第 XV 部

Explicit Multi-Unicast

第 15 部
Explicit Multi-Unicast

第 1 章　はじめに

1.1 XCAST-WG の取り組み

XCAST-WG は、インターネット上の多地点ビデオ会議や通信ゲームなど、たくさんの小さなグループ内でパケットの同報を行なうのに適した通信方式である XCAST（Explicit Multi-Unicast）と関連するマルチキャスト技術の研究開発および普及活動を行う WG である。

2006 年は、ハードウェア実装に向けた仕様である XCAST2.0 の構想、標準化および普及展開活動の基盤としての IRTF SAM-RG（Scalable Adaptive Multicast Research Group）の設立、普及活動として Matsuri と呼ぶネットワーク上のイベント等の開催、および各種 XCAST 関連技術の研究開発を行った。

これらの普及活動、および研究開発の概要を報告する。次の節で、主な普及活動/実証実験についてまとめ、次の章以後で、以下の項目について報告する。

- XCAST2.0 構想
- IRTF SAM-RG の設立
- 機能拡張性を考慮した XCAST グループ管理システム（XGMS）
- Xcast on Ad hoc Networks — Tree based Xcast Routing
- XCAST 通信における配送順序最適化について
- IPv6 on PlanetLab
- PlanetLab on StarBED

なお、本報告書への記載は割愛するが、これらの活動とは別に、名古屋大学の修士 1 名、早稲田大学の学士 1 名、AIT（Asian Institute of Technology）の修士 2 名が XCAST 関連技術の研究開発を行い、学会発表や修士学士論文発表を行った。

1.2 主な普及活動/実証実験

2006 年に実施した主な普及活動/実証実験を表 1.1 で報告する。普及開発活動は、WIDE XCAST-WG のみならず、XCAST Fan Club 主体でも行われている。

表 1.1.　主な普及活動/実証実験

実施時期	分類	実施内容
2006/1	普及活動	XCAST ゆく年くる年（Fan club）
2006/3	実証実験	WIDE 合宿での「隣の BOF の様子が見れるカメラシステムの運用課題抽出」
2006/3	標準化	第 65 回 IETF における IRTF SAM-RG 設立準備会議の主催および中継
2006/3	普及活動	FreeSBIE2 with XCAST のリリース（Fan club）
2006/5	普及活動	実証実験ネットワーク x6bone のアドレス移行（3ffe:→2001:）
2006/5	普及活動	18th Midnight XCAST Meeting (Fan club)
2006/6	普及活動	N+I BOF「BSD なひととき」の XCAST6 中継
2006/6	普及活動	オープンソースカンファレンス 2006 新潟の XCAST6 中継（Fan club）
2006/7	実証実験	NEMO、XCAST6 を用いた Tour de France の中継実験（nautilus6 と協業）
2006/7	普及活動	オープンソースカンファレンス 2006 北海道の XCAST6 中継（Fan club）
2006/8	普及活動	Matsuri「和田先生と語る Happy Hacking Keyboard の夕べ on XCAST6」
2006/9	標準化	第 66 回 IETF における第一回 SAM-RG 会合の正式開催
2006/9	研究支援	XCAST 若手の会発足。第一回 XCAST 若手の会開催
2006/10	実証実験	Matsuri「XCAST MATSURI（祭）2006」（配送順による損失率変動の計測）
2006/11	研究支援	第二回 XCAST 若手の会開催
2006/11	開発	Linux 版、XCAST6 ルータの実装完了
2006/11	実証実験	関西オープンソース 2006 の XCAST6 中継（packetiX の活用）
2006/11	普及活動	SEA & FSIJ 合同 Forum（GPLv3 conference）XCAST6 中継

第 2 章　XCAST2.0 (Eliminating HBH)

現在実験運用中の XCAST6 の semi-permeable tunnel（半透膜トンネル）の仕様には、IPv6 の Hop By Hop（HBH）オプションヘッダで、ルータが XCAST6 パケットを認知する仕様となっている。しかし、IPv6 ルータベンダは、HBH オプションヘッダの処理は、ソフトウエア処理として作りこんでしまっている。このため、XCAST6 パケットが流れるたびに、ソフトウエア処理が発生してしまい、結果として、XCAST6 の通信が増えるとルータ上のルーティングデーモンの動作を阻害してしまうという問題が指摘されていた。

