Title	グローバルコンピューティング環境におけるホスト情報取得および可視化の実現
Sub Title	Realizing acquisition and visualization of hosts in global computing environment
Author	塚原, 康仁(Tsukahara, Yasuhito) 杉浦, 一徳(Sugiura, Kazunori)
Publisher	慶應義塾大学大学院メディアデザイン研究科
Publication year	2010
Jtitle	
JaLC DOI	
Abstract	今日のインターネットでは計算機以外の機器も接続されるようになった. しかし, 計算機以外をインターネットに接続するには, それぞれ独自の設定環境が必要となる. そこで, 機器を管理, 制御, 通信可能なグローバルコンピューティング環境を構築することで, 統一された環境でインターネットに接続されたホストを管理, 制御, 機器間通信を可能とする. グローバルコンピューティングとはインターネットに接続された機器を自律的に発見し, 機器の制御, 管理, 機器間通信をすべての機器に対して行うことができる統合された環境である. グローバルコンピューティング環境にはホストの自律的発見, ホスト情報の取得, ホスト情報の提供機構が必要である. 本研究ではこれらの機構を有しホスト情報を可視化する機器管理システムを作成した. ホストの自律的発見機構は, ネットワークを流れているパケットを取得し, IPアドレス設定プロトコルを用いてホストを発見する. 同時にホスト情報もアドレス設定プロトコルから取得する. ネットワークに接続機器は必ずアドレス設定をするため, グローバルコンピューティング環境にも対応可能である. ヘッダフォーマットにしたがってホスト情報を提供し, 可視化する. 提案システムがグローバルコンピューティング環境に対応可能かについて評価した. 実験ではDHCPモジュールとDHCPログの対応性及び応答時間の計測を行った. 実験の結果, ホスト情報を全て取得できたが, 一定以上のスケーラビリティに課題を残した.
Notes	修士学位論文. 2010年度メディアデザイン学 第97号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40001001-00002010-0097

2010 年度　　修士論文

グローバルコンピューティング環境における
ホスト情報取得および可視化の実現

KEIO MEDIA DESIGN

慶應義塾大学大学院
メディアデザイン研究科

塚原　康仁

本論文は慶應義塾大学大学院メディアデザイン研究科に
修士 (メディアデザイン学) 授与の要件として提出した修士論文である。

塚原　康仁

指導教員：

杉浦 一徳 准教授　(主指導教員)

加藤 朗 教授　(副指導教員)

審査委員：

杉浦 一徳 准教授　(主査)

加藤 朗 教授　(副査)

古川 享 教授　(副査)

グローバルコンピューティング環境における
ホスト情報取得および可視化の実現

内容梗概

今日のインターネットでは計算機以外の機器も接続されるようになった．しかし，計算機以外をインターネットに接続するには，それぞれ独自の設定環境が必要となる．そこで，機器を管理，制御，通信可能なグローバルコンピューティング環境を構築することで，統一された環境でインターネットに接続されたホストを管理，制御，機器間通信を可能とする．グローバルコンピューティングとはインターネットに接続された機器を自律的に発見し，機器の制御，管理，機器間通信をすべての機器に対して行うことができる統合された環境である．グローバルコンピューティング環境にはホストの自律的発見，ホスト情報の取得，ホスト情報の提供機構が必要である．本研究ではこれらの機構を有しホスト情報を可視化する機器管理システムを作成した．ホストの自律的発見機構は、ネットワークを流れているパケットを取得し，IP アドレス設定プロトコルを用いてホストを発見する．同時にホスト情報もアドレス設定プロトコルから取得する．ネットワークに接続機器は必ずアドレス設定をするため、グローバルコンピューティング環境にも対応可能である。ヘッダフォーマットにしたがってホスト情報を提供し，可視化する．提案システムがグローバルコンピューティング環境に対応可能かについて評価した．実験では DHCP モジュールと DHCP ログの対応性及び応答時間の計測を行った．実験の結果，ホスト情報を全て取得できたが，一定以上のスケーラビリティに課題を残した．

キーワード

パケットキャプチャ, Plug and Play, ネットワーク可視化

慶應義塾大学大学院　メディアデザイン研究科

塚原　康仁

Realizing Acquisition and Visualization of Hosts in Global Computing Environment

Abstract

Appliances other than computers are challenging to be connected to the Internet. To connect those appliances to the Internet, the different configuration for each different device is required. It is needed to develop Global Computing(GC) Environment, which is the environment that everything can be connected to the Internet and users can easily manage, control and connect each host under unified specification, to plug and play. GC environment needs three functions. Autonomous host discover, analyze and provide host information. We develop acquisition and visualization system for GC environment which are including three functions. Autonomous discovery function and analysis function captures the packet of IP address configuration protocols. Proposed function has compatibility with GC environment, because all of the hosts are configured by IP address configuration protocols. We implemented a visualizing application which is using proposed system. We measure the response time of client and comparison between DHCP log and our system log. Experimental result shows our system can detect hosts accurately and indicate the problem of scalability.

Keywords:

Packet Capturing, Plug and Play, Network Visualization

Graduate School of Media Design, Keio University

Yasuhito Tsukahara

目　　次

図　目　次

表　目　次

第1章
序　　論

1.1. 研究背景

従来のインターネットでは計算機のみが接続されていた．近年インターネットに接続される機器は多様化した．例えば，家庭用電化機器があげられる．代表的な電化機器であるテレビは，インターネットを介して制御できるようになった．ネットワークオーディオシステムでは，インターネットを介して音楽ファイルを利用できるようになった．今日ではこのように身の回りの物がインターネットへと接続されるようになった．

こうした状況の中で，ネットワークに接続された機器をネットワークを介して横断的に使用できる環境の構築が試みられている．横断的に使用できる環境とは，ネットワークに接続しているホストを管理し，制御し，ホスト間の通信を行えるようにした環境を指す．そしてそのホストは我々の身の回りの家庭用電気機械器具（家電）だけでなく，自動車やあるいは社会的なサービスといったあらゆるものである．こうした試みは研究機関である KEIO-NUS Cute Center のグローバルコンピューティングプロジェクト [1] で行われている．グローバルコンピューティングプロジェクトは実世界に存在するものすべてがネットワークでつながれ，管理，制御，ホスト間通信を行える環境構築を目指している．Microsoft 社が提唱した Universal Plug and Play（UPnP）[2] では．ネットワークに接続するだけで，接続された機器が動作できる接続性を提供し，その上で様々な機器間の通信を行えるようにしたものである．Apple 社の Bonjour[3] ではネットワークへ機器が接続されたときに IP アドレスの割当から，ネットワーク内のサービスをマルチキャストし探索する機能を有している．Bonjour はネットワーク内のサービス

に焦点をおいた技術である．

このように機器や実在する物に対してインターネットへの容易な接続性を提供し，ホストの管理，制御，設定，機器間通信やサービスを提供する取り組みがなされてきた．しかし，こうした環境を構築する上で課題がある．例えば，ネットワークに接続しているホストの発見である．ホストとはネットワークに接続されている機器である．ネットワークに接続される物は多様となりインターネットは以前に増して複雑になっている．こうした状況で自律的な運用を行うためにはネットワークに接続されている機器を把握できなければ管理ができないからである．さらに，発見したホストを把握するためには発見したホストを可視化しなければならない．システムがホスト情報を保持していても，提示しなければ認識できないからである．そして，ホスト情報を提示するためにホスト情報を提供する機構も必要である．これら３つの課題が克服されなければ自律的な運用も管理もできない．ホストを管理，制御，設定，機器間通信やサービスを提供する基盤システムを実現するためにはこうした課題がある．

1.2. 研究目的

本研究の目的は，グローバルコンピューティング実現に向けた課題の１つであるホスト情報の自律的取得および可視化を実現するシステム提案である．実在するすべての物がネットワークへ容易にアクセスでき，それらホストを管理，制御，設定，機器間通信を行うことができるインターフェースを提供することで新たなサービスが生まれる創造基盤の構築を実現する．そのためには，自律的に接続されたホストを発見し，ホストの詳細情報を取得する．そして，その情報を提供し，可視化するシステムが必要となる．

本研究における，ホストの発見，情報取得，可視化を行う手法は多様なネットワークに適応できる手法とならなければならない．用途として MAN，WAN といった広域ネットワークまでもが想定される．そのため，各種ネットワーク環境下で自律的にホストの発見，情報取得可能になるような手法を選択する必要がある．ホストの発見および情報取得に用いることが可能な手法は，ネットワーク上

を流通するデータ (パケット) を取得，解析し情報取得を行うパケットキャプチャリングや，ICMP の利用などがあげられる．本論文ではパケットキャプチャリングに着目している．これによって，様々な環境のネットワークに適応でき，自律的なホストの発見，情報取得，可視化をが期待できる．

1.3. 本研究により期待される成果

ネットワーク管理者を含めたユーザは抽象化され，目に見えない世界のインターネットの中のホスト認識が困難となっている．グローバルコンピューティングではさらに多種多様な機器だけでなく実在するものが接続されることを想定している．ユーザのホスト認識はいっそう困難なものとなる．

本研究で期待される成果は，ユーザがホスト認識がいっそう困難な環境であっても自律的にホストを発見し，情報を取得する．取得された情報を可視化しユーザに対してホスト情報をリアルタイムに認識させる環境を構築する．本研究にて構築される環境は，グローバルコンピューティングを実現する上でネットワーク上の機器を可視化する機構を実現することである．ホストを管理，制御，設定や機器間通信を行うには，ホストを認識することが第一段階である．認識した状態で，次に管理や制御，設定，機器間通信が可能となる．したがって，本研究で構築される環境は管理や制御，設定，機器間通信等の機能を追加できるスケーラビリティを考慮した設計が期待される．スケーラビリティとは機能追加が容易にできる，機能拡張を指す．

1.4. 本論文の構成

本論文は全９章から構成される．第２章ではグローバルコンピューティングの構想に関して詳細に議論する．第３章ではグローバルコンピューティングの現状と課題に関して議論する．現状の議論では必要機能といったところまでの議論も行う．そして，現状の課題として関連する事例を挙げながら，実現に向けた課題を議論し，明らかにする．第４章ではグローバルコンピューティング実現に向け

たホスト情報取得および可視化システムの提案を行い，現状を議論しながら，複雑になったネットワークの可視化に最適な手法であり，グローバルコンピューティングを踏まえた手法を議論する．第５章ではシステムの設計について述べる．システム要件を定義し期待される成果を達成するための議論をする．第６章では実際の実装したものをまとめ，第７章では情報取得および可視化システムの評価を行う．第８章では第７章での評価結果をふまえ，今後の課題として述べる．第９章では最後に本論文の結論に関してまとめる．

第2章
グローバルコンピューティング環境の実現

本章ではグローバルコンピューティングについて述べる．グローバルコンピューティングは本研究が寄与する研究である．よって，グローバルコンピューティングの構想から本研究の関連を示す．また，グローバルコンピューティングに関連した．現状に関して示す．

2.1. グローバルコンピューティング環境

グローバルコンピューティング環境とは実在するすべてのものを IP(Internet Protocol) で構成されたネットワーク (インターネット) につなぎ，制御や設定，管理，ホスト間の通信を行うことができる環境である．今日，インターネットに計算機以外のものが接続されるようになった．ここでいう計算機とは，PC やラップトップ，携帯電話である．ウェブカメラや自動車，家電やテレビなど，多くのものが接続されている．しかし，計算機以外で接続されるようになったものは異なる仕様となっておりユーザはインターネットに機器を接続する場合には設定をしなければならないという問題がある．例えば，ウェブカメラをインターネットに接続する場合は専用のアプリケーションを使用して，IP の入力やユーザ名やドメイン名などを追加する必要がある．この専用のアプリケーションもベンダーによって異なるものとなり，ユーザはベンダーによって異なるアプリケーションを使って設定しなければならない．それぞれ機器の種類が異なるものであったり，機器のベンダーが異なる場合にはその種類やベンダー専用の設定方法を学ばなけ

