第 XXV 部

大規模な仮設ネットワークテストベッドの設計・構築とその運用

第 25 部
大規模な仮設ネットワークテストベッドの設計・構築とその運用

第 1 章　2007 年春合宿ネットワークに関する報告

1.1 2007 年春合宿ネットワーク

本章では、2007 年 3 月 5 日から 8 日まで静岡県浜名湖ロイヤルホテルにて開催された WIDE プロジェクト春合宿におけるネットワーク運用、およびそのネットワーク上で実施された実証実験の内容とその結果を報告する。

1.1.1 対外接続用回線

本合宿で構築したネットワークでは、合宿地から WIDE バックボーンおよびインターネットへの接続用回線として、FTTH、VSAT 衛星回線の 2 種類の対外接続用回線を使用した。合宿ネットワークで使用した対外接続用回線を表 1.1 に示す。

本合宿では、各通信回線で帯域や遅延などの特性が異なるため、広帯域な接続性を持つ FTTH 回線を主系統として、VSAT 衛星回線を予備系統としてそれぞれ運用した。しかし、本合宿では、VSAT 衛星回線のための衛星モデム（SIT）の故障により衛星を利用できなかった。そのため、FTTH 回線のみで運用を行った。NTT 西日本が提供するフレッツ速度計測サイトで、FTTH 回線の下りの速度を計測したところ平均 35.33 Mbps のスループットが得られた。WIDE バックボーンとの接続のために、cisco3.fujiawa と gre によるトンネル接続を行った。図 1.1、1.2 に、合宿ネットワーク設営時に行った合宿地のゲートウェイから、gre トンネル経由で、藤沢のホストへ iperf で計測した結果を示す。gre トンネル経由で 16 Mbps のスループットが計測された。合宿参加者が通常利用する分には問題ないスループットであった。

1.1.2 ネットワークの内部構成

合宿地に構築したネットワークでは、合宿参加者が接続するユーザセグメント、DNS や Web などのサーバ群が接続する NOC セグメントから構成された。図 1.3、1.4 に本合宿のネットワークトポロジを示す。

前回合宿時でのアンケート及び合宿参加者の申し込み時にアンケートを実施し、ほとんどの参加者が有線による接続性を必要とせず、無線 LAN の使用を想定していることを確認した。このため、本ネットワークでは無線 LAN を主とした接続性を提供し、有線 LAN 機器の設置が必要最小限となるようネッ

表 1.1.　本合宿で使用した対外接続用回線

回線名	回線数	通信速度
VSAT 衛星回線	1	Uplink/Downlink: 1 Mbps/5 Mbps
FTTH	1	100 Mbps（ベストエフォート）

```
root@gw1:~# iperf -c 203.178.137.93
------------------------------------------------------------
Client connecting to 203.178.137.93, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  3] local 203.178.158.33 port 47232 connected with 203.178.137.93
port 5001
[  3]  0.0-10.0 sec  19.4 MBytes  16.3 Mbits/sec
```

図 1.1.　iperf の結果（TCP）

```
root@gw1:~# iperf -u -c 203.178.137.93
------------------------------------------------------------
Client connecting to 203.178.137.93, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:   103 KByte (default)
------------------------------------------------------------
[  3] local 203.178.158.33 port 33518 connected with 203.178.137.93
port 5001
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.153 ms    0/  893
(0%)
```

図 1.2.　iperf の結果（UDP）

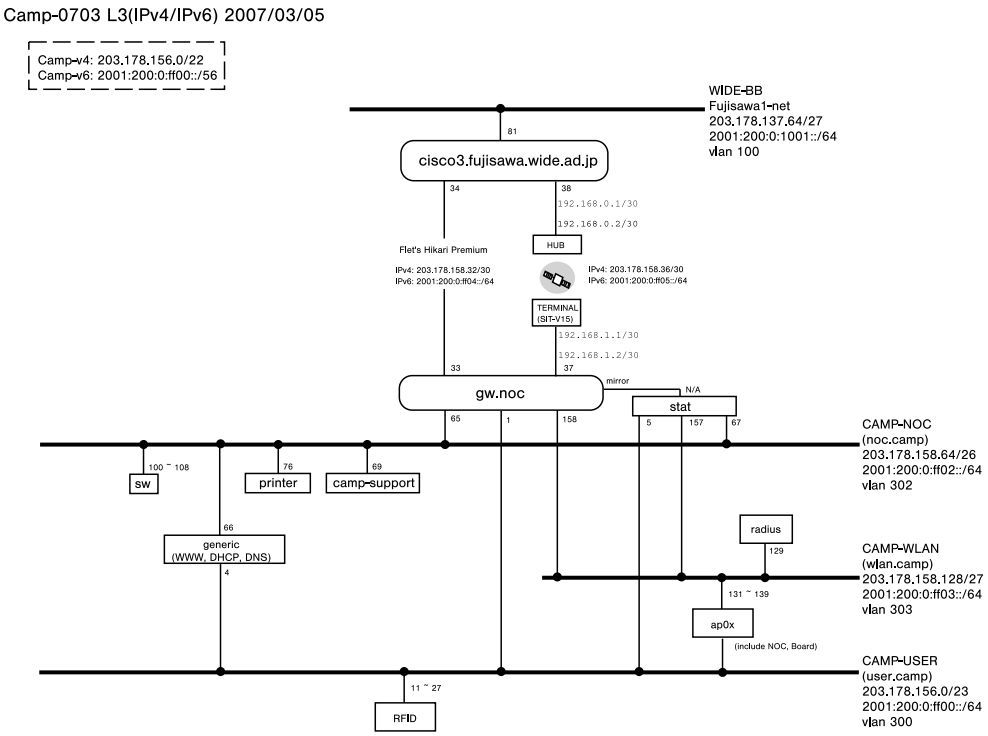

図 1.3.　WIDE 春合宿ネットワークトポロジ（Layer3）

トワーク機器を配置した。また、ユーザセグメントには合宿参加者が全員接続可能な大きなアドレス空間を持たせ、無線 LAN、有線 LAN ともにユーザセグメントに収容した。

1.1.3 合宿ネットワークでの問題点

合宿会期中に報告された問題点と行った解決策について述べる。

MTU と MSS

合宿地と藤沢間でトンネリングを行うため、MTU と MSS に注意が必要である。2007 年春合宿では、トンネルを設定する際に MTU サイズを 1500 以下に指定した。また、藤沢のトンネル受け側ルータで、DF ビットを 0 にすることでネットワークの安定化を図った。ほかにも、Windows サーバでは Path MTU Discovery を処理できないので、MSS を小さくした。

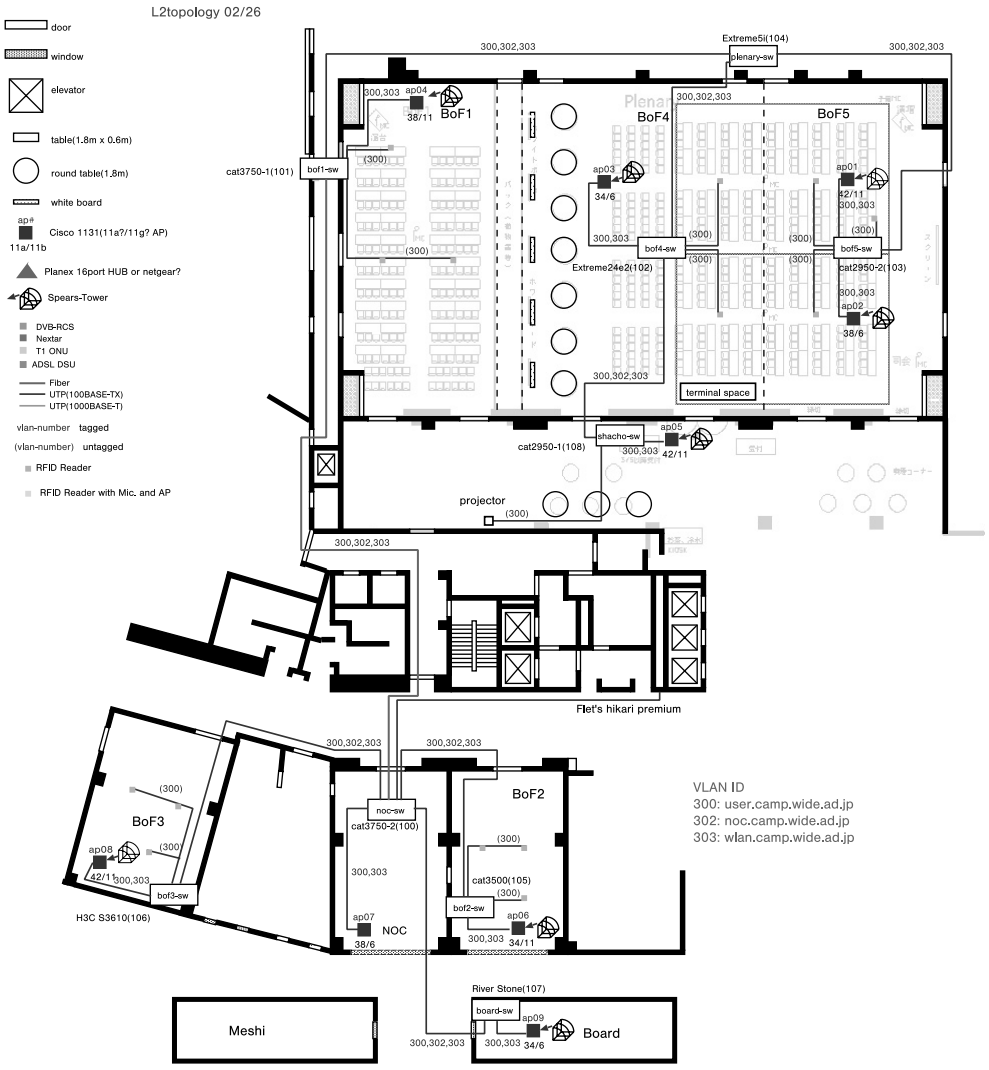

図 1.4. WIDE 春合宿ネットワークトポロジ（Layer2）

Mac OS X の ipfw

Mac OS X の Personal Firewall を有効にしている場合、IPv4 で外部へセッションが確立できないことが報告された。これは、Mac OS X の ipfw 実装に問題があるため、合宿ネットワーク側での対応は難しい。ベンダの改善が望まれる。

合宿ネットワーク用アドレスブロック

合宿ネットワークでは、例年同じアドレスブロックを利用している。2007 年春合宿では、利用しているアドレスブロック内の一部の IPv4 アドレスに対して、不特定多数のホストから ICMP Ping Flood による DoS 攻撃が検出された。今後の合宿ネットワークでは、ファイヤウォールなどの DoS 攻撃対策を検討する必要がある。

1.1.4 Camp Support System の運用

合宿中の議論を活発かつ円滑に行うために Camp Support System を構築した。第 1.2 節に詳細を示す。

1.1.5 合宿ネットワークを利用した実験

本合宿では以下の実験が行われた。

1. WIDE 証明書を利用したネットワークアクセスの提供

第 1.3 節に詳細を示す。

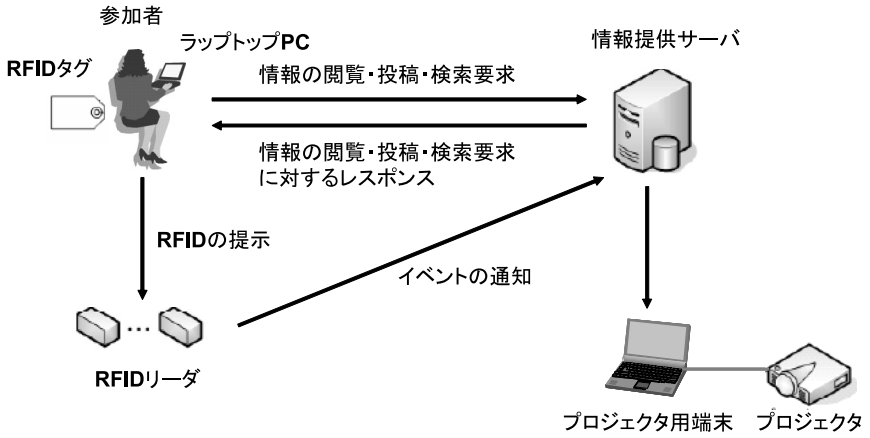

図 1.5.　Camp Support System の構成イメージ

1.2 Camp Support System の構築と運用

1.2.1 Camp Support System とは

Camp Support System は合宿支援システムとも呼ばれ、合宿中の議論の活性化を狙ったシステムの総称である。Camp Support System は、大別すると以下の 4 つから構成される。