XCAST2.0 とは、この問題を解決するために考案された新たな XCAST6 仕様である。

XCAST-WG では、昨年来、XCAST6 のような実験プロトコルを実フィールドで安全に実験するための実験番号空間を請求する活動を続けてきた。この活動の結果、XCAST6 の動作を定義するに十分な実験番号空間の定義を含んだ RFC4727[72] が Standard track として 2006 年 12 月に発行された。

XCAST2.0 は、この RFC で定義されている実験番号空間のうち、フローラベル、トラフィッククラス、マルチキャストアドレス、ルーティングヘッダを用いて XCAST6 の semi-permeable tunnel 機能を定義するものである。XCAST2.0 に基づいた XCAST6 仕様では、XCAST 非対応ルータの動作を阻害することなく安全に通信ができることが保証される。現在、XCAST2.0 の仕様の文書化を行っており、今後、実装の XCAST2.0 対応を進めていく計画である。

第 3 章　IRTF SAM-RG の正式設立

XCAST6 プロトコル及び関連プロトコルの研究開発の加速、および標準化を進めるために IRTF（Internet Research Task Force）に SAM-RG（Scalable Adaptive Multicast Research Group）を設立した。WIDE 賞の賞金を活用し、第 65 回 IETF 会合において、設立準備会議を開催した。準備会議で IRSG 等と交渉し、準備 ML での議論の後、チャータ案が IAB で承認され正式 RG の設立となった。WIDE XCAST-WG から、SAM Framework に関するインターネット草案を寄稿し、第 66 回 IETF 会場で第一回正式会合で発表した。今後、IEEE CCNC2007 で中間会議が開催される。この場でも、XCAST-WG から 3 つの発表を行う予定である。

第 4 章　機能拡張性を考慮した XCAST グループ管理システム（XGMS）

4.1 概要

本研究では、開発や保守などが容易で簡単に機能の追加ができる環境、またユーザからの様々な要求に柔軟に対応できるようなシステムを構築することを目的としている。これを実現する為に、Web アプリケーション開発基盤として比較的開発効率の高い Ruby on Rails を利用することにした。しかし、このフレームワークは Web アプリケーションを前提としており、ブラウザ以外の実時間性を考慮したアプリケーションへ適用が難しい。そこで、Ruby on Rails のアーキテクチャである MVC2 を拡張することで、フレームワークの適用範囲を広げることにし、拡張 MVC2 として提案する。これをグループコミュニケーションへ適用し、XCAST 環境を利用した機能拡張容易なグループ管理システム（XGMS）の構築を行った。グループコミュニケーション用の開発基盤を構築することで、機能要求に柔軟に対応し、拡張機能を比較的容易に実装できることを検証することが狙いである。

4.2 Web アプリケーションフレームワークと実時間性

4.2.1 背景

現在では誰もが Web を容易に閲覧できるようになった。それに伴い、当初は情報閲覧が目的であった Web をアプリケーションとして利用することが広まっている。特に実時間性が要求される分野においても使われるようになってきている。一方、アプリケーション開発においては、ユーザの多種多様な要求に対し、迅速かつ柔軟に応えなければならない。

その為には、生産性や開発効率が高く、変更に対し柔軟で保守もしやすいアーキテクチャ、または、開発基盤が必要であると一般的に言われている。

4.2.2 MVC Model2

Web アプリケーションの構築技術としては、サーバサイドプログラミングが発達してきた。特に MVC 構造を基にした Model2（MVC2）アーキテクチャが発達している。MVC2 は、次のような流れを繰り返すことで Web アプリケーションを実現する。まず、ブラウザに表示された View のページから HTTP リクエストがサーバサイドに渡される。サーバ側ではそれらのリクエストを単一の Controller が受け付ける。Controller は記述された定義に従い適切な Model を呼び出し、結果を受け取る。その結果をもとに次に表示させるページを決定し、クライアント側に HTTP レスポンスを返す。このように、MVC2 で想定されている通信形態は、クライアントとサーバの 1 対 1 のリクエスト/リプライ（PULL）型の同期的なシグナリングである。グループコミュニケーションなどのリアルタイム性を要するアプリケーションでは、多数に対する非同期なシグナリングが必要であり、この PULL 型のまま利用しリアルタイム性を求める為には、常時リクエストを送り続けなければならないという問題がある。

