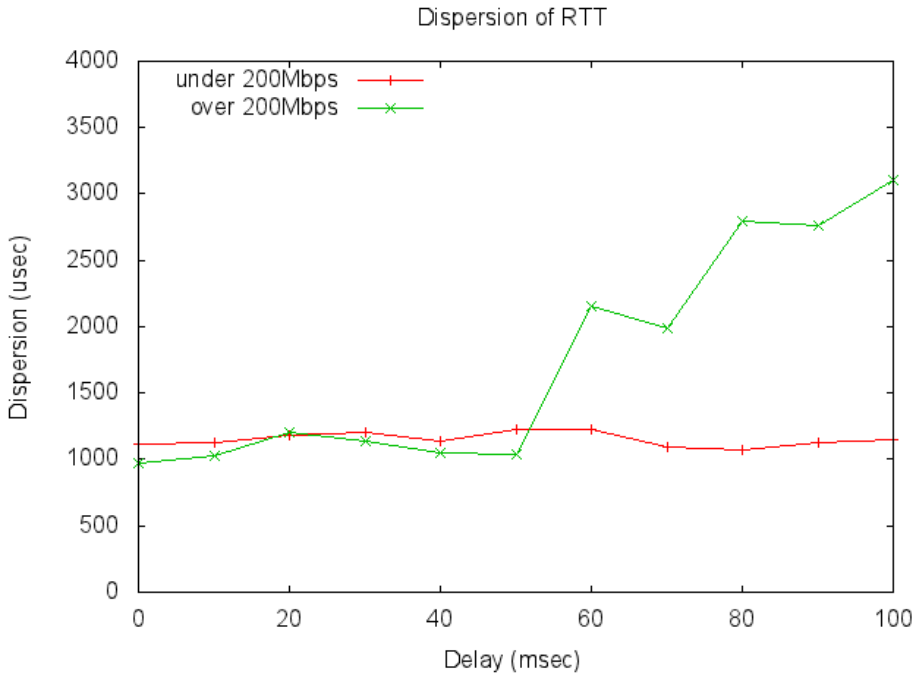

図 4.7: 遅延についての予備実験:ばらつき

4.4.4 予備実験から

以上の帯域と遅延に関する予備実験から、リングトポロジにおけるノードの並び順を決定するために、帯域と遅延をメトリックとして利用できることがわかった。リンクの性能に差があるノードを用いてオーバーレイのリングトポロジを構築する場合、ノードの並び順は、太いリンク帯域を持つノードから狭いリンク帯域を持つノードへと並べていくことで、配送網全体の転送性能を上げることができる。また、なるべくノード間の遅延の短い、ネットワーク的に近いノードを選んで並べていくことで、パケットが全ルータへと転送されるまでの時間を短くすると同時に、ジッタを抑制することができる。よって、OASM では、太い帯域のリンクを持つノードから細い帯域のリンクを持つノードへ、なるべくネットワーク的に近いノードを選択して並べていく。

4.5 効率的なリングトポロジ

4.4 節での予備実験の結果から、リングトポロジのオーバーレイネットワークを構築するにあたって、帯域と遅延がメトリックとして利用できることがわかった。網全体の転送性能を上げるためには、太いリンク帯域を持つノードから順にならべることが有効であること、網全体の遅延とジッタを抑えるにはネットワーク的に近いノード同士を選択して並べていくことが有効であることがわかった。OASM では、リングトポロジのオーバーレイ配送網を構築するにあたって、この 2 つの要件を実現するため、以下の 2 つのメトリックを導入する。

- Link Class：事前にユーザによって設定される帯域情報
- 仮想座標：vivaldi[17] によるノードのネットワーク上での座標

Link Class は帯域についての設定値である。ノード間のオーバーレイリンクの可容帯域を知るためには、可容帯域分のトラフィックを実際に流して計測する必要がある。しかし、数十台ものノード間で実際にトラフィックを流してオーバーレイリンクの帯域を計測することは、計測トラフィック量やコストの観点から現実的ではない。そこで OASM では、事前に OASM ノードの設置者が設定する値として Link Class を導入する。OASM はユーザを収容するエッジネットワークに設置される。そのため、OASM ノードにとっての帯域のボトルネックは、そのエッジネットワークのアクセス回線になることが予想される。そこで、ユーザは OASM ノードを設置する際に、あらかじめ接続形態などから推測できる可容帯域に従って Link Class を設定する。OASM では、事前に設定された Link Class を帯域のメトリックとして網構築に用いることで、実際に計測トラフィックを流す必要を無くし、余剰トラフィックを削減する。

一方、仮想座標は vivaldi[17] によって求められる、ネットワーク上でのノードの座標である。このノードの仮想座標を用いて、配送網構築の際に、近いノード同士を選択して並べることができる。vivaldi は、ネットワーク上に分散したノード同士が、ノード間の RTT を用いて自律的に自ノードの座標を推定するアルゴリズムである。vivaldi では、アプリケーショントラフィックを用いてノード間の RTT を計測し、その RTT をユークリッド距離として扱うことで自ノードの座標を推定する。また、ノード間の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 を計測する度に座標を修正していく。vivaldi の座標修正アルゴリズムの基本は以下である。

$$E = \sum_i \sum_j (L_{ij} - ||x_i - x_j||)^2 \tag{4.1}$$

E は推定される誤差、L_{ij} はノード i とノード j 間の実測した RTT、x_i はノード i の座標、x_j はノード j の座標である。vivaldi は、この E を最小にすることを目指して誤差を修正していく。バネの原理を利用した最も単純な vivaldi の誤差修正アルゴリズムは以下の通りである。

$$e = rtt - ||x_i - x_j|| \tag{4.2}$$

$$dir = u(x_i - x_j) \tag{4.3}$$

$$f = dir \times e \tag{4.4}$$

$$x_i = x_i + \delta \times dir \tag{4.5}$$

e はノード i とノード j の推定座標間の距離と、計測した RTT の差であり、推定した座標の誤差である。dir は誤差の方向、f は誤差を修正するベクトル、δ は定数である。最も単純な vivaldi の誤差修正アルゴリズムでは、RTT を計測する度に誤差の方向へ δ ずつ座標を移動することで誤差を修正していく。この δ の値は、大きい程座標が早く収

束する。しかし、大きすぎる場合座標が発振し、収束しない可能性が高まる。OASMでは、配送網構築のためのメッセージパケットを用いてRTTを計測することで、ノードの座標を更新していく。

OASMでは、Link Classとvivaldiを用いた位置情報の2つの情報を用いて一意なノードIDを決定する。そして、そのノードIDの順番にノードを並べることでリングトポロジの配送網を構築する。オーバーレイでリングトポロジの配送網を構築するにあたって、太いリンク帯域を持つノードから狭いリンク帯域を持つノードへ、かつネットワーク的に近いノード同士を選択して並べなければならない。そのためには、vivaldiによる2次元座標にLink Classという帯域情報を加えた3次元情報から、近傍性を保ったままノードIDを生成する必要がある。OASMでは、この3次元の情報を1次元のIDにマップするため、空間充填曲線の1つであるルベーク曲線[18]を用いる。ルベーク曲線は、図4.8の様に、座標をZ状に辿ることで1つの曲線で空間を埋め尽くし、多次元情報を1次元情報にマップすることができる。このルベーク曲線を用いて、Link Classと位置情報の近傍性を保ったまま1次元のノードIDを決定し、IDの順にノードを並べることでリングトポロジーを構築する。

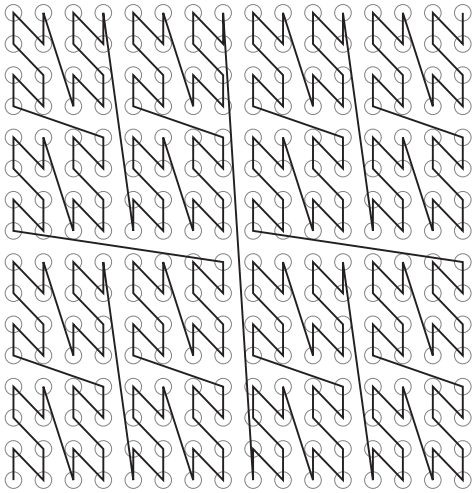

図 4.8: ルベーク曲線

4.6 動作概要

OASMは、用途に合わせて異なる2種類のオーバーレイネットワークを構築する。本節では、構築するオーバーレイネットワークの用途、またそれらのオーバーレイネットワークを構築、維持するためのプロトコルの挙動について述べる。

4.6.1 構築するオーバーレイネットワーク

OASMは、用途に合わせて以下に示す2種類のオーバーレイネットワークを構築する。その概要を図4.9に示す。

- All-Node Ring

  OASM 網に参加する全てのルータによって構築されるリングトポロジのオーバーレイネットワーク。あるルータが新規に Group Ring に参加する際に、参加処理を行うべき相手を発見するために利用される。

- Group Ring

  IP マルチキャストグループアドレス毎に、そのグループアドレスに参加しているルータによって構築されるリングトポロジのオーバーレイネットワーク。IP マルチキャストパケットの配送網として利用される。

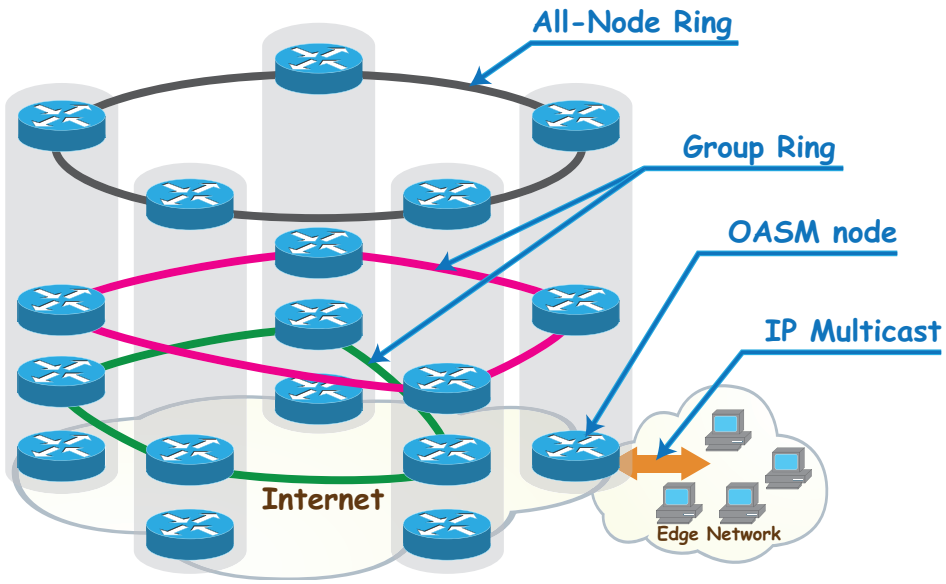

図 4.9: OASM の構築するオーバーレイネットワークの概要

All-Node Ring は、あるルータが新しく Group Ring に参加する際の宛先発見に利用される。All-Node Ring は、4.5 で述べた手法で生成された ID の降順にノードが並んだ、リングトポロジのオーバーレイネットワークである。ルータが OASM 網へ参加するには、まず All-Node Ring に参加する必要がある。All-Node Ring に参加するには、事前に OASM 網に参加しているルータのアドレスを静的に設定し、そのアドレスに対して参加処理を行う。また、ルータが新しく Group Ring へ参加するには、All-Node Ring に対して、参加したい Group Ring に既に参加しているノードを検索する要求を送信する。一方 Group Ring は、マルチキャストデータの配送網として利用される。Group Ring は、4.5 で述べた手法で生成された ID の降順にノードが並んだ、リングトポロジのオーバーレイネットワークで構築される。Group Ring の概要を図 4.10 に示す。

1 つの IP マルチキャストグループアドレスについての Group Ring は、図 4.10 のように構成される。ユーザを収容するサブネットに設置された OASM ルータは、ホストからの IGMP Report を受け取り、指定されたグループアドレスの Group Ring に参加する。Group Ring は、そのグループの IP マルチキャストグループアドレスに参加す