ればならない．インターネットへの接続設定が複雑で，ユーザは機器を容易にインターネットに接続することができない．このようにインターネットに計算機以外のものをインターネットに接続する場合は，異なる方法で設定をしなければインターネットに接続できないという問題がある．

グローバルコンピューティングはこうした問題を解決する環境を提供する．機器を起動すれば，自動的にネットワークへと接続し，使用できる環境である．そして，統一された環境下でユーザはネットワークを介して機器を制御し，管理し，ホスト間通信を容易に行える環境を提供する．このようなグローバルコンピューティングで実現する環境は計算機では実装されている．zeroconf(zero configuration) と呼ばれる概念の下で，PC やラップトップ，携帯などは，IP アドレス設定を自動的に行う環境が実装されている [4]．そして，PC やラップトップ，携帯電話間はインターネットを介して互いに通信できる環境となっている．グローバルコンピューティングでは，zeroconf の世界を計算機以外のものに広げ，身の回りの機器をネットワークを介して使用できる環境を提供するのである．今までは機器をネットワークへ接続する際，複雑な設定を強いられていたがこうした問題をグローバルコンピューティングは解決する．

グローバルコンピューティングでインターネットに接続される機器はすべてのものを想定している．例えば，電子機器ならばテレビやオーディオプレーヤー，センサー，ロボットなど多岐に渡る．また，これまでは非電子機器だったものも電子機器となってインターネットに接続される．例えば自動車である．WIDE Project の iCAR Working Group では，自動車をインターネットに接続する研究がなされている [5][6]．同様に，自動二輪車や自転車などもインターネットに接続されるようになった．他にも，家具，筆記具，調理器具などがある．これらはネットワークに接続されることで電子化される．このようにグローバルコンピューティングでは身の回りのものがインターネットに接続されることを想定している．

こうしたものが接続されるグローバルコンピューティング環境では，大別して３種類からなるネットワークが IP ネットワークを介して形成される．図 2.1 で，グローバルコンピューティング構想で形成される３種類のネットワークを示す．

これら３種のネットワークは従来から存在したものである．こうしたネットワー

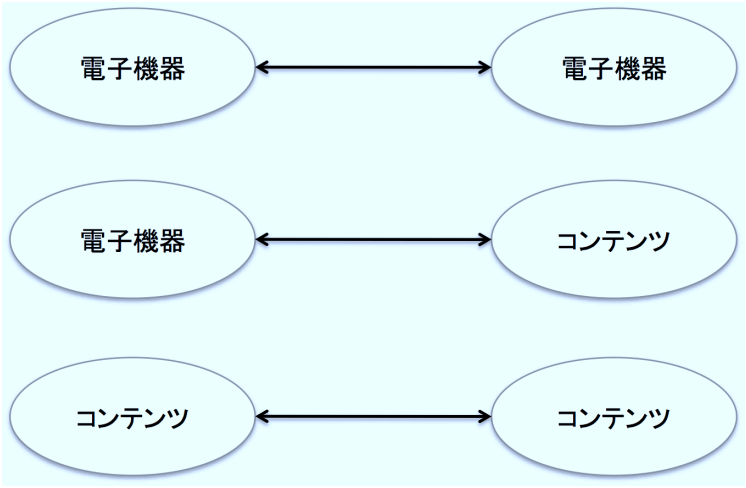


図 2.1 グローバルコンピューティングで形成されるネットワーク

クもまたグローバルコンピューティング環境によって機能は大きく広がることになる．

以下，図 2.1 に図示されたネットワークを順に論じる．

1. 電子機器 – 電子機器
   電子機器間のネットワークは，電化製品同士の IP ネットワークである．これは従来からも存在するものである．例えば，ホームネットワークがあげられる．AV 家電へ PC から映像データや音声データを IP ネットワークを介して出力することや，遠隔ビデオ監視用のウェブカム，その他冷蔵庫や空調といったもの，これらを管理するためのセンサーなどである．電子機器 – 電子機器のネットワークでは今までは計算機だけで形成されていた IP ネットワークに上記したような電子機器同士で形成されるネットワークが含まれるのである．

2. 電子機器 – コンテンツ
   電子機器 – コンテンツのネットワークではネットワーク上のコンテンツが有するデータを電子機器に出力するあるいは逆に，電子機器からインプットされたデータをネットワーク上のコンテンツへ出力するといったネット

ワークである．例えば，施設に設置された温度センサーやCO_2センサーの値に基づいて施設の空調制御などがある．センサーはウェブ上のデータとしてアップロードされていて，空調側のシステムがそのデータに基づいて調整するのである．他にもインターネット上の映像データや音楽データをテレビなどのディスプレイで利用することもできる電子機器 – コンテンツのネットワークはインターネット上のコンテンツのデータが電子機器へ出力されるネットワークあるいは電子機器によって得られたデータがコンテンツへ出力されるネットワークなのである．

3. コンテンツ – コンテンツ
   コンテンツ – コンテンツはインターネットの中のコンテンツ間の横断的やり取りである．例えば，OAuth など使って各コンテンツ毎に横断的にユーザー情報を利用したサービスがあげられる．twitter[7] はマイクロブログとして人気のコンテンツである．140 文字以内で「つぶやき」と呼ばれるコメントを投稿し，そのコメントをユーザと共有するサービスである．そしてこの「つぶやき」の蓄積されたデータを利用した外部サービスなどがある．また,「つぶやき」は他のサービスへと同期させることもできる．Facebook[8], Mixi[9] などへ投稿することもできる．Facebook や Mixi は Social Network Service(SNS) というもので，ユーザと写真や日記，コメントなどで交流することができるサービスである．他のサービスでも Google[10] のアカウントを使って他のサービスへ転用することもできる．Google とは検索エンジンサービスであり，その他にもメールサービスやカレンダーサービスなど多岐にわたったサービスを提供している．それで，Google アカウント情報を使って，SNS で Google ユーザーを探すことができる．コンテンツ – コンテンツは社会的なサービスから SNS のようなコンテンツまで連携したコンテンツで形成されるネットワークである．

これまでもこれら 3 種類のネットワークはそれぞれ作られてきた．しかし，こうしたネットワークはそれぞれが独自の規格のもとにネットワークにつながれている．それぞれの規格は互換性に乏しいために横断的に使用することができない．例えば，ECHONET[11] では，無線を使用して自宅の設備をネットワークに接続

している．Ethernet や Blututh などで自宅の冷蔵庫や電灯といった家電をネットワークに接続し，管理や制御ができるようになっている．その他センサーなどでも使用されている．一方で，UPnP[2] では AV 家電を中心にネットワークに接続すると HTTP を利用して AV 家電間での通信をインターネットを介して容易に行える環境が存在する．UPnP を基盤とした DLNA[12] は AV 家電や印刷機，NAS，PC などを相互通信させることきるようになっている．こうした技術は各々が特化した機器をもちすべての機器を対象とはしていない．そのため，あらゆる機器を横断的に使用することはできない．グローバルコンピューティングではあらゆる機器を対象として，インターネットを介して容易に機器の制御や管理，機器間通信を可能とする統合環境の構築を行う．グローバルコンピューティングという統合環境によって 3 種類のネットワークを容易に利用することができるようになる．そして，グローバルコンピューティングを実現するためにはインターネットに接続した機器を発見し，制御，管理，機器間通信を行うために機器の種類や通信に必要な情報を取得する．さらに，発見された機器の情報を提示する機構を構築することが必要である．

2.2. グローバルコンピューティング環境のインターネット

グローバルコンピューティングを実現するにはホストがネットワークにアクセスしたことを自律的に検知し，そのホスト情報を保持し管理する．そして，クライアントアプリケーションからの要求に応え，利用できるホストを提示し，ユーザが利用したいサービスや機器を自由に選択し，利用できる環境を目指す．言い換えれば，グローバルコンピューティングが実現するとユーザは機器をネットワークに接続するだけで，機器の管理や制御，機器間通信を行うことができる．互換性の乏しい環境下，機種毎で制御や管理，機器間通信が行われていたが，グローバルコンピューティングによって，統合環境のもとあらゆる種類の機器をインターネットを介して利用することができる．

図 2.2 はグローバルコンピューティングが実現されたインターネットを表した

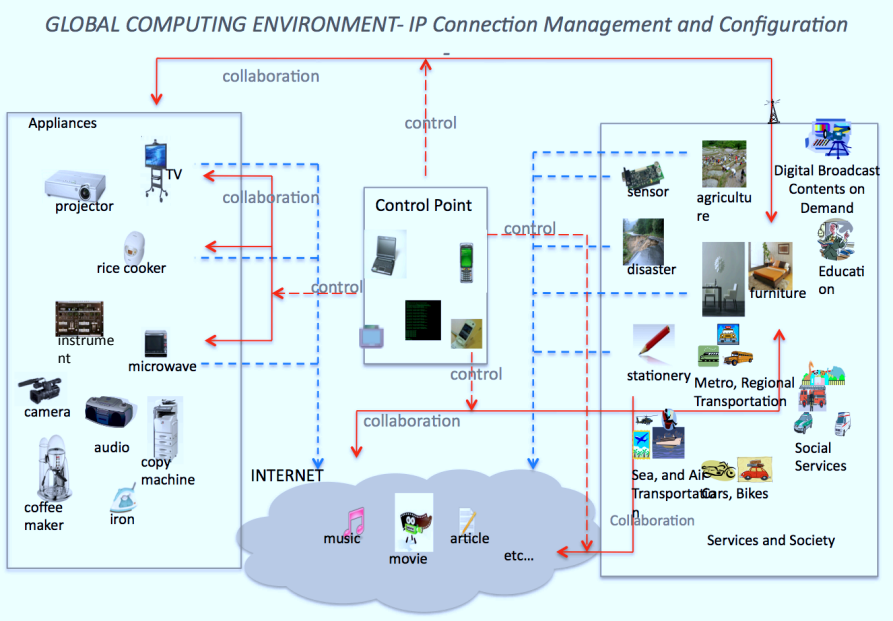


図 2.2　グローバルコンピューティングの全体像図

図である．図中央のコントロールポイントがグローバルコンピューティングの要である．左側群が電子機器，右側が従来は非電子機器だったものである，下部がウェブコンテンツである．これらすべてのものが IP ネットワークに接続されたグローバルコンピューティング環境を表している．こうした環境でコントロールポイントはそれぞれのホストを容易に設定でき，統一された環境のもとホスト同士のネットワークを形成できる環境を提供する．

2.3. グローバルコンピューティングに関連する現在の状況

グローバルコンピューティングのような取り組みはホームネットワークの範疇で取り組まれてきた．UPnP[2]，DLNA[12]，OSGi[13]，HAVi[14]，ECHONET[11]，Bojour[3] があげられる．情報家電ネットワークと通信放送連携 [15] によれば，グローバルコンピューティングの構想と同様の取り組みを行っている．例えば，UPnP

や DLNA はホームネットワークを中心に利用されてきた．

DLNA では，各種電子機器に対して，音楽，写真，動画などのデジタルコンテンツを電化製品や，PC，携帯端末などの電子機器間で容易に共有するための相互接続性のフレームワークを構築している．そして，業界標準をベースとしたガイドラインを設計し，開発を行い，製品などのデジタル環境を融合する取り組みを行っている．標準化対象はコンテンツのフォーマットである．

下位層では，UPnP を利用することを前提としている．その下位層である UPnP は，ネットワークに接続される各種機器の検出と各種の設定を自動化 (zeroconf) し，おもに一般家庭で利用しやすいネットワークプラットフォームを提供するための技術である．

OSGi は，Jini[16] や UPnP，IEEE1394 上の HAVi(HAVi に関しては後述する) などの PC 機器や AV 家電を接続するための様々なインターフェース間の相互変換の実現を目指していた．現在は，Java ベースの汎用的なソフトウェア部品化技術を策定し，携帯電話や自動車制御などのテレマティクスの領域での利用が増加している．