4.2.3 選択した方法

目的を達成する為の一つの方法として、高品質なフレームワークの上に非同期メッセージ通知機能を導入するという方法を考えた。既存の技術として、Web アプリケーションに PUSH 型機構を導入する Comet がある。例えば、Flex や Openl aszlo などはクライアントに対して Rich なインターフェースを構築する環境を提供し、Web アプリケーションにおける非同期メッセージ機構を提供する機構も備えている。また Ruby on Rails には、Juggernaut という Comet を提供するプラグインが公開されている。しかしながら、アプリケーション自体はいずれも Flash を利用しており、ブラウザ上で動作するアプリケーションを前提としていることになる。つまり適用範囲が狭いと言える。また Struts や Spring などのフレームワークは、ブラウザ上でのアプリケーションを作成する場合の開発基盤として主流となっているが、これもブラウザを前提としたものであり適用範囲が狭い。

そこで、高品質なフレームワークである Ruby on Rails を選択し、フレームワークの基盤となっている MVC2 の適用範囲をブラウザ以外のアプリケーションにも広げる方法を選択することにした。

4.2.4 拡張 MVC2

MVC2 の適用範囲を広げることを目的として非同期メッセージ通知機構を導入し、実時間性を考慮したアプリケーションへの開発基盤として、拡張 MVC2 を提案する。

クライアントがリクエストをしなければ、新しい情報が降りてこない PULL 型のシグナリングに対し、サーバ側から新しい情報を自動的に通知してくる PUSH 型のシグナリングを導入し、実時間性を考慮したアーキテクチャになっている。

クライアントとサーバの間の通信は http で行われ、データのやりとりは post を利用することで行っている。（図 4.1）post は比較的シンプルなプロトコルであり解析がしやすいということや、プロトコル自体の規約もまとまっている為、クライアント側でも実装がしやすいというメリットがある。この拡張 MVC2 を利用することにより、従来の MVC2 よりも機能を拡張する際の適用範囲が広がることになる。

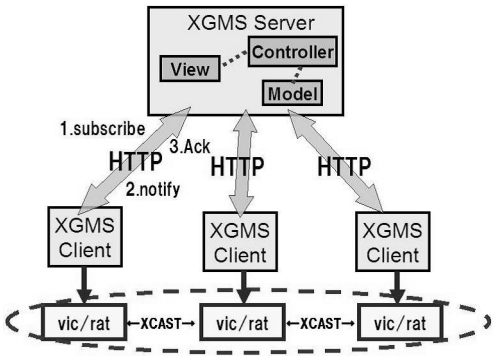

図 4.1. XGMS ビデオ会議システムの構造

4.3 XGMS (eXtensible Group Management System)

ネットワーク上のグループコミュニケーションシステムを、よりユーザの求める形に近づけるための基盤システムとして、MVC2 を用いた代表的な実装である Ruby on Rails に、通知予約/変化通知機能を実装し、さらに、グループ/グループメンバの管理、login したユーザにグループ情報を提供、グループへ

図 4.2.　XGMS を利用したミーティング風景

の参加の受け付け、他人を招待する機能等、グループコミュニケーションが最低限行える程度まで機能拡張を施した。これを機能実装の容易なグループ管理システム基盤 XGMS と呼ぶことにする。システムを構築する際に XGMS を基盤として用い、それに対して音声や画像の通信といったグループコミュニケーションのメインとなる部分を拡張してやるだけで、簡単にグループコミュニケーションシステムを実現することができる。(図 4.2) また、通知予約/変化通知機構が提供されている為、新たな機能を実装する際などにおいても、開発者は PULL 型通信と PUSH 型通信の適した方を適宜選択して利用することができる。

4.4 今後の課題

今後の課題としては、今回の評価対象であったリアルタイム性を損ねていた原因が、従来型システムのグループ管理を行っていた xcgroup 単体のシグナリング側の問題だけでなく、vic/rat のアプリケーション側の問題であることを考慮し、合理的に改善していくことも考えている。また、実際に機能を自ら実装し追加することの他に、様々な人に利用してもらうことで、拡張機能を実装・追加してもらい、様々な知見を得ることも予定している。これらの課題を行っていくことで、システムの拡張自体をより行い易い環境に向上させていくことが今後最大の課題となる。