- Introduction System
  発表者や発言者のプロフィールを表示させるシステム
- Ticker System
  PC からの連絡事項を表示させるシステム
- Log System
  分散ログを可能にし、ロガーの負担を軽減するためのシステム
- Portal System
  WIDE プロジェクト参加者のコミュニケーションを促進するためのシステム

図 1.5 に Camp Support System の構成イメージを示す。Camp Support System が提供する機能のユーザインタフェースは Web アプリケーションとして実装されている。そのため、参加者は Web ブラウザが利用できる端末さえ持っていれば、Camp Support System が提供する機能を利用可能である。情報を閲覧・投稿・検索したい参加者は、ラップトップ PC などから Web ブラウザを用いて情報提供サーバにリクエストを送信する。情報提供サーバは、参加者からのリクエストに応じてレスポンスを返す。また、Camp Support System を運用する際に、参加者には 1 人 1 枚 RFID タグを配布し、会場には複数の RFID リーダを設置する。話者となる参加者が RFID タグを RFID リーダにかざすことで、情報提供サーバにイベントが通知される。情報提供サーバは、会場に設置した情報表示用のプロジェクタを通じて、話者情報を会場へ通知する。そのほか、情報表示用のプロジェクタには会場で行われているイベント情報や、合宿 PC からのお知らせなどが通知される。

2007 年春合宿の Camp Support System については、Wiki[1]にまとめられているので、そちらも参照されたい。

次節以降では、Camp Support System の詳細を説明する。

1.2.2 Introduction System

1.2.2.1 概要

Introduction System は、

- BoF に関する情報提供
- 部屋に関する情報提供
- 人に関する情報提供

を行うことで、参加者や BoF 進行役、ロガーを支援するシステムである。具体的には、各部屋に置かれたプロジェクタを利用して、現在の話者の情報や次の BoF の情報を表示する。

1.2.2.2 システム構成

Introduction System は Camp Support System の CoreDB と Introduction System 専用のデータベース、PHP スクリプトから構成される。実際に運用する際にはブラウザから以下のアドレスにアクセ

1　https://camp-support.naist.wide.ad.jp/wiki/

スすることで、Introduction System と画面を共有する Ticker System がそれぞれ呼び出される。

- 各 BoF 部屋のプロジェクタに投影するページ
  - BoF1: `http://camp-support.camp.wide.ad.jp/disp/?id=1`
  - BoF2: `http://camp-support.camp.wide.ad.jp/disp/?id=2`
  - BoF3: `http://camp-support.camp.wide.ad.jp/disp/?id=3`
  - BoF4: `http://camp-support.camp.wide.ad.jp/disp/?id=4`
  - BoF5: `http://camp-support.camp.wide.ad.jp/disp/?id=5`
  - Plenary: `http://camp-support.camp.wide.ad.jp/disp/?id=6`

指定されている id は、CoreDB の部屋テーブルに登録されている各部屋の id である。

Introduction System を構成する主なスクリプトファイルは以下の通りである。

- show.php：ブラウザからアクセスするスクリプト
- intro_check.php：指定された部屋で現在何を表示すべきかを返す
- show_member.php：人の情報を返す
- show_timetable.php：BoF の情報を返す

処理の流れは以下の通り。

1. ブラウザから show.php にアクセスする。
2. show.php から呼ばれる JavaScript 内から、バックグラウンドで intro_check.php にアクセスする。この処理は定期的（現行では 2 秒毎）に繰り返される。
3. intro_check.php で、指定された部屋で BoF が開催されているかどうかを判定する。BoF が開催中の場合は、introdb にアクセスする。introdb には最後にタグをかざした人のデータ（を表示するためのアドレス）が格納されている。BoF 開催中でない場合は、状況に応じて次の BoF の情報を表示するためのアドレスなどが返される。
4. intro_check.php から返されたアドレスにアクセスし表示するデータを取得する。

1.2.2.3 今後の課題

現状のシステムの課題や運用上発生したトラブルと、改善案などは以下の通りである。

タグをかざしてからのレスポンスが遅い

ポーリング間隔は 2 秒であったので、もう少し間隔を短くしても良いと考えられる。タグリーダ回りを含めた根本的なチューニングが必要である。

プロジェクタの解像度が異なる問題

BoF 部屋によって解像度が異なるため、表示が画面内に収まらなかったり、文字が小さすぎて読めなかったりした。BoF 部屋ごとにスタイルシートを変更することで対処したが、あくまで応急処置であり、どのスタイルシートを使うかは、スクリプト内にハードコーディングしてある。可能であれば、プロジェクタ側で対処できると良い。スクリプト側で対応する場合は、プロジェクタの情報をデータベースなどに登録する必要がある。

日本語のフォントサイズが大きくならない問題

プロジェクタに接続していた PC に日本語の TrueType Font が入っていなかったので、kochi-ttf を ports からインストールし、X を再起動することで改善した。

1.2.3 Ticker System

1.2.3.1 概要

Ticker System とは、合宿期間中 Camp PC から参加者向けに様々な情報を表示する、電光掲示板のようなものである。各 BoF 部屋、Plenary 部屋にはメインのプロジェクタ以外に、多目的用のプロジェクタを設置してある。そのプロジェクタ（ブラウザ）の画面領域を Introduction System と共有して利用する。利用例を図 1.6 に示す。

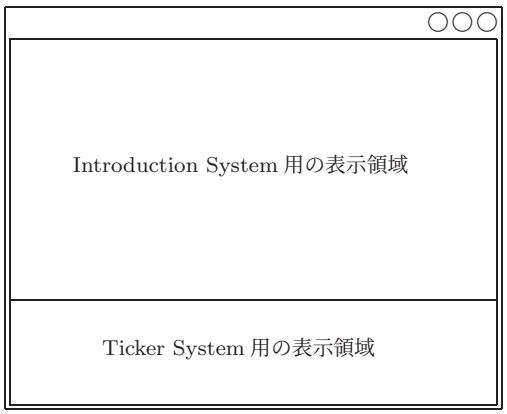

図 1.6. 画面配置

```
tickerdb=> select * from ticker where (current_timestamp, current_timestamp)
tickerdb-> OVERLAPS (start_time, end_time) AND \
                NOT((room & B'001010') = B'000000')
tickerdb-> AND NOT(cancel) ODER BY start_time desc;
```

図 1.7.　ticker での SQL 例

表 1.2.　Ticker System DB のテーブル

	名前	データ型	制約	デフォルト	その他
NUM	num	integer	primary key	serial	
入力者	input_user	text	not null		
メッセージ	message	text	not null		
配信部屋	room	bit(6)	not null		
開始時間	start_time	timestamp	not null		
終了時間	end_time	timestamp	not null		
緊急	asap	boolean		false	メッセージの緊急度
キャンセル	cancel	boolean		false	表示の終了

プロジェクタに表示される上位部分を Introduction System に使用し、下位部分を Ticker System で使用する。情報を表示するページは PHP などで作成し、そのページをプロジェクタに接続された PC からブラウザで読み込む。今回は Introduction System と Ticker System の担当が別々だったため、それぞれ独立して開発し、フレームページで表示だけ統合して描画させた。

1.2.3.2 システムの仕組み

仕組みは、データベース上にメッセージと表示期間が格納されており、表示用の Web ページ（PHP）はアクセスされると、現在の時間に表示されるべきメッセージを表示する。図 1.7 に SQL の例を示す。

これは BoF2 と BoF4 に表示すべきメッセージを受け取るクエリである。表示したい部屋のビットを GET または POST でこのページに送ると、メッセージなどの情報が今回は :: で接続されて返す仕組みにした。ブラウザ側でこのページに JavaScript を使ってバックグラウンドで 1 秒毎にアクセスし、:: で文字分割して表示メッセージを更新した。実際には JSON 形式などで渡した方が、ブラウザ側で様々な処理を行いやすい。

1.2.3.3 テーブル

Ticker System 用のテーブル構成を表 1.2 に示す。

入力者は、そのメッセージを入力した人の名前を格納する。配信メッセージは各部屋に表示されるので、その責任を明らかにするためである。

表 1.3.　配信部屋の bit 表現

MSB					LSB
0	0	0	0	0	0
Plenary	BoF5	BoF4	BoF3	BoF2	BoF1

表 1.4.　BoF1、BoF5、Plenary にメッセージを配信する場合

MSB					LSB
1	1	0	0	0	1
Plenary	BoF5	BoF4	BoF3	BoF2	BoF1

配信部屋に関しては複数の部屋を選んで配信する仕様にするため、ビットフィールドを使って表現している。通常合宿で使用される部屋は BoF1〜5 および Plenary の 6 部屋である。表 1.3 のように、6 bit のフラグで配信部屋を制御した。最下位 bit を BoF1 に、最上位 bit を Plenary に割り当て、bit が立っている部屋にメッセージを配信する仕組みになっている。表 1.4 は BoF1、BoF5、Plenary の 3 部屋にメッセージを表示するときの例である。

緊急とは、重要度の高いメッセージにつけるフラグである。これが ON になると実際画面に表示されるとき、表示優先度が高く表示される。今回は表示の関係上未実装のまま終えた。実際には boolean ではなく数段階のプライオリティにしておき、重要度の順にフォントを大きくしたり、色を変えて表示で

きるようなシステムが望ましい。

キャンセルとは、時間であっても表示しない場合につけるフラグである。配信するメッセージを削除したいとき、データベース上から削除すると完全に消えてしまい、後に見直すとき都合が悪い。そのためキャンセルフラグを立て擬似的に削除する。

1.2.3.4 Ticker System メッセージ編集ページ

編集ページはCamp PCメンバのみが扱えるようにした。サーバにmoCA認証が組み込んであるので、WIDE番号でCamp PCメンバか判断し、ページを表示させる。編集ページではメッセージ、表示期間、配信部屋などを入力する。またmoCA認証より自動的に入力ユーザ名を取得し、同時にデータベースに格納する。今回は実装しなかったが、表示の優先度変更、色、フォントサイズの変更など様々な追加機能が考えられる。

1.2.3.5 moCA 認証

moCAによるクライアント認証を組み込んでいる場合、SSLで認証を用いると、以下のような環境変数に各種情報が格納される。PHPなどでこれらを処理するスクリプトを書き、ユーザの識別を行うことができる。

- `$_SERVER["SSL_CLIENT_VERIFY"]`
  クライアントデータが取得できた場合はSUCCESS、失敗した場合はNOが格納されている。この値をみてユーザがクライアント証明書を使ったか判断する。
- `$_SERVER["SSL_CLIENT_S_DN_CN"]`
  スペース区切りでクライアントデータが格納されている。
  (WIDE番号) (名前) (苗字) WIDE
- `$_SERVER["SSL_CLIENT_S_DN_Email"]`
  メールアドレスが格納されている。

1.2.3.6 Ticker System メッセージ表示ページ

実際にメッセージを表示するページである。プロジェクタ側PCのブラウザで、このページにJavaScriptを使ってバックグラウンドで1秒毎にアクセスし、メッセージごとに::で分割された文字列を受け取る。このメッセージはSQLから返る時点で新着順に並んでいるので、::で単純分割して列挙表示した。XMLやJSON形式などで渡した方が、ブラウザ側での汎用性が高まる。

1.2.4 Log System

camp0703では、前回のcamp0609におけるintro systemに含まれていた分散ログシステムを運用した。

1.2.4.1 システム構成

主なシステム構成は前回2006年秋合宿のものと同じである。詳細については、2006年秋合宿のstaff wiki[2]に詳しいのでそちらを参照されたい。ここでは、2007年春合宿の変更点について述べる。

主な構成要素は2006年秋合宿と同様であり、以下のようなものを利用している。

- データベース：PostgreSQL
- HTTPサーバ：Apache 2.0
  - 言語：PHP 5
  - RFID：Java 1.5
- User Interface：Webブラウザ（Ajax)

1.2.4.2 主な変更点

2007年春合宿での主な変更点は以下の3点である。

- 分散ログシステム部の独立化
  - パス指定方法の変更
  - 必要機能の切り出し
- データベース構成の変更
  - 2007年春合宿システム用データベースアクセス関数の構築
- ユーザインタフェースの改良
  - 新規発言者登録機能の追加
  - 発言者登録ボタンの位置の変更

1.2.4.3 Log DB

今回のシステムにおいて利用したLog System用のデータベースののSQLは図1.8のようになる。

今回のシステムでは各機能ごとに別々のデータベースを作成した。これにより、各機能ごとの切り分けが簡単にできるようになったのに対し、データベースへのアクセスが煩雑になった。