HAVi の目的はホームネットワークに接続された AV 機器を分散コンピューティングのプラットフォームとみなし，この上で動作する分散アプリケーションがそれぞれ協調してタスクを実行する環境を提供することである [14]．また，異なった仕様の製品間の相互接続および相互運用を実現する環境でもある．そして，HAVi のアーキテクチャの特徴は，こうしたサービスを提供するソフトウェアを定義したミドルウェアであり，分散アプリケーションを開発する API セットの提供である．

ECHONET は，外部の社会システムと宅内機器，センサなどの連動に寄る状態監視，計測，制御によるサービスを実現する．特徴として 4 点述べる．まず，配送不要な無線や電灯線などの多様な媒体をネットワークとして選択でき，施設への設置が容易にできる点である．また，システム構成要素のオブジェクト指向モデル化を採用しているおかげで，機器制御を「機器オブジェクトのプロパティの設定の形で定式化でき，機器開発，相互接続保証が容易である．さらに，ミドルウェアの API 定義もまた行っており，アプリケーション開発を容易にしている．

最後に ECHONET 機器と DLNA 機器との相互接続の実装も進められている点である．

Bonjour では，代表的な機能としてサービスの自動探索を提供している．Bonjour の特色は，デバイスではなくサービスに探索を中心に据えている点である．特定のデバイスが提供しているサービスをとらえるのではなく，目的にしているサービスを提供しているデバイスを特定するのである．例えば，FTP サービスを提供しているサーバを特定し，そのサーバを通知する．

これまで，グローバルコンピューティングに関連した技術の現状を述べてきた．各プロトコルが電子機器をつなぐ機能を実装し，電子機器 – 電子機器，電子機器 – コンテンツなどのネットワークを形成し，ネットワークを介して様々なサービスを提供する環境を構築している．そして，それぞれのプロトコルが独立で動いている場合もあれば，プロトコル間の相互接続性や相互運用を実装し，提供しているプロトコルも存在する．

次章では，これら既存する技術を考慮し，グローバルコンピューティング実現に向けた必要機能を割り出し，現状と課題に関して議論していく．

第3章
グローバルコンピューティング実現のための現状と課題

第２章では，グローバルコンピューティングの構想に関して述べてきた．本章では，現在のグローバルコンピューティングを実現する上でどのような機能が必要なのか，そして課題となることを既存のミドルウェアの仕様を交えながら述べる．機器をネットワークに接続し，利用できるようにするためには４つの機能が必要である．1. ホスト発見機能，2. ホスト分析機能，3. ホスト情報提示・提供機能，4. ホスト管理・制御機能である．

1. **ホスト発見機能**
   自律的にホスト発見

2. **ホスト分析機能・ホストアクセス制御**
   ホスト情報の取得（機器の種類，通信に必要な情報），ホスト情報の保持，アクセス制御

3. **ホスト情報提示・提供機能**
   情報提示アプリケーション（クライアント），情報提供用のフォーマット・トランザクション処理

4. **ホスト管理・制御，相互作用機能**
   ホストのパラメータ（状態や使用できるサービス）管理，ホスト間相互作用

ネットワークに機器が接続したことを把握できなければ，どのような機器を利用できるのかユーザは知ることができない．そのため自律的にホストを発見する

ことが第一段階となる．次に，発見したホストがどのようなサービスを提供できるか，システム上ホストと通信するためにはホスト情報が必要である．それが，ホスト情報分析である．加えて，利用して問題ないホストなのか，ネットワークに障害をもたらすホストかなどネットワークへの認証なども必要である．それがホストアクセス制御である．これらが第二段階である．ホストを発見することができ，アクセス制御を終え，必要なホスト情報を取得した段階で，次に利用できる機器，サービスの提示が必要である．提示する機能がなければユーザとの接点が存在せず，利用できるものの認識ができなくなる．そのためホストの提示を行う．それが第三段階である．第三段階を終えて初めてユーザは機器の制御や管理，機器間相互作用を行うことができる．そして機器のパラメータを管理し，ユーザが加えた変更や，ユーザが利用するサービスを通知する機能が必要である．それが第四段階である．これら４つの機能がなければグローバルコンピューティングは実現されない．

3.1. グローバルコンピューティング環境を構成する機能

グローバルコンピューティングにおけるコントロールポイントとはホスト情報を保持し，ホストの状態の更新を適宜検知し保持する．新たにネットワークにアクセスしてきたホストを自律的に検知し，そのホスト情報を保持する．そして，クライアントアプリケーションからホスト情報の送信要求があった場合は，その都度保持しているホスト情報をクライアントアプリケーションへ送信する．さらに，クライアントアプリケーションからホスト間通信の要求があった場合は，対象となるホスト同士を通信させる機構である．自律的にホストを検知するとは，常に新たにネットワークにアクセスしたホストがいないか監視し，アクセスがあった場合はそのホストの情報を取得する．

ホストの設定や管理，制御，ホスト間の通信を行うためにはまず第一に IP ネットワークにどのホストがアクセスしているのか把握しなければならない．そして，それはリアルタイムにアクセスしてきた機器を把握できなければならない．さらにリアルタイムホスト発見ではスケーラビリティを考慮に入れなければならない．

スケーラビリティとはここでは各種ネットワークへの適応性である．現在のネットワークは複雑である．広帯域ネットワークやマルチホーム，二重 NAT や VPN と NAT の組合わさったものなど多岐にわたる．そのためこうした環境に適応できることが課題に挙げられる．

第２段階はホスト情報の取得とアクセス制御である．ホストの種類がわからなければ，その仕様にあわせて適切な設定などを行えない．そのため，アクセスしてきたホストを自律的に発見すると同時にホストを分析してどの種類のホストなのかや後に通信を行う時に必要な情報を取得する必要がある．加えて，アクセス制御も要求される．不正をおこなうホストや誰もが使用してよいホストなどをホスト情報取得の段階で分析をし，適宜アクセスを制御できることが必要である．ユーザに対して提示できない機器である場合は，ユーザに提示しないようにホスト情報を加える必要がある．また，不正なホストの場合はアクセスを制限するなどの措置も必要である．

第３段階が情報の提供と提示である．これら取得されたデータはホストの設定や管理，制御，ホスト間の通信を行うためのユーザーが使用するクライアントアプリケーションやコントロールポイントへとデータが提供されなければならない．クライアントアプリケーションやコントロールポイントへデータを提供するためのフォーマットや実際にデータをどのような処理で提供するのかなどの機能を有することになる．また，実際にユーザに対して利用できるホスト情報を提示する機能も有する．そして，第３段階が実現されて初めてホストの設定や，制御などが実現できる．

第４段階では，ホスト間の相互作用の通信でデータの交換や，ホスト利用できる状態かなどのパラメータ，ホストの状態が変化した時のコントロールポイントへの通知が必要である．情報交換するにあたって，どのような情報交換なのかの定義が必要となる．さらに，ホスト間での通信やコントロールポイントとホスト間での通信において IPv4 を利用する場合などは NAT 越えの課題がある．WAN 側が NAT の外側，NAT によってアドレスがローカルアドレスに変更されて通信される LAN 側を NAT の内側とすると，NAT の内側からの通信は可能だが，外側からの通信ができない．したがって NAT 越えの機能も必要となる．

3.2. 関連技術の現状と課題

3.2 ではグローバルコンピューティングを実現するために，４つの機能に該当する関連技術の仕様に関して議論していく．議論の中で関連技術に不足する機能を挙げ，グローバルコンピューティング実現のための課題に関して議論する．

3.2.1 Universal Plug and Play

UPnP は機器のアドレス設定として DHCP(Dynamic Host Configuration Protocol) を使用している [17]．DHCP によってアドレス設定が完了すると SSDP(Simple Service Discovery Protocol)[18] を使って，ホストがコントロールポイントへマルチキャストで通知する手法をとっている．これがホスト発見の機能である．DHCP が使用されていないネットワーク下では Auto-IP[19] という技術が使われる．Auto-IP では，ホストがリンクローカルアドレスのうち（169.254.1.0 から 169.254.254.255) 自分が使いたい IP アドレスを 1 つ決めて使用する．リンクローカルアドレスが決定されると同様にホストが自身がネットワークへアクセスしたことをマルチキャストでコントロールポイントへ通知する．ホスト発見の後，ホストは HTTP(Hyper Text Transfer Protocol) を使ってホスト詳細情報をコントロールポイントへ通知する．取得後，ホスト情報提示もまた HTTP を使って情報をコントロールポイントと交換する．取得した URI にアクセスすることで使用したいホストの情報を見ることができる．ホスト情報提供では，XML(Extensible Markup Language) と SOAP(Simple Object Aceess Protocol)，その他に GENA(General Event Notification Architecture) と組み合わせて HTTP を使用した場合と，ホスト提示やホスト発見の時のように HTTP のみで行われるようになっている．ホスト情報管理・制御では XML で定義しているフォーマットに沿って，ホストの状態が変化した時に変化した値をコントロールポイントへ通知している．UPnP の概略は以上である．UPnP の仕様書 Device Architecture[20] を参照した．

UPnP の課題となることに関して述べる．まず，ホストのアドレス設定の手法が不足している．グローバルコンピューティングはインターネットを介して世界中のホストを利用できる環境である．そのためグローバル IP を使用する．しか

し，UPnP は DHCP しか対応していない．IPv4 のアドレス設定手法には PPP[21] といった手法もある．加えて，IPv4 枯渇問題が深刻になってきている [22] 現状とグローバルコンピューティングが対象とする機器の多さを考慮すると IPv6 への対応が課題である．もちろん，UPnP は IPv6 にも対応している [23]．しかし，RA(Router Advertisement)[24] のみ対応している．あまり多く使用されていないが，DHCPv6[25] や 6LoWPAN[26] もあり，各アドレス設定の手法に対応しなければならないという課題が UPnP にはある．また，HTTP を使用しているためホスト情報を容易に閲覧することが可能である．そのため，不正に閲覧することも容易に可能である．セキュリティ面で課題を有している．

3.2.2 Bonjour

Bonjour[3] のアドレス設定手法は DHCP を使用している．UPnP 同様，DHCP を使用していないネットワークの時は Auto-IP を使ってリンクローカルアドレスを代わりに用いる仕様となっている．Bonjour のホスト情報提供とホストの管理・制御は，multicastDNS(mDNS) が使用されている．利用できるサービスを探索する時に使われている技術である．探索しているサービスと関連するホストを一覧できるリストを得るためにここでは，ホストが mDNS のクエリを送信する，適合するサービスがホストの名前と共に応答がかえる仕組みとなっている．具体的には Service Record(SRV)，Text Record(TXT)，Pointer Record(PTR) の 3 つある．SRV の登録情報は，サービスの名前を保持する．そして，サービスを実際に使用するときに，DNS を使って，ホストの名前とポート番号を取得する．PTR はサービスの発見を担っている．サービスのリストにマッピングすることによってそれを実行する．TXT は SRV と一致するサービスの名前を保持している．TXT は複数のホストを使用するときに，使用されている．そして，特徴としてポート番号を動的に割り振って使用できるよう設計されている．情報提示は PC 上の GUI を使用している．以上は Bonjour の仕様書を参照した [27].