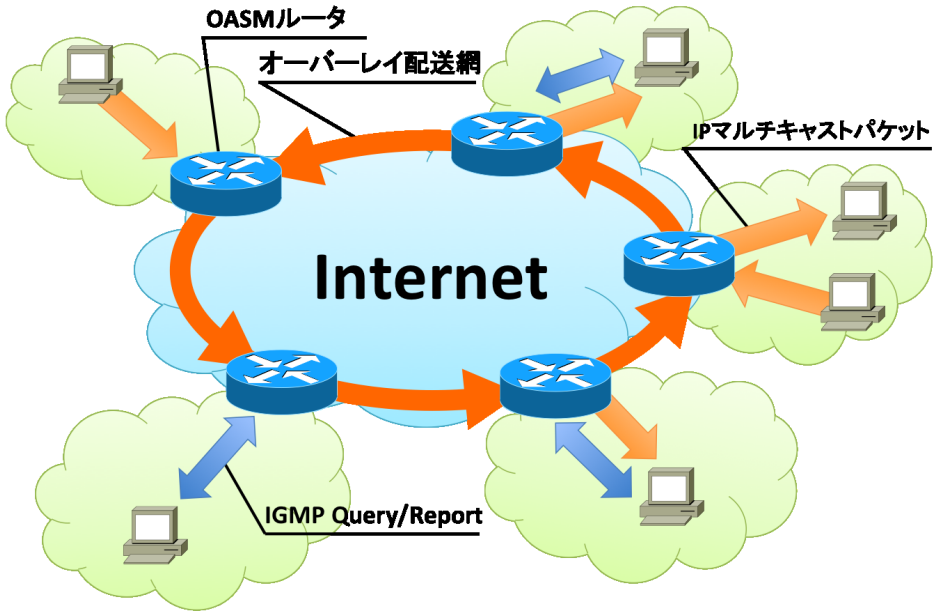

図 4.10: OASM の構築するマルチキャスト配送網の概要

る OASM ルータによって構築される。エンドホストの送信した IP マルチキャストパケットは、まず OASM ルータに受信され、カプセル化されて Group Ring におけるネクストホップへと転送される。Group Ring を通してカプセル化されたパケットを受信した OASM ルータは、カプセル化を解き、受信ノードが接続するサブネットへと転送する。

OASM は、これらのオーバーレイネットワークを構築、維持、利用するために以下のプロトコルを用いる。Group Ring への参加を行う際の宛先を発見する Discovery、リングトポロジに於いてリングの始点となるルータを決定、広告する Designated Router Advertisement、All-Node Ring 及び Group Ring を構築、維持、復旧するためのプロトコルである Stabilize、IP マルチキャストデータの配送を行う Forwarding の 4 つのプロトコルである。これらのプロトコルは、全て UDP を用いる。OASM が用いるメッセージの共通ヘッダフォーマットを図 4.11 に示す。

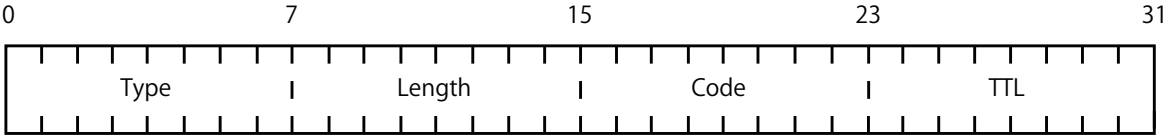

図 4.11: 共通ヘッダフォーマット

Type フィールドはヘッダに続くパケットの種類を示す。Type の種類は MESSAGE(0)、STABILIZE(1)、PACKET(2) の 3 種類である。MESSAGE は Discovery、DR Advertisement メッセージを示す。STABILIZE は Stabilize メッセージを示す。PACKET はカプセル化された IP マルチキャストパケットを示す。Length フィールドはパケット全体の長さを示す。Code フィールドは各メッセージによって使用される。TTL フィール

ドは、パケットのループを防ぐためのものである。OASM ルータを経由する度に 1 ずつ減少し、TTL が 0 になったパケットは破棄される。以下に、各プロトコルの詳細について述べる。

4.6.2 Discovery

OASM ルータが新規に Group Ring に参加する際、参加要求を行うべき相手を知るために All-Node Ring を用いた Discovery を行う。Discovery メッセージのメッセージフォーマットを図 4.12 に示す。

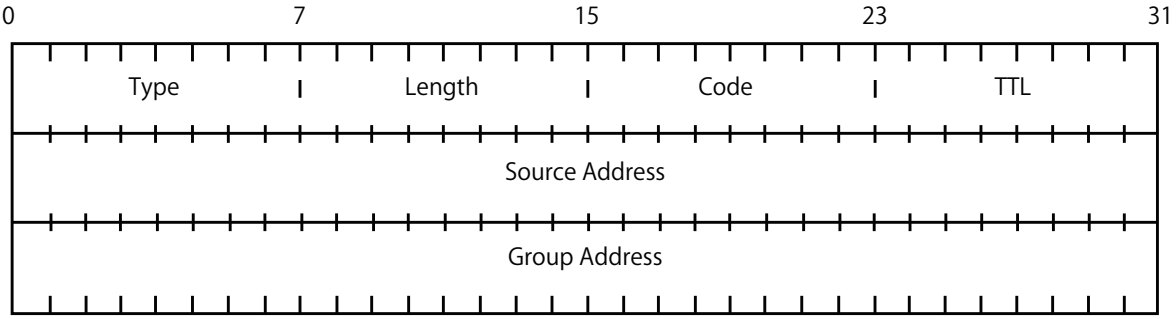

図 4.12: Discovery メッセージのパケットフォーマット

あるルータが新しく Group Ring に参加するには、Source Address に送信ルータのアドレスを、Group Address に参加したい Group Ring のアドレスを入れ、Code に DISCOVERY_GROUP(0) を入れて All-Node Ring のネクストホップへ送信する。Discovery メッセージを受け取ったルータは、自身がこのグループアドレスの配送網に参加していた場合は、Source Address に返信するルータのアドレスを、Code に DISCOVERY_REPLY(1) を入れて、送信元ルータへの返答を行う。参加していない場合は、All-Node Ring のネクストホップへと順次転送していく。もしその Group Ring に既に参加しているルータが存在しない場合、参加ルータは検索要求が All-Node Ring を 1 周して戻ってくることでこれを検知する。返答を受け取ったルータは、4.6.4 節に示す Stabilize を用いて Group Ring へ参加する。

4.6.3 Designated Router Advertisement

OASM では、All-Node Ring 及び各 Group Ring において、4.6.4 節で述べるノード ID の変更に伴うトポロジの変更のため、Designated Router(以下 DR) を選出する。DR は、各 Ring において最大のノード ID を持つノードが選出される。また、DR には生存時間が設定されており、各ルータは生存時間が過ぎた DR はその Ring から離脱したものとして、新しい DR の選出を行う。DR 広告メッセージのメッセージフォーマットを図 4.13 に示す。Group Address フィールドが 0.0.0.0 の場合、All-Node Ring に関する

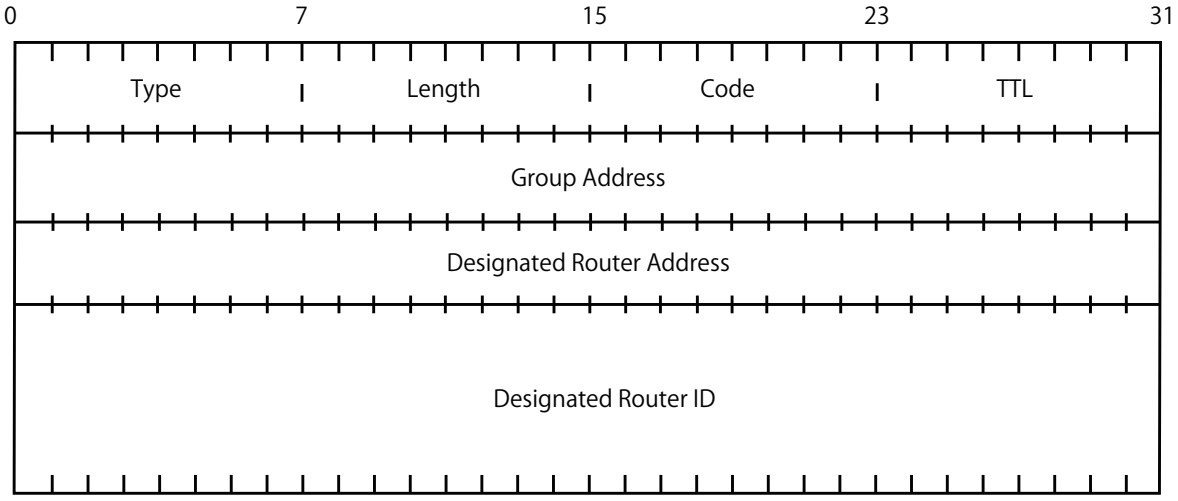

図 4.13: DR 広告メッセージのパケットフォーマット

メッセージ、それ以外の場合はそのアドレスに対応する Group Ring に関するメッセージとして処理される。

新しく Ring に参加したルータは DR を知らないため、DR Address フィールドに自身のアドレス、DR ID に自身の ID を入れて Ring のネクストホップへメッセージを送信する。DR 広告メッセージを受け取ったルータは、自身の知る DR の ID とメッセージの ID を比較し、既知の DR の方が大きい場合はメッセージの DR Address と DR ID のフィールドを既知の DR のものに変更してネクストホップへ転送する。一方既知の DR の方が小さい場合、メッセージに含まれる DR を新しい DR として記憶し、メッセージをネクストホップへ転送する。また、ある Ring の DR となったルータは、定期的に自身のアドレスと ID を入れた DR 広告メッセージをその Ring へと送信する。既知の DR からのメッセージを受け取った場合と、既知の DR より ID の大きい DR 広告メッセージを受け取った場合、各ルータはその DR の生存時間を最大にする。

4.6.4 Stabilize

All-Node Ring 及び Group Ring の構築、維持、復旧には、DHT の 1 つでありオーバーレイのリングトポロジを構築する Chord[19] を参考にした Stabilize というプロトコルを用いる。Stabilize メッセージのメッセージフォーマットを図 4.14 に示す。Stabilize では Code として、STABILIZE_HELLO_NEXT(0)、STABILIZE_REPLY_NEXT(1)、STABILIZE_HELLO_PREV(2)、STABILIZE_REPLY_PREV(3)、STABILIZE_PREV_SET(4) を用いる。また、Group Address フィールドが 0.0.0.0 の場合は All-Node Ring を、それ以外の場合はそのアドレスに対応した Group Ring のメッセージであることを示す。Seconds フィールドと Micro Seconds フィールドにはそれぞれ、送信時の紀元からの経過時間のうち、秒とマイクロ秒が格納される。この 2 つのフィールドは、返信メッセージに同じ値をいれて返信することで、ノード間の RTT を計測し、vivaldi による座標の推定に用いる。