1.2.4.4 課題

合宿参加者への通知が不十分だったため、あまり利用されていなかった。また、プレナリやBoFでの

2 https://member.wide.ad.jp/wide-confidential/camp/06autumn/staff/pukiwiki-1.4.6/index.php?intro%20system

```
CREATE TABLE log (
  log_id integer PRIMARY KEY DEFAULT SERIAL,
  room_id integer,
  title varchar NOT NULL,
  status varchar
);

CREATE TABLE log_transaction (
  transaction_id integer PRIMARY KEY DEFAULT SERIAL,
  log_id integer REFERENCES log ON DELETE CASCADE,
  command varchar NOT NULL,
  p_box_id integer,
  box_id integer NOT NULL,
  wide_no integer NOT NULL,
  time timestamp NOT NULL DEFAULT current_timestamp,
  body varchar
);
```

図 1.8.　Log DB

ログは、個人によって取りやすい環境が異なるため、Log System の運用自体について今後検討が必要である。

1.2.5 Portal System

1.2.5.1 Portal System 導入の経緯

合宿中には合宿 Web、Log System、Introduction System、その他様々な Web サービスが提供される。Camp Support System のミーティングで話し合う中で、それらのサービスに容易にアクセスできるようなポータルサイトを作りたいという事になった。また同時に合宿参加者同士で情報交換や交流を支援するシステムにしたいという要望が出た。さらに Introduction System で表示するプロフィールの管理や、Ticker System の情報表示、メッセージをやりとりする機能も盛り込むことにした。システムのすべてを 1 から作り上げることも考えたが、オープンソースで提供されている SNS「OpenPNE」というものがあることがわかり、今回の合宿では OpenPNE をベースにしたポータルサイトを立ち上げることになった。

1.2.5.2 OpenPNE

OpenPNE[3]はオープンソースの SNS であり、mixi によく似たインタフェースを持つ。日記、メッセージ、コミュニティ、フレンド、プロフィールなどの SNS の基本的な機能を備えている。OpenPNE を動かす為には Apache、MySQL、PHP の環境を用意する必要がある。現在様々なバージョンが提供されているが、2007 年春合宿時点で最新版の OpenPNE-2.6.4 を利用した。

1.2.5.3 セットアップ（環境構築）

MySQL のインストール

FreeBSD上に ports を利用して mysql-client-5.0.33 及び mysql-server-5.0.33 をインストールした。当初、MySQL4.1 を導入したが、Introduction System と連携のために view を利用したいということで、MySQL5.0.33 に変更した。

phpMyAdmin のインストール

MySQL の管理を容易にするためにデータベースを GUI で管理可能な phpMyAdmin[4]を ports からインストールした。

PHP のインストール

使用した PHP のバージョンは php-5.2.1.2 である。ports からインストールした後、PHP モジュールを追加インストールした。

```
#cd /usr/ports/lang/php5-extensions/
#make config
#make install
```

OpenPNE を動かすために必要となるモジュールは以下の通りである。

```
[X] GD GD library support
[X] MCRYPT Encryption support
[X] MYSQL MySQL database support
[X] MYSQLI MySQLi database support
```

OpenPNE のインストール

ソースは SourceForge.net よりダウンロードした。tar.gz 圧縮を解凍すると、中にインストール方法に関する詳細な html ドキュメントがあるので参照する。また、次のサイトが参考になった。

- SNS 構築（OpenPNE）`http://fedorasrv.com/openpne.shtml`

3　`http://openpne.jp/`
4　`http://www.phpmyadmin.net/`

つまづきやすいポイント

- http://camp-support.naist.wide.ad.jp/openpne/?m=setup にアクセスしても以下のように表示される。

> 現在、サーバが混み合っているか、メンテナンス中です。
> ご迷惑をおかけいたしますが、しばらく時間を空けて再度アクセスしてください。

このメッセージが表示される場合は、MySQLへの接続に問題が生じていることを示している。プロセスを確認して MySQL が動作しているか、phpMyAdmin を利用して OpenPNE のテーブルは正しく作られているか確認する。phpMyAdmin でも接続できない場合は PHP から MySQL への接続が出来ていないため、phpinfo スクリプトで MySQL モジュールが導入されているか確認する。

- 一部の画像が表示されない
  OpenPNE では画像を DB 負荷軽減のために画像をリサイズしてキャッシュする機能を備えている。その際に GD ライブラリを利用する。phpinfo スクリプトで GD ライブラリが利用可能か確認する。
- ログイン画面は表示されるがログインしようとしてもログイン画面に戻される。
  PHP の認証を行うときのセッション情報を保存できない場合に生じる。phpinfo スクリプトで session.save_path を確認し、書き込み権限があるか確認する。
- Admin 画面でメールアドレスだけ文字化けする
  phpinfo スクリプトで MCRYPT モジュールが導入されているか確認する。また、OpenPNE の設定ファイル内の define('ENCRYPT_KEY', 'hoge'); が変更されてしまうと暗号化されているメールアドレスを復元できない。特にサーバを移行する場合などは注意が必要である。

カスタマイズ

https://camp-support.naist.wide.ad.jp/wiki/index.php にカスタマイズした内容やソースコードを公開した。

1.2.5.4 Portal System の運営を振り返ってみて

OpenPNE の利用について

OpenPNE は SNS の基本的な機能を備えているため、ベースのシステムとして利用するには有効だと思われる。盛んに開発が進められているため、最新の情報をチェックするようにするとよい。WIDE 合宿に合わせた機能の追加、削除、カスタマイズを行えるとよい。

ユーザーの事前登録

合宿参加者の情報を事前に登録した。アンケートパスワードによるログインはできれば証明書によるログインと併用できるとよい。また、一部ユーザの登録漏れがあった。合宿中に登録することで対応したが、事前に確認をすることが望ましい。

Introduction System との連携

合宿参加者のプロフィール情報を OpenPNE で参照、変更できるようにした。Introduction System では、OpenPNE のテーブルを view を用いて変換したものを参照するようにした。

その他合宿情報の提供

information 欄に合宿関連ページへのリンクを表示、右側バナー部にアンケート記入のお願いと Board Plenary での宿題コミュニティの情報を表示した。

Ticker System との連携

Ticker System で提供される情報をインラインフレームで Portal System の information に表示されるようにした。Ticker System を用いて手軽に情報を参加者に提供できたのはよかった。

DB の負荷

今回の合宿の規模では特に負荷の影響による接続問題などは生じなかった。Portal 登録者数は約 200 名。ログイン者数は約 150 名（登録者数の 75%）。今後、ユーザー数が増加する場合や、サーバのスペックによっては負荷分散なども検討する必要がある。なお OpenPNE では画像だけ別サーバのデータベースを利用する機能が実装されているので、それを利用することも可能である。

1.2.5.5 アンケート結果

今回、合宿での Portal System 運用について、参加者アンケートにて「面白い」、「交流ができた」というような声が多数寄せられた。また、常時運用を希望する声もあり、合宿参加者だけでなく、WIDE メンバーが利用できる SNS としての常時運用を検討していく必要がある。

Portal System の運用や利用について、参加者へのアナウンスが遅くなってしまった。事前に Portal System の存在を知ってもらい、できるだけ多くの人に合宿前から利用できるサービスとして提供できると良い。

「画像のアップロードだけでなく、文章ファイルも使えるようにしてほしい」、「証明書を使ってログインできるようにしてほしい」、「携帯電話からも利用できるようにしてほしい」などといった機能追加の要望も寄せられているので、WIDE Portal System として必要な機能を検討し、今後の運用につなげていってもらいたい。

1.3 合宿ネットワークを用いた実験

1.3.1 WIDE 個人証明書を利用したネットワークアクセスの提供

本実験では、各 WIDE メンバーに配布される個人証明書と WPA-EAP を組み合わせた認証・暗号化に対応した無線ネットワークアクセスの提供を行い、次の点についての評価を行った。

- 導入や運用に必要なコスト
- クライアント側における利用可能性
- 100 から 150 ノード程度の利用環境におけるシステムの安定性

1.3.2 構成と設定の要点

本システムは、無線アクセスを提供する 9 台のアクセスポイントと、証明書検証を行う認証サーバシステムで構成されている。今回は、アクセスポイントとして、Cisco 社製の Aironet シリーズを利用し、また、認証サーバシステムとして、x86 サーバ上に FreeBSD および、FreeRadius を導入したシステムを使用した。

アクセスポイント側における、WPA-EAP 対応として、以下の設定を通常の設定に加え、

設定例

```
aaa group server radius rad_eap
 server 192.168.0.1 auth-port 1812 acct-port 1813
!
aaa authentication login eap_methods group rad_eap

dot11 ssid wide
   vlan 209
   authentication open eap eap_methods
   authentication key-management wpa
   guest-mode
!
radius-server host 192.168.0.1 auth-port 1812 \
acct-port 1813 key 7 SHAREDKEY
```

運用を行った。なお、認証サーバ（FreeRadius を稼働）の IP アドレスは 192.168.0.1 である。

1.3.3 運用結果

運用に要するコストは、RADIUS を利用した機器ノードの MAC アドレス検証を実施していたため、運用上大きく変わらなかった。また、MAC アドレス検証についても、同一サーバ、同一ソフトウエア上で、平行して実施することが可能であった。

次に、利用可能なクライアントの調査結果は、148 ノード中、45 ノードが WPA-EPA を使用した。利用可能な OS は、

- NetBSD
- FreeBSD
- Linux 2.6
- Nokia E60
- Mac OS X (10.4/10.3)
- Windows XP
- Windows Vista
- Windows Mobile 5.0

であった。使用されたサプリカント（通常は、無線 LAN カードなどとともに提供される）は、

- OS 付属のもの
- IBM (Lenovo) Access Connections
- wpa-supplicant
- NEC ワイヤレスクライアントマネージャー 4.1
- Cisco Aironet

などであった。一方、接続できない事例として、27 名からの報告があり、接続できない環境は以下の通りであった。

- Windows 上で、Corega 製/バッファーロー製のサプリカント

- FedoraCore 6
- Windows Vista

そのほかに、搭載されているサプリカントバージョン次第では、接続できない事例、サプリカント自体がWPA-EPA への対応ができない事例などがあった。

次にシステムの安定性について述べる。運用中におけるアクセスポイントのロードアベレージは、0–7パーセント程度であり、また、システムが不安定になることや予期しない停止などは発生しなかった。したがって、全体の利用者が 150 名程度、その内、WPA の利用者が、45 名程度のネットワークでは、負荷の面における問題点を見いだすことはできなかった。

このため、より高い負荷をかけた環境における検証が必要と考えられるが、年々アクセスポイントの性能が向上しているため、WIDE 合宿規模の運用では、アクセスポイントに対して、高い負荷をかけることができなくなっている。このため、大規模運用における問題点（ハンドオーバや無線 LAN のコンジェスチョンなどにともなう影響など）を表面化させることが難しい。よって今後、大規模運用における検証を行うためには、異なるアプローチの検討が必要である。

第 2 章　2007 年秋合宿ネットワークに関する報告

2.1 2007 年秋合宿ネットワーク

本節では、2007 年 9 月 11 日から 14 日まで長野県信州松代ロイヤルホテルにて開催された WIDE プロジェクト秋合宿におけるネットワーク運用、およびそのネットワーク上で実施された実証実験の内容とその結果を報告する。

2.1.1 対外接続用回線

本合宿で構築したネットワークでは、合宿地から WIDE バックボーンおよびインターネットへの接続用回線として、VSAT 衛星回線、そして ADSL の 2 種類の対外接続用回線を使用した。表 2.1 に本ネットワークで使用した対外接続用回線を示す。

本合宿では、各通信回線で帯域や遅延などの特性が異なるため、広帯域な接続性を持つ ADSL 2 回線を主系統とし、遅延が大きく帯域も十分ではない VSAT 衛星回線を予備系統としてそれぞれ運用した。また、ADSL 1 回線では合宿全体のトラフィックを捌くには不十分であるため、トラフィックの特性に応じて通信回線を選択する Policy-based Routing を行い、各回線の負荷分散を実現した。

2.1.2 ネットワークの内部構成

合宿地に構築したネットワークでは、合宿参加者が接続するユーザセグメント、DNS や WEB などのサーバ群が接続する NOC セグメント、そして、後述する無線マルチホップ通信を用いた合宿バックボーンネットワーク構築実験を実現するため、この実験のためのセグメントの 3 つのセグメントによって構築した。