第 5 章　Xcast on Ad hoc Networks — Tree based Xcast Routing

5.1 概要

近年、テレビ会議やストリーミング中継などの利用目的から、効率のよいグループ間通信プロトコルが求められている。本研究では、アドホックネットワークにおいて効率よく動作するグループ間通信プロトコル——Tree based Xcast Routing（TXR）を提案する。

TXR では独自ツリーテーブルをヘッダに埋め込み、宛先ノードまでソースルーティングをする。宛先アドレスなどをヘッダに埋め込むという考え方は XCAST から由来する。また、TXR は独自グループメンバー管理プロトコルを使うことにより、アドホックネットワーク特有の変動性などの問題を解決している。

5.2 背景とアプローチ

本研究では、ネットインフラが整備されてない状態でのグループ通信を考える。たとえば、学会などにおいての一時的なネットワーク利用、災害などの緊急時や街中での使用が考えられる。学会では意見交換やファイル交換、災害時では緊急連絡や非難情報の提供、街中では待ち合わせやお店などの宣伝、などといった利用例が考えられる。

このような要求に対して、本研究では XCAST ライクなアプローチを取る。つまり、宛先アドレスなどをヘッダに埋め込み、個々のパケットに自由性を持たせる。これにより、XCAST 同様、少人数で構成されるグループ通信に最適となる。しかし、既存の XCAST は有線での利用を想定していて、そのままアドホックネットワークに移行できない。そこで本研究では、アドホックネットワークの性質に合わせて、独自グループメンバー管理プロトコル FGF と配信プロトコル TXR を提案する。また、TXR で使われるツリーテーブルは途中ノードでの計算量が最小になるように定義している。

5.3 グループメンバー管理——Flooding based Group Forming (FGF)

FGF ではフラッディングすることにより、グループ名を全ノードに宣伝し、参加ノードから ACK をもらう。これにより、中心サーバなしでメンバー管理することが可能となる。参加ノードが返す ACK の数は最小になるように工夫されている。つまり、同じルートからの ACK は最大で 1 つだけである。また、TTL やタイムアウトを定義することにより重複やループ問題を解決している。

5.4 コンテンツ配信——Tree based Xcast Routing (TXR)

TXR では後述する独自ツリーテーブルをヘッダに埋め込み、宛先ノードまでソースルーティングする。途中ノードはヘッダ中のツリーテーブルを参照して、ネクストホップを決定する。これにより、ほかのアドホックプロトコルを使うことなくマルチホッピングを提供でき、かつ、途中ノードではルーティング情報を一切持つ必要がない。また、XCAST 同様、個々のパケットに自由性を持たせることができる。

5.5 TXR 独自ツリーテーブル

TXR では独自のツリーテーブルを定義している。これは、途中ノードでの計算量がより少なくなるようにという目的に帰する。

例として、図 5.1 のようなツリー構造を考える。これは FGF により得られるものだと仮定する。まず、深さ優先で全ノードに番号をつける。

図 5.1 のツリー構造は図 5.2 のようなツリーテーブルとして表せる。

これにより、途中ノードでネクストホップを決める際に、テーブル中の自分の行（C なら 3 行目）だけ参照すれば十分で、他の行は完全に無視できる。

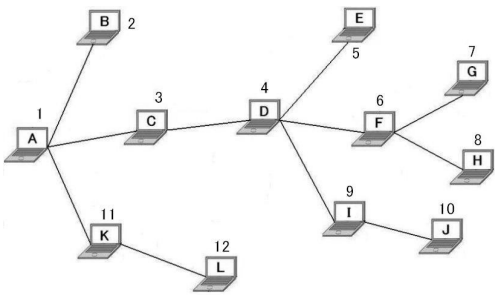

図 5.1. Example for Tree Table

ID	Address	Sub Nodes
1	A	2 3 11 LAST
2	B	ϕ
3	C	4 11
4	D	5 6 9 11
5	E	ϕ
6	F	7 8 9
7	G	ϕ
8	H	ϕ
9	I	10 11
10	J	ϕ
11	K	12 LAST
12	L	ϕ

図 5.2. Tree Table for the example

5.6 今後の課題

現在は一定期間にフラッディングを行いアドホックネットワークの変動性に対応しているが、今後は無駄なフラッディングを減らす方法を考案する。また、パケットロスを検出したり、メンバー管理のセキュリティ問題なども考慮するつもりである。さらに、現在の実装ではテキストベースの配送に限定しているが、将来音声や動画などの配信に対応できると利用範囲が広がると考えられる。