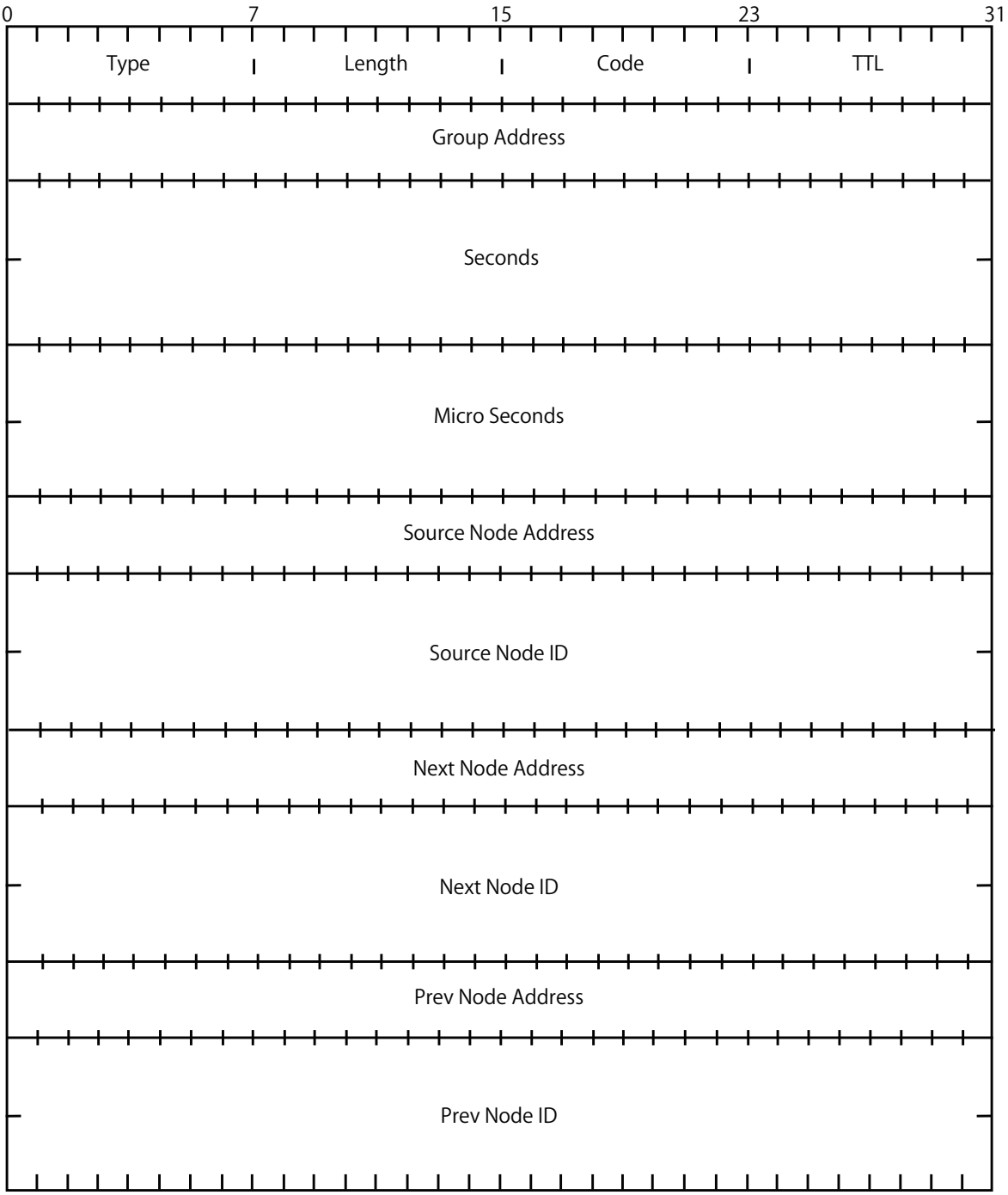

図 4.14: Stabilize メッセージのパケットフォーマット

OASM では、ルータの離脱を検知し、復旧するために、各ルータは All-Node Ring と各 Group Ring ごとに、自身の次ルータと前ルータに対して定期的に Stabilize メッセージを送ることで生存確認を行う。次ルータに対して生存確認を行う際は、Code に STABILIZE_HELLO_NEXT(0) を入れて送信する。これに対する返信は、Code に STABILIZE_REPLY_NEXT(1) を、返信するルータの次ルータのアドレスと ID、自身のアドレスと ID、前ルータのアドレスと ID を入れて返信する。これにより、全ての

ルータは自身の参加する Ring 毎に、1 つ先と、2 つ先のルータのアドレスを記憶する。同様に前ルータに対しても定期的にメッセージをやり取りし、生存確認を行う。

Stabilzie メッセージのやり取りによって、次ルータの離脱を検知した場合、検知したルータはネクストホップを Stabilize によって学習した 2 つ先のルータに変更する。そして、STABILIZE_HELLO_NEXT を送信し始めることで、リングトポロジを復旧する。Group Ring において、もし 2 つ先のルータまで同時に離脱した場合、ルータは再度 Discovery を行うことで次ルータとなるべきルータを発見する。All-Node Ring の場合、事前に OASM 網に参加するために静的に設定されたルータのアドレスに対して参加処理を行う。

配送網への参加も、Stabilize プロトコルによって行われる。あるルータが新しく Group Ring へ参加する際、Discovery によって、既にその Group Ring へ参加しているルータを発見する。次に、発見したルータに対して STABILIZE_HELLO_NEXT メッセージを送信する。STABILIZE_HELLO_NEXT メッセージを受け取ったルータは、自身の前ルータの ID とメッセージの送信元ルータの ID を比較し、自身の前ルータとして正しい ID を持つルータを前ルータとして記憶する。この前ルータを判断する動作を擬似コードで下記の図 4.15 に示す。

```
rtr correct_previous_router (rtr_a, rtr_b, rtr_m, rtr_dr) {

  // rtr_a は既に登録されている前ルータ
  // rtr_b は STABILIZE_HELLO_NEXT の送信元ルータ
  // rtr_m は自ルータ

  // 前ルータが登録されていない場合、rtr_b を選択
  if (IS_EMPTY (rtr_a)) {
    return rtr_b;
  }

  // 送信元ルータの ID が自分より大きく前ルータより小さい場合、rtr_b を選択
  if (rtr_a > rtr_b && rtr_b > rtr_m) {
    return rtr_b;
  }

  // 自身が DR であり、送信元ルータの ID が前ルータより小さい場合、
  // または送信元ルータの ID が自身より大きい場合、rtr_b を選択
  if (rtr_m == rtr_dr && (rtr_a > rtr_b || rtr_b > rtr_m)) {
    return rtr_b;
  }

  return rtr_a;
}
```

図 4.15: 前ルータの判断処理

STABILIZE_HELLO_NEXT を受け取ったルータは、図 4.15 のアルゴリズムを用いてメッセージの送信元ルータが前ルータとして正しいか判断する。STABLIZE_REPLY_NEXT を受け取ったルータは、メッセージの Prev Node Address が自身のアドレスであるかを確認し、自身のアドレスであった場合はネクストホップを STABILIZE_REPLY_NEXT の送信元アドレスとする。もし違う場合、STABILIZE_REPLY_NEXT の Prev Node Address フィールドのアドレスに対して、STABILIZE_HELLO_NEXT メッセージを送信する。このように、リングを遡ってルータに問い合わせて行くことで、新規参加ルータは自身が参加すべき箇所を発見し、Group Ring へ参加する。

また、4.5 節で述べたように、ルータの ID は vivaldi によって推定された座標から決定される。vivaldi による座標は Stabilize メッセージのやり取りで計測した RTT によって常に修正される。そのため、ID も座標の修正によって変更されることがある。この ID の変化によって、ID の順番に並んでいるリングトポロジが不正となる場合がある。ID の変化によって発生するトポロジの不整合は、各ルータが自身の ID、次ルータの ID、2 つ先のルータの ID を比較することによって検知し、修正を行う。もし 2 つ先のルータの方が次ルータの ID よりも大きく、かつ 2 つ先のルータが DR で無い場合、2 つ先のルータをネクストホップとして STABILIZE_PREV_SET(4) を送信する。STABILIZE_PREV_SET を受け取ったルータは ID を比較することなく、前ルータを STABILIZE_PREV_SET の送信元ルータに変更する。

4.6.5 Forwarding

IP マルチキャストパケットの配送は、カプセル化され、パケットのグループアドレスの Group Ring を 1 周することで行われる。カプセル化された IP マルチキャストパケットのパケットフォーマットを図 4.16 に示す。

ユーザの送信した IP マルチキャストパケットは、まず OASM ルータによって受信される。OASM ルータはパケットを IP ヘッダからカプセル化し、そのパケットの宛先グループアドレスに応じた Group Ring のネクストホップへと転送する。Source Address フィールドにはカプセル化したルータのアドレスをオリジネータとして付与する。Group Ring からカプセル化されたパケットを受け取ったルータは、カプセル化を解いてユーザを収容するエッジネットワークへ送信する。また、カプセル化されたパケットを TTL を 1 減らして Group Ring のネクストホップへと送信する。TTL が 0 であった場合、または、オリジネータである Source Address フィールドのアドレスがネクストホップのルータであった場合、そのパケットを廃棄する。これにより、パケットのループを防止する。

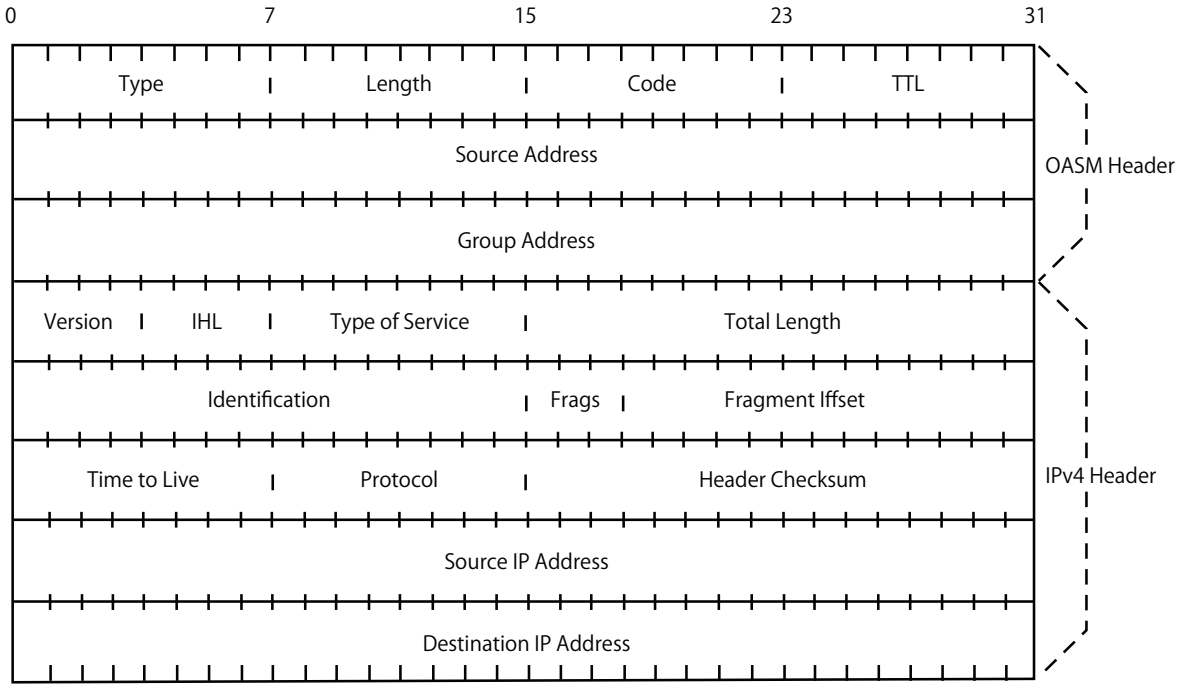

図 4.16: カプセル化されたのパケットのフォーマット

第5章　実装

本章では、OASM の実装について述べる。本章では、本実装のことを oasmd と呼ぶ。

5.1　実装環境

表 5.1 に oasmd の実装環境を示す。

表 5.1: oasmd の実装環境

使用 OS	Debian 6.0.3 Squeeze
使用カーネル	Linux Kernel 2.6.32-5
使用言語	C 言語

5.2　oasmd の概要

表 5.2 に oasmd の各機構を構成するモジュールを示す。また、各モジュール間の関係について図 5.1 に示す。

表 5.2: oasmd のモジュール

機構	モジュール
IGMP プロキシ	gproxy oasm_gproxy
トポロジ管理機構	ring message stabilize
経路表	route fdb
パケット転送機構	interface forward