図 2.1、図 2.2 に本合宿のネットワークトポロジを示す。

前回合宿時でのアンケート及び合宿参加者の申し込み時にアンケートを実施し、ほとんどの参加者が有線による接続性を必要とせず、無線 LAN の使用を想定していることを確認した。

このため、本ネットワークでは無線 LAN を主とした接続性を提供し、有線 LAN 機器の設置が必要最小限となるようネットワーク機器を配置した。また、後述する無線マルチホップ通信を用いた合宿バックボーンネットワーク構築実験ではプレナリを実験会場とし、有線 LAN によって接続されない独立の無線 LAN ルータを用いていたため、合宿の半分のアドレス空間をこの実験で用いた。プレナリ以外の会場については合宿参加者の大半が接続可能なアドレス空間をユーザセグメントとして用い、無線 LAN、有線 LAN ともにこのユーザセグメントに収容した。

2.1.3 合宿ネットワークでの問題点

本合宿期間中に報告された問題点とそれに対して行った解決策について述べる。

表 2.1.　本合宿で使用した対外接続用回線

回線名	回線数	通信速度
VSAT 衛星回線	1	Uplink/Downlink: 768 Kbps
ADSL	2	ベストエフォート

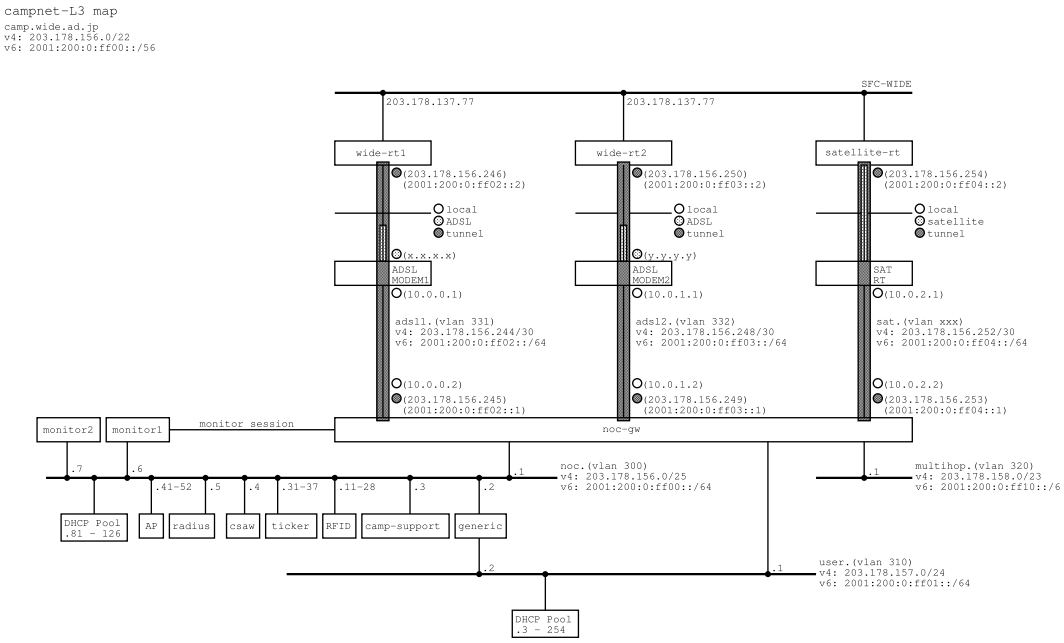

図 2.1.　WIDE 秋合宿ネットワークトポロジ（Layer3）

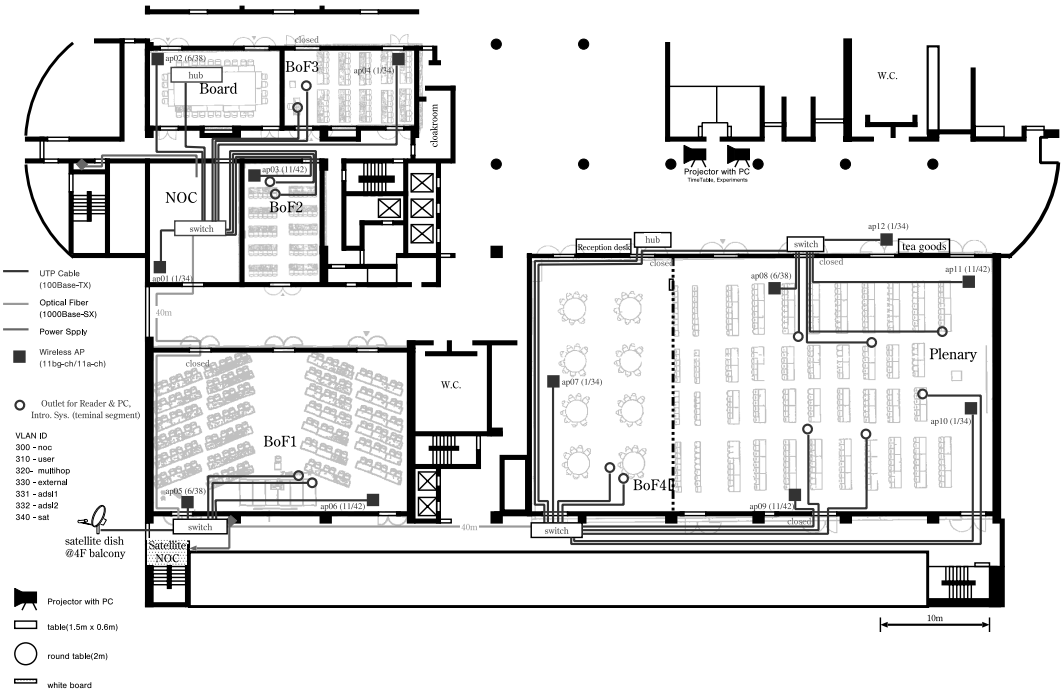

図 2.2.　WIDE 秋合宿ネットワークトポロジ（Layer1/2）

MTU と DF bit

合宿地において WIDE ネットワークのアドレスを利用してネットワークの運用を行うため、合宿地と藤沢との間でトンネルを用いて接続を行っていた。そのため、対外線として利用している ADSL の MTU サイズよりさらに MTU サイズが小さくなってしまい、DF bit を立ててデータを送信してくる一部のサービスを合宿地で利用できない状態になっていた。そこで、藤沢側のトンネル接続ルータにおいて DF bit を落とすことにより一部の利用できなかったサービスを利用することができるようになった。

2.2 合宿ネットワークを利用した実験

本合宿では以下の実験が行われた。

1. The Wireless Multi-Hop Multi-Channel Network for WIDE Camp Backbone
2. Overlay GHC Prime Challenge
3. iTunes on a P2P Network
4. 複数セッションに着目したボットや P2P ファイル交換ソフトの検知手法の実験

また、合宿中の議論を活発かつ円滑に行うために合宿支援システムを構築した。

2.2.1 合宿支援システム

合宿支援システムは合宿中の議論の活性化を狙ったシステムの総称である。特に発表者や質問者のプロフィールを会場前方のスクリーンに表示するイントロシステム（通称、誰が喋ってるのシステム）は基本サービスとして提供することが PC に求められている。今回の合宿では、合宿支援システムは CSAW ワーキンググループの構築した WIDE メンバ用 SNS である CSAW（`https://csaw.wide.ad.jp/`）をメンバの情報データベースとして利用した。

2.2.2 The Wireless Multi-Hop Multi-Channel Network for WIDE Camp Backbone

2.2.2.1 背景

災害地での救助活動にロボット、コンピュータ技術を利用する研究が近年活発に進められている。従来、被災地での情報収集作業は、生身の人間が危険な地域に乗り込んでいく必要があった。このような現場は、特に災害が発生した直後は危険であり、二次災害を引き起こす可能性が高い。また自然災害だけでなく、近年ではテロなどの人為的な災害の発生も考えられ、その手段として使われる爆発物や有毒ガスなどは、被災者のみならず救助者の生命をも危険にさらす可能性がある。

この問題を解決する手段として、被災地域の情報収集を救助ロボットを用いて行う試みが続けられている [143]。救助ロボットを用いて情報を事前に把握することで、被災者が存在しない地域に救助者が進入していく危険を減らしたり、救助者の生命に関わるような危険な場所をあらかじめ知ることができるようになる。

現在の多くの救助ロボットは有線接続、あるいは短距離無線通信を用いて遠隔操作される設計になっている。しかしながら、このような接続方式では、ロボット運用者が救助ロボットとともに危険な区域に進入していかなければならない。

我々は、この問題をインターネット技術を用いて解決することを試みる。既存の通信インフラストラクチャが存在しない、あるいは破壊されて利用できない状況であっても、短期間にネットワークを構築し、そのネットワークを用いて救助ロボットの操作、および被災地の情報収集を可能とする技術を実現する。有線ネットワークの敷設は困難と考えられるため、ネットワーク構築の基本は無線ネットワークとなる。被災地が屋外ならば、広域の無線通信技術の利用も考えられるが、屋内や地下での被災地での運用を考慮にいれると、短距離の無線通信を組み合わせたマルチホップ無線通信がより適切であると考えられる。また、被災地の地理形状は複雑になっていることが予想されるため、ネットワークの構成や規模は被災状況に応じて柔軟に対応できる能力を持ち、かつ必要に応じて拡大できなければならない。これらの作業は、それまで構築したネットワークを足がかりに、救助ロボットが新たな無線基地局を増設することで対応できると考える。我々は、このような被災地救助ロボット用のネットワークを Robohoc ネットワークと名付けた。Robohoc ネットワークの要求項目に関しては、[144] を参照してほしい。

2.2.2.2 実験の目的

Robohoc ネットワークを実用的に運用するためには柔軟な無線アクセスポイントの配置、通信可能距離の確保、障害対応、通信品質の確保など、様々な要件を満たす必要がある [144]。本実験では、Robohoc ネットワークの基礎となる、マルチホップ無線ネットワークの運用経験蓄積と基礎データの計測を目的とする。

具体的には、複数の無線ネットワークインターフェースを装備した無線ルータを製作し、それらをメッシュ上に並べ、既存の経路制御プロトコルでパケットの流れを制御した上でパケット転送性能を計測する。

2.2.2.3 問題点

マルチホップ無線ネットワークを構築する場合、各ノード間の通信チャネルとして同一チャネルを用

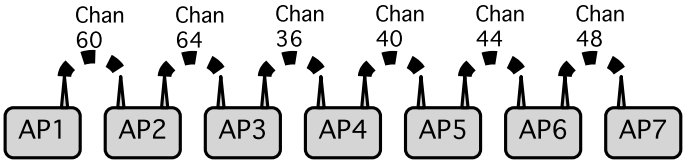

図 2.3. 事前実験で用いたトポロジ。7 台の無線ノードを 1 列に接続した直線的なトポロジを構成。各無線リンクには異なるチャネルが割り当てられている。

いる場合と、異なるチャネルを用いる場合に分類できる。MANET（Mobile Adhoc Network）のように、無線メッシュネットワークを構成しているノードが移動する場合、同一チャネルを用いた通信路を前提とすることが多い。これは、MANET が自律分散ネットワークであり、かつ移動によってノード間の近接度が変化するため、状況に応じて通信範囲に含まれるノードが出現と消滅を繰り返すためである。各ノードが異なるチャネルを用いて通信していると、ノードが近接した場合の通信確立手順が複雑になる。これに対して、IEEE802.15 などの省電力無線メッシュネットワークなどでは、ノード間の通信チャネルは通信時にネゴシエーションによって決定される。このようなネットワークにはコーディネータと呼ばれるノードが存在し、ある 2 点間の通信路の確立はコーディネータによって管理される。メッシュネットワーク全体のトポロジはコーディネータが把握しており、マルチホップ通信の経路上のタイムスロットがコーディネータによって決定されるため、事前に通信チャネルの情報を共有しておく必要がない。

我々が提案する Robohoc ネットワークは、自立分散して運用されることを目標としている。それと同時に、被災地からの情報をできるだけ多く獲得するために、広い帯域も必要としている。そのため、データリンク層としては IEEE802.11 系の広帯域無線通信路を選択した。IEEE802.11 技術を用いてメッシュネットワークを構成する場合、同一チャネルを利用することが多いが、すべてのノードが同一チャネルを用いる設計では、チャネル干渉により、どこかのリンクの通信が、その通信とは無関係な他のリンクの通信を抑制してしまうと考えたため、自立分散運用を維持しつつも、ノード間の通信に複数の無線チャネルを用いることができるように設計しようとしている。

