第 20 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

1999 春の合宿

本章では 1999 年 3 月 24 日から 27 日に静岡県伊東市で開催された合宿において構築したエンドユーザサービスまでを含む実験ネットワークについての議論を進める。

1.1 実験概要

本ネットワークでは想定するモデルとして、エンドユーザまで IPv6 を前提としたネットワークを設計し、接続としては IPv4 ユーザも想定する。対外線を流れる IP パケットは IPv4 でのアプリケーションを前提とする実験と、DNS を除いて全て IPv6 とする。このため、合宿内ネットワークでの IPv4 パケットは、合宿外へ流れる手前で IPv6 へ変換される必要がある。

エンドユーザが利用する合宿内のネットワークは全て一つのセグメントとなり、IPv6 および IPv4 それぞれ一つのネットワークがそのセグメント上で構成さる。また、階を隔てる接続は、運用実験として富士通研究所によって VDSL を利用し、その性能評価を行う。

対外接続は、2M ビット/秒の双方向衛星回線と 128K ビット/秒の専用線 2 本が用意される。合宿でのエンドユーザ用として、双方向衛星回線と専用線 1 本を利用し、専用線の 1 本は School of Internet (以下 SOI) での中継実験専用とする。合宿での対外トラフィックでは衛星回線と専用線の特性を活かすため、ネットワークの上で利用されるアプリケーションによってそのどちらを通るかを選択する首振りの実験を行う。

1.2 ネットワーク構成

1.2.1 物理接続

目的

ここでは、物理的構成を設計の目的を述べる。

- 10/100 イーサネットの透過性の実現

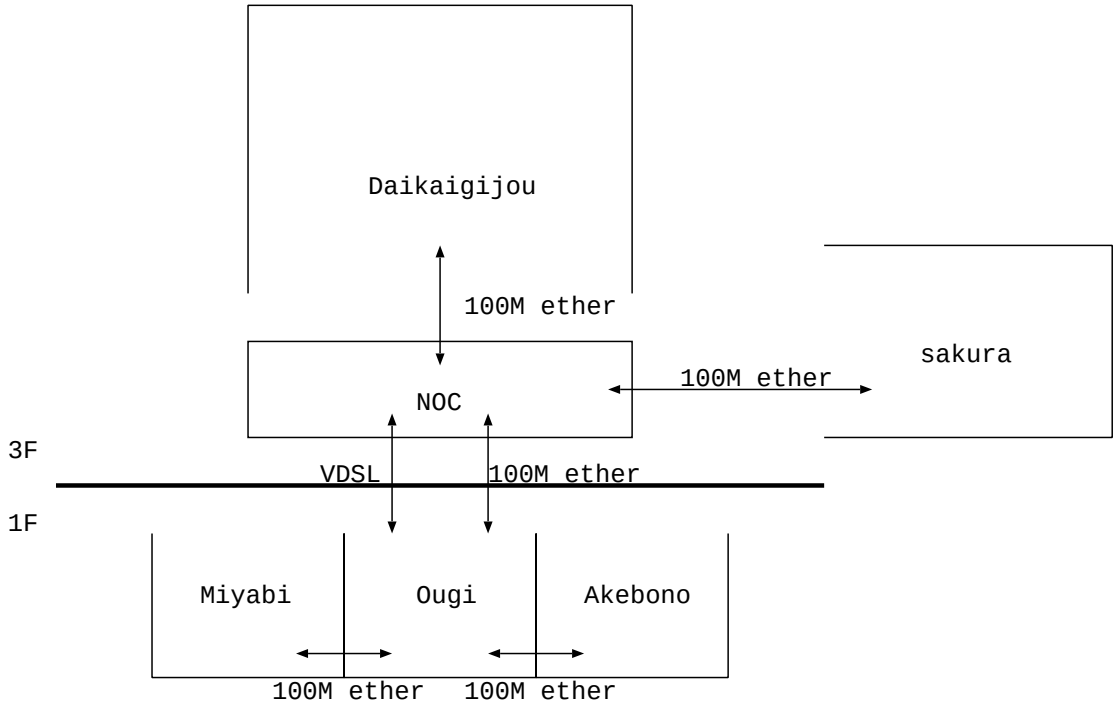

図 1.1: 施設ネットワーク

ユーザが利用するデータリンク層プロトコルには、10M イーサネットと 100M イーサネットが予想される。両方をサポートし、ユーザが気にする必要の無いネットワーク構築が必要である。

- サービスセットアップのコスト削減

  サービスのセットアップは各セグメントにおいて行わなくてはならない。無用な設定コストを省くため、セグメント数は最小にとどめる。

- 耐故障性の向上

  ネットワークの耐故障性を向上させるため、接続距離が比較的長くなる階を隔てる接続は二本の物理接続を用いて異種のデータリンク層プロトコルで構成する。

トポロジ

今回のワークショップにおいて構築したネットワークの物理接続を図 1.1に示す。

大会議場と扇の間との接続は、VDSL と 100M イーサネットという二つのデータリンク層プロトコルを二本の物理線で提供する。それ以外の各会議室間のネットワークは、100M イーサネットを用いて接続する。

評価

ユーザに提供されるネットワークの接続には、10/100M イーサネット自動認識をサポートしたスイッチを利用した。これにより、ユーザが自分の接続するポートを気にせず通信を行う事が可能となった。

ユーザの利用するネットワークは一セグメントで実現したため、どの部屋においても同一のセグメントに接続される。これにより、最小の管理コストで各部屋にサービスを提供する事を実現した。

200 人以上のサービスを一つのセグメントで提供したため、サービスのスケイラビリティに問題が生じた。具体的には、DHCP アドレスが割り振られないという問題が生じた。

1.2.2　VDSL

実験の目的

富士通研究所 Comet チームは Comet によるネットワーク負荷発生および性能測定機能を利用した VDSL モデムの評価実験を行った。

実験の概要

1. VDSL の性能 (距離、遅延、帯域等) を調べる。
2. Comet でネットワークに負荷をかけつつ、負荷パケット落ちを精密に観測することで、ネットワークの特性を調べる。
3. 一般の使用者が測定負荷に影響を受けるのか、測定が一般の使用者のトラフィックに影響を受けるのかを調べる。

実験環境

3F 大会議室と 1F BOF 部屋とを、Switch、LR ルータ、VDSL モデムを経由して接続する。それぞれ Splitter 箱までの設備機器が各会議室にあり、その間を２芯銅線で接続する。この環境で 3F 大会議室に置いた Comet より下り線に DV パケットを流すことで定量的な負荷をかけつつ、1F 利用者に対して外部接続のサービスを提供した。