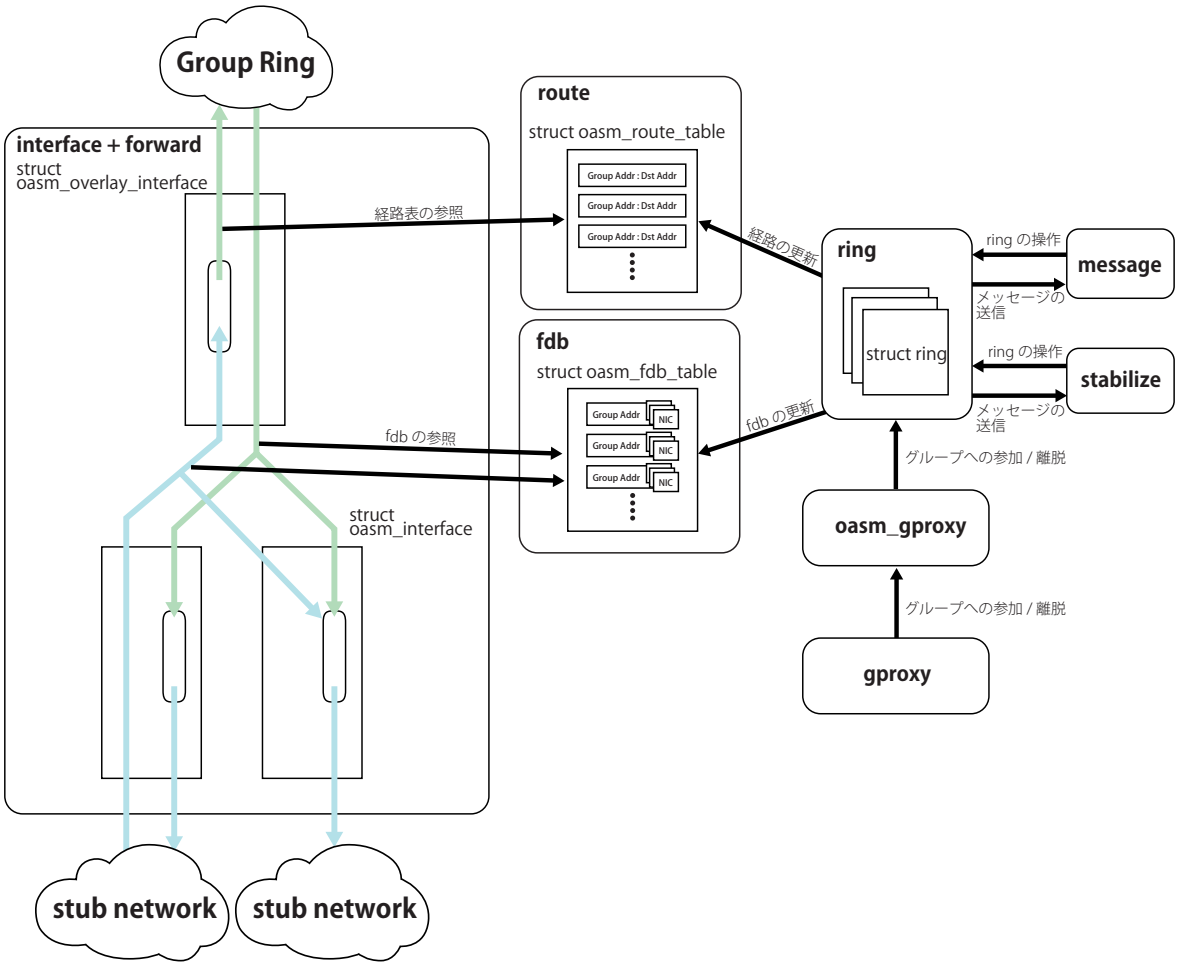

図 5.1: モジュール間の関係

5.2.1 IGMP プロキシ

IGMP プロキシは、gproxy モジュールと oasm_gproxy モジュールから構成される。gproxy モジュールは、oasmd 以外のアプリケーションでも汎用的に使用できるように設計された IGMP プロキシ本体のモジュールである。RFC2236 に準拠して実装されており、IGMPv1、IGMPv2 に対応している。IGMP プロキシは、ホストからの IGMP Report を受信し処理するスレッドと、タイマの管理を行い、IGMP Query の送信やグループからの離脱を処理するスレッドによって構成される。そして、この 2 つのスレッドがインターフェース毎に生成される。この IGMP プロキシの API は、グループへの参加時と離脱時に実行される関数へのポインタを渡す形で提供される。両関数のプロトタイプを以下に示す。

```
void join_function (struct gproxy_message);
void drop_function (struct gproxy_message);
```

join_function は新しいグループへの参加が発生した際に、drop_function はグループからの離脱が発生した際に実行される。これらの関数が呼ばれる際の引き数として、そ

のイベントが発生したインターフェース名、IP マルチキャストグループアドレスを含む gproxy_message 構造体が渡される。

oasm_gproxy モジュールは、汎用 IGMP プロキシである gproxy モジュールを oasmd から利用するためのモジュールである。oasm_gproxy は、IGMP プロキシとして動作すべきインターフェース毎に gproxy モジュールのスレッドを起動する。そして、新しいグループへの参加が発生した際には新しく Group Ring へ参加する処理を行う。また、離脱が発生した際には Group Ring から離脱する処理を行う。

5.2.2 トポロジ構築機構

トポロジ管理機構は、ring モジュール、message モジュール、stabilize モジュールから構成される。ring モジュールは、All-Node Ring と各 Group Ring を ring 構造体で抽象化し、各リングの管理を行う。ring 構造体は、リング毎に次ノード、2 ホップ先のノード、前ノードを保持し、ring 構造体群のタイマ管理を行うスレッドが定期的にそれらのノードに対して stabilize メッセージを送信する。また、ring 構造体は各リングの DR の情報も保持しており、DR 広告メッセージを用いて DR ノードの lifetime 管理や、DR ノードの lifetime が 0 になった際の DR メッセージの送信などを行う。

message モジュールは、Discovery メッセージや DR 広告メッセージを ring モジュールが送信するための関数群を提供する。また、それらのメッセージを受信し、メッセージ毎に該当するグループの ring 構造体の操作を行う。メッセージの受信には専用のスレッドを起動し、スレッドがメッセージ用のポートを待ち受け、受信したメッセージ毎に処理を行う。Discovery メッセージを受信すると、ring モジュールの提供する関数を用いて現在参加している Group Ring を検索し、Discovery に対する返答を行う。また Stabilize メッセージを受信すると、stabilize モジュールへパケットを引き渡す。

stabilize モジュールは、stabilize メッセージを送信、処理するための関数群を提供する。ring モジュールは、stabilize モジュールの提供する関数を用いて、管理する ring のタイマに応じて Stabilize メッセージを送信する。また、message モジュールは Stabilize メッセージを受信すると、メッセージパケットを stabilizie モジュールへ渡す。stabilize モジュールは、そのメッセージのグループアドレスに該当する ring 構造体を検索し、隣接ノードの lifetime を再設定や前ノードの判断を行い、stabilize に対する返答を行う。

5.2.3 経路表

経路表は、route モジュール、fdb モジュールから構成される。OASM は、Group Ring を管理し、オーバーレイの経路表を作成する。この経路表は、IP マルチキャストグループアドレス毎に、カプセル化したパケットを送信する宛先を記憶する。route モジュールは、この経路表を構築、操作するための関数群を提供する。API として、経路表の作成、更新、削除、検索を行う関数が提供される。トポロジ構築機構は、新しい Group Ring への参加や離脱、経路の変更といったイベントが発生すると、route モジュールの

提供する関数群を用いて経路表を更新する。また、パケット転送機構は、この経路表を検索することでパケットを転送すべきルータを知る。

一方、OASM は OASM ルータとして接続される複数のエッジネットワークに対して IP マルチキャストルータとして動作するため、IP マルチキャストの Forwarding Data Base を持つ。Forwarding Data Base は、IP マルチキャストグループアドレス毎に、そのグループアドレスに参加している NIC のリストを記憶する。fdb モジュールは、この Forwarding Data Base を構築、操作するための関数群を提供する。API として、fdb の作成、更新、削除、検索を行う関数が提供される。トポロジ構築機構は、IGMP プロキシからの新しいグループへの参加をトリガとして新しい Group Ring へ参加する際に、IGMP Report を受信した NIC をそのグループアドレスへ追加する形で fdb を更新する。また、同様にグループからの離脱をトリガとして離脱したグループアドレスから離脱した NIC を削除する。

5.2.4 パケット転送機構

パケット転送機構は interface モジュールと forward モジュールから構成される。interface モジュールは、物理インターフェースを oasm_interface 構造体で抽象化し、インターフェースごとのキューや、インターフェースの検索、操作などを提供する。また、オーバーレイ網へパケットを転送、受信するための仮想インターフェースを oasm_overlay_interface 構造体として抽象化し、キューや操作を提供する。

forward モジュールは、インターフェースごとにパケット転送を行うスレッドを提供する。インターフェースごとに作成された送信スレッドは、そのインターフェースに該当する oasm_interface 構造体のキューを監視し、パケットがキューに入るとそのパケットをエッジネットワークへと送信する。また受信スレッドは、エッジネットワークから IP マルチキャストパケットを受信し、仮想インターフェースの送信キューへ追加する。また、fdb を検索し、他のインターフェースがそのパケットのグループアドレスに参加していた場合、パケットをコピーし、そのインターフェースの送信キューへ追加する。また、仮想インターフェース用の送信スレッドは、oasm_overlay_interface 構造体の送信キューを監視し、経路表に従ってパケットをカプセル化し、Group Ring の次ノードへと転送する。また、Group Ring からカプセル化されたパケットを受信すると、まずカプセル化されたまま経路表に従って Group Ring の次ノードへと転送する。そして、カプセル化を解き、fdb に従ってエッジネットワークへと送信する。

第6章　評価

本章では、本研究の提案手法である OASM の評価を行う。

6.1　評価方針

4.2 節で示した通り、機能要件は、1. マルチキャスト配送網の構築、2. 送信者が増加しても負荷が上昇しない配送網、3. 送信者が送信するストリームを配送網中のルータがコピーすること、4. 任意の送信者から複数受信者へのデータ配送、5. 配送網のアプリケーション透過性、の 5 つである。このうち、1、3、5 は、1 対多型の通信を実現するアーキテクチャである OASM が持つ機能として実現される。そこで本研究では、2 の送信者数の増加に対する配送網全体の負荷を 1 つめの評価軸とする。そして、任意の送信者から複数受信者へのデータ配送の実現、つまり、複数の送信者と受信者を許容できるだけの規模性があるかどうかを 2 つめの評価軸とする。以上の 2 点を評価軸から、具体的に以下の 2 点の計測項目を評価として用いる。

- 経路の収束時間

  複数のノードで OASM 網を構築し、ノード数に対してオーバーレイトポロジが安定するまでの時間。

- Round Trip Time

  複数のノードで OASM 網を構築し、ノード数に対してパケットが全ての OASM ノードを通過して配送されるまでに要する時間。

OASM では、トポロジを決定する ID の生成に vivaldi を用いる。vivaldi はノード間の RTT を計測する度に座標を修正する。そのため、OASM 網が安定するには、まず vivaldi による座標が収束しなければならない。しかし、OASM において RTT を計測し合うノードは Stabilize による生存確認を行うノード同士であり、任意のノードの次ノードと前ノードのみと限定的である。そのため、vivaldi が収束するにはある程度時間がかかることが予想される。そこで、実効的な時間で収束できるノード数の限界を知るため、ノード数の増加に対して網が収束するまでの時間を評価し、これを規模性の指標とする。

また、ノード数の増加に起因してパケットの配送遅延が増加することが予想される。OASM は配送網をリングトポロジのオーバーレイネットワークで構築する。そのため、ノード数の増加に従ってパケットが通過すべきノードも増え、パケットが全ての受信者