過去 IEEE802.11a をデータリンク層として行った実験では、単純に複数の無線ネットワークインターフェースを装備した機材でネットワークを構成した

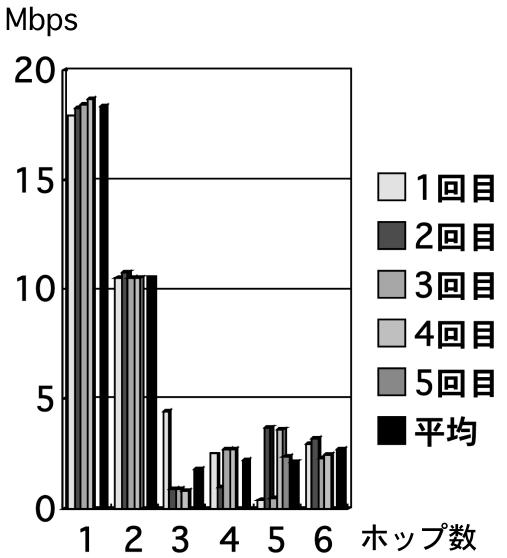

図 2.4. 事前実験結果。無線リンクが 2 つになると、1 つの時と比較して転送性能がおよそ半分に落ちている。以後、ホップ数が増えるごとに著しい転送性能劣化がみられる。

場合、たとえチャネルが異なっていても無線リンク間で干渉が発生し、スループットが低下する現象が確認されている。図 2.3 に事前実験時のネットワーク構成を示す。

図 2.3 のトポロジを用いてスループットを計測した結果が図 2.4 である。

各無線区間には、802.11a で利用できる異なるチャネルを割り当ててある。この状態で、AP1 を始点とし、AP2 から AP7 まで終点とした状態で netperf による UDP のスループット測定を実施した。その結果、無線区間がひとつしかない場合（1 ホップ）は実効帯域として 18 Mbps 近い転送性能が出ているが、ホップ数が増えるに従って転送性能が落ちていく。

調査の結果、転送性能低下の原因は、チャネル間の無線干渉の問題であるらしいことが判明してきた。図 2.3 に示しているとおり、各無線リンクは異なるチャネルを利用しているものの、各ノードに実装されている無線ネットワークインターフェースの距離が

表 2.2. 無線ノードの構成

CPU	Pentium M 1.2 Ghz
Memory	512 MB
NIC	Atheros (miniPCI) × 4
OS	NetBSD 4 に FreeBSD の Atheros ドライバを組み込んだもの

近いため（近接して実装された PCI バスブリッジに 4 つの miniPCI 型無線ネットワークインターフェースを装備）、他チャネルの送信電波をノイズとして検出し、その結果、パケット転送時の性能低下につながっていると考えられる。

そこで本実験では、異なるチャネル間の干渉を抑えるため、無線ノードのアンテナを無指向性から指向性に変更し、実効転送性能の向上を目指す。

2.2.2.4 ネットワーク構成

本実験で利用した無線ノードの写真を図 2.5 に、ノードの仕様概要を表 2.2 に示す。

事前実験で判明した問題に対応するため、今回無線ノード間は Corega の指向性アンテナ（CG-WLANT02I、図 2.6）を用いた。

無線メッシュネットワークは合宿地のプレナリ部屋に敷設した。プレナリ部屋のバックボーントポロジを図 2.7 に示す。各無線ノードは IPv4/IPv6 の両方をサポートし、3 つの無線ネットワークインターフェース（rh4 に関しては 2 つ）で他の無線ノードと接続している。無線ノード間の通信には IEEE802.11a を用いた。各無線リンクのチャネルはなるべく重ならないように調整したが、リンクの総数が 10 となっており、利用できる 802.11a のチャネル数（36、40、44、48、52、56、60、64 の 8 つ）を超えているため、36 と 64 チャネルは 2 カ所で用いられている。

各リンクは IPv6 ではリンクローカル・アドレスを用い、IPv4 では 203.178.158.0/26 から切り出した/30 のネットワークを用いてリンク間通信を実現している。参加者が利用するネットワークは、各無線ノードの 4 つ目の無線ネットワークインターフェースを IEEE802.11b/g で運用して提供した。なお、実験の過程で、802.11g をアクセスポイントモードとして運用している際に無線ノードが異常終了してしまう問題が発見されたため、実験後半は 802.11b のみの運用に移行している。

経路制御は、初日は RIPng と RIPv2 で、二日目以降は OSPFv3 を用いた。

図 2.5. 無線ノード

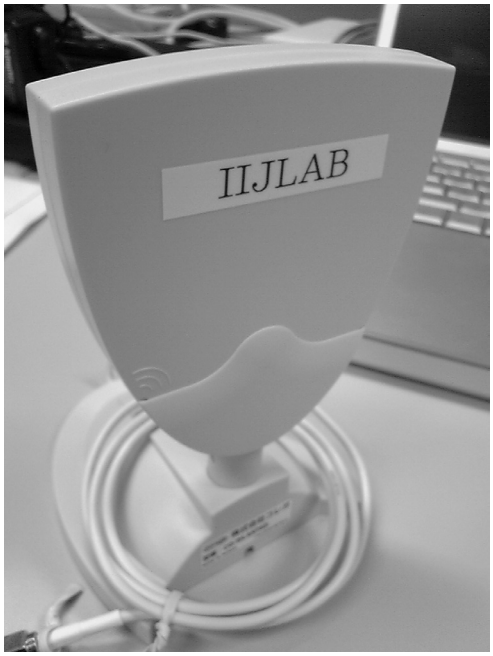

図 2.6. 指向性アンテナ（Corega CG-WLANT02I）

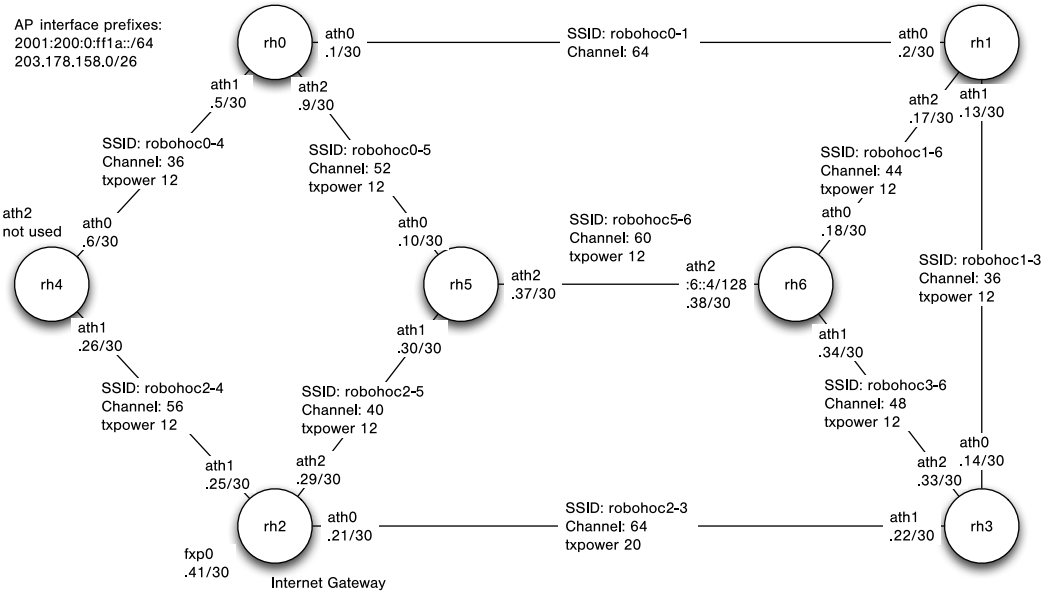

図 2.7.　プレナリ部屋バックボーントポロジの設定。各無線リンクは IEEE802.11a を用いて構成。各リンクに異なる無線チャネルを割り当てることで干渉を軽減。

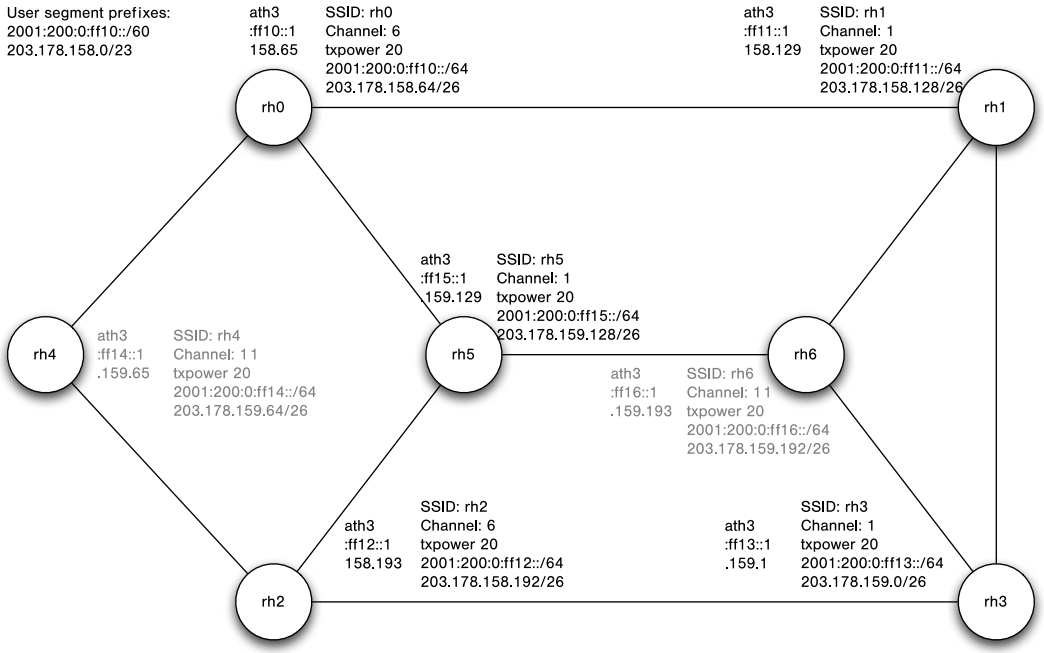

図 2.8.　プレナリ部屋ユーザーセグメントの設定。各無線ノードは IEEE802.11b/g（後半は 802.11b 専用として運用）アクセスポイントとして運用され、それぞれ異なるサブネットを提供。無線ノード rh4 と rh6 は、当初運用予定に入っていたが、干渉軽減のために運用を中止。

インターネットへの接続は無線ノード rh2 のイーサネットインターフェース経由で実現した。よって、このトポロジでの最大ホップ数は無線ノード rh1 から rh2 へ至る 3 ホップということになる。

2.2.2.5 考察

図 2.7、図 2.8 のトポロジでの転送性能計測結果を図 2.9 および図 2.10 に示す。

図 2.9 は無線ノード rh1 から他の無線ノード rh3、rh2、rh4 に対して netperf で TCP 転送性能を計測

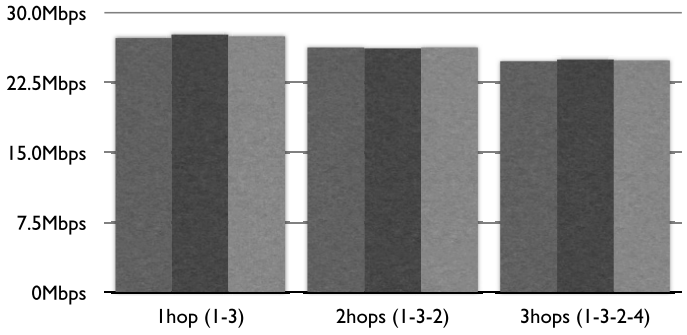

図 2.9. 図 2.7 のネットワークで、無線ノード rh1 と、無線ノード rh3、rh2、rh4 の間で nerperf による TCP スループットを測定した結果。それぞれの無線ノードに対して、各 3 回ずつ計測。

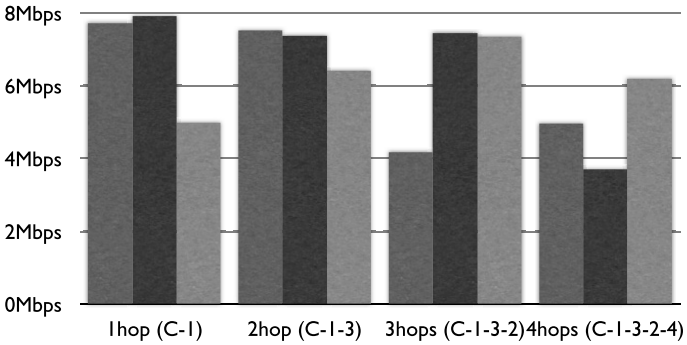

図 2.10. 図 2.7 のネットワークで、無線ノード rh1 に接続した 802.11g クライアントと、無線ノード rh1、rh3、rh2、rh4 の間で nerperf による TCP スループットを測定した結果。それぞれの無線ノードに対して、各 3 回ずつ計測。