なお VDSL モデム (www.vdsl.org) は以下の特徴をもつ。

- ２芯銅線 RJ-45 で接続する。

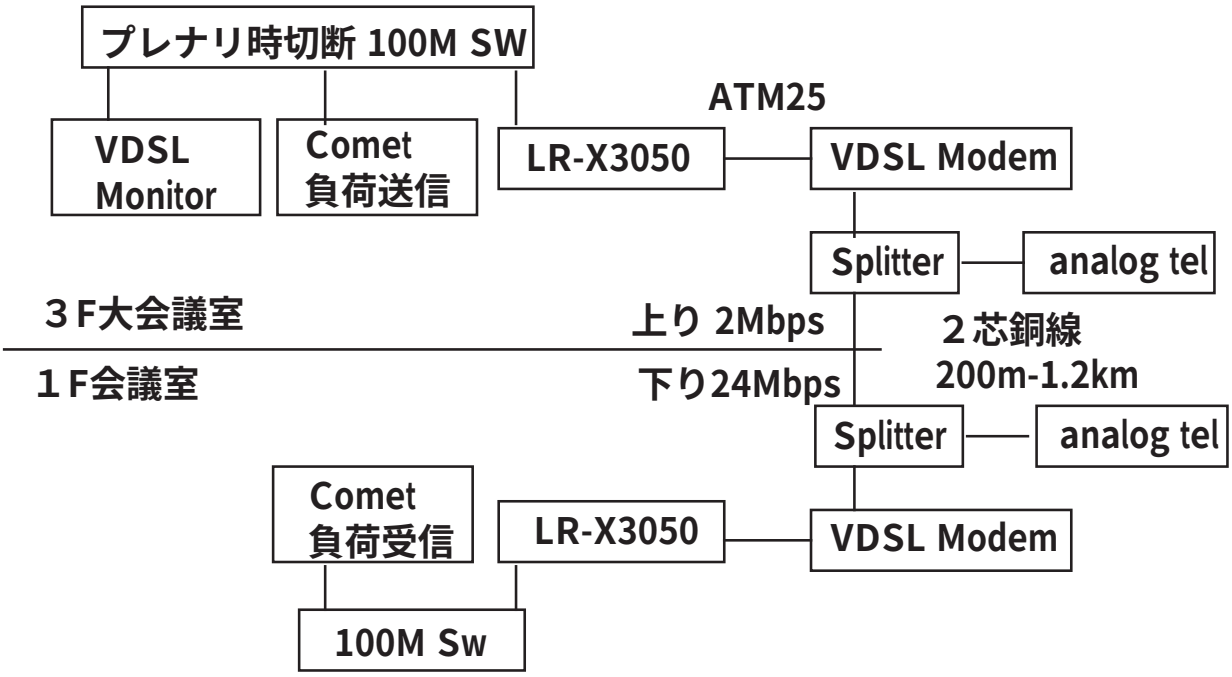

図 1.2: VDSL 実験環境

- 非対称性 (公称 下り 24 Mbps、上り 2 Mbps)
- VDSL のインタフェース と 25Mbps ATM * 2
- 局用モデム (ONU) で VDSL の帯域の指定、同期を取る。

LR-X3050:www.fujitsu.co.jp/hypertext/Products/telcom/lan/lrx3050.html
VDSL:www.orckit.com/orckit_product.html

結果

1. VDSL 間距離 800 m まで通信可能

   距離の延長は 200 m 単位で行った。1km に延ばすと、モデム間で同期が取れなくなり、不通になった。但し、今回距離延長用のケーブルは巻取り状態のため、敷設した場合に比べて伝送条件は悪い。

2. DV (UDP) は 25M 下り線で 19M 程度まで通過

   LR-X3050 の SNMP で ATM と Ethernet インタフェースのフレーム数を採取して合宿期間中、MRTG にてインタフェースの利用率を計測した。19Mbps のレート (単方向 DV/IP の負荷 39Mbps) でパケットが落ちはじめることを確認した。落ちはじ

める直前の ping の RTT が最大 800 〜 1000ms とばらけたが、telnet の応答性に問題は認められなかった。Comet により定量性のある負荷を発生させることで、安定した評価実験を行なえた。

3. VDSL の性能評価

DV が流れていない環境で、Comet (BSD/OS 3.1 + 100M Ether) から netperf を行った。netperf 添付のストリーム用スクリプトによるスループットを得た。

TCP: Send, Rcv Socket Size=57344 byte, Message size=4096 byte

UDP: Send Socket Size=57344 byte, Message size=1472 byte

条件		スループット (Mbps)
LR 直結	下り TCP	20.08
VDSL 200m	下り TCP	13.82
VDSL 400m	下り TCP	13.82
VDSL 200m	下り UDP	95.71
VDSL 200m	上り TCP	1.75
VDSL 200m	上り UDP	95.69

UDP はカーネルで廃棄しているので性能的に意味はない。TCP のスループットが VDSL 接続で 13.8 と低下する理由は VDSL の非対称性にあると想像されるが、未解析である。

遅延について
ping の RTT は 20-30 msec、 DV を流すと 20-1000,2000,3000 msec の揺れが観測された。VDSL を外して LR 直結でも同様な現象を確認した。

4. VDSL 性能評価

19Mbps DV が流れる Comet の上から、netperf を行った。Comet は DV パケットを処理しつつ、同時に netperf のホストからのパケットも処理する。netperf 添付のストリーム用スクリプトを原則 2 回行って出た最高値 (Mbps) と、そのときの受信画像の状態を示す。

25 Mbps 下り線で UDP の netperf を行うと、netperf の出すパケットが優位なのか、DV データがあってもなくても大体 21Mbps のスループットが出た。次に、2Mbps 上り線を使う場合、DV データなしでは TCP, UDP ともに 1.7Mbps - 1.8Mbps くらいのスループットを得た。DV を送信しつつ、netperf が UDP を使用する場合では、DV のストリームと回線を奪いあい、DV の画面はノイズが載っていたり、全く画像

	DV 無	19Mbps DV 有	受信画像の状態
下り TCP	13.8	2.2	そこそこ安定
下り UDP	21	21	ノイズ有り、画像出ない
上り TCP	1.7	–	–
上り UDP	1.8	–	–

が届かなかった。Netperf が TCP を使っていた場合、DV の UDP パケットで TCP パケットは殆んど流れないようで、DV の画像はそこそこ安定していた。

更に 受信側 Comet で DV パケットのシーケンス番号を調べて、VDSL 上で落ちパケット数を定量的に測定した。パケット落ちがごく稀に発生する (1 分に落ちるパケット数が 0-2) 程度の DV 負荷をかけて、その状態で合宿ネットワークの一般ユーザーからのトラフィックも VDSL 上に混在させて落ちパケット数がどうなるかを観測した。結果は落ちパケット数が 10 〜 15 に増えた。

落ちパケット数が、それほど伸びない原因は

- １Ｆのユーザー数が意外に少なかった。
- telnet のような TCP トラフィックしか発生してない。

が考えられる。

まとめ

VDSL は、結構利用できる印象を得た。特に 2 芯の銅線 800 m の距離まで延長することができた。ホスト間の距離を必要とする場合は重要である。また TCP で下り線 14 Mbps、上り線 1.7Mbps 程度出せることを確認した。

DV (UDP) は 下り 25Mbps では、19M 程度まで通過させることができた。下り線に、激しいトラフィックがある状態でも、下り線 TCP で 2M 程度のスループットを得ることができた。合宿期間中、一般の利用者は通常通りに利用することができた。

1.3　IPv6 ネットワークの構成

ここでは、WIDE 合宿で運用された IPv6 ネットワークの概要とその運用結果を述べる。

1.3.1　概要

今回の合宿ネットワークは、前回の WIDE 合宿 (99 年 9 月) で運用されたネットワークを発展させたものとなった。前回の合宿では、一般参加者の合宿地外へのネットワークの接続性は、IPv4 と IPv6 の双方について用意された。よって、この実験環境は、既に IPv4 での対外接続をもった組織に IPv6 を導入するということ実現していた。そこで、今回の合宿では、この環境を発展させて、外部への接続性は IPv6 のみによることとした。

これにより、合宿ネットワークは、IPv6 のみの接続性が確保された組織のネットワークを実現する。そして、このネットワークを利用して、トランスレータによる IPv4 ネットワークとの相互接続の検証や、IPv6 ネットワークの運用上の問題点の調査、OS やアプリケーションの IPv6 実装の相互接続実験をおこなった。

1.3.2　構成

今回の合宿ネットワークは、SFC と合宿会場の 2 か所の NOC と両者間の 128kbps 専用線と双方向 2Mbps の衛星回線で構成されている。また、FreeBSD 2.2.8 に IPv6 スタックである KAME を導入した PC を中心とした構成になっている。その構成を図 1.3に示す。

NOC

- SFC-NOC

  SFC-NOC では、合宿ネットワークを対外ネットワークへの接続と、合宿会場へのルーティング、IPv6 → IPv4 トランスレータ faith、IPv6 DNS サーバ NEWBIE の運用を行った。

- 合宿会場-NOC

  合宿会場内は、セグメント #3,#4,#5 で構成されており、セグメント #5,#4 においては、プライベート IPv4 アドレスを割り当てた。ルーティングは、IPv6 的には、pc2 と pc4 が、IPv4 的には、pc4 が行い、、IPv4 → IPv6 トランスレータや SOCKS64、WWW、DNS、DHCP が運用された。

  セグメント #5 は、一般参加者は、PC を接続するために用意され、DHCP によるプライベート IPv4 アドレスと RA によるグローバル IPv6 アドレスが提供された。