へ配送されるまでにかかる時間は増加する。OASM は、この配送遅延をなるべく小さくするため、vivaldi による近接ノードの選択を行う。そこで、提案手法である vivaldi による近接ノードの選択がどの程度有効であるかを評価すると同時に、どの程度のノード数まで配送網全体の遅延を実効的な範囲に収めることができるかを評価し、これをノード数に対する配送網負荷の指標とする。

6.2 実験環境

4.1 節で述べた通り、OASM は世界規模で動作し、膨大な数のユーザを収容するため、相当数のルータで配送網を構築する。そのため、規模性と遅延を指標として実験を行うにあたって、実験環境に対して以下の要求があげられる。

- 世界規模のインターネットトポロジ環境の再現
- 多数の実機による実験ノード
- 実験ノード間に実環境に即した遅延が存在すること

OASM は、配送網をリングトポロジのオーバーレイネットワークで構築する。また想定環境として世界規模を想定しているため、OASM を動作させる実験環境の IP 層のトポロジには、世界規模の実インターネットのトポロジが再現される必要がある。さらに、世界規模で動作できるだけの規模性を評価するため、相当数の実験ノードが必要である。そして、この実験ノードは実機でなければならない。本実験では、評価項目として配送網全体の遅延を用いる。そのため、仮想マシンでは、仮想化環境に起因する遅延が発生してしまい、遅延に関する計測結果は実験結果として意味を成さないものになってしまう。また、OASM はオーバーレイトポロジの構築にノード間の RTT を用いる。そのため、実験ノード間には、再現される実インターネットのトポロジに応じた遅延が再現される必要がある。

以上のように、本実験では、多数の実機材を用いて実インターネットの IP 層のトポロジを再現し、さらに再現したトポロジに即したノード間の遅延が存在する実験環境が必要である。そこで本実験では、各要件を満たした実験環境を構築するため、大規模実証実験環境である StarBED[20]、インターネットの AS 間トポロジを模倣する AnyBed[21]、そして、AS 間の遅延を再現するために世界中の DNS サーバ間の遅延を計測した King Data Set[22] を用いた。本節では、まず実験環境の構築に用いた StarBED、AnyBed、King Data Set について述べ、この 3 つを用いて構築した実験環境について述べる。

6.2.1 StarBED

StarBED[20] は、北陸リサーチセンターに開所され、情報通信研究機構 (NICT) によって運営されている大規模実証実験環境である。StarBED は、960 台の実ノードと、実験環境構築支援システムである SpringOS によって構成される。StarBED の概念的な

トポロジを図 6.1 に示す。StarBED を利用するユーザは、グループ F から L までのハードウェアの異なる実験ノード群からノードの割り当てを受ける。そして、SpringOS を利用して数百台規模の実験ノード群を操作し、実験を行う。SpringOS の機能として、実験用スイッチでの vlan の割り当て、boot ディスクの変更、ipmi を用いたノードの電源操作等を行うことができる。これらの機能を用いて実験ネットワークの構築、ノードの操作を行うことで、利用者は多数の実機を用いた大規模実証実験環境での実験を円滑に進めることができる。

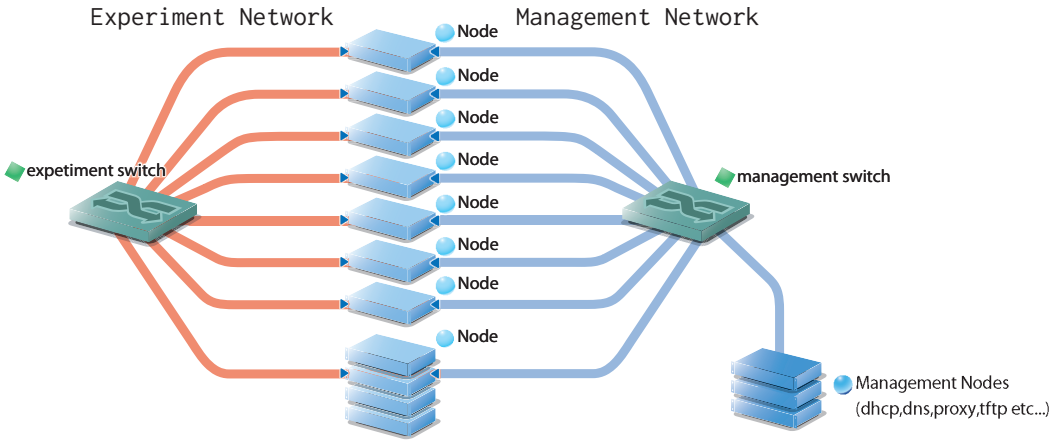

図 6.1: StarBED の概念的なトポロジ

本実験では、マイクロ秒単位で遅延を計測するため、仮想マシンではなく、実機で実験を行う必要があった。そのため、StarBED から 100 台の実機ノードの割り当てを受け、実験環境を構築した。StarBED における実験環境の設定は以下の通りである。まず全てのノードの実験用 NIC を同一セグメントに所属させた。次に、100 台のノードを、1 台の NFS サーバを NFS Root として pxeboot した。この状態で、6.2.2 節で述べる AnyBed を用いた AS 間トポロジの構築を行った。実際に実験に使用したグループ I のノードの詳細を表 6.1 に示す。

表 6.1: StarBED における実験機材

ハード	CPU	メモリ	NIC
Cisco UCS	Intel Xeon CPU X5670 2.93GHz	48G	NetXtreme II BCM5709

6.2.2 AnyBed

AnyBed[21] は、奈良先端科学技術大学院大学と NICT によって開発されている、ネットワークを用いた実験に必要なトポロジを構築するためのツールセットである。AnyBed によって構築される L3 トポロジは、テストベッドの構成に依存しない。AnyBed では、複数のノードを同一ネットワークに所属させ、そのネットワークをマルチホームとして

扱うことで L3 トポロジを構築する。AnyBed によって再現できるトポロジは、1 ノードを 1AS とした BGP による AS 間トポロジと、1 ノードを 1 ルータとした OSPF による AS 内トポロジである。AnyBed の大まかな構成を図 6.2 に示す。

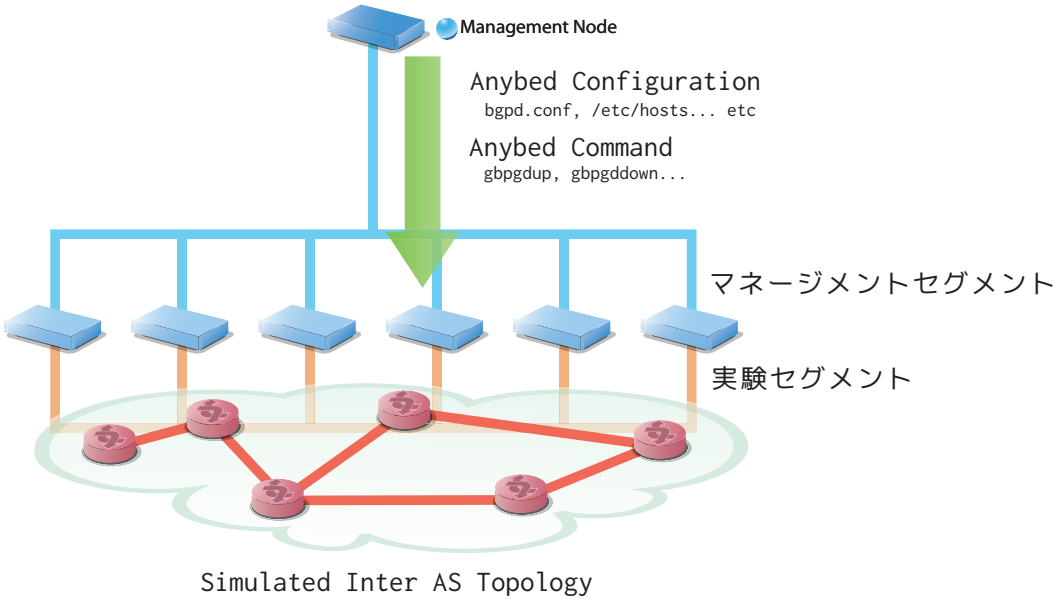

図 6.2: AnyBed の構成

Anybed は、1 台の管理ノードが複数台の実験ノードを操作する。実験ノードは管理インターフェースと実験用インターフェースを持ち、それぞれ管理用セグメントと実験用セグメントに接続される。管理用セグメントでは、pxeboot を行うための DHCP や tftp サーバ、NFS Root を行うための NFS サーバなどが用意される。数百、数千台規模にのぼる実験ノードを、1 台ずつインストールして設定するのは現実的ではない。そのため、AnyBed では、管理用サーバに NFS Root 用の Root ディレクトリを用意し、実験ノードは pxeboot から NFS Root Mount によって起動する。また、AnyBed に関連する設定ファイルの投入も、NFS 経由で管理サーバから行われる。管理サーバから実験サーバ群へのコマンドの発行は、dsh (dancer's shell) や gsh (Global Shell run commands in parallel to multiple machines) といった ssh のラッパープログラムによって発行される。

AnyBed は、L3 トポロジを構築するためにルーティングデーモンとして Quagga[23] を用いる。AnyBed による AS 間トポロジの再現では、1 ノードが 1AS として動作し、IP Alias を用いてピア先の AS 毎に/24 の IP アドレスを実験インターフェースに設定する。そして、自身のピア用の Alias アドレスを用いて複数のノードと BGP ピアを結ぶ。このマルチホームを用いた手法によって、実験用インターフェース 1 つだけを用いて、複数 AS との接続関係の模倣を実現している。模倣する AS のトポロジ情報には、CAIDA[24] の公開している世界中の AS ピアリング情報を用いる。

6.2.3 King Data Set

King Data Set[22] は、インターネット上でのエンドホスト間の RTT を推定したデータセットである。King は、世界中の DNS サーバ間の遅延を計測することで、エンドホスト間の RTT を概算する。King の手法では、DNS の再帰問い合わせを利用して DNS サーバ間の RTT を計測する。計測の概要を図 6.3 に示す。まず、計測クライアントは再帰検索を許可する DNS リゾルバサーバまでの RTT を計測する。そして、ns.example.com へ転送される検索クエリをリゾルバサーバへ送信する。クエリは、リゾルバサーバから example.com の権威サーバである ns.example.com へ転送され、回答はリゾルバサーバを経由して計測クラアントへ到達する。これにより、再帰クエリが返るまでの時間からリゾルバサーバまでの RTT を引くことで、リゾルバサーバと権威サーバ間の RTT を推定することができる。

DNS サーバは、ユーザからのクエリに高速に返答するため、ネットワーク的にエンドユーザの近くに設置されることが多い。そのため、DNS サーバ間の RTT は、エンドホスト間の RTT に近いものとなる。King の手法では、エンドユーザとアドレスプレフィックスの近い DNS サーバを選択することによって、推定する RTT を、よりエンドホスト間の RTT に近いものにしている。この King のデータセットは、vivaldi[17] の評価にも用いられている。OASM は、エンドユーザを収容するエッジネットワークに設置される。そのため、実験を行うにあたって、OASM ルータ間の遅延として King Data Set を用いることによって、より実際のインターネットに近い遅延を実験環境に再現することができる。