Bonjour もまた，UPnP と同様で IP アドレス設定の手法への対応が不足していることが課題である．IPv6 にも対応しているが，DHCPv6 や 6LoWPAN などへは対応していない．また，IPv4 も PPP には対応していない．加えて，mDNS と

いう独自の仕様を採用しており，基本的にはLAN内を想定した技術である．そのため，LANだけでなくWANを前提としているグローバルコンピューティングに応用する場合は機能が不足している．

3.2.3 Home Audio/Video Interoperability

HAViはIEEE1394上で機器を接続し，設定をすることなく家庭内のAV機器を使えるようにした技術である．JAVAで提供されているのでOS依存しない特徴をもっている．しかし，設定することなく機器を接続するだけで使用できる環境はグローバルコンピューティングに類似する．したがって，ホスト情報管理・制御などを示す．ホスト情報制御・管理では，HAViはEvent Managerと呼ばれる機能で，ホストの管理を行っている．あるソフトウェアの状態が変化したときに，そのソフトウェア自身が他のソフトウェアに通知できる仕様となっている．機器の利用できるサービスはRegistryによって管理されている．ソフトウェア内にあるデータベースにソフトウェアエレメントのSEIDとその属性情報が登録されていて，Registryがそのデータベースを管理している．HAViの問題点はインターネット接続の環境を提供していない問題がある．それを補完するために，UPnPの相互互換が実装されている．しかし，UPnPにはIPアドレスの設定の各手法への対応など問題がある．また，HAViはAV機器に焦点を当てているため，グローバルコンピューティングが想定する機器すべてには対応することができないという問題がある．

3.3. グローバルコンピューティング環境の実現に向けて

3.2ではグローバルコンピューティングに関連する技術の現状と課題に関して述べてきた．グローバルコンピューティングは，ホームネットワークだけではなくIPネットワークを介してネットワークに接続されたすべてのホストを管理や設定，制御，ホスト間の通信を行うための基盤環境である．ホームネットワーク内だけでなく，他のIPネットワーク，例えば異なるAS(Autonomous System)間，セグメント間等でホスト間通信もおこなえること想定した環境である．しかし，現在

のホームネットワークを対象とした既存技術では，グローバルコンピューティングを実現する上で機能が不足していると言える．例えば，UPnP では DHCP を使用することを前提としている．DHCP が使用できない場合はリンクローカルアドレスを使用する仕様となっている．リンクローカルアドレスを使用することは，ホームネットワーク内で使用する場合は問題はない．しかし，グローバルコンピューティングのようにあらゆるネットワーク上のホストを使用する場合は適していない．リンクローカルアドレスはルータを原則として通過することはない．LAN 内だけで使用されるアドレスだからである．また，IPv4 ネットワークに関してだが，DHCP だけがアドレス設定の手法ではない．家庭では PPPoE(Point-to-Point over Ethernet) を使用している．IPv6 ネットワークに関しても，“UPnP Device Architecture V1.0 Annex A - IP Version 6 Support”[23] によれば，UPnP などは RA(Router Advertisement) を前提としている． しかし，IPv6 のアドレス設定の手法は RA だけではない．ステートフルアドレッシングである DHCPv6 を使用したアドレス設定の手法もある．グローバルコンピューティングを実現する上で各ホストがリンクローカルアドレスではなく，グローバル IP を使用して接続されていなければならない．したがって，既存する技術ではアドレス設定手法が DHCP 以外の環境ではグローバルコンピューティングを実現できない．他のアドレス設定手法を持つネットワークにも適応できる基盤システムを開発する必要がある．

グローバルコンピューティングにおいて IP アドレス設定を行い，コントロールポイント・クライアントアプリケーションが，アクセスしたホストを認識する段階はホスト発見である．よって，各アドレス設定手法をもつネットワークに適応できるような手法を使ってホスト発見機構を設計，実装する．そして，グローバルコンピューティングに適したホスト発見に合わせて，ホスト情報分析，ホスト情報提示・提供を同様に設計，実装する．

次章ではグローバルコンピューティングに適したホスト発見，ホスト情報分析，ホスト情報提示・提供を使ったホスト情報取得および可視化システムとして提案を行う．

第4章
ホスト情報取得および可視化システムの提案

本章においてグローバルコンピューティングを実現する上で必要な基盤システムとしてホスト情報取得および可視化システムの提案を行う．第3章で述べたように，ネットワークによってホストへのアドレス設定の手法は複数あった．そして，その各手法に適用するリアルタイムにホストを検知するシステムがグローバルコンピューティングには必要である．本研究はこの問題を解決するためにホスト情報取得および可視化システムを提案する．

4.1. ホスト情報取得および可視化システムの概要

ホスト情報取得および可視化システムは，グローバルコンピューティングにおけるホスト発見機能，ホスト分析機能・アクセス制御，ホスト情報提示・提供機能までの実装である．クライアントアプリケーションへ表示されるものは，実際にネットワークに接続されているホストをリアルタイムに表示する機能である．一見するとホストをリアルタイムに可視化するためのツールでしかない．一種のネットワーク管理ツールである．しかし，システムの仕様はグローバルコンピューティングを実現するために必要な機能を有したホスト情報可視化システムなのが，本研究で提案するものである．必要な機能は，自律的にホストを発見し，ホスト情報を取得する．ホスト情報に加えてホストの現在の状態を示すパラメータを有している．自律的にホストを発見するとは，常に新たにネットワークにアクセスしたホストがいないか監視し，アクセスがあった場合はそのホストの情報を取得

することである．また，クライアントアプリケーションにデータを送信するフォーマットや各ホストが状態に変更があった場合に状態変化を通知するフォーマット，さらにクライアントとデータを交換する時のトランザクション処理である．さらに詳細に述べると，ホスト発見ではリアルタイムにホストを発見する．それは，IPv4 ネットワークでも IPv6 ネットワークの両者に対応し各ホストを発見するシステムである．ホスト分析では，各ネットワークにおけるアドレス設定手法に適応し，各環境下でグローバルコンピューティングに必要な情報を取得できる機能を有している．具体的には，ホスト間通信に必要な IP アドレスや MAC アドレスである．さらに，ホストの種類が判定できる情報などである．ホスト情報提示・提供でも同様にコントロールポイントからクライアントアプリケーションへホスト情報を送信できるだけでなく，ホスト管理できることを想定して設計する．UPnP が各ホストの状態を把握するために実装したパラメータの交換するための仕組みのように，グローバルコンピューティングでもそういったことを想定したインタフェースを有するシステムである．このように，本研究におけるホスト情報および可視化システムはグローバルコンピューティングの基盤システムとなる機能を有したホスト情報および可視化システムである．

ここで，ホスト発見に関しての手法を検討する．ホスト発見の手法によって，ホスト分析，ホスト情報提示・提供の設計が異なるものとなるからである．例えば，第 3 章で述べたように，UPnP や Bonjour のクライアントからの通知の手法を採用するならばコントロールポイントを中心として各ホスト毎にコントロールポイントが情報交換を行ってホスト情報の取得をおこなう．これを行うためには各ホストもまた，高機能なクライアントシステムを有していなければならない．各ホストが状態変化したときの通知など，クライアントであるホストが主体となってコントロールポイントとのコミュニケーションをとらなければならない．逆に，コントロールポイントが主体的にホストを発見しホスト情報を取得する場合はクライアントであるホストはコントロールポイントの要求に応える機能をもてばよい．ホストは高機能でなくともコントロールポイントが主体となって管理することができる．例えば，ネットワークトラヒックの監視からホストを発見し，同時にそのホストの IP アドレスを取得しておく．取得した IP アドレスを使ってその

表 4.1 既存のホスト発見手法

能動的手法	受動的手法
クライアントに対してユニキャスト	クライアント側からのアドバタイズ
マルチキャスト（ブロードキャスト含む）	ネットワークトラヒック監視
	IP アドレス設定のサーバとの連携

ホストへ状態を伝える応答を要求すればよい．ホストの発見の手法を決めなければ，他の機能の仕様を決められない．従って，既存のホスト発見の手法を検討する必要がある．

4.2. ホスト発見手法の検討

4.1 で述べたように，ホスト発見の手法によって他の機能の仕様が異なるものになる．そこで，ホスト発見の手法に関して議論する．

サーバ・クライアントモデルで考えると，サーバ側からみてサーバが能動的にホストを見つけにいく手法と，受動的にホストを見つける手法と大きく分けて 2 つに分けられる．能動的な手法はサーバ側が行動をし，見つける．受動的な手法とはクライアント側からの行動によっての発見やトラヒックの監視などである．能動的な手法をさらに分けると，サーバがユニキャストして見つける手法とマルチキャスト（ブロードキャストを含む）して見つける手法がある．ユニキャストではネットワークで使用されている IP アドレスへ ping ツールなどのように，アドレスを順に指定してすべてへ送信する．マルチキャストでは同様にネットワークで使用されている IP アドレスから任意に指定して送信する．それぞれサーバからのパケットを受け取ったクライアントはネットワーク接続していることを通知するパケットを応答する．クライアントが数が非常に多い場合はサーバ側で輻輳が発生して，パケットが破棄される場合があるが，TCP が使用しているスロースタートアルゴリズムを利用して対策することができる．受動的な手法は，クラ

イアント側から通知するクライアントからの通知手法やネットワークトラヒックの監視である．クライアントからの通知では，クライアントがサーバへ送ったパケットを傍受することによってサーバが，ホストが新たに接続されたことを知ることができる．しかし，コントロールポイントに接続する前のクライアントは完全にブートしている状態とは言えない場合があり，コントロールポイントに十分な応答ができない場合がある．例えば，ROMMON で tftp ブートする場合である．また，また、コントロールポイントをどうやって発見するか問題が残っている。クライアントはコントロールポイントの IP アドレスをあらかじめ設定して把握させるか，あるいはブロードキャストなどで送信することになる．IP アドレスをあらかじめ設定する場合は，ユーザに初期設定を強いることになる．また，ブロードキャストの場合はコントロールポイントが別のセグメントに存在する場合はルータにリレーさせなければならない．そのためルータにあらかじめそういった機能を持たせる必要がある．ネットワークトラヒック監視では L2 スイッチから流れてくるデータを取得してパケットの解析を行いホストを発見する方法がある．さらに，IP アドレス設定手法のサーバ側と連携することでホストを発見することもできる．ホストを管理しているメモリやログファイルなどと連携してホスト情報を取得することでリアルタイム発見が可能である．以下，各手法に関連する手法を例に挙げながら議論し，グローバルコンピューティングに適したホスト発見の手法を検討する．

1. クライアントに対してユニキャスト
   図 4.1 で示したように，サーバ側からパケットを送信し，クライアント側がそのパケットに応答する手法が，このユニキャスト手法である．この手法を使用する場合は ICMP(Internet Control Message Protocol)[28] を用いることができる．ICMP はネットワークの通信の情報やエラーなどを通知するときに使用するプロトコルである．このプロトコルを用いて，ping[29] ツールなどがホスト発見に使用することができる．ping コマンドの後に IP アドレスを指定して実行することでホストとの通信が可能かどうかのメッセージを受信することができる．この原理で，ネットワーク内で使用されている IP アドレスのすべてへ送信すればホストを発見できることができる．こ

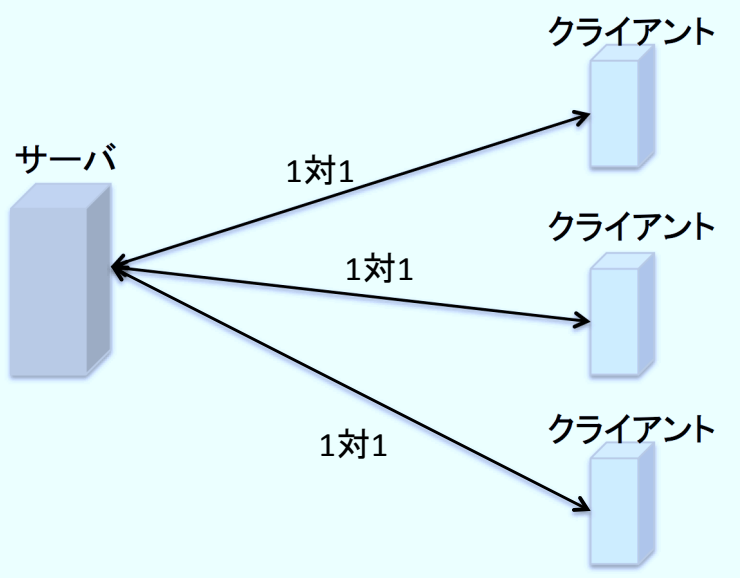


図 4.1 ユニキャスト手法

の手法は各アドレスにパケットを送信して確認するため，処理しなければならないパケットの数がアドレス空間の広さに比例して増加するとともに，ネットワークトラヒックも増加するという問題点がある．さらに，ネットワーク内で使用されている IP アドレスのすべてへ送信してホストを発見するこの手法は，事実上 IPv6 では使用できないことになる．IPv6 ではアドレスの多さや，モバイル IPv6 のように，ホストが別のネットワークへ移動した場合も同一のアドレスを使用することができるからである．