上位層サービス

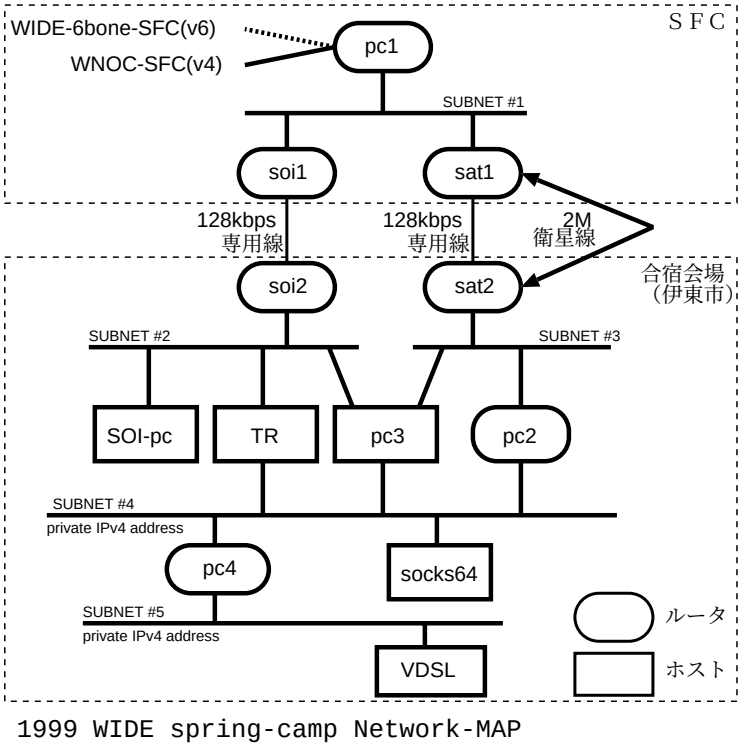

図 1.3: ネットワーク構成図

ここでは、運用を行ったサービスについて述べる。

- IPv6 → IPv4 トランスレータ

  pc1 上で、KAME に含まれる FAITH を運用した。FAITH は、アプリケーション型トランスレータで、IPv6 ホストからの IPv4 転写アドレス (IPv4 アドレスを組み込んだ IPv6 アドレス) への通信を一旦終端し、FAITH が IPv4 転写アドレスに含まれる IPv4 アドレスのホストに通信し、両者を中継することにより、IPv6 ホストと IPv4 ホスト間の通信を可能にする。この運用結果を、1.4.2節に示した。

- IPv4 → IPv6 トランスレータ

  本合宿では、IPv4 → IPv6 トランスレータとして、ヘッダ変更型トランスレータと SOCKS64 の 2 種類の運用を行った。

  ヘッダ変更型トランスレータは、pc3 上で運用した。このトランスレータは、IPv4 パケットのヘッダ部分を IPv6 ヘッダに変更することにより、IPv4 ホストから IPv6 ホストへの通信を可能にする。この運用結果を、1.4.1節に示す。

  また、socks64 上で、SOCKS5 プロトコルを利用した IPv4 → IPv6 トランスレータを運用した。この運用結果を、1.4.3節に示す。

- DNS

  本合宿では、newbie と bind8 をもちいて、IPv6 での問い合わせ用と IPv4 での問い合わせ用にわけて DNS を運用した。

  IPv6 クライアント用の DNS サーバとして、newbie を pc1 上で運用した。newbie は、クライアントから DNS サーバへの問い合わせを IPv4 および IPv6 で受け付けることができる。そして、NEWBIE の特徴に、FAITH フレンドリー機能がある。この機能は、NEWBIE が、IPv4 アドレスしか持たないホスト名から IPv6 アドレスへの問い合わせの要求に対し、IPv4 転写アドレス形式の IPv6 アドレスを返答するというものである。よって、IPv6 ホストは、シームレスに IPv6 → IPv4 トランスレータ FAITH を利用することができる。

  一方、IPv4 クライアント用の DNS サーバとして、bind8 を用いたスプリット・DNS を運用した。スプリット・DNS とは、本合宿のネットワークのように、IPv4 アドレス的に、グローバル・アドレス部分とプライベート・アドレス部分に分かれている場合、プライベートアドレスからの問い合わせ用の DNS とグローバルアドレスからの問い合わせ用の DNS の 2 つにわけて運用することである。

  これによって、同じホスト名に対して、異なる IP アドレスを返答することが可能となる。本合宿では、グローバル用 (プライマリ DNS サーバ) として、pc3 プライベート用として、pc4 で、それぞれ、bind8 の運用を行った。

- 公共サービス

  合宿期間中のお知らせの掲示や配布物の公開のために WWW と FTP の運用を行った。このサーバには、KAME 付属の IPv6 化された apache と wu-ftpd を利用した。

IP 層サービス

ここでは、運用を行った IP 層のサービスについて述べる。

- ルーティング

  合宿ネットワークにおけるのルーティングは、IPv4 については、静的なルーティングとし、IPv6 については、RIPng によるルーティングとした。なお、ルーティング・デーモンには、KAME に付属の route6d を利用した。

- アドレス

  IPv4 アドレスについては、定常的に接続されるホストに、IPv4 アドレスを割り当てた。それ以外のホストは、DHCP によって、IPv4 アドレスを割り当てた。また、IPv6 アドレスについては、RA によって IPv6 サブネットアドレスを通知した。DHCP によるアドレス割り当ては、セグメント#2,#4,#5 で行い、RA は、セグメント#2 以外のセグメントで行った。

接続性

以上に示したサービスにより、参加者は、そのホストの種類によって、次に示す外部への接続性が得られる。

- IPv4 ホスト

  DHCP による IPv4 アドレスの取得、IPv4 → IPv6 トランスレータもしくは SOCK64 と IPv6 → IPv4 トランスレータの併用による IPv4 ホストへの接続性

- IPv6 に対応したホスト

  RA による IPv6 アドレスの取得、IPv6 ホストへの接続性、IPv6 → IPv4 トランスレータによる IPv4 ホストへの接続性、

以上より、IPv6 での接続性のみであっても、IPv6 に対応しないホストが外部へ通信することが可能である。

1.3.3　運用結果

最後に運用結果を述べる。なお、トランスレータの運用結果については、1.4.1節,1.4.2節,1.4.3節に示す。

- IPv6 アドレスの逆引き問題

  IPv4 では、DHCP を用いた場合、あらかじめ用意されたアドレス空間から IP アドレスを割り当てる。そのため、DHCP 用に確保した IPv4 アドレスの逆引きをあらかじめ DNS に登録することができる。しかし、IPv6 では、ホストが、RA と MAC アドレスからホストの IPv6 アドレスを生成する。そのために、事前に MAC アドレスを把握することは、不可能であるから、IPv6 アドレスの逆引きを DNS に登録できない。ホストの信用性のチェックに逆引きを利用するサーバを利用する場合、通信の拒絶や遅延という問題が生じる。この問題の解決には、安全なダイナミック・DNS の運用が必要である。

- NEWBIE の連携問題

  NEWBIE は、IPv4 転写アドレスと呼ばれる IPv6 アドレスを用いることにより、トランスレータの利用をシームレスした。よって、IPv4 の DNS としても、運用すれば DNS を統一することができたが、newbie は、開発中の DNS 実装であり、プライマリ・DNS として使うには力不足であった。このために、複数の DNS サーバが存在し、利用者の混乱を招いた。

- DHCP アドレス取得の問題

  セグメント #5 は、/16 の IPv4 プライベートアドレスを割り当てられ、IP 層的に一つのフラットなネットワークとした。そのために、DHCP サーバによる IPv4 アドレスの取得がうまく行かない場合があった。これは、DHCP がステートフルなプロトコルであるため、そのコストが高く、アドレス要求が集中した場合に、今回利用した DHCP サーバが処理しきれなかったために発生した。一方、IPv6 においては、RA によるステートレスなアドレス設定がされるので、このような問題は発生しなかった。

1.4 IPv4/IPv6 変換

1.4.1 v4/v6 Translator

実験の概要 Translator の接続を図に示す。 合宿地と慶應 SFC 側とを結ぶ線は二本あり、一本は v4 専用、もう一本は v6 専用とすることになった。

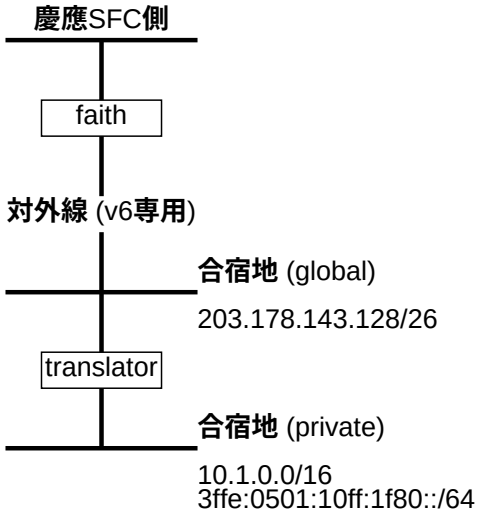

図 1.4: Translator の接続

合宿地のユーザが利用する segment は v4/v6 が共存する。しかし IPv4 専用の対外線は SOI が Video Stream に利用する。このため、SOI が中継をしている時間帯は IPv4 用の対外線はほとんど帯域が消費され利用できなくなる。 そこで v4 の packet を v6 に変換して v6 専用の対外線を通し、慶應 SFC 側で動いている faith を通じて対外の IPv4 Host との通信を行う。