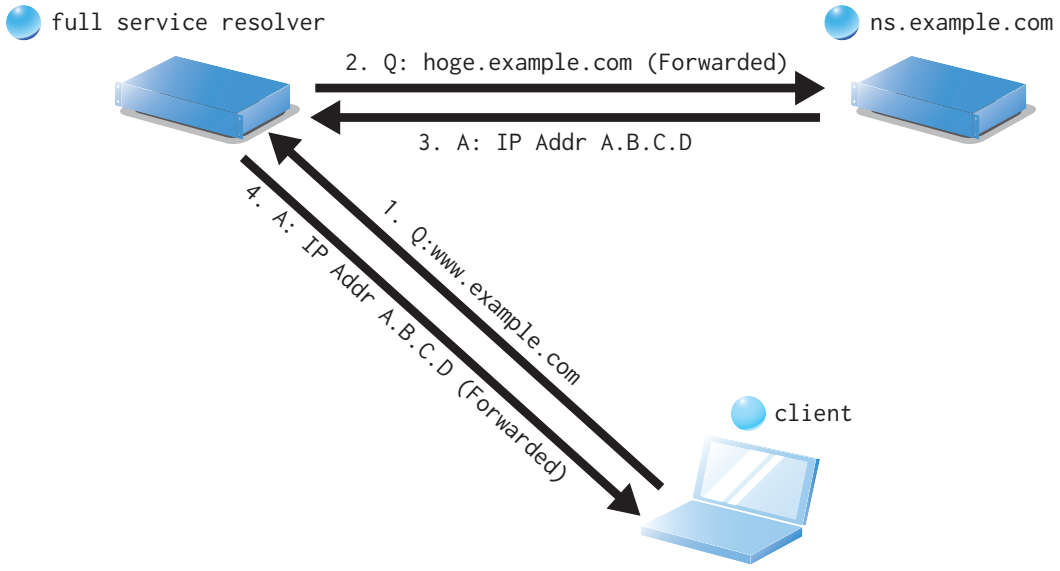

図 6.3: King の手法による DNS サーバ間の RTT の計測

6.2.4 OASM の実験環境

OASM は、オーバーレイネットワークを用いて AS に制限されない広域な配送網を構築する。また、遅延をメトリックとしてオーバーレイトポロジを構築する。そのため、実験環境では広域の AS 間接続環境を模倣し、かつノード間の遅延を実現する必要がある。そこで、StarBED の実機ノード群上に AnyBed を用いて AS 間トポロジを構築し、King Data Set をもとに AS となるノード間のリンクに iproute2 を用いて遅延を発生させた。この実ノードによる遅延を含めた AS 間接続再現環境の構築手順を以下に示す。

1. Route Views から取得したフルルートを用いて、King Data Set に出てくる DNS サーバの AS 一覧を作成する。

2. 1 で生成した AS 一覧から、AnyBed を用いて上位 100AS をリスト化し、このリストを元に AS 間トポロジを生成する。

3. AnyBed によって生成されたトポロジからピア情報を取り出し、AS 間ピア情報に相当する DNS サーバ間の遅延を King Data Set から取り出す。

4. Linux の iproute2 を用いて、全ルータにピア先ルータとなる実験ノードの MAC アドレス毎に 3 で King Data Set から取り出した遅延を発生させるように設定する。

5. OASM ルータのアドレスとして用いるため、全て実験ノードの実験インターフェースに/32 のアドレスを設定し、AnyBed によって構築された BGP 網へ広告する。

これによって、実機 100 台を用いて実インターネットの AS 間接続環境と遅延を再現した実験環境を構築した。

6.3 実験結果

6.3.1 収束時間

収束時間に関する実験としてまず、同時に複数台のノードで OASM 網を構築し、経過時間毎に配送網全体で経路の変更が何回発生したかを計測した。ノード数は 10 台から 100 台まで、10 台ずつ増やして実験を行った。各ノード数について 600 秒経過するまでに、配送網に参加している全 OASM ノード内で経路変更が発生した回数を計測した。4.5 節で示した vivaldi の誤差修正式における、係数 δ が 0.02 の場合を図 6.4 に、0.03 の場合を 6.5 に示す。vivaldi の誤差修正アルゴリズムにおける δ は、座標を修正する際の移動距離の係数である。この δ が大きいほど、座標を修正する際の移動は大きくなる。そのため、δ が大きいほど座標が早く収束する。しかし、δ が大きすぎると座標が発振し、座標が収束しない、または収束するまでに多くの時間がかかる。

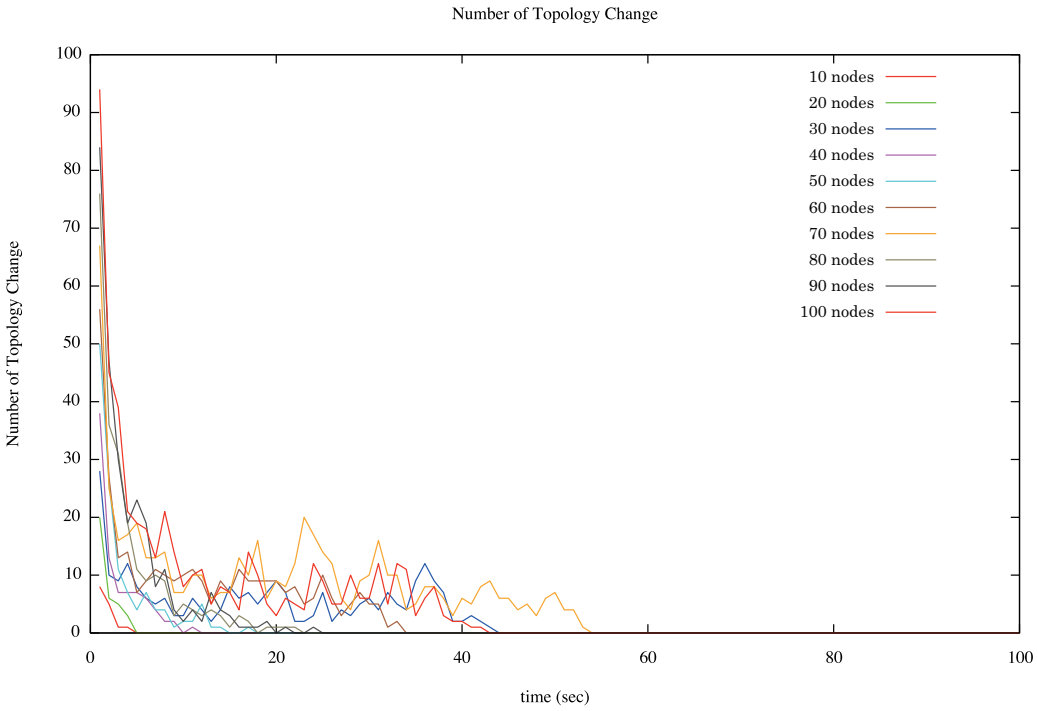

図 6.4: 経過時間毎の経路変更回数：$\delta = 0.02$

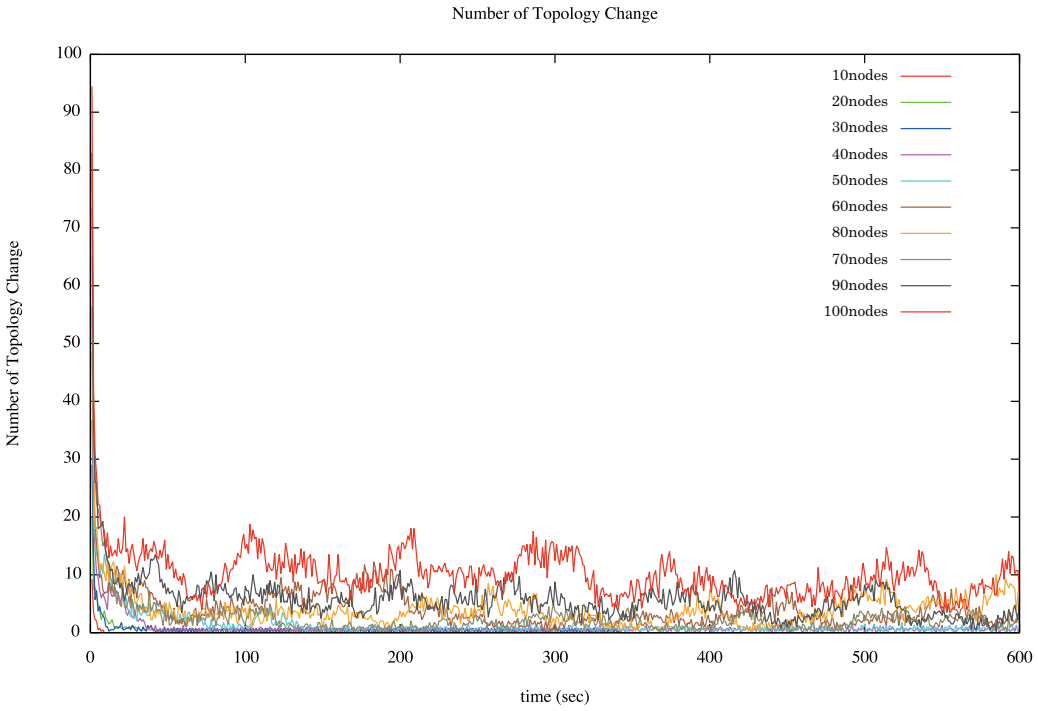

図 6.5: 経過時間毎の経路変更回数：$\delta = 0.03$

δ が 0.02 の場合、100 台までの全てのノード数において 60 秒以内には網が収束している。しかし、トポロジの収束に最も時間がかかったのが 70 台のとき、次いで 30 台、100 台、60 台、80 台と、収束に掛かる時間とノード数が多ければ収束に時間がかかる傾向が見られるものの、30 台に時間がかかる等ばらつきも大きい。これは、初期座標の値が原因と考えられる。OASM では、OASM 起動時のノードの初期座標として乱数を取る。乱数である初期座標の誤差が大きい場合、vivaldi による座標に時間がかかる場合がある。本実験で、収束時間がノード数によらなかったことから、この初期座標の誤差がノード数以上に収束時間に影響を与えていることがわかった。

一方 δ が 0.03 の場合、30 台までは収束したものの、40 台以上では 600 秒を経過してもトポロジが収束しなかった。[17] によれば、OASM の実装に用いた最も単純な vivaldi アルゴリズムでは、座標が発振せずに収束する δ の最大値は 0.05 である。しかし、vivaldi の収束時間や、座標が発振するかどうかは、RTT を計測するアプリケーショントラフィックの性質にも依存する。今回の 0.03 が収束しなかったという結果から、リングの両隣のノードのみ RTT を計測するという OASM のメッセージングトラフィックの形態では、δ の値は 0.02 から 0.03 の間でチューニングする必要があるということがわかった。

次に、1 台のノードが参加してから経路が収束するまでの時間を計測した。実験方法は、最初に 2 台のノードで OASM を起動して網を構築し、経路が収束する度に 1 台ずつ参加させていき、各ノード数について、経路が収束するまでの時間を計測した。この実験の結果を図 6.6 に示す。本実験では、4.5 節で示した vivaldi のアルゴリズム内の係数 δ を 0.03 とし、ノード数 2 台から 100 台までの測定を 16 回を行った。この実験の結果から、ノード数が増加する程、ノードが新規に参加した際に経路が収束するまでに要する時間が増加することがわかる。OASM では、あるノードが RTT を計測するノードは、リングにおける両隣の 2 ノードのみである。そのため、新しくノードが参加した際には、そのノードのランダマイズされた初期座標の修正が全ノードへ伝搬し、全ノードの座標が再度収束するまでに要する時間は、網に参加しているノード数に応じて増加する。

また、ノード数が 45 台を超えると、収束にかかる時間にばらつきが出始める。ノード数が 45 台を超えると 30 秒以上かかる場合が出始め、70 台を超えると 60 秒以上、97 台の際には 700 秒程かかる場合がある。これは、新規参加ノードのランダマイズされた初期座標によるものである。初期座標が大きく外れていた場合、網全体のノードの座標が収束し、経路が収束するまでに要する時間も増加する。しかし、71 台までは、経路の収束時間を 60 秒以内に収めることができている。今回の実験から、ノードが 1 台ずつ参加した場合、実効的な時間でトポロジが安定するノード数の限界は、70 台程度であることがわかった。

OASM ノードは、IP マルチキャストルータとしてユーザを収容するサブネットに設置され、定常的に運用されることを想定される。そのため、多数の OASM ノードが同時に配送網に参加する可能性は低い。しかし、ある特定のコンテンツに突然人気が集中し、受信者が殺到する場合も想定できる。同時に多数の受信者が参加する場合を想定すると、前述の実験から δ は 0.02 に設定する必要がある。一方で、現実的には受信