したときの結果である。事前実験の結果と異なり、ホップ数が増加しても転送性能がほとんど低下していないことが確認された。また、事前実験で得られた転送性能（およそ 18 Mbps）よりも、本実験でのスループットの方が向上している。これは、指向性アンテナを用いたことによる無線リンクの高品質化や、環境の変化（事前実験は無線ノイズの多いオフィス内で実施された）が影響していると考えられる。

図 2.10 は、無線ノード rh1 に IEEE802.11g を用いて接続したノート PC から、無線ノード rh1、rh3、rh2、rh4 に対して netperf を用いて TCP 転送性能を計測した結果である。無線ノード間の通信（図 2.9）の場合と異なり、計測結果にばらつきがみられる。また、IEEE802.11g の実効転送性能として予想される値（20 Mbps 以上）を大きく下回る値が計測されている。性能がでない原因は無線ノードのアクセスポイント機能の実装の問題ではないかと推測される。その後の解析で今回利用した Atheros カードの hostap モードでは送信時のアンテナポートがランダムに切り替わってしまっていることが判明した。従来のハードウェア構成では 2 つあるアンテナポートの 1 つにしかアンテナが付いていなかった為、相当数のフレームがアンテナの付いていないポートから送出されパケットロスを起こしていたはずである。

2.2.2.6 結論

本実験は被災地救助ロボットネットワーク（Robohoc ネットワーク）の実現に必要な、無線メッシュネットワークの運用経験の蓄積と、基礎データの取得を目的として実施した。事前実験の結果から、単純に無線ネットワークインターフェースを複数装備した機材を無線ノードとして利用した場合には、マルチホップ通信において著しい性能劣化がみられることがわかっていた。その原因と考えられた異なるチャネル間の影響を軽減するため、指向性アンテナを用いて無線ノード間通信を実現した。その結果、マルチホップ通信での性能が向上することが確認された。

今後は、Robohoc ネットワークを現実のものとするための、自動ネットワーク設定技術、容易なネットワーク拡張技術、耐障害性技術などを実現していく。

2.2.3 Overlay GHC Prime Challenge

2.2.3.1 目的と背景

2.2.3.1.1 実験の目的

この実験は、IDEON ワーキンググループにて開発している Overlay GHC 言語のリファレンス実装による分散計算の耐故障性を検証すると同時に、同言語の WIDE プロジェクト内における普及を図ることを目的としていた。

ただし、実際には、実装が遅れたため、耐故障性の検証までには至らず、分散計算の基本機能の実証に留まった。

2.2.3.1.2 実験の概要

この実験では、参加者が任意のグループを形成し、それぞれが Overlay GHC により記述された「エラトステネスの篩」のアルゴリズムの実装によって分散計算を行い、素数列を求めた。実験終了時に最大の素数を求めたグループが勝利する[5]という競争の形式をとり、NAIST の学生グループが優勝した。

実験には、wija（`http://www.media-art-online.org/wija/`）のプラグインとして斉藤が開発している Overlay GHC のリファレンス実装を用いた。

2.2.3.1.3 実験の背景

Overlay GHC（`http://www.media-art-online.org/ghc/`）は、IDEON ワーキンググループにて開発している、P2P オーバレイネットワークプログラミング言語である。この言語は、1985 年に現早稲田大学理工学部教授である上田和紀博士により開発された GHC（Guarded Horn Clauses; ガード付きホーン節）に基づいている。

この言語は、地球規模 OS の外殻（shell）言語として設計を進めており、GHC から受け継いだ大規模な並行性への対応能力に加え、特に分散処理への適応性を高くすることを目的としている。ネットワーク上に分散したリソースの組み合わせのための糊（glue）言語として利用できることを想定しており、共有変数によって既存のアプリケーションを組み合わせることで、新しい機能を実現する。

Overlay GHC では、分散システムをプロセスのネットワークから成るひとつのプログラムとして記述できるため、オーバレイネットワークの鳥瞰的プログラミングが可能となる。

現在、マルチコア時代、高速ネットワーク時代に対応する強力な並行プログラミング言語が求められており、様々な言語やプログラミングパラダイムが提案されている。今回の開発は proof-of-concept 的な意味合いが強く、必ずしも性能を追求していないが、将来的には、例えば、現在、同様な分野で注目されている Erlang を、記述しやすさと性能の両面において凌駕することが期待できる。

今回の実験は、Overlay GHC の初期試験的意味合いを持つ。

2.2.3.2 理論

2.2.3.2.1 エラトステネスの篩

今回の実験では、図 2.11 に示したような「エラトステネスの篩」のアルゴリズムを用いた。

このアルゴリズムでは、自然数を生成する Generator プロセスと、それを素数で篩にかける Sifter プロセスを最初に生成し、Sifter プロセスが動的に

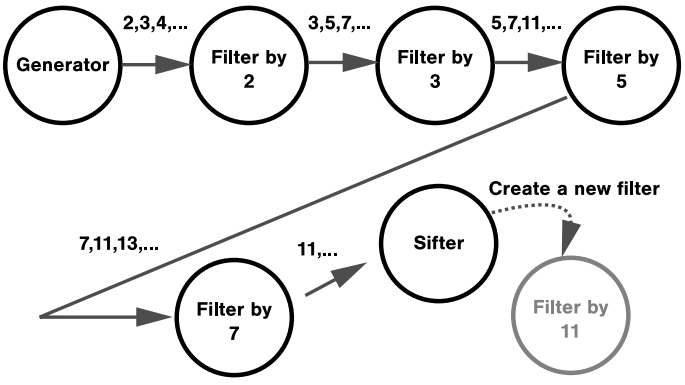

図 2.11.　エラトステネスの篩

5　優勝グループと第 2 位のグループには、斉藤からのささやかな贈り物として Billy's Bootcamp Elite — Mission 2: Maximum Power と Billy's Bootcamp Elite — Fat Blasting Cardio の DVD をそれぞれ差し上げた。

Filter プロセスを生成することで、素数によるフィルタのチェインを形成していくという計算方式をとる。Sifter は、自然数を受け取ると、それは素数であるので出力し、かつその素数で割り切れない数しか通さない Filter プロセスを新たに生成し、自らの手前に挿入する。それぞれのプロセスはデータ駆動で動作し、パイプライン結合しているため、別々のプロセッサに割り当てて並列実行させることができる。

2.2.3.2.2 GHC による並行計算

図 2.12 は、上田博士により書かれた GHC プログラム [171] と同等なもので、今回の実験でも基本的に同じプログラムを用いた（多少の最適化を行っている）。

このプログラムでは、計算の開始時に、求めたい素数が取り得る最大の値（図では 100）を指定する。このことにより、Generator プロセスが過度に先行して計算を進めることを防ぎ、全体を制御・計測しやすくしている。

Overlay GHC では、任意のグループの中でプロセスを均等に負荷分散させた分散計算を行わせることができる。また、特定のプロセスを特定のプロセッサで実行させることができるが、今回の実験の場合、出力プロセスは分散計算を開始させたプロセッサ上で実行することにした。

2.2.3.3 実験手続き

2.2.3.3.1 準備

実験参加者は、以下のソフトウェアと前提項目を用意する必要があった。

ソフトウェア：

- *wija* version 0.13 + Overlay GHC プラグイン
- Prime Challenge プラグイン

前提：

- Java 2 Standard Edition 1.5+[6]
- Jabber ID
  - media-art-online.org、jabber.org、jabber.jp の ID は動作確認済み
  - Gmail/Google Talk ID も使用可能
  - Openfire サーバを利用している場合は不可[7]

合宿 2 日目 10:30 からの BoF によりインストールをサポートした。

2.2.3.3.2 システム

実験では、図 2.13 で示したような XMPP（Extensible Messaging and Presence Protocol）の枠組みを用い、XMPP の in-band 通信をピア間のランデヴーに、out-of-band ストリーミングを実際の分散計算上の通信に用いた。

実験のシステム構成を図 2.14 に示す。この構成では、Prime Challenge プラグインが Overlay GHC のコンソール出力に仕掛けたフックを用い、参加ピアに

```
:-main(100).
main(Max) :- primes(Max, Ps), output(Ps, Os), io:outstream(Os).

output([P|Ps], Os) :- true | Os = [write(P), nl|Os1], output(Ps, Os1).
output([],     Os) :- true | Os = [].

primes(Max, Ps) :- true | gen(2, Max, Ns), sift(Ns, Ps).

gen(N, Max, Ns) :- N =< Max | Ns = [N|Ns1], N1 := N + 1, gen(N1, Max, Ns1).
gen(N, Max, Ns) :- N >  Max | Ns = [].

sift([P|Xs], Zs) :- true | Zs = [P|Zs1], filter(P, Xs, Ys), sift(Ys, Zs1).
sift([],     Zs) :- true | Zs = [].

filter(P, [X|Xs], Ys) :- X mod P =:= 0 | filter(P, Xs, Ys).
filter(P, [X|Xs], Ys) :- X mod P =\= 0 | Ys = [X|Ys1], filter(P, Xs, Ys1).
filter(P, [],     Ys) :- true          | Ys = [].
```

図 2.12. エラトステネスの篩の GHC による実装

6 Windows プラットフォーム以外では J2SE 1.4.2+ で動作する。

7 Openfire サーバでは、XMPP と XEP（拡張プロポーザル）で明示的に定義されている以外のストリーミングプロファイルを通さないため。

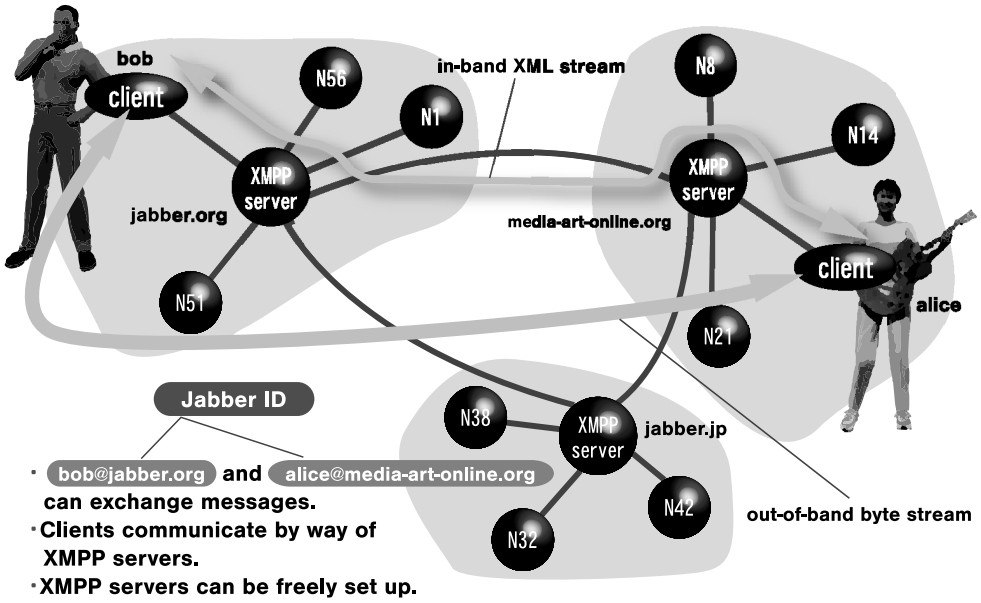

図 2.13.　Jabber/XMPP の概要

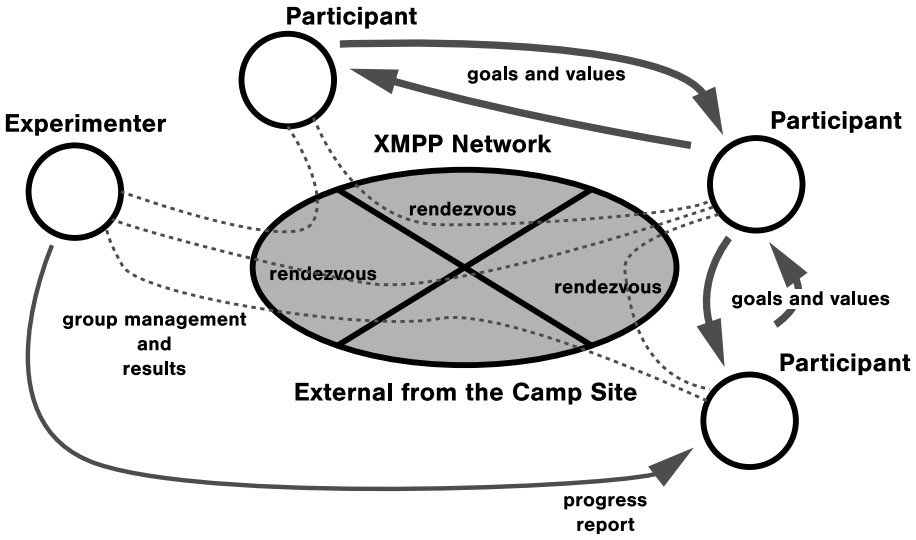

図 2.14.　実験のシステム構成

より生成された素数は、XMPP の in-band 通信を通して Experimenter ノードに送られる。Experimenter は素数が生成された時刻を記録するとともに、web サーバ[8]として、実験の進捗を示す Web ページを提供する。