慶應 SFC 側で faith が動いているという前堤で、translator は合宿地の v4 address を faith prefix (3ffe:501:10ff:2000::/64) を付けた v6 address に変換して faith に向けて packet を送信する。faith は、その prefix の付いた v6 packet を受け取ると v4 address を取り出し、必要な相手と v4 で session を張る。

評価

合宿の最初の頃は translator は安定しなかったが、合宿の三日目以降安定して動作した。今回の translator は TCP しか変換の対象とせず、しかも payload の変換をしなかったため、ftp は通らなかった。しかし payload の変換を必要としない protocol (ssh,telnet,smtp,pop3,irc

等) は通ったので、v4 で生活していた人達も これらの protocol では 途中で v6 に変換されている ということを意識せず、使えたようである。

今回の translator は payload を変更しないので stateless な header 変換を行った。 状態を持っていないので translator が落ちても reboot 後 変換が可能になると、v4 の session は落る前に張っていたものが有効に使えた。

今回の translator の使い方は、対外線を流れる packet は IPv6 だけにしたい という約束と、慶應 SFC 側で faith を動かしている という点で実際の translator が使われると思われる状況とは異なっているので、この IPv4 → IPv6 translator が有効だ とは言えないが、実装面では 安定した動作ができるようになってからは 特に問題無く動作した。

課題

今回の合宿で使った translator は単純に IPv4 → IPv6 への header 変換をするだけのもので 機能的にはまだ未対応のもの (ICMP echo/echoreply 以外、UCP 、fragment …) が多くある。だが、ワークショップ内ではユーザの利用に対して最低限サービスの利用が提供出来ることが判った。現状において実現されていないサービスへの対応は payload の変換をしなければいけない protocol すべてに対応する必要はないが、ftp は需要が多いので payload の変換はしなければいけないだろう。

IPv6 が普及する初期段階では 小数の IPv6 host が IPv4 の network に継がる と予想されるが、そこでの translator の動きは 合宿で行った変換とは異り IPv6 → IPv4 への変換となる。 動きとしては faith と同じ (IPv6 → IPv4) となるが、ICMP UDP も変換の対象とし 必要なら TCP を faith に上げることで faith と共存することができる。 こうすると kernel 内での ftp payload の変換は不要となる。

1.4.2　faith

目的

今回の合宿ネットワークでは、合宿地からの対外線が 2 本と、衛星回線が 1 本施設された．対外線の 1 本は、 SOI の中継用回線として利用され、もう 1 本の対外線と衛生回線が、合宿地からの通常の通信を転送するための回線として利用された．さらに、今回の合宿では対外線を IPv6 専用線として利用し、IPv4 の通信を転送しないことが決定された．

したがって、合宿地から合宿地外の IPv4 ホストに通信する場合には、対外線の対向となる地点である SFC において、v6-v4 トランスレータを利用する必要があった。

そこで、SFC 側にて v6-v4 トランスレータである faith を設置した。これによって、合宿地からの IPv4 ホストへの通信性を確保するとともに、faith のトランスレータとしての機能と性能を検証した。

ネットワーク構成

合宿地から faith を利用して、合宿地外の IPv4 ホストに通信を行う場合には、2 種類の形態が存在した。合宿地にあるホストが IPv4 のみ利用可能な場合と、IPv6/IPv4 の両方が利用可能な場合によって、通信形態が異なった。

- 合宿地のホストが IPv4 ホストの場合

  合宿地のホストが IPv4 のみ利用可能な場合には、合宿地の出口で v4-v6 トランスレータによって、通信が IPv6 に変換される。変換された通信は、IPv6 パケットとして対外線を通過し、SFC にて faith を通過して、IPv4 の通信に戻される。

- 合宿地のホストが IPv6/IPv4 ホストの場合

  合宿地のホストが IPv6/IPv4 両方利用可能な場合には、IPv6 によって通信を行う。IPv6 のパケットは対外線を通過し、SFC にて faith を通過して、IPv4 の通信に変換される。なお、faith では、通信先の IPv4 アドレスを IPv6 アドレスとして表現するための手法として、IPv4 アドレスを IPv6 アドレス中に埋め込むという手法が採用されている。

faith の設定

SFC 側に設置した faith では、以下のプロトコルに関して v6-v4 変換を行った。

- telnet
- ftp
- pop3
- ssh
- smtp
- irc

さらに、faith を利用するためには、faith 専用の IPv6 プレフィクスが必要である。そのプレフィクスとして、今回は 3ffe:501:10ff:2000::/64 を利用した。

通信解析

faith の利用状況を、3 月 24 日の合宿開始時から 27 日の合宿修了時まで測定した。faith を利用した通信セッション数の、最大値は 387 であった。同時に 378 個の faith プロセスが立ち上がっていたことになる。しかし、合宿当時に利用していた faith には、通信が何らかの理由で中断した場合に、プロセスが消えずに残ってしまう場合があるという不都合があった。したがってこの値はおおよその目安程度に考えられる。

プロトコル	最大値
telnet	129
ftp	8
pop3	9
ssh	238

表 1.1: 通信プロトコル毎の最大セッション数

また、各通信プロトコルごとのセッション数の最大値を表 1.1に示す。

さらに、各プロトコルの時系列におけるセッション数を、図 1.5に示す。横軸は時間 (分) である。途中でセッション数がリセットされている個所がある。これは、前述した faith の不都合のため、無効となった faith のプロセスが残ってしまっているのを除去するために、一度 faith を立ち上げ直したためである。

まとめ

faith による v6-v4 プロトコル変換は、非常に有効に機能した。TCP のアプリケーションはほぼ変換できるものと考えられる。前述の、プロセスが残ってしまう不都合は、合宿後に解消された。したがって、faith による v6-v4 プロトコル変換は、合宿規模のネットワークでも、問題なく実用に耐えると考えられる。

1.4.3　SOCKS64

実験の目的

- SOCKS64 の運用実験
- SOCKS64 と FAITH の組み合わせによる IPv4–IPv6–IPv4 接続の検証

実験の概要

SOCKS64 は SOCKS5 プロトコルを使用して IPv4–IPv6 相互接続を実現する方式の富士通研究所による実装である。IPv4 ホストに通常の SOCKS5 ライブラリ (libsocks5.a や SocksCap32 等) を導入し、SOCKS5 サーバとして SOCKS64 サーバを設定することで IPv4 ホストから IPv6 ホストへの通信が実現できる。IPv4 ホストが接続先の FQDN を SOCKS5 プロトコルで SOCKS64 サーバに渡し、SOCKS64 サーバがそれを解決すると共にその接続先への通信を中継する仕組みである。

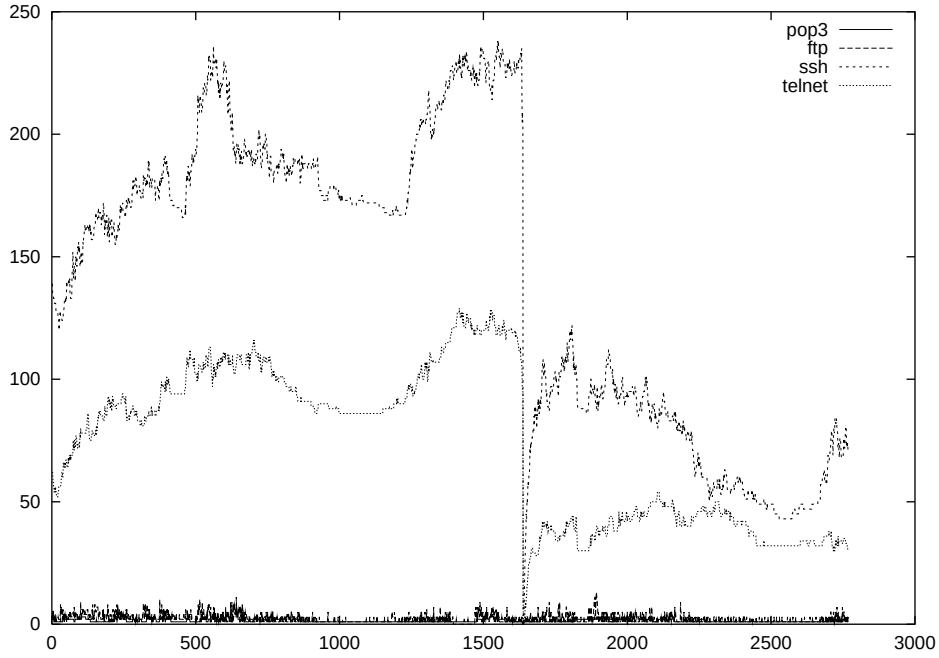

図 1.5: 各プロトコルセッション数

今回の合宿ネットワークでは合宿地外の IPv6 ホストへの接続はもちろん、SOCKS64 サーバを FAITH のクライアントとして動作させることで合宿地外部の IPv4 ホストへの接続もできるようにした。この場合、合宿地から FAITH を動作させていた慶應義塾大学 SFC までが IPv6 による接続となる。