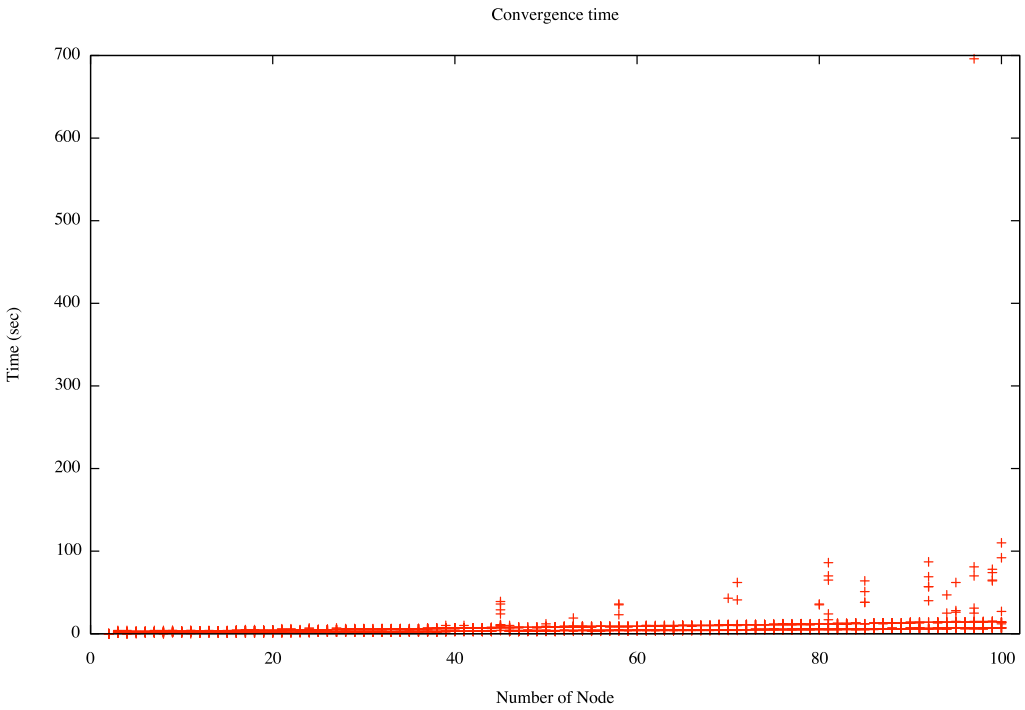

図 6.6: 1 台ずつ参加した際の経路収束時間

者はある程度時間のばらつきを持って参加することが予想される。δ が 0.03 で 1 台ずつ参加した際の経路収束時間の累積密度分布を示した図 6.7 から、2 台から 100 台までのノード数において、新規ノード参加時の経路収束時間の 95%以上が 14 秒以下に収まっている。このことから、14 秒以内に 40 台以上のノードが参加することを最悪時として想定し、δ の値を 0.02 から 0.03 の間で設定するのが適切であるといえる。

6.3.2 遅延

配送網の遅延を評価するための実験として、2 台から 100 台までのノードで構成した配送網を、パケットが 1 周するまでに要した時間 (RTT) を計測した。その結果を図 6.8 に示す。この実験では、δ を 0.02 とした。そして、10 台から 100 台までの各ノード数について、RTT を 100 回計測した。図 6.8 中の Vivaldi Ring は OASM が構築したトポロジで実測した RTT、Random Ring は、各ノード数でランダムに構築したリングロポトジでパケットが 1 周するのに要する時間を、AnyBed による AS 間接続情報と King Data Set を用いてシミュレートしたものである。

図 6.8 から、vivaldi を用いて構築したリングトポロジでは、ランダムに構築したリングトポロジよりも総じて遅延を抑制することができたことがわかる。特に、ノード数が増えるに従って、vivaldi を用いた近接ノード選択によって配送網全体の遅延を抑制する効果が大きいことがわかった。ランダムに構築したリングでは、網に参加するノードが増えるほど、ネットワーク的に遠いノードが並ぶ確率が増え、配送網全体の

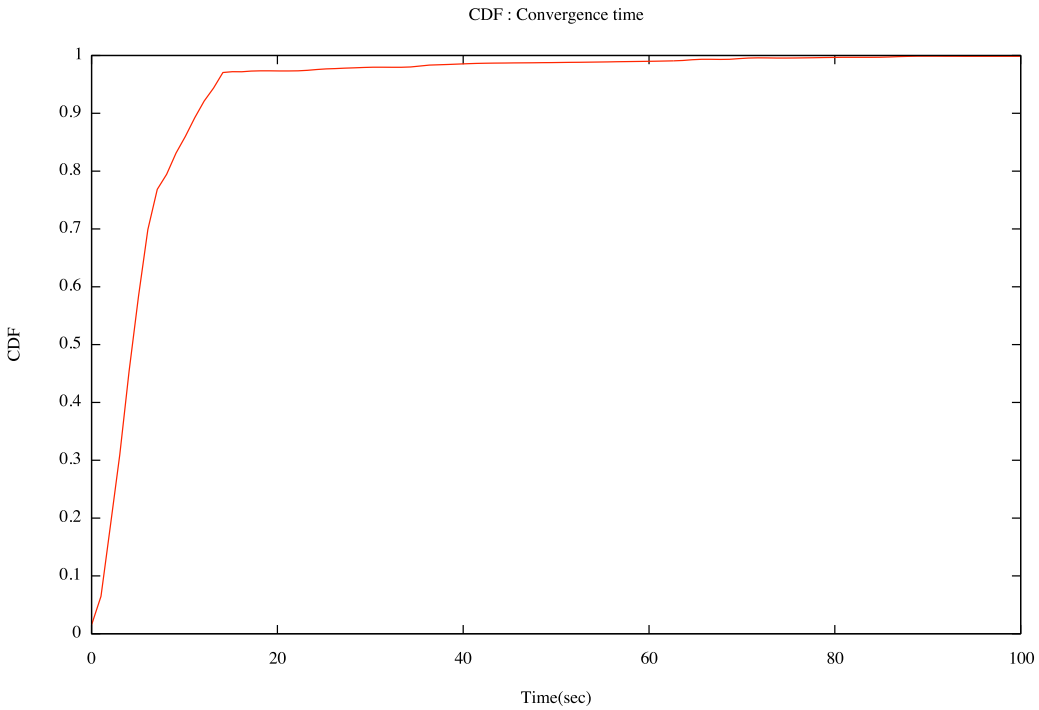

図 6.7: 1 台ずつ参加した際の経路収束時間：CDF

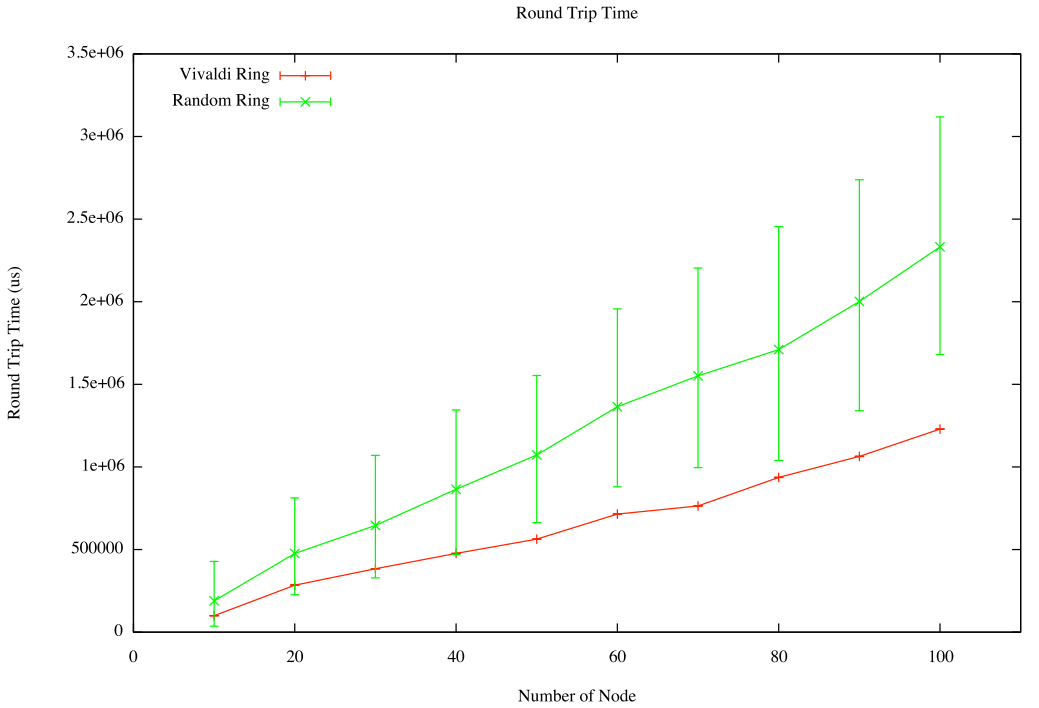

図 6.8: パケット配送にかかる時間

遅延も増加する。一方、vivaldi を用いて構築したリングでは、ノード数が増加してもネットワーク的に近いノード同士を選択してリングを構築し、網全体の遅延を抑制することに成功している。しかし、vivaldi を用いてトポロジを最適化しても、ノード数の増加に従って、配送網全体の遅延はほぼ線形に増加することがわかった。

また、網全体での配送遅延が 400ms 以内に収まるのは 30 台までであることがわかった。国際電気通信連合の勧告 G.114[16] によれば、一方向通信のマルチメディアアプリケーション全般において、許容されるネットワーク遅延の最大値は 400ms とされている。そのため、配送網全体の遅延を実効的な範囲に収めることができるノード数は 30 台までといえる。しかし本実験環境は、インターネットの AS 間トポロジを模倣し、1AS に 1OASM ノードという環境である。実際の環境では、多くのエンドユーザを収容する AS 内に多数の OASM ノードが設置される。また、同一 AS 内の遅延は AS 間を超える場合の遅延よりも短いと考えられる。そのため、実際の環境では同一 AS 内で複数の OASM ノードが配送網に参加し、そのノード間の遅延は短いことが予想されるため、ノード数が 30 台を超えてもある程度までは配送遅延を 400ms 以内に遅延を収めることができると考えられる。

6.4 実験のまとめと考察

本研究では、OASM 実装の規模性に関する評価と、ノード数に対する配送網負荷に関する評価について、実インターネットの AS 間接続と遅延を再現した実験環境において評価実験を行った。規模性に関する収束時間についての実験から、OASM 実装の実効的に動作できるノード数は 70 台程度であることを示した。OASM で用いた vivaldi のアルゴリズムでは、δ が 0.02 の場合は 100 台まで 60 秒以内にトポロジが収束し、δ が 0.03 の場合は 30 台を超えると 600 秒経過しても収束しない。しかし、0.03 の場合でも、ノードが 1 台ずつ参加する場合は 70 台程度まで実効的な時間で収束する。以上のことから、本実験によって、リングトポロジを構築する OASM の実効的な動作規模の限界は、δ が 0.02 から 0.03 の間で 70 台程度であるといえる。

また収束時間についての実験において、ノード数に対する経路収束時間のばらつきがみられた。このばらつきの原因は、vivaldi の初期座標として乱数を用いているためである。OASM では、RTT を計測し座標を修正し合うノードは、あるノードの前ノードと次ノードのみである。そのため、ランダマイズされた初期値による誤差が大きい場合、その誤差が網内のノードへ伝搬し、全体で座標が収束するまでの時間が長くなる。そして、この初期座標の誤差は収束時間に対してノード数よりも大きな影響を与えていることが実験からわかった。そこで、ノードがネットワーク上の位置を何らかの手法で類推し、初期値の誤差が小さくなるような仕組みを導入することによって、トポロジの収束に要する時間をより短くできると考えられる。