2. マルチキャスト（ブロードキャスト含む）
   図 4.2 で示したように，ネットワークへパケットを複数送信し，受信したクライアントがそのパケットへ応答する手法である．この手法は Bonjour などで使用されている mDNS(multicast DNS) で用いられている．名前解決したいホスト名を UDP でポート番号 5353 に対してネットワークにマルチキャストする．該当するホストはそのパケットに対して応答し，名前解決をする手法である．この原理を応用してネットワークへマルチキャストしてクライアントからの応答を受信することでホストを発見することができ

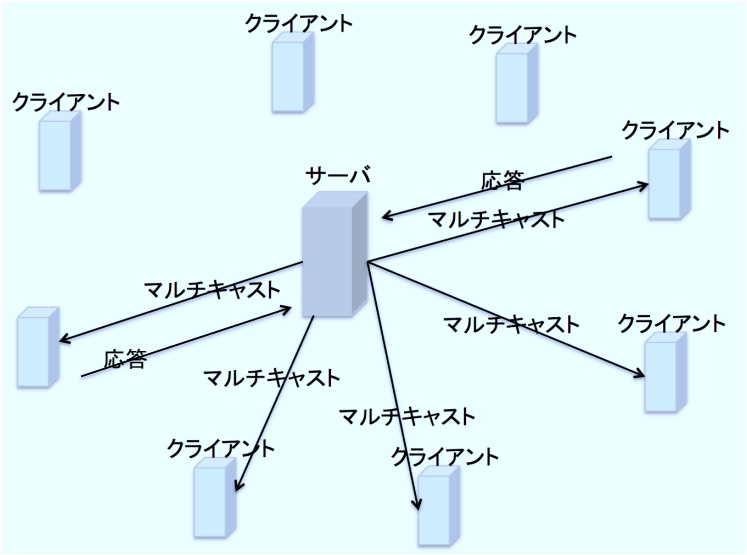


図 4.2 マルチキャストの手法

る．実際に DHCP では DHCP クライアントがネットワークへブロードキャストし，DHCP サーバを探索している．この手法の問題点はリアルタイムにホストを発見することができないことである．複数の IP アドレスへマルチキャストしてホストを発見する場合，絶えずパケットを送信し続けなければならない．想定する送信間隔は数秒間隔から数分間隔まで考えることができるが，リアルタイム性を重視すると送信間隔は短くなっていく．その場合は，不必要なパケットが送信される可能性が高いためネットワークへ負荷をかけることになる．また，同じ IP アドレスを使用して別のホストがネットワークにアクセスした場合を考慮すると，前のホストの IP アドレスをリースする時間より短い時間でホストが変わっていないか確認する処理なども発生する．

3. クライアント側からの通知
図 4.3 で示した手法である．クライアントがネットワークへアクセスすると同時にサーバへネットワークへアクセスしたことを通知するパケットを送信し，サーバ側はそのパケットを受信後新たにクライアントがネットワー

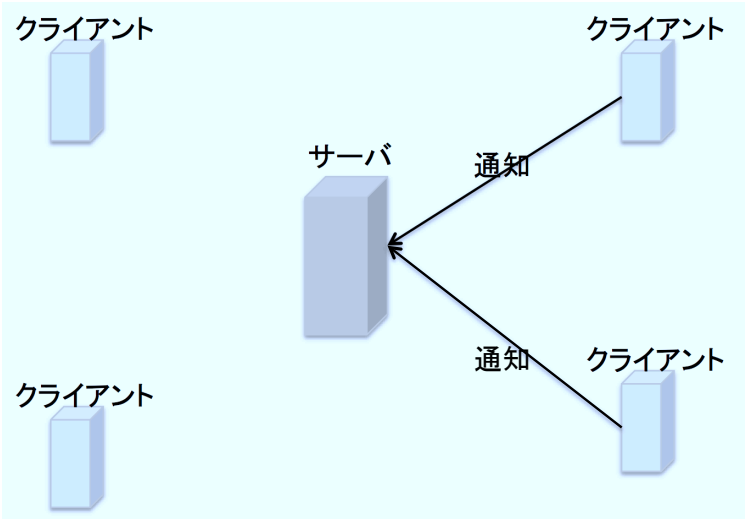


図 4.3 クライアント通知の手法

クにアクセスしたことを知ることができる手法である．この手法は UPnP などで使用されている．この手法の問題点として，設定が必要となることである．UPnP などの例のようにあらかじめ，クライアントであるホストが定義されているフォーマットにしたがってそのホスト自身のパラメータをクライアントが保持していなければならない．したがって，zeroconf な環境ですべてのホストを自律的に通信し合う環境を構築するグローバルコンピューティングにおいて適さない．

4. ネットワークトラヒック監視
   図 4.4 で示した手法である．この手法はネットワーク管理ツールでホスト発見手法として多く使用されている．一般的なツールとして IDS(Intrusion Detection System) があげられる．パケットの情報を監視し，不正侵入を見抜き，そしてネットワーク管理者に通知するシステムである．その他のネットワークツールとして NetGrok[30] がある．NetGrok はネットワークトラヒックからホストの IP やそのホストの使用帯域，ホスト間の接続を可視化するシステムである．ネットワークトラヒックの監視からホスト情報を取得する場合はリアルタイムにネットワークトラヒックの監視ができるため，

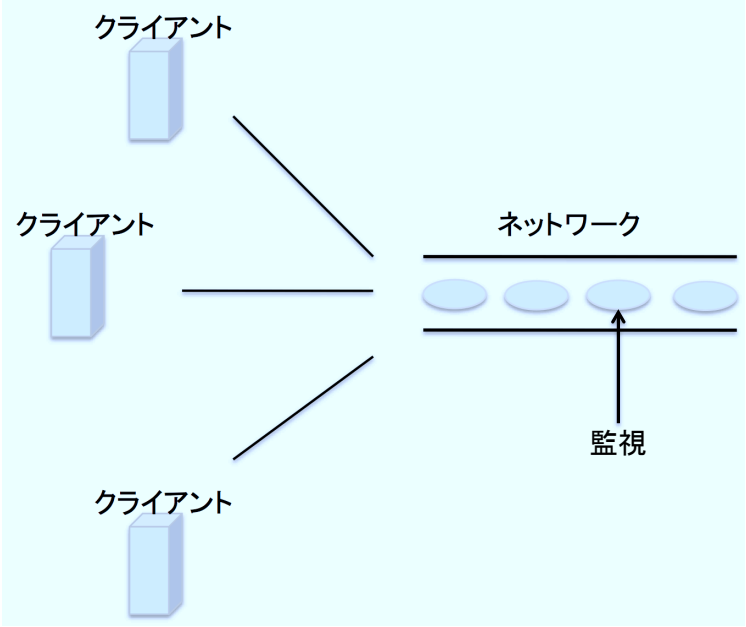


図 4.4 パケット監視の手法

ホスト情報もリアルタイムに取得することができる．また，ブラックボックスとなっているルータでも適用可能などを考慮すると様々なネットワークへの適応性は高い．この手法の問題点として，ネットワークトラヒックの監視を行うためには各ネットワークの管理者の許可が必要である．そのため，ネットワークによっては設置ができない場合もある．場合に寄っては拡張性を損なう場合がある．

5. IP アドレス設定のサーバとの連携

図 4.5 にて，IP アドレス設定のサーバとの連携の手法を示した．この手法を使ってホスト情報を取得している．ツールの一つとして NetReg[31] があげられる．初めてネットワークにアクセスしてきた DHCP クライアントの情報を取得して，データベースに登録しホスト情報を保持するツールである．ネットワークに障害が発生した場合などに使用される．具体的なホスト情報の収集は DHCP の Request メッセージがクライアントから送信されたときにルータが DHCP Request メッセージパケットを NetReg のサーバ

に中継し，ホスト情報を取得している．実際にはNetRegはクライアントのMACアドレスとホスト名を取得して，認証に利用している．そのため厳密にはIPアドレスなどは取得していない．しかし，DHCPACKもルータに中継させればIPアドレスも取得することができる．この手法を使用すれば，ホスト情報をリアルタイムに取得することができる．NetRegの手法以外にも，例えばDHCPサーバがメモリに保持しているホスト情報やDHCPログファイルの情報をリアルタイムに参照することによってホストを発見することができる．また，PPPoEなども同様でサービスを行っている機器と連携することによってホスト情報をリアルタイムに取得することができる．この手法の問題点として，ネットワークのアドレス設定の手法の違いによって連携するサーバが異なるため，各ネットワークへの適応性が著しく損なわれることである．DHCPの場合はDHCPサーバと連携しなければならない．加えてIPv6への対応が難しい．DHCPv6のアドレス設定手法を採用しているネットワークなら連携しやすいが，RA(Router Advertisement)の場合は，ルータとホストと連携しなければならない．このように各種ネットワークによって導入の設定などが大きく異なることになる．スケーラビリティが問題になる．

これまでにホスト発見手法に関して議論してきた．各手法には問題点があったが，各手法の特徴を考慮し，グローバルコンピューティングインフラストラクチャとして適したホスト情報発見手法を決定する議論を行う．

4.3. グローバルコンピューティングに適したホスト情報取得手法

グローバルコンピューティングではすべての機器がネットワークに接続される環境である．そして，ネットワークを介して制御や管理，ホスト間の自律的な通信を行うための環境である．そのためには，グローバルIPアドレスが必要である．現在のIPアドレスはIPv4とIPv6の２つある．現在では，IPv4枯渇問題[22]

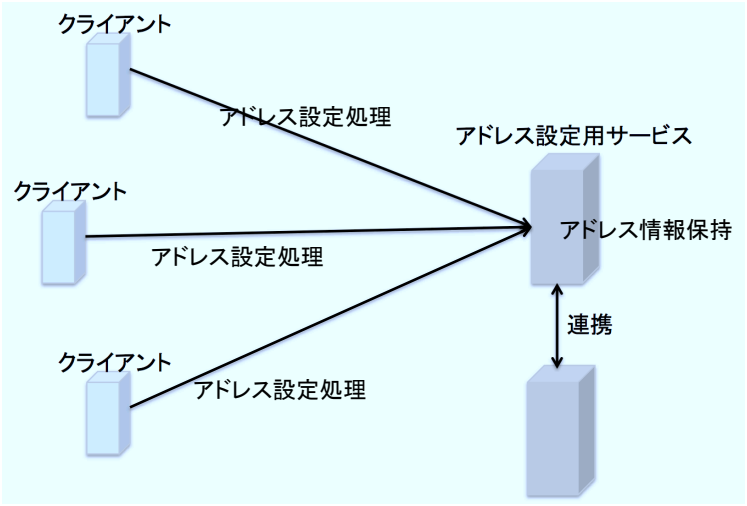


図 4.5 アドレス設定サーバとの連携手法

が深刻になりつつある．IPv4 アドレスはあと残り 2%だと言われている．すべての機器をネットワークに接続するためには，IPv4 アドレス枯渇問題を考慮すると，グローバルコンピューティングでは IPv4 と IPv6 の両者に対応したホスト情報取得が求められている．

IPv4 と IPv6 の両者のネットワークと対応した手法であることを考慮しながら，4.2 の手法のうち，どの手法を使用するか決定する．まず，ユニキャストについて考える．ユニキャストでは IPv4，IPv6 両者で対応している．しかし，ホストを探すための手法としては効率が悪い．ユニキャストでホストを発見するためには，サーバが属するネットワークのアドレス一つ一つにパケットを送信し，応答してもらう必要があるが非効率でありスケーラビリティ上の問題がある．ホスト発見手法としては非効率を得る．次に，マルチキャストについて考える．マルチキャストでもまた IPv4 と IPv6 の両者に対応している．しかし，リアルタイムにホストを発見するのに問題点がある．リアルタイムにネットワークにアクセスするホストを発見するためには，絶えずマルチキャストにおけるパケット送信を続ける必要がある．一度発見したホストがネットワークから切断される可能性もあるので，ホストをリアルタイムに発見するために一度送信したアドレスにも送信し続