Prime Challenge プラグインは、この Web サーバへのアクセスを提供すると同時に、図 2.15 に示したような設定ウィンドウを通して、ユーザがグループを作成したり、グループへの参加・脱退ができるようにユーザインタフェースを提供する。分散計算の開始も、このプラグインから指示する。

2.2.3.4 結果

2.2.3.4.1 参加者とグループ

実験には 27 名が参加した。この値は Experimenter ノードに記録された参加者数であり、合宿アンケート

8　この表現は概念的なものであり、実際には Experimenter は out-of-band ストリーミングによる HTML ファイルの提供を行い、各参加ピアの *wija* がローカル環境で web サーバとして動作する。参加者の Web ブラウザは、*wija* に内蔵されたローカルな web サーバにアクセスする。

図 2.15. Prime Challenge プラグインの環境設定画面

表 2.3. 参加者の構成（プラットフォーム別）

プラットフォーム			人数	
Linux	(i386)	2.6.20-16-generic	1 名 (4%)	2 名 (8%)
	(i386)	2.6.18-4-686	1 名 (4%)	
Mac OS X	(i386)	10.4.10	11 名 (40%)	12 名 (44%)
	(ppc)	10.4.10	1 名 (4%)	
Windows	XP (x86)	5.1	12 名 (44%)	13 名 (48%)
	Vista (x86)	6.0	1 名 (4%)	
合計			27 名	(100%)

表 2.4. 参加者の構成（ドメイン別）

ドメイン	人数
media-art-online.org	11 名 (40%)
gmail.com	8 名 (30%)
jabber.org	6 名 (22%)
jabber.jp	2 名 (8%)
合計	27 名 (100%)

によれば、17 名が実験に参加した自覚を持ち、22 名が参加する意思はあったものの、何らかの原因により参加できなかったという。

実際の参加者の構成をプラットフォーム別に分類したものを表 2.3 に、XMPP サーバのドメイン別に分類したものを表 2.4 に、利用された Java VM 別に分類したものを表 2.5 に、それぞれ示した。

プラットフォーム別では、少なくとも（この種の実験に興味を示すような）WIDE メンバの間でのポピュラーな環境が Mac OS X (i386) と Windows XP であることが分かる。今後の開発の参考としたい。

ドメイン別では、この実験のために新たに Jabber ID を取得したのでない場合（ある場合は media-art-online.org を利用する例が多い）、Gmail/Google Talk がポピュラーに用いられていることが分かった。これも今後の開発の参考とし、Gmail/Google Talk のサーバを経由するテストを強化していきたい。また、合宿アンケートの回答からは、jabber.org の利用者が *wija* でログインできない問題が 1 件、発生していたことが分かった。これは、おそらく SSL の利用を指定していたためではないかと推測される。jabber.org は検証可能な証明書を用いていないと考えられ[9]、現在の *wija* の実装は、そのような例に対応していないからである。この問題は *wija* の将来のバージョンでは解決していく予定である。

Java VM 別では、Mac OS X 以外では Sun

9 No trusted certificate found 例外が発生する。

表 2.5.　参加者の構成（Java VM 別）

Java VM		人数	
Java HotSpot (TM) Client VM (Sun Microsystems Inc.)	1.6.0_02-b06	8 例　(30%)	15 例　(56%)
	1.6.0_01-b06	3 例　(11%)	
	1.6.0-b105	1 例　(4%)	
	1.5.0_11-b03	3 例　(11%)	
Java HotSpot (TM) Client VM (“Apple Computer, Inc.”)	1.5.0_07-87	12 例　(44%)	
合計		27 例　(100%)	

表 2.6.　参加グループ（終了時）

グループ名	人数
パラダイス和尚	4 名
CAP とそのしもべ	2 名
DS1	2 名
JAIST と愉快な仲間達	2 名
JAIST の愉快な仲間たち	2 名
NAIST な人たち	2 名
New	2 名
cap-test	2 名
hoge	2 名
大王先生とそのしもべたち	2 名
ayumin 神さまと信者たち	1 名
おっぱっぴー	1 名
んがも探検隊	1 名
test	0 名

Microsystems Inc. の VM が用いられているが、GCJ を利用しようとして失敗した事例が 1 件、確認されている。この事例での失敗は、Prime Challenge プラグインのみが *wija* に組み込まれなかったというもので、おそらくはコンパイルされたバイトコードのバージョンが Prime Challenge プラグインのみで異なっていたことが原因ではなかったかと推測している（*wija* では GCJ は未サポートだが、今後サポートしていく予定である）。

また、実験終了時の参加グループと人数を表 2.6 に示した。1〜4 名の小さなグループでの参加だったことが分かる（実験の仕様上、1 名のグループでは分散計算が開始できないことにしていたため、実際に計算を行えたグループはこれより少ない）。

2.2.3.4.2 計算負荷

合宿アンケートによれば、実験に参加した自覚を持つ 17 名のうち、10 名が計算の負荷を感じず、7 名が多少の負荷を感じたという。負荷を感じなかった程度の計算しか行われなかったことが失敗ではなかったか、との指摘もあった。

負荷の計測を行っていた参加者の報告としては、PentiumM 1.3 GHz で、CPU への平均的な負荷は 20〜40% だったという。

2.2.3.4.3 計算結果

分散計算の競争の結果は図 2.16 に示した通りである。

第 1 位「NAIST な人たち」は NAIST の学生グ

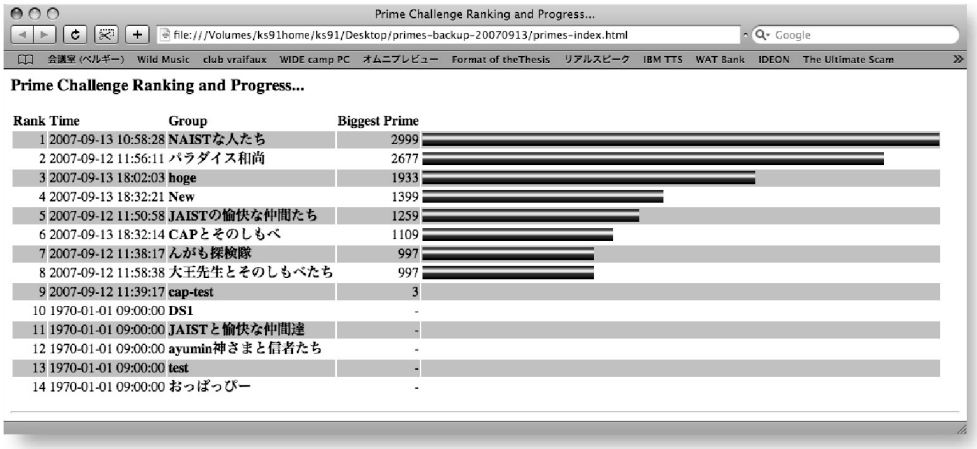

図 2.16.　順位および進捗ブラウズ画面（終了時）

ループであり、第 2 位「hoge」は、斉藤の記憶が正しければ東大の学生グループだった。「パラダイス和尚」グループは実験担当者本人が参加していたため、最終的な順位にはカウントしていない。

2.2.3.5 考察

2.2.3.5.1 素数生成イベントのプロットから

日付・時刻と生成された素数の値をプロットしたグラフを図 2.17 に示した。

このグラフから、今回の実験における分散計算は、決して高速ではないものの、数十分を超える長さの計算セッションは無かったことが分かる。今回は、Overlay GHC のリファレンス実装のメモリ効率上の制限から、高々 3,000 程度の値の素数しか求めることができなかったため、仮に耐故障性の実装ができていたとしても、その実証のためにはやや不十分な環境だっただろうことが伺える。

一部、ほかと比較して低速な計算セッションが見られる（素数の分布に傾きが見られるもの）。実験時の実装では、コネクションの確立部分に不具合があったこと、また、単一プロセッサ内でのマルチスレッド処理が必要以上に低速であったことを確認しており、その影響があったと考えられる。

2.2.3.5.2 分散計算の詳細とオーバヘッドについて

第 1 位となったグループの計算セッションについて、時刻と生成された素数の値をプロットしたグラフを図 2.18 に示した。

このグラフから、速度自体の問題はあるものの、ほぼ一定の割合で素数を生成することができていたことが分かる。セッションの立ち上がり時に低速なのはコネクションを確立しているためであり、最後に高速になっているのは Filter プロセスが続々と終了して計算負荷が低下しているためと推測できる。この現象自体はスタンドアロンで計算している際にも同様に観測されており、分散計算によるオーバヘッド以外に、分散化が計算全体に与える影響は無いと考えられる。

分散計算は非常に低速であり、ログからは、2,999 までの素数を求めるのに 604.259 秒（約 10 分 4 秒）かかっていたことが分かる。この低速さは、主に以下に起因すると考えられる。

1. 共有変数の値のアップデートが、必ずプロセスの生成元のプロセッサを経由して行われるような原始的な設計となっている。このため、素数の生成の例では Sifter プロセスを動かすプロセッサに通信が集中することに加え、オーバレイホップ数が増大する。
2. 共有変数の値を GHC ソースで記述したものを XML で符号化して転送している。このため、変