第 6 章 XCAST 通信における配送順序最適化について

6.1 研究の背景と目的

XCAST 通信においては IP マルチキャストにおけるグループアドレスは存在しないが、グループ内のメンバ間で通信を実現するためには、各ノードがグループに参加しているメンバの IP アドレスを知る必要がある。そこでグループのメンバシップを管理するシステムが必要となるが、現在の XCAST 通信におけるグループメンバ管理システムでは配送順序に基準が設けられておらず、グループに参加したメンバから順番に宛先リストが構成されている。

本研究では XCAST 通信における配送順序の決定方法に基準を設け、宛先リストを最適化することで効率的な XCAST 通信を実現することを目的とする。

6.2 提案システムの概要

本研究では XCAST 通信のグループメンバ管理において、宛先リストを操作し配送順序を効率化するシステムを提案する。提案システムの動作概要は以下のとおりである。

1. グループのメンバシップを管理しているサーバ（以下中央サーバと表記）がグループメンバを宛先リストとして提供
2. 各参加ノードが RTT をフルメッシュに計測
3. 中央サーバに RTT 情報を報告
4. 中央サーバは受け取った RTT 情報をコストとして最適な配送順序を計算し、宛先リストを操作
5. 各ノードは新しい宛先リストに従って XCAST 通信を実行

6.3 検討事項――配送順序最適化手法について

XCAST における配送順序最適化問題の解は以下の 2 つの条件を満たす必要がある。

- 全ノードを一度ずつ通る
- 配送コストが最小になる

これは経路探索問題に帰着することができ、実質的にハミルトン路問題を解けばよいことになる。

中央サーバはノード間の各リンクに対しコストを RTT とした隣接行列を保持する。その隣接行列から考えられうる全ての配送順序についてコストを計算して、全経路の内で最もコストが低いものを最適経路とする。この際配送順序の列挙には深さ優先探索を用い、枝刈りを効率的に行うことで計算量を下げる。

コストの決め方の指標として提案システムでは RTT を用いているが、XCAST がビデオカンファレンスや動画配信といったアプリケーションを想定していることを考慮すると、帯域の狭いホストが上流にいた場合にその経路がボトルネックとなりサービス全体の品質が下がってしまう。このように経路のコストを定める評価関数の決め方についてはより細かい議論が必要であろう。

6.4 XCAST MATSURI（2007 年秋）におけるトラフィック実験

2007 年秋に実施された XCAST MATSURI において実施した配送路最適化に関する実験の結果と考察を行う。

6.4.1 実験目的

宛先リストを変更した時の XCAST 通信への影響を調査し、リストを変更することの有意性を確認する。

6.4.2 実験環境

- XCAST ルータ（v4/v6 トンネル機能提供）
- 参加ノード（最大 12 個）
- 利用帯域：～50 kbps 程度（1 ノードの送出データ量）

6.4.3 実験方法

- スーパーノードにおいて宛先リストを 5 分おきにランダムに変更
- MATSURI 参加ノードの内の数地点において RTP パケットロスと jitter を計測
- 当日は大多数の参加者が XCAST ルータに v4/v6 トンネルを張っていたため、XCAST ルータにおいてパケットが複製されてネットワーク的にスター型のトポロジとなっていた。

6.4.4 実験結果

- 特定の拠点（sohgo、名古屋大学）からの送信 RTT パケットのみにロスが全拠点から観測された。
- sohgo（無線での接続）からの送信パケットは全拠点で定常的にロスしていた。
- 名古屋大→ XCAST ルータ間のリンクにロスが観測された。

6.4.5 考察

- ロスの原因がネットワーク的な影響からなのか、宛先リスト変更の影響なのかの切り分けが十分行えなかった。これは数地点でしかロスを計測していなかったことにより、同時刻における宛先リスト中のパケットロスが生じているリンクを分析できなかったことによる。
- 本実験では多数人数が一箇所の XCAST ルータにトンネルを張っていたため、トポロジ的に配送経路が短くなった。将来的に IPv6 が普及した状況を想定すると、各ノードは v4/v6 トンネルを張って v6 アドレスを取得する必要がなくなり、XCAST 通信のホップ数が増えることが考えられる。今後の実験として配送順路最適化の議論を行うためには、よりたくさんの v4/v6 トン