けなければならない．したがって無駄な処理が多いためにユニキャスト同様，非効率であり，ホスト発見に適さない．続いて，クライアント側からの通知の手法である．この手法も同様にIPv4とIPv6の両者対応している．この手法は多くのネットワーク技術で使われている手法である．DHCPやRA，DHCPv6などはクライアント側からの通知が一般的であり，その他多岐にわたったて使用されている手法である．グローバルコンピューティングを行う上でも有効な手段の一つと言える．ホストがネットワークへアクセスした時に，コントロールポイントへ通知を行えばリアルタイムにホストを発見することができる．また，ホスト情報を同時に送信してコントロールポイントへ通知できるので条件を満たす．したがって他に適する手法と比較する必要がある．次にネットワークトラヒックの監視手法について考える．ネットワークトラヒックの監視手法もまた，IPv4，IPv6に対応できる．各アドレス設定のパケットを取得することでリアルタイムにホストの検出が可能である．この手法もまた，グローバルコンピューティングに必要なホスト発見を行うことができる．したがって，クライアント通知と比較する必要がある．最後に，アドレス設定用のサービスとの連携手法である．この手法は，一つのアドレス設定手法を対象とする場合は，十分な手法である．例えば，DHCPならばDHCPを使用しているネットワークで自律的にホストを発見することができる．しかし，アドレス設定の手法はIPv4とIPv6を合わせるといくつもある．DHCPv4やPPP，IPv6ならばDHCPv6やRAなどである．したがって，各ネットワークへの適応性が低いため，グローバルコンピューティングには適さない．グローバルコンピューティングにおけるホスト発見手法はクライアントからの通知手法か，ネットワークトラヒックの監視手法である．

クライアントからの通知手法の場合はコントロールポイントのIPアドレスをあらかじめ設定している場合はユニキャストで通知することができる．しかし，この場合は設定を強いるので適さない．ユニキャストではない場合はマルチキャストないしはブロードキャストでコントロールポイントへ通知することができる．通知を受け取ったコントロールポイントがクライアントに対して，コントロールポイントのIPアドレスを返信することで，クライアントはコントロールポイントのIPアドレスを有する．マルチキャストやブロードキャストは通常UDPが使

用される．DAD(Duplicate Address Detection) は UDP で 3 回ほど同一のパケットを送信する手法を採用している．パケットロスなどによってコントロールポイントに通知が届かない場合はクライアント側は時間をおいて再送信するのが通常の手法である．しかし，仮に何らかの理由で数回続けてパケットロス，パケットドロップが発生した場合には，クライアント側が，コントロールポイントはネットワーク上にないという判断を下すことになる．加えて，クライアントからの通知手法はスケーラビリティを損なうことになる．クライアントが増加するとコントロールポイントへ送信されるパケット量が増加する．そのため，サーバ側で処理が限界に達する場合も考えられ，スケーラビリティが低いという問題がある．クライアント側の通知手法はこのような問題を抱えている．

反対にネットワークトラヒックの監視手法の場合はパケットドロップの可能性がある．NIC でパケットがバッファメモリのオーバーフローする可能性がある．しかし，ネットワークトラヒックの監視手法では，広帯域なネットワーク環境でなければ，既存の一般的な PC でパケットドロップすることなくパケットの取得が可能である．また，取得するパケットを変更するだけで他のアドレス設定手法に対応できる．拡張性が高い利点がある．こうした観点からクライアント側から通知手法よりも問題を回避できる可能性が高い．したがって，グローバルコンピューティングとしてのホストのリアルタイム検知はネットワークトラヒックの監視手法を使用する．

4.4. ホスト情報取得および可視化システム提案のまとめ

グローバルコンピューティングとしてのホスト情報取得および可視化システムの提案をした．そしてネットワークにアクセスしたホストをリアルタイムに検知するための手法を検討した．ホスト発見手法には，クライアントへユニキャスト，クライアントへマルチキャスト（ブロードキャストを含む），クライアント側からの通知，ネットワークトラヒック監視，IP アドレス設定のサーバとの連携があった．各手法の問題点を検討して，ネットワークトラヒックの監視手法からリアル

タイムにホストを検知する手法を採用した．次章では，この手法をもとに具体的なシステムの設計を行う．その設計はグローバルコンピューティングの基盤システムとして考慮した設計となる．

第5章
ホスト情報取得および可視化システムの設計

本章では，グローバルコンピューティングを想定したホスト発見機能，ホスト分析機能，ホスト情報提示・提供の機能の設計を行う．ホスト発見から，必要情報を取得するホスト分析，そして取得された情報を提供するホスト情報提示・提供の設計である．

5.1. システム要件

システム要件に関して議論する．本研究ではグローバルコンピューティングのインフラストラクチャとしてホスト発見機能，ホスト分析機能，ホスト情報提示・提供を設計する．以下，各機能ごとでシステム要件をまとめる．

1. ホスト発見のシステム要件
   各ネットワークのIPアドレスの設定手法に適用できるインターフェースを持つことやIPv4ネットワーク，IPv6ネットワーク両者に対応させること．

2. ホスト分析のシステム要件
   各ホスト毎に通信できる情報．ホストの種類の把握が可能な情報

3. ホスト情報提示・提供のシステム要件
   クライアントアプリケーション対してホスト情報の提供方法．ホスト管理・制御の段階も見越した，メッセージの設計

表 5.1 現在のアドレス設定手法

IPv4	IPv6
DHCP	DHCPv6
PPPoE	RA

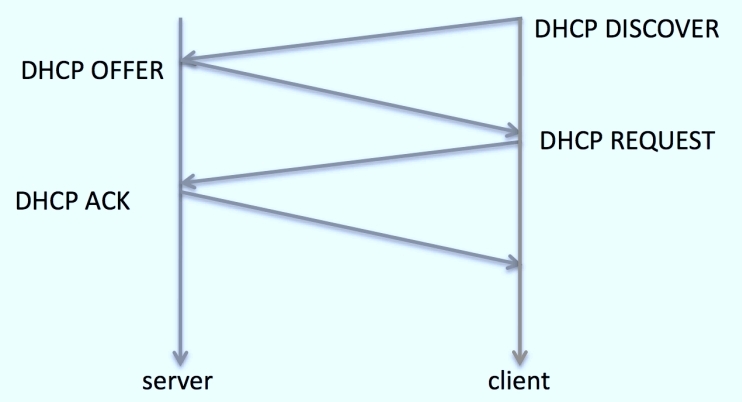


図 5.1 DHCP のトランザクション

5.1.1 ホスト発見機構の設計

ホスト発見の手法としてパケットキャプチャを採用した．したがって，IP アドレスを設定するための各ネットワークの環境をまとめる．zeroconf のもとに IP アドレスを配布する制御システムはいくつかある．IPv4，IPv6 で分けて考えると，表 5.1 である．

まず，DHCPv4 は最も一般的なアドレス設定の機構である．クライアント側から Discover メッセージを送信し，サーバと少なくとも 4 回メッセージを交換し，IP アドレスを取得する．DHCP は多くのネットワークで使用されている IP アドレス設定手法である．

PPPoE は家庭環境で IP アドレスを取得する際に多く使われている．PPPoE は Ethernet 上で PPP を使用してアドレス設定をホストに行うプロトコルである．PPP は LCP(Link Control Protocol) と NCP(Network Control Protocol) がある．

LCP では認証を行う．そして PPP セッションが確立される．その後，NCP でアドレスの設定がなされる．実際にアドレスを設定するプロトコルは IPCP(Internet Protocol Control Protocol) である．IPv6 では IP アドレスの取得方法には大きく分けて２つある．ステートフルアドレッシングとステートレスアドレッシングである．前者は DHCPv6[25] である．後者が RA(Router Advertisement) である．DHCPv6 は DHCPv4 同様にクライアント側から Solicit メッセージを送信し，アドレスを取得する．RA では，定期的にパケットをマルチキャストしてアドレスを設定をしている．ホスト側はそのパケットを受けて IP を設定する．クライアント側から通知してアドレス設定する機能も有している．ホスト発見では 6LoWPAN などもアドレス設定手法としてあるが，今回は IPv4 では DHCPv4，PPPoE，IPv6 では DHCPv6 と RA の計４つのプロトコルのパケットを取得する．各々のプロトコルに対してデカプセル化するモジュールを作成し，異なるアドレス設定の方法を持つネットワークに適応できるよう設計した．

次にホスト発見行うためのシステム設計概略図 5.2 に示す．DHCPv4 の例である．DHCP パケットをリレーするルータに隣接した L2 スイッチなどでポートミラーを行い，パケットキャプチャサーバがパケットを取得する．DHCPv6 も同様である．RA では，RA パケットを送信するルータの手前に設置すればよい．図 5.2 で言えば，同様の位置に設置する．PPPoE では家庭用ルータで直接ポートミラーしてネットワークトラヒックを監視すればよい．

5.1.2 ホスト分析機構の設計

ホスト分析の設計を行う．ホスト分析では，グローバルコンピューティングで必要となるホスト情報を取得する機能である．5.1.1 で述べたように各４つのプロトコル毎で必要な情報を取得するモジュールをここで設計する．

はじめに情報取得する手法に関して述べる．パケットキャプチャでは PCAP[32] を使用する．ネットワークトラヒックの監視手法として，SNMP(Simple Network Management Protocol)[33] などの手法もあるが，パケットのペイロードのデータから必要な情報を取得することが目的なので，もっとも一般的な PCAP ライブラリを使い，raw packet データを収集する．PCAP ライブラリの他にも BPF(Berkley

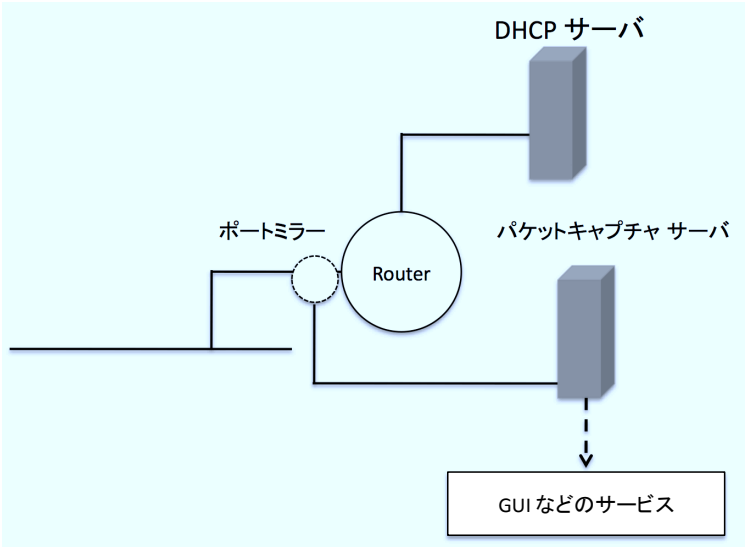


図 5.2 システム設計概略図

Packet Filter) も使用することが可能である．BPF は，FreeBSD などで実装されている．しかし，Linux などでは BPF は実装されておらずその代わりに同様の機能を提供するが API としては互換性がない LPF(Linux Packet Filter) が実装されている．PCAP 以外の API は互換性が乏しく，OS 依存してしまう．こうした点で，Unix，Linux，Mac OS，Windows で利用できる PCAP ライブラリを使用することにした．

PCAP を利用して，パケットキャプチャするプロトコルは DHCP，PPP，DHCPv6，RA の 4 種である．グローバルコンピューティングで使用する情報はシステム要件に従うと以下になる．