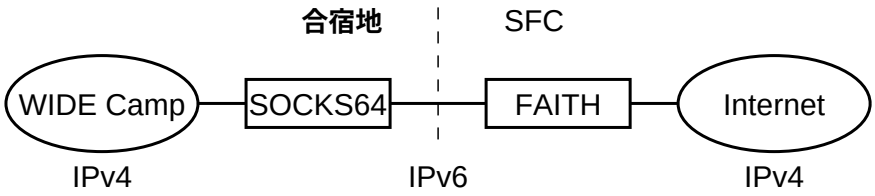

図 1.6: SOCKS64 と FATIH による IPv4–IPv6–IPv4 接続

実験環境

SOCKS64 サーバは MMX Pentium 266MHz のノート型 PC に BSDI BSD/OS 3.1 と IPv6 プロトコルスタック KAME をインストールしたものを使用し、10Mbps Ethernet で合宿ネットワークの NOC セグメントに接続した。IPv4 アドレスは固定で割り当て、IPv6

アドレスは IPv6 ルータからの RA メッセージによる自動設定とした。FAITH のクライアントとして動作させるため、FAITH 向けレコードを返す DNS サーバを DNS サーバとして指定した。DNS の問い合わせにも IPv6 を用いた。IPv6 に関しては単なるクライアントホストとしての設定とまったく変わりは無い。

結果

SOCKS5 プロトコルを利用して IPv4–IPv6 相互接続する方式はクライアント側でライブラリを導入する必要がある。FreeBSD や Windows ではダイナミックリンクライブラリを利用して簡単に導入できるが、ユーザ側で何もする必要がないトランスレータと比べるとユーザにとっての敷居が比較的高いようである。これまでに数回にわたって WIDE 合宿での実験を行ってきたが、回を重ねる毎に参加ユーザ数が増え、運用も安定してきた。

今回の WIDE 合宿では合宿地外への線が IPv6 専用となったため、IPv4 しか使えない人が外部に接続するためには IPv4–IPv6 トランスレータを利用するか SOCKS64 を利用するしかなかった。このため、今回の WIDE 合宿ではたくさんの実験参加者が得られた。今回の利用状況を表 1.2にまとめる。

今回の実験により、多くのクライアントに対して SOCKS64 が安定して動作できることを確認できた。

クライアント数	127
TCP コネクション数	35052
利用プロトコル	http, pop, ssh, telnet, smtp, ftp 他

表 1.2:

ユーザは DHCP により IPv4 アドレスを取得しているので、クライアント数はユーザ数とは異なるが、参加人数 200 数十名と比べてもかなり多く利用されていることが分かる。

5 分間隔で計測したコネクション数のグラフを示す。グラフの左端が 3 月 26 日午前 9:00 である。最大で 52 コネクション同時に利用されていることがわかる。

会期中、SOCKS64 のサーバが停止したりするようなことはなかったが、DNS サーバが不安定になることがあった。この DNS サーバは FAITH 向けレコードを返すため、実際の DNS に登録されているのが A レコードのみであっても必ず AAAA レコードを返す。しかし、ときどき AAAA レコードを返さなくなる現象が起こった。このようなときには DNS サーバの再起動を依頼したが、完全には対応できなかった。

また、IPv6 ルータが RA メッセージを出さなくなり、SOCKS64 サーバの IPv6 アドレスが無効になってしまうというトラブルもあった。これは IPv6 ルータで RA メッセージを出すようにすることで解決した。

今後の課題

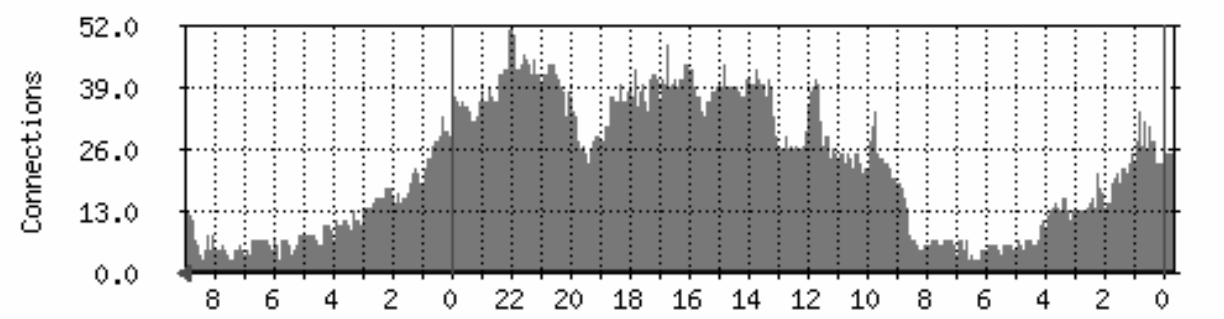

図 1.7: SOCKS64 利用コネクション数

今回は FAITH と組み合わせて IPv4–IPv6–IPv4 接続を実現したが、他の相互接続技術との組み合わせで試してみることは今後の課題である。

今回のログを解析した結果、IPv4 アドレスに対して接続している例がかなり見受けられた。上述の通り、SOCKS64 サーバは FAITH 向けレコードを返す DNS を引いているため、IPv4 アドレスに対して接続するのは次のようなケースしか考えられない。

1. アプリケーションが FQDN ではなく IPv4 アドレスを直接使用している
2. SOCKS5 ライブラリの設定を間違え、最初にクライアントで名前解決してしまってから SOCKS64 サーバに接続している
3. FAITH 向けレコードを返す DNS サーバが誤って A レコードを返している

残念ながら、今回採集したログからは上記のどれが起こっているのか判定できない。1. と 2. はクライアントの設定を確認しないと判別不能だが、3. は SOCKS64 サーバでより詳細なログを採集することで判別できる。今後はより詳細なログを取ることを検討する必要がある。

上記 1. や 2. を無くす方法として、SOCKS64 サーバが IPv4 アドレスをクライアントから受け取った場合にそれを DNS で逆引きして FQDN を得、その後に AAAA レコードを引きなおすという方法が考えられる。このような方式を実装し実験することは今後の課題である。

1.5　対外トラフィック制御

1.5.1　ネットワーク構成

今回の合宿における対外線ネットワークは、128K の地上線が二本と 2M の衛星線により構成される (図 1.8)。

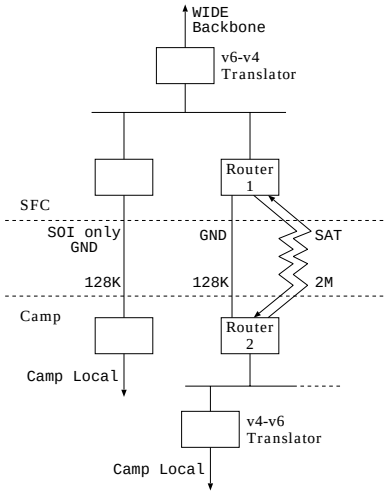

図 1.8: ネットワーク構成

地上線のうち一本は SOI 専用とし、地上線一本と衛星線を合宿の生活線として用いる。

今回の実験は IPv6 への移行後の IPv4 との互換性に主眼を置いているため、生活線を流れるパケットは IPv6 パケットのみとする。IPv4 パケットは SFCNOC と合宿地にそれぞれ設置された v4-v6 トランスレータにより IPv6 パケットに変換されて対外線を流れる。またトランスレータの他にも、合宿地に SOCKS64 サーバを設置し、v4-v6 トランスレータとしてこれを用いることも可能とした。