ネルサーバを用意してノードのインターネット上の接続環境を分散させる必要があるであろう。

第 7 章　IPv6 on PlanetLab

7.1 概要

PlanetLabは世界中に分散したノードを用いたネットワークの実験場である。現在、PlanetLab の実験環境において IPv6 はサポートされていない。そのため IPv6 に関連する実験を PlanetLab 上で行なうことは出来ない。

XCAST wg では User Mode Linux を利用して IPv6 パケットを PlanetLab ノード上でルーティングする手法を提案し実装した。また、提案手法を容易に実装するためのエージェントを開発した。

本研究の目的はPlanetLab ノード上における IPv6 パケットのルーティングを可能にし、XCAST6 パケットを PlanetLab 上で転送することである。本研究の目的が達成されることにより、PlanetLab 上で IPv6 に関連した実験が可能となる。

7.2 PlanetLab

本節ではPlanetLab とは何かを説明し、PlanetLab の抱える問題点をあげる。また、関連研究として VINI プロジェクトを説明する。

7.2.1 PlanetLab とは

PlanetLabは世界中に分散したノードを用いたネットワークの実験場であり、参加者によって提供されたノードで構築されている。PlanetLab の参加者は 2 台以上の計算機を PlanetLab ノードとして提供する。そしてそれらのノードは全ての PlanetLab の参加者によって共有される。PlanetLab の参加者は大量のノードを用いて実装実験を行なうことが出来るため、ネットワークアーキテクチャのスケーラビリティを測定するのに適した環境であると言える。

PlanetLab は研究者に対して仮想マシンを実験環境のリソースとして割り当てる。PlanetLab のノードは Linux VServer という仮想化技術を用いており、研究者は Linux VServer によって構築された仮想マシン上で実装実験を行なう。

PlanetLab では研究者の実装実験ネットワークを無償のサービスとして提供することが許可されている。これにより、実用段階に近い状態での評価が可能となる。

7.2.2 PlanetLab の問題点

PlanetLab ノードを利用する際には様々な制限がある。その一つはネットワーク層プロトコルである。現在、PlanetLab ノードでサポートされているネットワーク層プロトコルは IPv4 のみである。そのため、PlanetLab ノードは IPv4 以外のパケットを扱うことが出来ない。

別の制限は kernel である。Linux VServer の特性により、PlanetLab ノード上の全ての仮想マシンは kernel を共有している。そのため、PlanetLab の参加者が kernel に変更を加えることができない。

これらの制限により PlanetLab 上で実装可能なネットワークアーキテクチャが限られている。

7.2.3 VINI

PlanetLab のユーザ環境で IPv4 パケットのルーティングをする VINI というプロジェクトがある。VINI は仮想マシン上に User Mode Linux を配置し、User Mode Linux 上でルーティングエンジンを動かすことによってユーザ環境でのルーティングを実現している。User Mode Linux とは Linux 上で仮想的に Linux を動かす技術である。VINI では OSPF や BGP による IPv4 パケットの動的ルーティングが可能であるが、IPv6 パケットのルーティングを行なうことはできない。

7.3 PlanetLab 上での IPv6 パケットルーティング

本節では PlanetLab ノード上で IPv6 パケットをルーティングするための提案手法と、提案手法の実装を容易にするためのトポロジ構築エージェントについて説明する。また、PlanetLab 上での XCAST6 パケットの転送について説明する。

7.3.1 User Mode Linux を用いた IPv6 静的ルーティング

XCAST wg は User Mode Linux を用いて IPv6 静的ルーティングを行なう手法を提案した。また提案手法を実装し、各 PlanetLab ノード上にある User Mode Linux 上で IPv6 の到達性が得られたことを