- IP アドレス（IPv4，IPv6）
- ホスト名
- MAC アドレス
- ベンダー，機種情報

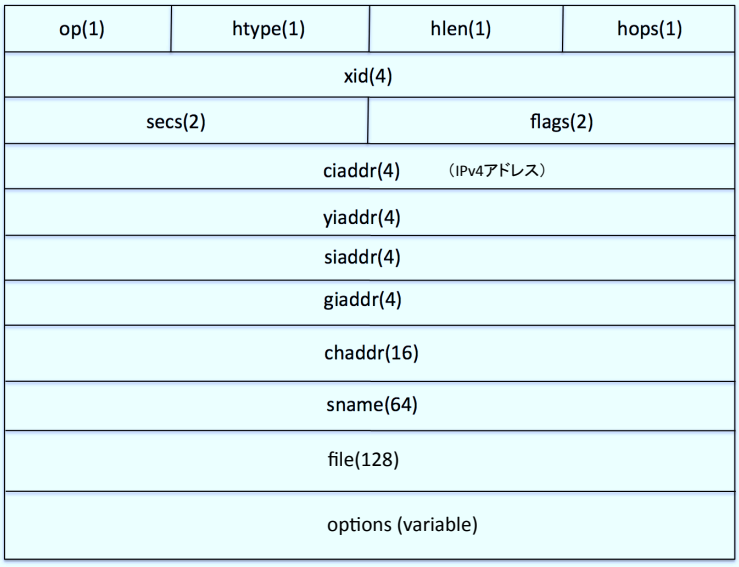


図 5.3 DHCPv4 のパケットフォーマット

これらのホスト情報を各 4 種のパケットから取得する．IP アドレスと MAC アドレスは必ず必要とする情報である．IP アドレスはホスト間通信を行うために必要な情報であり，MAC アドレスも同様である．また，ベンダー情報，機種情報を取得するにあたって MAC アドレスは有用である．OUI(Organizationally Unique Identifier)[34] を参照することによって NIC(Network Interface Card) のベンダー情報を知ることができる．ベンダーの特定によってホストの種類などが予測できる．最後にホスト名であるが，必ず必要な情報ではないが，ベンダーの特定同様，ホストの種類の特定に役立つ．可能な場合は取得することとしたい．

ホスト情報の取得にあたって，各 4 種のパケットのヘッダ情報を示す．ヘッダ情報を参考にして，どのようにして必要情報を取得するか設計する．まず，各プロトコルのヘッダを示す．図 5.3 は DHCPv4 のフォーマットである．図 5.4 は DHCPv6，図 5.5 は PPP と IPCP，図 5.6 は RA である．以下順にみていく．

- DHCPv4
  DHCPv4 では，DHCPDISCOVER，DHCPOFFER，DHCPREQUEST，DHCPACK が順に交換される．図 5.3 において，option は各メッセージ毎で異な

る情報が挿入されて送信されている．IP アドレス (IPv4) は ciaddr から取得することができる．また，別の手段として図 5.7 の IP ヘッダフォーマットを参考にして，DHCPACK パケットの IP のヘッダの Destination Address から IP アドレスを取得することも可能である．MAC アドレスは option に格納されているので，option から取得する．ホスト情報は DHCPREQUEST メッセージの MAC アドレス同様 option の中に格納されているので，そこから取得すればよい．

- DHCPv6
  DHCPv6 は IPv6 用のステイトフルアドレッシングで用いられる．DHCPv4 と同様でクライアント，サーバ間でメッセージの交換によってアドレスを設定する．メッセージは DHCPv4 とは異なっている．SOLICIT，ADVERTISE，REQUEST，REPLY である．DHCPv6 のパケットフォーマットを図 5.4 に示す IP アドレス（IPv6）は，REQUEST か REPLY メッセージのパケットを取得することで各パケットのフォーマットの option 内に格納された IPv6 のアドレスを取得することができる．MAC アドレスは，Ethernet のヘッダから取得する．

- PPP
  PPP での IP アドレス設定の手順は，PPP によってクライアントーサーバ間のセッションが形成され，認証すると，Network-Layer Protocol の設定が開始される．そのときに使用されるのが NCP(Network Control Protocol) で適切にクライアント側に IP アドレスが設定される．具体的には，IPCP(IP Control Protocol) で適切に IP の設定がクライアントに対して行われる．

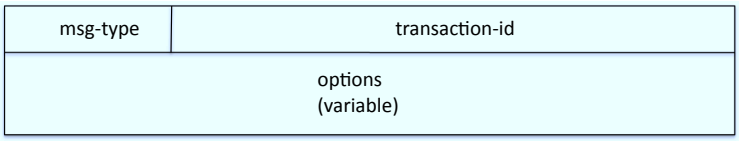


図 5.4 DHCPv6 のパケットフォーマット

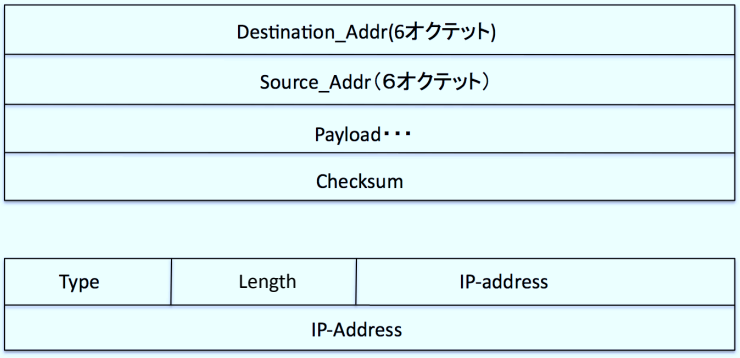


図 5.5 PPPoE と IPCP パケットフォーマット

PPPoE（PPP over Ethernet）が一般的に使用される．PPPoE でセッションするにあたって，ホスト発見の段階が最初にある．そのときのパケットが図 5.5 フォーマットである．このパケットをキャプチャし MAC アドレスを取得する．一方，IP アドレスは前述したように PPP のセッションが確立した後の Network-Layer Protocol の設定が行われる段階で取得できる．具体的には NCP の IPCP を取得することによって IP アドレス（IPv4）を取得することができる (図 5.5)．

- RA
  RA は IPv6 ネットワークにおける，IP アドレス設定手法である．RFC4862[24]

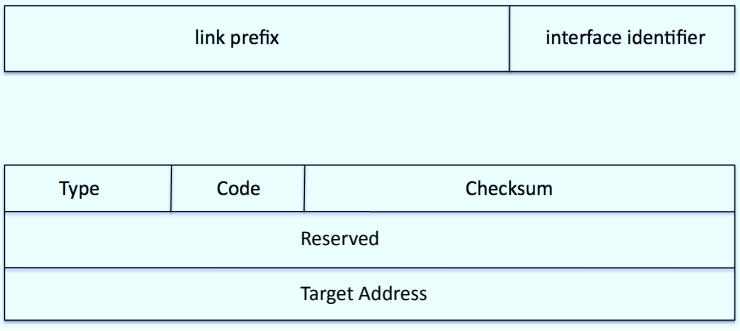


図 5.6 Router Advertisement のパケットフォーマット

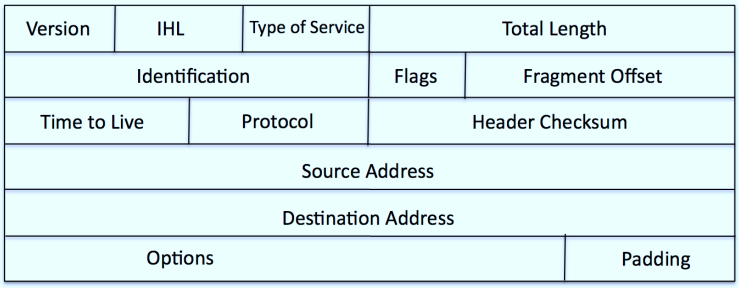


図 5.7 Internet Protocol のヘッダーフォーマット

によれば RA における IPv6 アドレスの設定手法は 2 種類に分けることができる．1 つは，クライアント側から RS(Router Solicitation) メッセージをルータへマルチキャストする．そして，ルータがクライアントへ RA(Router Advertisement) メッセージを送信する．受け取ったクライアントは自身のリンクローカルアドレスと RA のプレフィックスと組み合わせて IPv6 のグローバルアドレスを設定する．

もう一方は，ルータ側が定期的にに RA をマルチキャストしているので，クライアント側はその RA を受け取ることで，同様に自身のリンクアドレスと，RA 内のプレフィックスを組み合わせてグローバルアドレスを設定する．

実際に，IP アドレス，MAC アドレスを取得する際には RA のトランザクションにしたがって取得すればよい．RA メッセージのパケットを取得してプレフィックスを取得する．そして，その前に NS メッセージを送信している場合はそのパケットを取得してリンクローカルを取得する．その後，RA の IP アドレスの設定手法と同様に，プレフィックスとリンクローカルアドレスを組み合わせて IPv6 のアドレスを取得すればよい．また，RS を送信

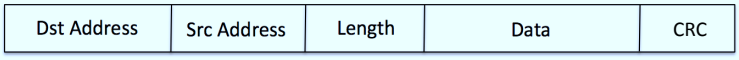


図 5.8 Ethernet のフォーマット

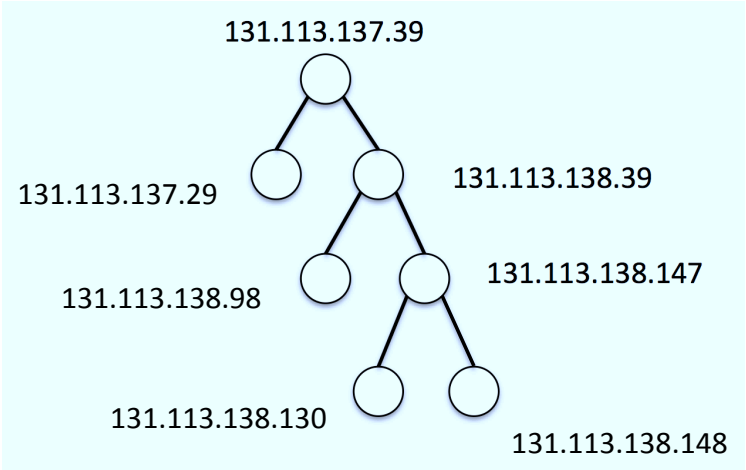


図 5.9 バイナリツリーを利用した例

する場合は RS のパケットを取得して同様に Target Address から取得して RA のプレフィックスと組み合わせれば IPv6 のアドレスを取得することができる．MAC アドレスは Ethernet のヘッダ情報から取得することができる．図 5.8 を参照してほしい，Ethernet のフォーマットである．

5.1.3 ホスト情報保持のためのデータ構造

取得した情報の格納は 2 分木を採用した．２分木 (バイナリーツリー) は基準とする数値を決定し，その基準値より大きいならば右側，小さいならば左側へと各ノードに保存していく．探索の場合は，基準値より大きい場合は右側の枝を探し，小さい場合は左側を探索していけば，格納された中で求める数値を得ることができる．この場合は単純な配列にデータを格納した場合と比べ，探索時間が短縮できる．図 5.9 は実際に IP アドレス（IPv4）をバイナリツリーで格納し，保持した例である．

“131.113.137.39” の IP アドレスが頂点にあるがこれをルートノードという．このルートノードを基準に大小を比較し左右のノードへ IP アドレスが格納されていく．すべての親ノードが左右にノードを保持している完全なツリーとなってい

る場合に検索処理速度が,

$$O(\log_2 N)$$

となる．しかし，どちらか左右に偏っていたりする場合は処理速度が遅くなる．どちらか左右に偏るとはこの場合は，“131.113.137.39” より大きい値の IP アドレスをもつホストが多くアクセスした場合は図 5.9 ように右に偏ったりすることである．

バイナリツリーは配列でホスト情報を保持するよりも探索等の処理が速く，重複する値は保持できないという特徴からホスト情報保持の手法として適していると考え採用した．IP アドレスもまた重複することのない値である．時間的な観点では，別のホストが同じ IP アドレスを使用する場合はある．そのときはリース時間を参照して，IP アドレスが使用されているか使用されていないかの判定を行うことで対処ができる．したがって，IP アドレスの格納や管理に適している．