1.5.2　首振り

生活線として使用する 128K の地上線と 2M の衛星線は、前者は帯域が狭いが遅延は小さい、後者は帯域は広いが遅延は大きいという特徴を持つ。そのため、通常のトラフィックは地上線を通り、流量の大きいバルクデータのトラフィックは衛星線を通るように、地上線と衛星線の両端にプロトコルルーティングを行う首振りルータを設置した (図 1.8の Router 1、Router 2)。なお、衛星線は上下線あるため、プロトコルルーティングは地上線、衛星線の両端で双方向で行う。さらに、地上線では CBQ(Class Based Queueing) を用いて帯域制御を行った。

首振りルータ上では、経路的には全てのトラフィックは地上線を流れるように設定し、バルクデータのトラフィックだけを選別し衛星線に流すようにした。バルクデータの選別法としては、ポート番号による選別法を用いる。今回、バルクデータとして衛星線に流したパケットのポート番号を表 1.3に示す。

表 1.3: 衛星線に流したパケットのポート番号

種類	ポート番号
ftp-data	20
http	80
pop	110
squid	3128
http-proxy	8080

IPv4 のプロトコルルーティングの実装は様々存在し、これまでの合宿でも利用してきたが、今回の合宿では対外線を流れるのは IPv6 トラフィックのみであり、プロトコルルーティングも IPv6 で行う必要がある。そのため、今回新たに IPv6 のプロトコルルーティングの実装を行った。

実装は、基本的には前回の合宿で用いた mpath [1] というテクノロジを IPv6 に移植して実現した。mpath は、IP 層での防火壁機能である ipfw のルールセットを利用してプロトコルルーティングを行う技術である。ipfw は IP 層において送受信する各パケットに対し、あらかじめ設定されたルールセットを満足するか調べ、満足したパケットに対しては廃棄や通過などの動作を行う。mpath ではこのルールセットをそのまま利用し、あるルールセットを満足するパケットの次ホップを上書き変更できるように拡張することで、プロトコルルーティングを実現する。

1.5.3 結果

ネットワークについては、対外線ネットワークは合宿中一度も不通にならずに安定したサービスを提供していた。特に、IPv6 の首振りルータの実装が合宿中のトラフィック負荷に対しても安定で正常に動作していたことは特筆すべき点である。

1.5.4 トラフィック解析

合宿中、3 月 26 日 20:38 から 22:28 の間に、対外線を流れる全トラフィックのログを取った。なお、本論文で示しているトラフィックは、合宿中のある部分のトラフィックを抜き出しただけであり、一般的なトラフィックの様子を示しているとは必ずしも言えない。

[1] ftp://ftp.flirble.org/pub/unix/hacks/FreeBSD/mpath/mpath.b4.tgz

まず、対外線の SOI 専用線、地上線、衛星線の各上下線のトラフィックの様子を図 1.9に示す。ここで、内向き、外向きの SOI 専用線のトラフィックをそれぞれ SOI-in、SOI-out と表し、地上線のそれを GND-in、GND-out、衛星線のそれを SAT-in、SAT-out と表す。

図 1.9から、合宿中の全トラフィックのうち大部分は合宿地へ流れるトラフィックであったことがわかる。これは、対外線を流れるトラフィックはほとんどが合宿地内のユーザによるコネクションであり、内向きにデータが流れ、外向きには確認応答が流れていたことを示している。

そして、地上線と衛星線のそれぞれにおいて、IPv6 にトランスレートされている IPv4 トラフィックと IPv6 トラフィックの割合を、図 1.10、図 1.11に示す。

図 1.10 より、地上線に関しては IPv6 にトランスレートされた IPv4 と IPv6 のトラフィックはほぼ同程度であった。衛星線では図 1.11より、IPv6 にトランスレートされた IPv4 が IPv6 よりもはるかに多かったが、これは図 1.15より squid が大多数を占めており、インターネット上の web サーバの大多数が IPv6 に対応していないために squid サーバが IPv4 を利用したことに起因すると考えられる。しかし、web トラフィックを除けば IPv6 と IPv4 のトラフィックはほぼ同程度であり、今後の IPv6 の普及のためには、いかに web サーバの v6 化を広めていくかが焦点となると言える。

次に、地上線の上下線、衛星線の上下線それぞれにおける主要なアプリケーションの割合を図 1.12、図 1.13、図 1.14、図 1.15 に示す。

図 1.12と図 1.13、図 1.14と図 1.15より、合宿地から外へのトラフィックは内向きのトラフィックとはオーダが異なっており、確認応答が大部分を占めていると言える。そこで、内向きのトラフィックに着目すると衛星線は大部分が squid であった。一方、地上線は ssh や telnet により占められていた。また、図 1.9を見ても明らかなように、衛星線でのピーク値は 300Kbps 程度であり、2M の帯域を埋め尽くすほどのトラフィックではなかったことがわかる。逆に、地上線は 128K 付近まで常にトラフィックが流れており、パケット落ちが頻発し、ssh や telnet などの interactive セッションは非常にレスポンスが悪くなっていた。これはつまり、今回の合宿で首振りルータはバルクデータの選別法としてはポート番号による手法を利用していたが、この手法では首振りルータがバルクデータを選別しきれなかったと言える。

その理由は、近年の ssh の普及により、バルクデータ転送などのセッションを ssh の Port Forwarding 機能を用いて使用するユーザが増えたことが第一に考えられる。このようなトラフィックは、全てのパケットに ssh の Well Known ポート番号が割り当てられるため、純粋な遠隔ログインセッションとしての ssh パケットなのか、それとも ssh の殻に覆われたバルクデータセッションのパケットなのか区別がつかない。

また、ftp のデータ転送コネクションに Well Known ポート番号 (20) を使わない ftp サーバが増えてきたことも理由として挙げられる。この場合、パケットには一時的なポート番号が割り当てられ、首振りルータはどれが ftp データ転送パケットなのかを判別できなくなってしまう。

上記の理由により、首振りルータではバルクデータをバルクであるという判断ができずに地上線に流してしまったため、衛星線は空き、地上線は非常に混雑するという状況が起こってしまったと考えられる。