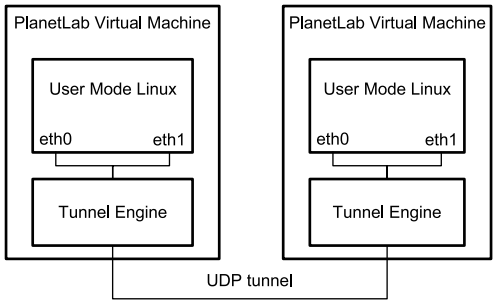

図 7.1.　Overview of the proposal method

確認した。図 7.1 に提案手法の概要を示す。

提案手法では UDP トンネルを用いて IPv6 パケットを PlanetLab ノード間で送受信する。PlanetLab ノードは IPv6 プロトコルスタックを持っていないため、IPv6 パケットを処理することができない。しかしながら、IPv6 パケットを UDP パケットでカプセリングすることで PlanetLab ノードでも IPv6 パケットを受け取ることが可能となる。

提案手法では、PlanetLab の仮想マシン上に User Mode Linux とトンネルエンジンを配置する。利用する User Mode Linux は IPv6 プロトコルスタックをサポートしている。トンネルエンジンとは umlswitch というデーモンに UDP トンネリングの機能を付加したものである。トンネルエンジンは各ノード上の User Mode Linux の仮想ネットワークインターフェース同士を UDP トンネルで接続する。つまり、User Mode Linux の各インタフェースは point-to-point リンクで別の User Mode Linux のインタフェースと接続される。このとき、User Mode Linux 間で IPv6 のオーバレイネットワークが構築される。ここで、User Mode Linux の経路表を書き換えることによって IPv6 パケットのルーティングが可能となる。また、実験者が IPv6 ネットワークのトポロジを決定できる。

実装後、ping6 を用いて各 User Mode Linux の IPv6 パケットの到達性を確認した。

提案手法によって構築される IPv6 ネットワークは User Mode Linux 間に閉じたネットワークである。そのため、インターネットとの接続性がない。インターネットとの接続性を得るために、提案手法に L2TP サーバを組み込むことを予定している。PlanetLab ノードは L2TP によってカプセリングされたパケットを IPv4 パケットとして受け取ることができる。そこで L2TP サーバを仮想マシン上もしくは User Mode Linux 上に配置することによって、提案手法によって構築される IPv6 ネットワークとインターネットの相互接続が可能となる。

提案手法では User Mode Linux を用いている。そのため、User Mode Linux の kernel を書き換えることによって様々なプロトコルスタックを PlanetLab 上で実装することが可能である。本研究では User Mode Linux を XCAST6 対応にすることによって PlanetLab 上で XCAST6 パケットの転送に成功した。

7.3.2 トポロジ構築エージェント

提案手法では静的ルーティングを行なうため、各ノードに対して手動で経路表を設定しなければならない。手動での経路表の設定では設定ミスが起こる可能性がある。また、大量のノードに対して手動で経路の設定を行なうことは現実的ではない。

そこで、経路の設定作業の負荷を低減させるためにトポロジ構築エージェントを作成した。トポロジ構築エージェントは YAML というデータ形式で記述されたトポロジの設定ファイルを元に、各ノードの User Mode Linux に対して経路の設定を行なう。YAML は構造化されたデータ形式であり、人にとって読みやすいようにスペースのインデントで階層を表現する。

提案手法とトポロジ構築エージェントを用いることによって、PlanetLab 上で容易に IPv6 パケットのルーティングを実現できる。

7.3.3 XCAST6 on PlanetLab

XCAST wg では提案手法を用いて、PlanetLab 上における XCAST6 パケットの転送を実現した。実装は User Mode Linux を XCAST6 に対応させることによって行なった。User Mode Linux 上から XCAST6 パケットを送信し、宛先の User Mode Linux において XCAST6 パケットの受信を確認した。今後、提案手法を用いることによって様々なトポロジで XCAST6 の実験を行なうことが可能である。

PlanetLab 上での XCAST6 の実装は提案手法の User Mode Linux のみに変更を加えて行なった。これと同様に XCAST 以外の IPv6 を用いるプロトコルスタックも User Mode Linux に変更を加えることによって PlanetLab 上に実装可能である。

7.4 今後の課題

提案手法は VINI のアーキテクチャを参考に IPv6 のルーティングを実現している。今後は VINI との差別化をどのように行なっていくか考えていく必要がある。差別化の方向性の一つとして、トポロジ構築エージェントの設定をより容易にするという点があげられる。また VINI と差別化をせず、VINI との統合を図るという方向性もある。

提案手法は現時点において、インターネットとの相互接続ができない。つまり、ユーザに対するサービスとして IPv6 ネットワークを提供できない。これでは PlanetLab の利点を活かしきれないため、インターネットとの接続性の確保は必要である。