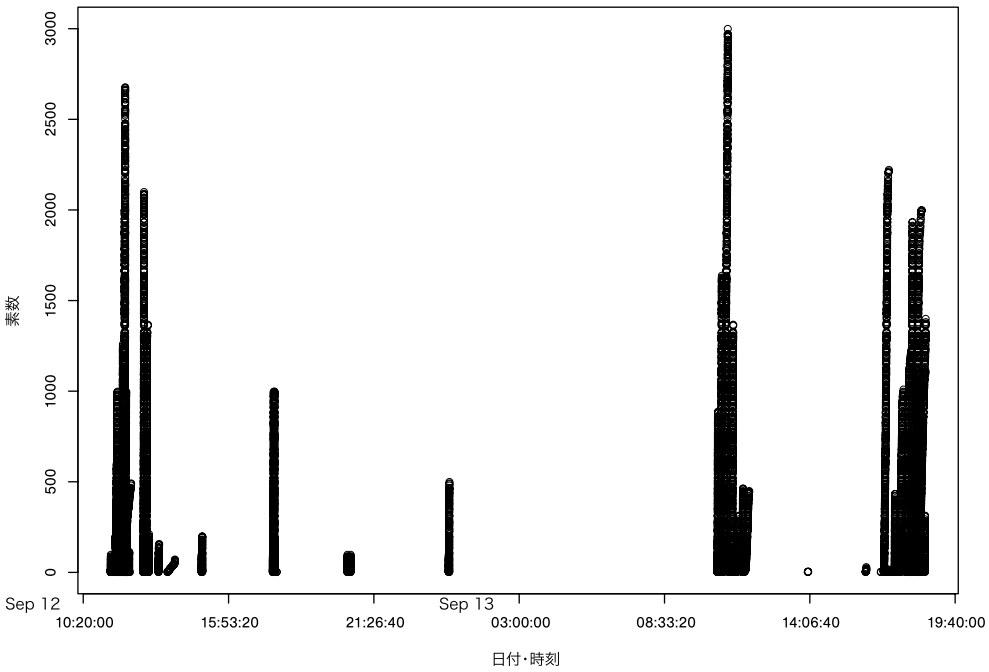

図 2.17. 日付・時刻と生成された素数

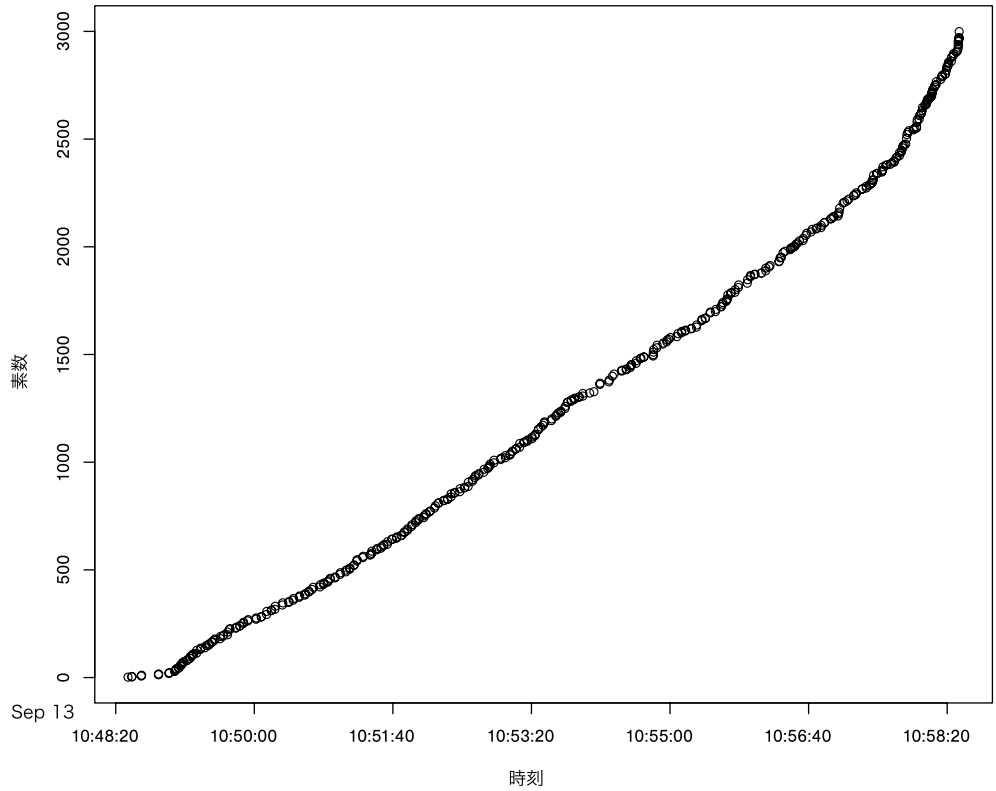

図 2.18.　時刻と生成された素数（第 1 位グループ）

数の値が届く度に字句解析と構文解析が二重に発生する。

3. 以上から共有変数の値の到着が遅延するが、その間、多数のプロセスがビジーウェイトしているため、届いた値の処理に至るまでに更に遅延が発生する。

これらは遅くとも次回の実験までには改善されている予定である。

プロセッサ数が増えることの効果は、今回の実験では、参加ピア数が 2〜4 と変化に乏しかったため、分析を割愛した。

2.2.3.6 結論

2.2.3.6.1 今回の実験で得られたこと

今回の実験は、特に耐故障性の検証という点において当初の目的を果たさなかったが、初めて Overlay GHC を多くの技術者に利用してもらうことで得られたフィードバックは大きかった。

この報告書を執筆している現在、処理系は大幅に改善され、スタンドアロン環境での性能を 2〜5 倍に向上させている他、実験時に得られたフィードバックに基づいて、デバッグ環境、エディタ環境などの改善が行われている。分散計算時のオーバヘッドも次回合宿までには大幅に削減されている予定である。

2.2.3.6.2 次回の実験への指針

今回の実験では、耐故障性の実装が遅れたことは最大の反省点だが、それ以外にも、素数を計算するという実験題材が、地球規模 OS の外殻(shell)言語としての Overlay GHC の有用性を検証する上で適切であったか、という反省もある。

Overlay GHC の研究開発で今後取り組んでいくテーマには次があるが、次回の実験では、地球規模 OS のアプリケーションを記述するチャレンジといった、参加者がより興味をもって取り組める内容を盛り込むことで、参加者数の増大と、Overlay GHC 自体の有用性の検証を同時に図りたい。

理論：

- 分散実行のフェアネスの実現
- 耐故障性のある分散実行

仕様と実装：

- 外殻(shell)言語としての要求の実現
- 実行効率性（proof-of-concept として成立する程度の）

応用と検証：

- 地球規模 OS のアプリケーションを記述するチャレンジ
  - ヒッチハイク問題
  - ランチ難民問題

2.2.4 iTunes on a P2P Network

2.2.4.1 実験内容

WIDE 合宿ネットワークと JAIST ネットワークの間で仮想ネットワークを構築し、遠隔地にいながらにして、JAIST に設置された iTunes サーバから視聴を行う実験を行った。この仮想ネットワークは、分散ハッシュネットワーク技術を用いて構築したものである。分散ハッシュテーブルとは P2P ネットワーク技術の一種であり、すなわち、本実験では完全に分散化された仮想ネットワーク環境の構築の実現を目指す。

2.2.4.2 目的と意義

2.2.4.1 項のの実験内容で述べたように、本実験は、分散化された仮想ネットワークの構築が第 1 の目的となる。従来、仮想ネットワークといえば OpenVPN などに代表されるような、Point-to-Point 型かクライアントサーバ型のものしか存在しなかった。しかしながら、インターネットはそもそも、分散化された耐故障生の高いネットワークであるのにもかかわらず、OpenVPN のような中央集権型のネットワークに依存した利用形態では、インターネットの持つ最大の特徴を殺してしまうことになる。そこで、本実験では、仮想的なネットワークにおいても、インターネットと同質なネットワークを構築可能とすることを目指す。

2.2.4.3 実際の詳細

本実験では実験参加者に対して、P2P 型の仮想ネットワーク構築ソフトウェアである P2P@i を提供し、実験参加者は、本ソフトウェアを用いて仮想ネットワークに接続する。また、あらかじめ JAIST には、仮想ネットワークに接続された iTunes サーバが設置されており、実験参加者は P2P@i と iTunes を起動することで、JAIST に存在する楽曲を遠隔地にいながらにして視聴することが可能となる。

iTunes とは、Apple 社が開発したマルチメディアプレイヤーである。しかしながら、iTunes はネットワーク機能も備えており、本機能を利用すると、同一セグメント内にあるコンピュータ同士で楽曲を共有することが可能となる。すなわち、本来ならば同一セグメント内にあるコンピュータ同士でしか楽曲を聴けないため、JAIST に存在する楽曲を合宿地で視聴することは不可能なはずであるが、本実験ではそれを可能にした。これはもちろん、従来のトンネリング技術でも可能であったことだが、ネットワークを構築して実現したことの意義は大きい。

2.2.4.4 実験結果

実験結果としては、利用者側の意見として、十分動作したという意見があったので本実験は成功したと言える。しかしながら、世界的にデスクトップアプリケーションは死につつあるという情勢の中、得体の知れないソフトウェアをインストールしたくないという心理が働いたのか分からないが、実験参加者はごく少数にとどまった。

本結果から、実験を行う上で最も留意すべき事は、いかにして参加者の注目を集めるかということと、いかにして参加障壁を下げるかであると痛感した。特に、デスクトップアプリケーションをインストールするタイプの実験の場合、その参加率は著しく低下すると考えた方がよいだろう。実験を行うならば、ネットワーク利用者が意識せずに参加するような、ネットワークインフラに直結したものか、それが出来なければウェブに対応した物が望ましいのではないかと感じた。

2.2.5 複数セッションに着目したボットや P2P ファイル交換ソフトの検知手法の実験

2.2.5.1 目的

本実験は複数セッションに着目したボットや P2P ファイル交換ソフト検知手法の有効性を示すために、合宿ネットワークを監視することで誤検知の発生率を調査した。従来の IDS などでは 1 つのパケットやセッションから情報を得ることであるソフトウェアの動作を検知していた。しかし、ボットや P2P ファイル交換ソフトは検知を回避する方法を導入しており、検知が難しくなっている。これらのソフトウェアを確実に検知するため、複数のセッションに着目することで高い精度をもつ検知手法を提案・実装した。検知手法の有効性を示すための指標として、検知精度があげられる。検知精度の高さを示すためには、

誤検知（False Positive）や見逃し（False Negative）の少なさを示さなければならない。False Positive はネットワークの環境やトラフィックの傾向が変化することによって、発生率が変動してしまう。そのため、多様なネットワークの 1 つとして WIDE 合宿におけるネットワークを監視し、本実装による False Positive 発生率の少なさを検証する。

2.2.5.2 手法概要

本手法はセッション間の相関関係を検知条件として利用している。セッションとはプロトコル毎に分けたホスト間の通信を指す。そして、セッション間の相関関係として「セッションの出現規則」と「セッションに含まれる情報の相互利用」の 2 つが挙げられる。セッションの出現規則とは、ある特徴を持つセッションの発生順序をルール化したものである。数種類のセッションが発生する順序や同種類のセッションの繰り返しを認識できる。セッションに含まれる情報の相互利用とは、あるセッションに含まれる情報を他のセッションの発見に利用することである。例えば DNS の応答レコードに含まれる A レコードを記憶し、同じホストから発生する他のセッションの宛先 IP アドレスを見ることによって、DNS セッションとの関係を認識できる。このようなセッションの相関関係による条件と、従来の IDS に実装されているパターンマッチによる条件を組み合わせることによって、より高精度な検知が可能となる。

2.2.5.3 実験条件

WIDE 合宿で運用されているネットワークのトラフィックを監視し、本実装による検知結果を検証した。ネットワークのトラフィック監視ではルータに接続されているスイッチのポートをミラーリングし、外部と送受信される全てのトラフィックを収集した。本実装の稼働には専用のホストを用意し、ミラーリングしたトラフィックを送信した。検知された結果は同ホストに蓄積した。実験は合宿開始から終了までの 4 日間、継続して実施した。

検知にはボット検知用のルールと P2P ファイル交換ソフト検知用のルールの 2 種類を用意した。ボット検知用のルールでは DNS の問い合わせと IRC サーバへの接続を検知条件としている。まず、DNS による Spyware、ボットで使用されるドメインの問い合わせを検知する。検知対象となるドメインは BleedingEdge Snort で公開されているルールを参考に、3322.org、cjb.net、maxonline.com、vendaregroup.com、seznam.cz、botstealer、dawnsoul のいずれかの文字列を含むドメイン名と設定した。これらを検知した場合、当該問い合わせに対する応答の A レコードを、保持する。120 秒以内に同ホストから IRC サーバへの接続として、記録していた A レコードへの接続があった場合、ボットの活動として検知する。P2P ファイル交換ソフトウェアの検知ルールとして、winnyp の検知ルールを用意した。winnyp は winny と異なりプロトコルが公開されておらず、検知手法が一般化していない。Winnyp のトラフィックを調査したところ、1) 接続確立後の通信手順がほぼ同一である 2) 複数のノードに接続し情報を送受信する、の 2 点が確認できた。また、Winnyp が発生させるセッションの大部分が TCP セッション開始直後の 5 パケットが同じサイズのセグメントを持つと分かった。以上の点を踏まえ、まず TCP の接続確立後から、5 パケット連続して同じサイズのパケットが送受信されるセッションを検知する。これらのセッションが 5 回検知された場合に、winnyp の活動として検知する。

2.2.5.4 実験結果

実験した時期が本実装の完成直後であったためにテストが不十分であり、期待された結果を得ることができなかった。しかし、動作検証のために必要なデータを多く収集でき、多くの実装上の問題点を修正できた。実装上の問題は、主にトラフィックの解析部分において多く発生しており、特に TCP の解析における不備が多かった。

ボットの検知ルールではボットなどが頻繁に利用するドメイン名の問い合わせは多数発見できた。ただしこれらを調査したところ、ほとんどがメールサーバから発生した問い合わせであった。これは SMTP によって接続した際、HELO コマンドによって渡されたドメイン名を正引きした際に発生したものと推測される。また、エンドユーザが利用しているホストからの問い合わせも発見したが、DNS の問い合わせ結果に含まれる A レコードに対して IRC で接続したセッションは発見されなかった。この事から、DNS のドメイン名問い合わせを監視する場合と比較して、本手法ではボットの振る舞いであるかを正確

に判断できることを示した。

winnyp の検知において多くの検知結果が出力されたが、検知結果を目視で確認したところ結果は全て実装上の不備による誤検知であった。winnyp の検知には特に TCP のセグメントサイズやセグメントを含むパケットの出現回数が条件となっていたため、誤検知を多発していた。そのため実装を修正し、提案手法を正しく実行できるように変更した。