これはつまり、ポート番号によるセッションの識別はもはや限界であり、新たな手法が今後は必要とされていくことを示している。

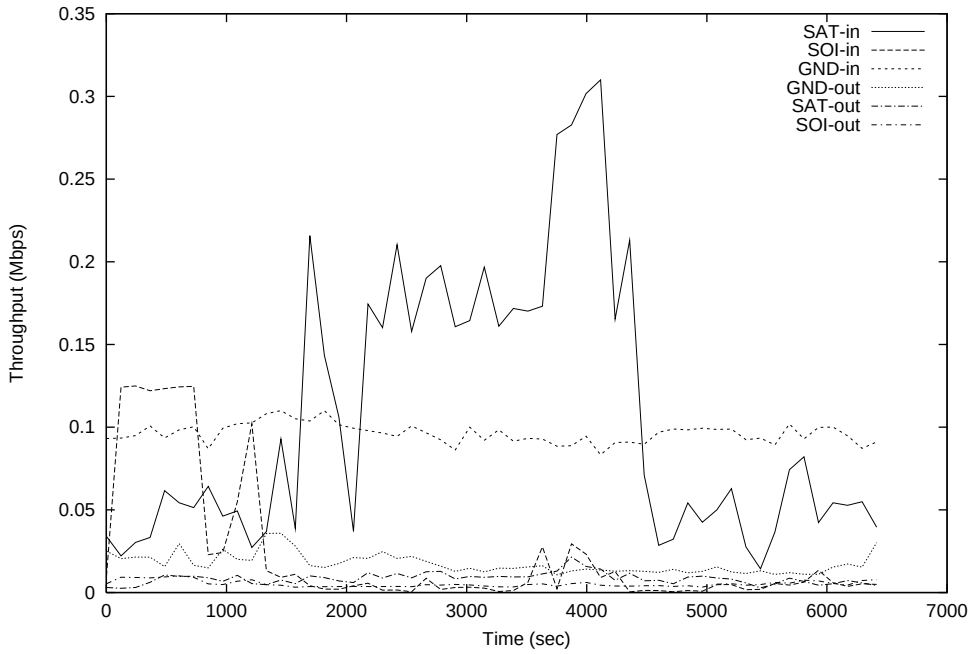

図 1.9: 各線におけるトラフィックの様子

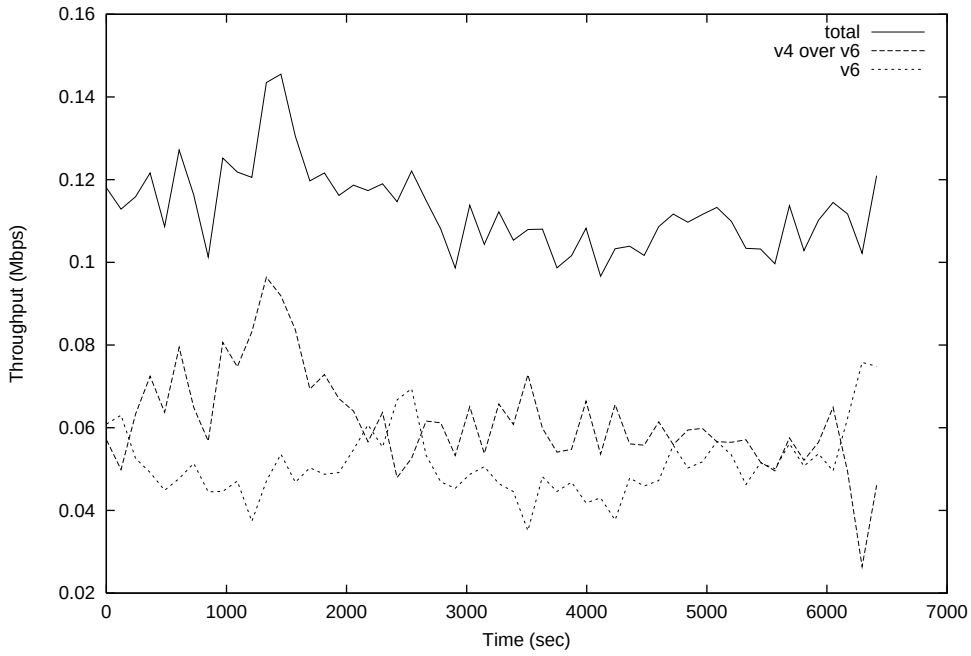

図 1.10: 地上線における v4、v6 トラフィックの割合

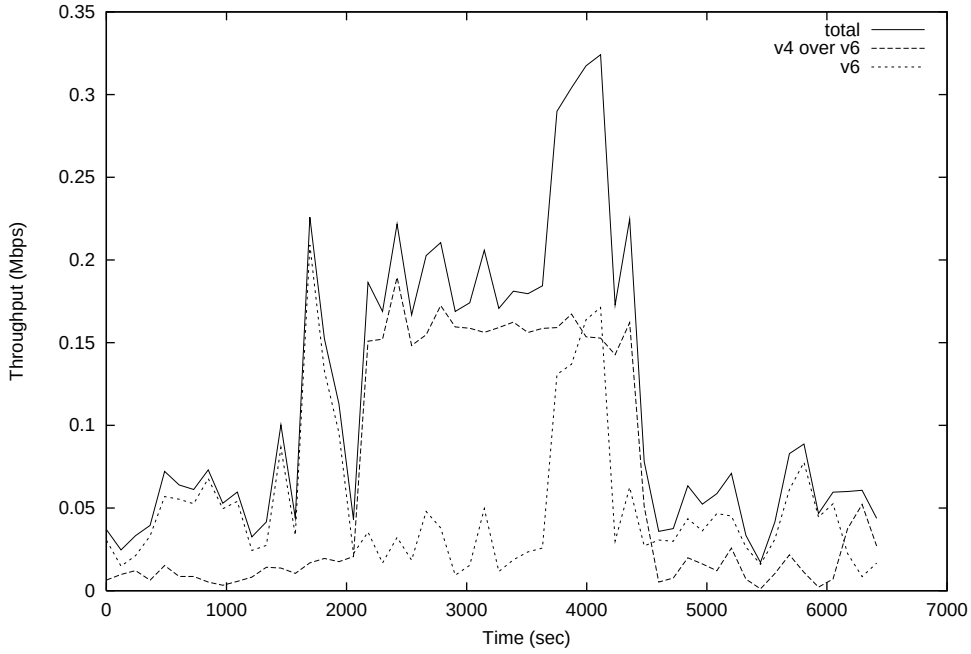

図 1.11: 衛星線における v4、v6 トラフィックの割合

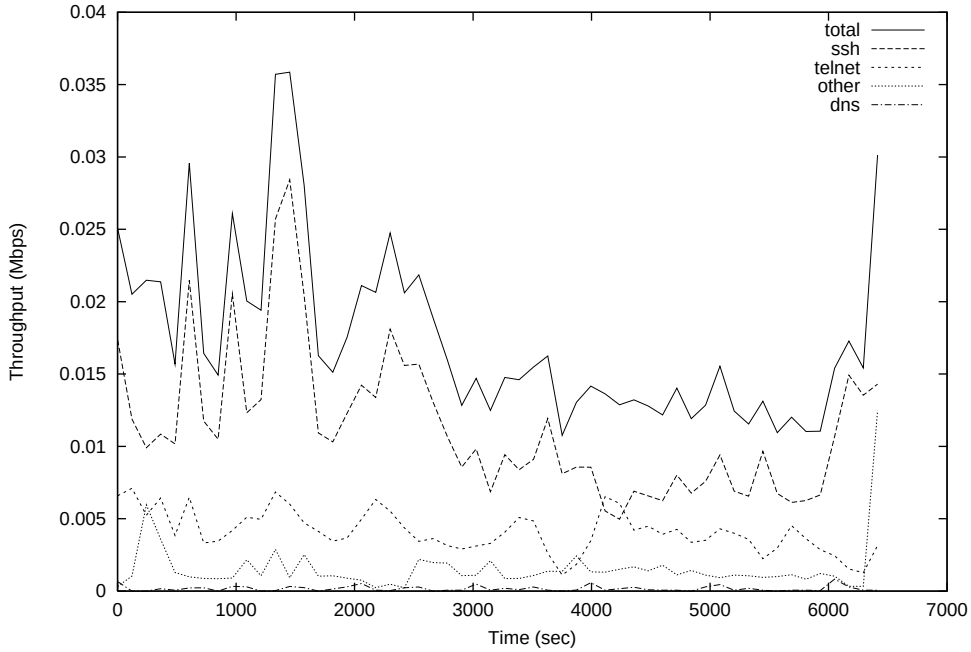

図 1.12: 地上線外向きのトラフィックにおける各アプリケーションの割合

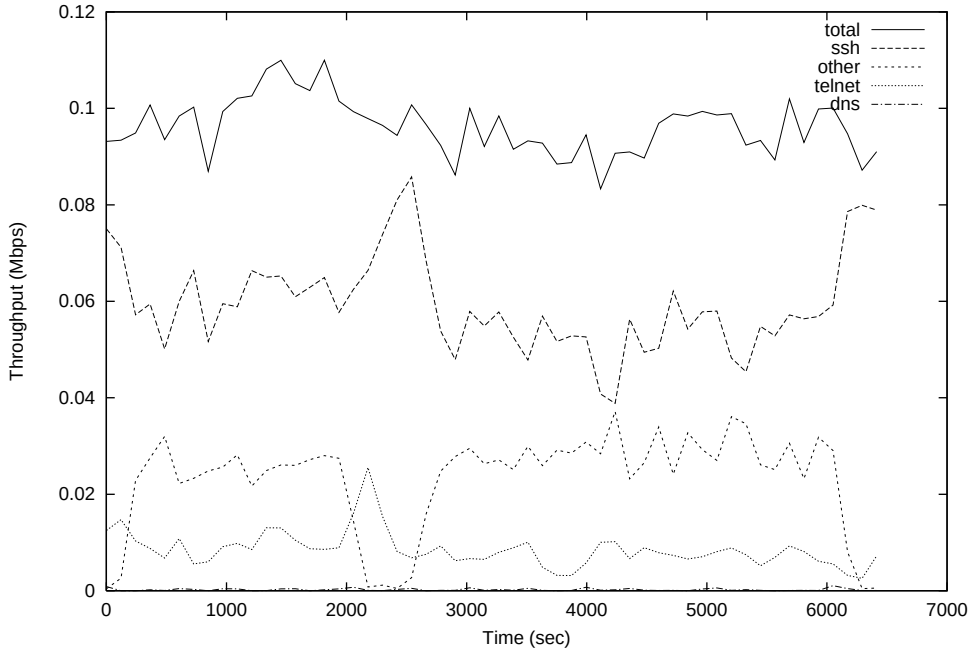

図 1.13: 地上線内向きのトラフィックにおける各アプリケーションの割合

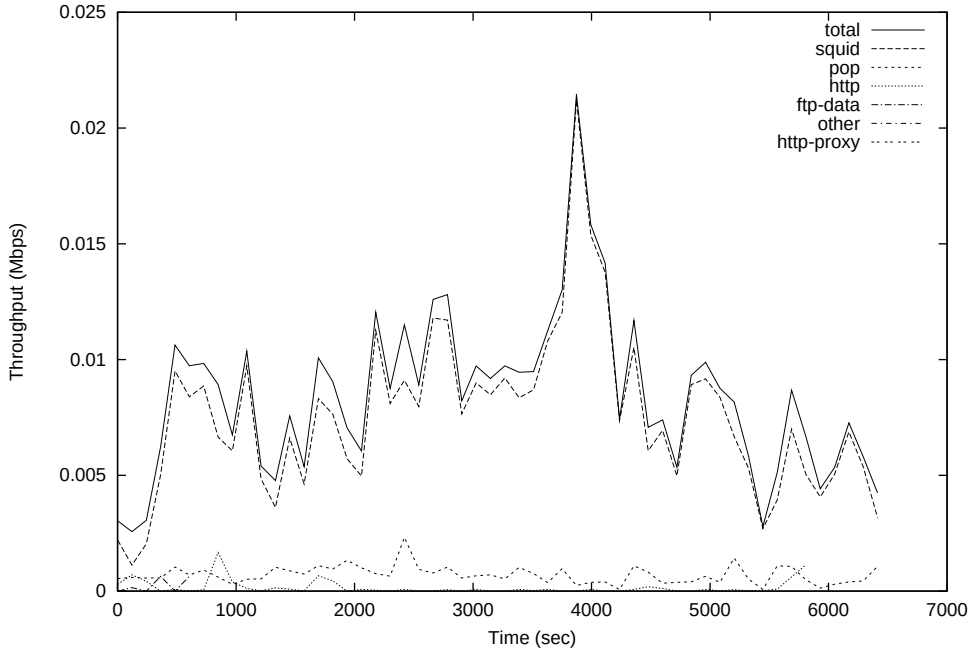

図 1.14: 衛星線外向きのトラフィックにおける各アプリケーションの割合

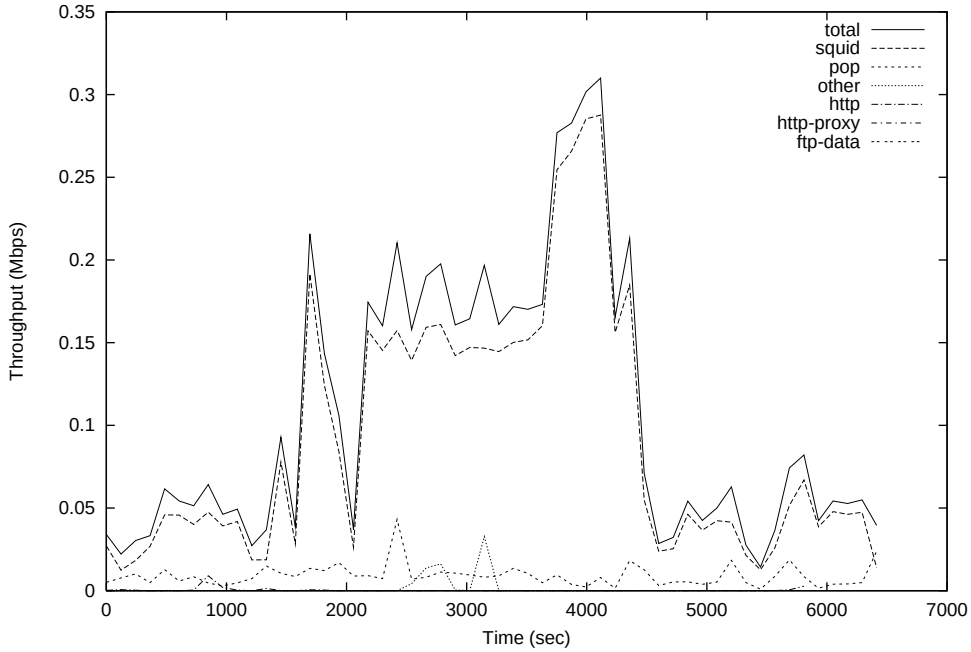

図 1.15: 衛星線内向きのトラフィックにおける各アプリケーションの割合

1.6 Y2K

1.6.1 実験目的

2000 年問題 (以下 Y2K) に対する合宿内での実証実験

1.6.2 実験概要

Y2K の検証は、実際に合宿の参加者が個々に自分の持つ端末などの機材の日付を 1999 年の 12 月 31 日に設定する。合宿内では、実験参加の機材の時刻を出来る限り同期させる。各機材はこの状態で年を越し、その時実際の利用にどのように影響するかを検証すると言う手法を取る。

時刻の同期は NTP (Network Time Protocol) を利用する。NTP サーバはクライアントに対して 1999 年 12 月という未来の時間を提供し、合宿のネットワーク上に用意される。各クライアントは、各ユーザによって ntpdate 等をした時点で実験の参加となり、機材の時計が 1999 年の 12 月 31 日に設定される。

この状態で時間を経過させ、2000 年 1 月 1 日を迎える。その後、timezone に配慮し 10 時間以上経過させた後、NTP サーバが提供する時刻を現実の世界の時刻に戻し、各機材で時刻合わせを行う。

1.6.3 結果

実験後その結果は実験参加者に対して特筆すべき現象とそれが起きた要因と考えられる内容に関してのアンケートを行った。