第 8 章　PlanetLab on StarBED

8.1 概要

XCAST wg では XCAST6 に対応した PlanetLab を提案し、UML を組み合わせる方法で実装した。インターネット上に実際に XCAST6 対応の PlanetLab を配置しても問題ないかを検証するため、StarBED という実験施設の上で PlanetLab ネットワークを構築し、検証を行う予定である。本稿ではその進捗と今後の予定について報告を行う。

8.2 StarBED

StarBED は NICT が設立した大規模な実験施設である。680 台のノードを備えており、インターネットのような大規模ネットワークを再現し、検証を行うことができる。また、StarBED では実験用のネットワークと管理用のネットワークに分け、作業によるトラフィックが実験結果に影響を与えないようになっている。具体的には各ノードに複数のインターフェイスがあり、管理ネットワーク側では DHCP を用いて管理用の IP アドレスや設定を与えている。

8.3 ネットワークトポロジ

ネットワークトポロジは実環境に近いものが好ましいため、Abilene を意識したトポロジで構成した。今回の実験では 11 台の PlanetLab ノードを用意して実験ネットワークを構築した。なお XCAST6 のルータ機能の検証のため、PlanetLab ノードがルータも兼ねている。

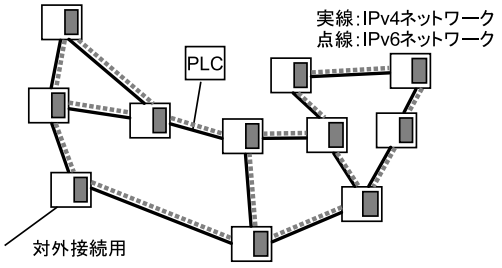

図 8.1.　ネットワークトポロジ

8.4 PlanetLab のインストール

PlanetLab のインストールシーケンスは通常の OS に比べて複雑である。この章では StarBED 環境でのインストールについての報告とより大きな規模の場合についての考察を行う。

8.5 PlanetLab のインストールシーケーンス

まずハードウェアとしては BootCD とノード毎の固有の設定ファイルが入ったフロッピーディスクが必要である。更にネットワーク環境もインストール時に必要であり、PlanetLab ネットワークを管理している PLC というノードと通信を行った上でインストールが行われる。インストール後の起動時にも利用されるため、PlanetLab ノードの数だけ必要になる。

今回は手作業で扱える範囲であったため実際に PlanetLab ノードの台数分の BootCD を用意したが、より大規模な PlanetLab ネットワークを構築する際にはこれを支援する仕組みが必要になるだろう。

8.5.1 マルチホーム環境での PlanetLab のインストール

StarBED で提供されているノードでは管理用、実験用とでインターフェイスが複数あるが、PlanetLab のインストール時にはネットワーク設定を行うインターフェイスを指定できず、自動的に行われるため管理用ネットワーク側に設定を行われる可能性がある。今回は実験ネットワーク用のインターフェイスが指定されたため問題は起こらなかったが、どのインターフェイスに対して設定が行われるかを予め把握しておく必要がある。

8.6 今後の課題

現在は UML での XCAST6 機能の動作は確認したが小さな規模でしか確認していないため、大規模なネットワークでの検証を StarBED 上で行う予定である。また、XCAST の分岐配送機能が有効である場合、無効である場合それぞれについても検証を行う。

その際に必要となる作業としては UML イメージの配置、オーバレイの IPv6 ネットワークの構築などがあるがそれら作業を効率的に行うための手法についても検討していく。

第 9 章　まとめ

XCAST-WG では、本年は、様々な標準化・普及展開に関する活動や XCAST に関する様々な研究開発を展開できた。ハードウェア実装に向けた仕様である XCAST2.0 の構想を 12 月研究会で発表し関係各位のフィードバックを得られた。標準化および普及展開活動の基盤としての IRTF SAM-RG（Scalable Adaptive Multicast Research Group）を正式設立することができた。普及活動として Matsuri と呼ぶネットワーク上のイベントを開催した。また、アドホック通信への XCAST の適用に関する研究開発、拡張が容易なグループ管理システムに関する研究開発、配送順の最適化に関する研究開発、および PlanetLab を用いた普及展開手法に関する研究開発を実施した。2007 年もこれらの活動を継続する予定である。