配送網負荷に関する遅延時間についての実験から、vivaldi を用いたトポロジの構築が配送網負荷の軽減に対して有効であることが示せた。δ が 0.02 の vivaldi で構築したトポロジでは、ランダムに構築したリングよりも平均値で半分ほど遅延を抑制することができた。また、ノード数が 80 台まではパケットが全受信者まで到達するのに要す

る時間を 1 秒以下に抑えることができた。また、30 台までは配送遅延を実効的な遅延時間である 400ms 以内に収めることができた。以上の実験から、本研究では、OASM 実装の実効的な動作規模の限界は 70 台程度であることを示した。また、1AS に 1 ノードで配送網を構築した場合、網全体の配送遅延を実効的な時間内に収めることができるノード数は 30 台であることを示した。

6.5 配送網の構造化

次の段階として、オーバーレイネットワークのリングトポロジである配送網の構造化を行う必要がある。本実験の結果、単一のリングを構築する場合の動作ノード数の限界は 70 台であり、また配送網全体の遅延を実効的な時間に収めることができるのは 30 台までであることがわかった。しかし、このノード数では、世界中のユーザを収容する 1 対多型の配送網を構築することはできない。また、単一のリングではどんなに最適化しようともノード数の増加に従って遅延は増加する。そこで、本実験で示した最大ノード数である 70 ノードまでのリングを 1 つの単位として、複数のリングを階層的に接続することで構造化し、より規模性のある配送網を構築する必要がある。

本実験では、1AS に 1OASM ノードという環境で実験を行い、実効的な遅延時間に収めることができるノード数は 30 台という結果を得た。しかし、実際の OASM の動作環境では、1AS 内に多数のノードが設置される。一般的に、AS 内のノード間と AS の異なるノード間では、AS 内のノード間の方が遅延が短いと考えられる。例えば、WIDE Project[25] の運営する WIDE BackBone 内では、関東地方の拠点間の遅延は 1ms 以内、東阪では約 5ms 程度である。WIDE BackBone 内で OASM を用いてオーバーレイ配送網を構築した場合、動作限界である 70 台全てが東阪を超えるオーバーレイリンクを構築したとしても、配送網全体の遅延は 400ms である。実際には、vivaldi による近傍ノードの選択が行われ、さらに配送遅延は小さくなる。このように、1 つのリング内のノードをなるべく同一 AS 内のノードになるように選択し、リング内で AS を超えるオーバーレイリンクを減らすことによって、配送網全体の遅延を抑えることができると予想される。

また、リング同士を接続するノードには、潤沢な可容帯域を持つノードを選択する必要がある。単一のリングトポロジでは、全てのノードはユーザを収容するエッジネットワークとリングの次ノードの 2 つの宛先に向けてパケットをコピーする必要がある。しかし、リングを接続するノードは、接続する両方のリングからデータを受け取り、エッジネットワークに加えて両方のリングへパケットをコピーして転送する必要がある。このように、接続ノードはダウンリンクとアップリンク双方に 2 倍の負荷がかかる。そのため、接続ノードには潤沢な可容帯域を持つノードが選択される必要がある。また、単一リング内の AS 数は減らした方が良いため、接続ノードは、参加するリングの一方においては別 AS のノードとオーバーレイリンクを構成した方が良い。

以上のように、リング同士を階層的に接続するように配送網を構造化する際には、単一のリング内で AS をまたぐオーバーレイリンクを減らし、かつリング同士を接続するノードとして潤沢な可容帯域を持つノードを選択することが必要である。また、コ

ピーすべきトラフィック量の増える接続ノードは、潤沢な可容帯域を持つノードが選択される必要がある。このように、AS と可容帯域を指標としてリングトポロジのオーバーレイ配送網を構造化することによって、配送網全体の遅延を抑え、さらに多くのノード数で動作する配送網を構築することが可能になる。そして、この構造化した配送網を構築するシステムを実装、評価した上で、リングの階層の深さを決定する。階層が深くなる程、最も離れた 2 つのノード間のホップ数は増加し、それに伴って配送遅延も増加する。そのため、現在のインターネットの規模では、どの程度の階層の深さが適切であるかを評価し、その上で必要があれば最大 70 ノードというリングのサイズを再検討する。

第7章　結論

本章では、本論文のまとめと今後の課題を示す。

7.1　本研究のまとめ

本研究では、誰もが自由に送信者となることができる 1 対多型の通信を実現するための配送網を構築するシステムを設計、実装した。本研究では、1 対多型のマルチキャスト通信を実現するための配送網として、リングトポロジのオーバーレイネットワークを選択した。さらに、オーバーレイのリングトポロジの最適化を行うため、vivaldi によるネットワーク座標系とルベーク曲線による座標からの ID の生成を導入した。そして、これらを用いたシステムとして OASM を提案、設計し、Linux 上に実装した。また、提案したシステムの有用性を示すため、実機ノードを用いて AS 間トポロジと AS 間遅延を模倣した大規模実験環境を構築し、評価実験を行った。その結果、本研究の OASM 実装では、実効的に動作するのは 70 台程度までであることを示した。また、配送網の遅延を実効的な範囲に抑えることができるのは 30 台程度までであることを示した。

7.2　今後の課題と展望

本研究では、リングトポロジのオーバーレイネットワークを用いて 1 対多型の通信を実現するシステムの設計、実装を行った。そして、規模性、配送遅延に関する評価を行った。しかし、本論文では提案したシステムの帯域に関する議論を行っていない。本研究で導入した Link Class が、リンクの帯域性能に差のあるノードを用いてトポロジを構築した際に、帯域性能によってノードを並べることができているかを確認する実験を行う必要がある。そして、リングトポロジのボトルネックとなる極端に狭い帯域を持つノードが配送網全体の転送性能に与える影響を評価する必要がある。

帯域に関する評価を行った上で、オーバーレイリングトポロジの構造化を行う必要がある。本研究では、vivaldi による仮想座標からルベーク曲線を用いて ID を生成することでトポロジを最適化し、配送遅延を抑制した。しかし、ノード数の増加に伴って配送遅延は線形に増加する。また、実効的な動作ノード数が 70 台では、世界規模での自由な 1 対多型の通信を実現するという本研究の目的を実現することはできない。そこで、オーバーレイネットワークで構築される配送網の構造化を行う必要がある。本論文で示した単一のリングトポロジで実効的に動作する規模である 70 台のリングを最

小リングとして、複数のリングを階層的に接続する形で配送網を構造化する。これによって、ノード数の増加に伴う配送遅延を抑制するとともに、より多くのノードで配送網を構築することが可能になる。

謝辞

本論文の作成にあたり、ご指導いただきました慶應義塾大学環境情報学部教授村井純博士、同学部教授中村修博士、同学部准教授楠本博之博士、同学部准教授 Rodney D. Van Meter III 博士、政策・メディア研究科特任講師吉藤英明博士、同研究科特任講師斉藤賢爾博士、株式会社アールシーエー重近範行博士に感謝します。

常にご指導いただきました、政策・メディア研究科博士課程岡田耕司氏に感謝します。学部 1 年時から、インターネットについて研究と運用の両方の側面から絶えず多くのご指導をいただきました。そして、学部 2 年時から始めた本研究を卒業論文としてまとめることが出来たのも岡田耕司氏のおかげです。重ねて感謝申し上げます。

また、同じく学部 1 年時からご指導いただきました政策・メディア研究科博士課程堀場勝広氏に感謝します。研究室に入った当初から様々な現場に連れて行っていただき、運用、研究をはじめ多くのことを学ばせていただきました。重ねて感謝申し上げます。

研究室を通じた生活の中で多く示唆を与えてくださった政策・メディア研究科博士過程空閑洋平氏、松谷健史氏、修士過程波多野敏明氏、メディアデザイン研究科博士課程遠峰隆史氏、環境情報学部村上滋希氏、上野幸杜氏、山岸祐大氏に感謝します。

そして、徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・武田合同研究プロジェクトの諸氏皆々様に感謝します。

参考文献

[1] M.Mahalingam, D.Dutt, K.Duda, P.Agarwal, L. Kreeger, T. Sridhar, M.Bursell, and C.Wright. draft-mahalingam-dutt-dcops-vxlan-00.txt. ID, IETF, Aug 2011.

[2] Steven C. Dake, Christine Caulfield, and Andrew Beekhof. The corosync cluster engine. *Linux Symposium*, Jul 2008.

[3] bonjour. `http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/NetServices/Introduction.html` 最終確認日 2012/1/20.

[4] Cisco Visual Networking Index: Forecast and Methodology, 2010-2015. Jun 2011. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf` 最終確認日 2012/1/20.

[5] Ustream asia press release, 民主党代表選挙の延べ視聴者数が100万人を突破. `http://www.ustream-asia.tv/news_20110830.html` 最終確認日 2012/1/20.

[6] John Buford. Survey of ALM, OM, Hybrid Technologies. *IETF SAMRG*, July 2006.

[7] A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense mode (PIM-DM): Protocol Specification (Revised). RFC 3973, IETF, Jan 2005.

[8] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (Rivised). RFC 4601, IETF, August 2006.

[9] R. Boivie, N. Feldman, IBM, Y.Imai, WIDE / Fujitsu, W. Livens, ESCAUX, D. Ooms, and OneSparrow. Explicit Multicast (Xcast) Concepts and Options. RFC 5058, IETF, Nov 2007.

[10] D. Thaler, M. Talwar, A. Aggarwal, Microsoft Corporation, L. Vicisano, Qualcomm Inc., and T. Pusateri. draft-ietf-mboned-auto-multicast-10. *IETF*, Mar 2010.

[11] Beichuan Zhang, Wenjie Wang, Sugih Jamin, Daniel Massey, and Lixia Zhang. Universal ip multicast delivery. *Comput. Netw.*, 50:781–806, April 2006.

[12] Marc Brogle, Dragan Milic, and Torsten Braun. Supporting ip multicast streaming using overlay networks. In *The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness & Workshops*, QSHINE '07, pages 49:1–49:7, New York, NY, USA, 2007. ACM.

[13] M. Castro adn P. Druschel, A. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEEE Journal on Selected Areas in Communications Vol.20 No.8*, Oct 2002.

[14] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. Splitstream: high-bandwidth multicast in cooperative environments. *SIGOPS Oper. Syst. Rev.*, 37:298–313, October 2003.

[15] Yatin Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Syst.*, 9:104–118, July 2003.

[16] ITU Recommendation : G114 One-way transmission time. `http://www.itu.int/rec/T-REC-G.114/recommendation.asp?lang=en&parent=T-REC-G.114-200305-I` 最終確認日 2012/1/20.

[17] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. *SIGCOMM Comput. Commun. Rev.*, 34:15–26, August 2004.

[18] J. A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems*, PODS '84, pages 181–190, New York, NY, USA, 1984. ACM.

[19] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[20] Toshiyuki Miyachi, Ken-ichi Chinen, and Yoichi Shinoda. Starbed and springos: large-scale general purpose network testbed and supporting software. In *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, valuetools '06, New York, NY, USA, 2006. ACM.

[21] Hiroaki Hazeyama, Mio Suzuki, Shinsuke Miwa, Daisuke Miyamoto, and Youki Kadobayashi. Outfitting an Inter-AS Topology to a Network Emulation TestBed for Realistic Performance Tests of DDoS Countermeasures. In *Proceedings of Workshop on Cyber Security Experimentation and Test (CSET'08)*, August 2008. `http://www.usenix.org/event/cset08/tech/` 最終確認日 2012/1/20.

[22] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: estimating latency between arbitrary internet end hosts. *SIGCOMM Comput. Commun. Rev.*, 32:11–11, July 2002.

[23] Quagga routing suite. `http://www.quagga.net/` 最終確認日 2012/1/20.

[24] The Cooperative Association for Internet Data Analysis. `http://www.caida.org/home/` 最終確認日 2012/1/20.

[25] WIDE Project. `http://www.wide.ad.jp` 最終確認日 2012/1/20.