2000 年を迎えた時点での現象は、Perl で記述された CGI Script 等で、時刻の表示が 00 年ではなく 0 年と表示されたという報告があった程度であった。しかし、時刻を一時的に狂わせた影響は大きく、未来に飛躍した時点では、認証のための鍵が期間切れとなり削除されると言った現象や過去に戻った時点では、時刻表示プログラムが暴走したといった現象や、file の時刻の整合性が取れなくなり、バージョン管理が破綻したと言う現象が報告されている。

この実験結果は、合宿の特性上既に Y2K に対して対応済の物が多くあった事が考えられる。また、合宿がこの問題に対する検証期間があまりにも短期だったため、まだ潜んでいる問題も多く存在すると考えられる。

実験時または後の影響となる、未来や過去への時間の飛躍は実験後、様々なシステムに大きく影響し、様々な問題を引き起こす。

1.6.4　まとめ

今回の実験では、極めて限られた合宿という状況の中で非常に短期間の実験であっても何らかの問題が発生している。この事は今現在、現実に稼働しているシステムのスケールで考えるとどの程度の問題が潜んでいるかを予測する事は不可能である。

合宿開催後のアンケートの Y2K に関する内容とその結果は表に示す。

表 1.4: 2000 年問題を取上げたことについてどのように思われましたか。

あまり興味がわかなかった。	27 人 (23.89%)
非常に重要な問題だと認識できた。	56 人 (49.56%)
今更遅い。	17 人 (15.04%)
その他	11 人 (9.73%)
無回答	0 人 (0.00%)

表 1.5: プレナリ前と後で 2000 年問題についての考えが変わりましたか。

YES	26 人 (23.01%)
NO	83 人 (73.45%)

表 1.6: 今後さらに、WIDE 内で 2000 年問題対応を組織的に行う必要があると思いますか。

YES	79 人 (69.91%)
NO	30 人 (26.55%)

この結果から実験参加者の意識では Y2K の問題の重要性に対するに対する認識の変化が見られる。

今回の合宿での実験では残念ながら本質的な Y2K の問題を見出すまでには至らなかった。しかし、この規模の実験でさえ Y2K の問題が無かったわけではなく、現実にそれが来た時に何が起こるかを予測するのは不可能である。Y2K の問題に対しては、現実にこのような小規模でなく実際にもっと大規模なシステムにおいての実証実験を綿密な計画のもと、極めて早急に行う必要がある。