5.1.4 ホスト情報提示・提供機構の設計

ホスト情報提示・提供の設計を行う．本研究では，グローバルコンピューティングとしてのホスト情報取得および可視化を行っている．必要なデータを受け渡すためのインターフェースはクライアントアプリケーションだけである．しかし，本研究の目的はグローバルコンピューティングとしてのホスト情報取得，可視化なのでホスト管理・制御を考慮に入れた設計をする．

まずはじめに使用するプロトコルの選定を行う．使用するプロトコルは UDP を使用して Socket 通信によって行う．UPnP のように HTTP GET と XML を使ってホストやクライアント，コントロールポイントとのデータの交換をすることも可能である．3.2.2 で議論したように，ウェブクライアントなどに対しては互換性が高い．しかし，テキスト情報であるため内部情報を容易に見えるという問題がある．

UPnP がそうであるように，マルチキャストを使ってコントロールポイントを複数に変更等を通知する時には UDP を使用している．現状の UPnP のプロトコルスタックの一部を UPnP Device Architecture の仕様書 [20] を参考にして示す．

図 5.10，表 5.11 は UPnP のプロトコルスタックである．ここで示した意外でも各処理毎で，異なるプロトコルスタックはある．示した２つを比較してもわかるように各処理によって使用されているプロトコルは異なっている．

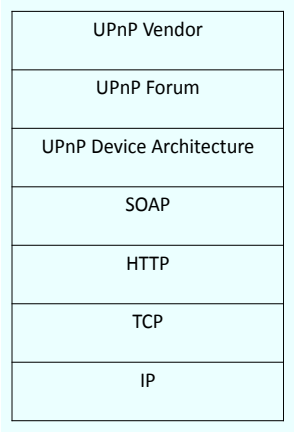


図 5.10 UPnP のプロトコルスタック – コントロールプロトコル –

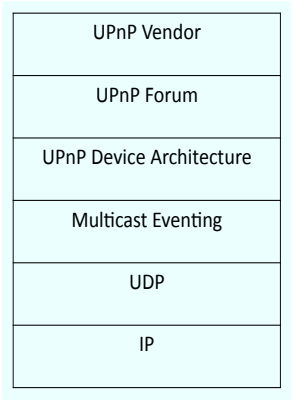


図 5.11 UPnP のプロトコルスタック – マルチイベンティング –

UPnP が異なるプロトコルスタックを処理毎に使用しているようにグローバルコンピューティングの基盤システムもまた同様にプロトコルスタックを考えなければならない．そこで，UDP を利用することにした．HTTP GET を利用して

XML を使用した場合はテンプレートに沿った形なら柔軟にパラメータを変化させ，ホストの状態をコントロールポイントなどへ送信することもできる．あるいは，実際のコントロールポイントがウェブクライアントの場合に互換性が高いといったメリットもある．しかし，トランザクション処理には向かない．5.1.2 で述べたように各 IP アドレス設定手法のプロトコルはパケットフォーマットを定義し，厳密なトランザクションを実行することで zeroconf の世界を実装している．UDP はトランザクション処理に適したプロトコルである．グローバルコンピューティングとしてのホスト情報取得および可視化システムも同様に，メッセージやメッセージ毎のパケットフォーマットを定義し，トランザクション処理を定義し，クライアントアプリケーションのデータ提供やホスト管理・制御を考慮した設計を行う．

図 5.12 が実際に設計したグローバルコンピューティングの基盤システムとしてデータの交換を行うときのパケットフォーマットである．順に説明する．Total Length はパケットのトータルサイズを指し，2byte の field である．パケット全体のバイト数が必要な場合に参照するものである．次に，Msg Type，Message に関してである．Msg Type には 2byte が格納されている．Message はどの処理かを表すメッセージである．Msg Type と Message は一体で，Message を数字で表したのが Msg Type である．各アドレス設定手法がメッセージを定義してトランザクションを組んでいたように，グローバルコンピューティングにおける情報提供用のフォーマットもメッセージタイプによってどのような処理なのかをサーバ，クライアントが認識できるようになっている．そして，このメッセージの組み合わせでトランザクション処理を作る．options は各 Msg Type，Message によって，

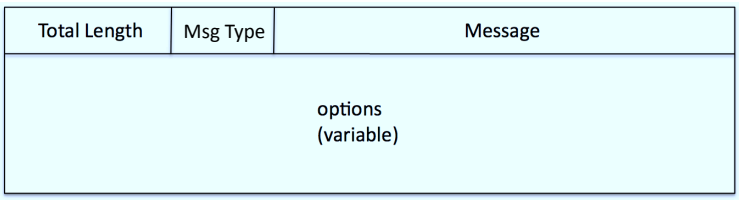


図 5.12 グローバルコンピューティングのパケットフォーマット

表 5.2 メッセージ

サーバ Message(Msg Type)	クライアント Message(Msg Type)
Provide(2)	Request(1)
Exit(0)	

図 5.13 Request メッセージフォーマット

そのフォーマットは変化する．Message によって処理が決まるのでその処理に必要な情報を各 Message 毎で別途定義される．次に，トランザクション処理を実際に形成するメッセージ群の設計を行う．ホスト情報取得および可視化システムにおける，サーバサイドとクライアントサイドでのデータの送受信のメッセージとトランザクションを設計する．

表 5.2 は実際にホスト情報をクライアントへ送るときのメッセージである．クライアントアプリケーションが起動し，サーバへ **Request** メッセージ (1) を送信する．受信したサーバは **Provid**e(2) を送信しホスト情報をクライアントに対して送信する．**Request** メッセージのフォーマットは図 5.13 である．サーバは **Request** メッセージを受け取ったら， **Provide** メッセージをクライアントへ送る．

Provide メッセージのフォーマットは図 5.14 である．一番上の左から順に，パケット全体のサイズ，Msg Type は **Provide** は 2 なので，2 が格納される．Message には Provide が入る．Options であるが，ここからが **Provide** メッセージの独自のフォーマットである．各ホスト情報の本データの前には Length を付与している．Length の後には，それぞれホスト情報分析で取得したデータが格納されている．順番は，IP アドレス，MAC アドレス，ベンダー名である．さらに，その下の optional data であるが，ホスト名が取得できているホスト情報を送信する場合は，図 5.15 のフォーマットで送信され，ホスト名が取得できないホストの場合は図 5.16 のフォーマットで送信される．各 field のバイト数だが，Length はすべ

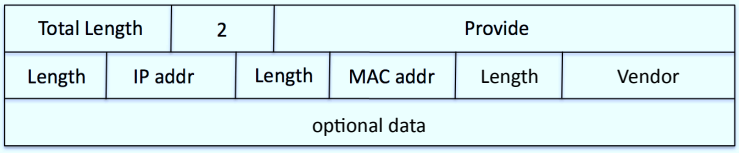


図 5.14 Provide メッセージフォーマット

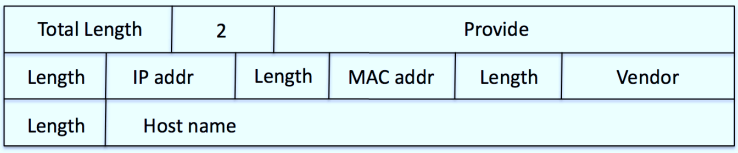


図 5.15 Request メッセージフォーマット (ホスト名があるとき)

て 2byte である．ホスト名とベンダー名の field のみ値が変化するのでそれに合わせて field サイズもまた変化する．

サーバ側のこのフォーマットのパケットを UDP を使ってクライアントに送信する．UDP は通信の信頼性に欠けるので，同一メッセージを 3 回ほど送信することにした．サーバのキャッシュ内のホスト情報をすべて送り届けたら最後にサーバは **Exit**(0) メッセージを送信して通信を終了する．**Exit**(0) のフォーマットは **Request** 同様で，Msg Type が 0 で，Message が **Exit** となる．このようにして，クライアントのデータ提供を行う．

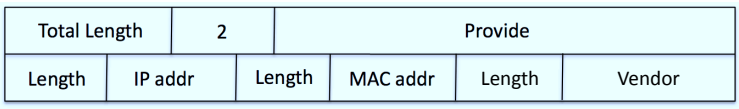


図 5.16 Provide メッセージフォーマット (ホスト名がないとき)

5.2. システム設計のまとめ

本章では，グローバルコンピューティングとしてホスト情報取得と可視化システムの設計を行った．ホスト発見では，パケットキャプチャ手法を採用した．パケットキャプチャによって，各ネットワーク毎で採用されている IP アドレス設定手法に対応することができ，ホストはグローバル IP を各々の環境で，利用することができるようになった．ホスト分析では，取得するプロトコルのパケットに合わせ，モジュールの設計を行った．そして，各 IP アドレス設定のトランザクションを考慮し，必要情報を含むメッセージのパケットを取得することにした．ホスト情報提示・提供では，各処理を実行するためのパケットヘッダを定義した．そして，メッセージに基づいてトランザクションを定義し，各処理に対応できる設計を行った．そして，実際にホスト情報をクライアントに提供する処理を設計した．次章ではグローバルコンピューティングとしてのホスト情報取得，可視化システムの実装に関して述べる．

第6章
ホスト情報取得および可視化システムの実装

本章では，グローバルコンピュティングとしてのホスト情報取得および可視化システムの実装について述べる．まず，実装環境，次にシステム全体像，ホスト発見機能，ホスト分析機能，ホスト情報提示・提供機能の順に述べ，クライアントアプリケーションの実装例を述べる．最後にシステムのまとめを示す．

ホスト発見では，ネットワークトラヒックの監視の実装から，実際にパケットをキャプチャする機能を実装する．ホスト情報分析では，取得する各プロトコルのモジュール化を行う．そして，DHCPv4 モジュールの実装を行う．取得したホスト情報のうち，MAC アドレスからベンダーの情報を取得する機能の実装を行う．OUI データからベンダー情報は参照する．取得したホスト情報は 5.1.2 で述べたようにバイナリツリーを使用する．実際の，バイナリツリーの構造化とホスト情報格納，探索の実装もまた行う．ホスト情報提示・提供の実装では，ホスト情報提供用のフォーマットの実装と，データ提供用サービスの実装を行う．データ提供用のサービスは UDP サーバである．最後にクライアントアプリケーションの実装を行う．5.1.2 で定義したフォーマットにしたがって，各フォーマットの実装およびホスト情報取得後のホスト情報可視化までの実装を行う．

6.1. 実装環境

実装環境は OS(Operating System) は，FreeBSD8.0，開発言語は C 言語である．使用したライブラリは PCAP ライブラリを使用した．その他，pthread[35],

表 6.1 実装環境一覧

	項目	種類
サーバ	OS	FreeBSD8.0(64bit)
	開発言語	C 言語
	ライブラリ	PCAP
	ライブラリ	pthread
	DB	Mysql
ネットワーク	スウィッチ	catalyst 3750E
クライアント	開発言語	C#，WPF

データベースに Mysql を使用した．ネットワークトラヒックの監視対象の L2 スイッチは Cisco Systems 社の Catalyst3750E を使用する．以上がサーバーサイドの実装環境である．クライアントサイドでは.NET で C#を用いた WPF(Windows Presentation Foundation) を使用した．以下，実装環境等を表 6.1 にまとめて示す．

6.2. システム全体像

実際に実装したシステムのアーキテクチャに関して述べる．図 6.1 は，システムアーキテクチャ図である．機器がインターネットにアクセスするとキャプチャサービスがパケットを取得する．第 5 章で述べたように，対象とするプロトコルのパケットが流れているとき，そのパケットを取得する．ここまでがホスト発見である．対象のプロトコルのパケットをキャプチャしたら，必要なホスト情報を取得し，インタープリターサービスでシステム全体で統一されたフォーマットに修正し，サーバのメモリに格納する．さらに，MAC アドレスをもとに，ベンダーを割り出す処理を行う．ホスト情報はメモリに格納されると同時に，ログ情報としてデータベースサービスに，データが転送され，データベースに保存される．ここまでがホスト分析である．ホスト情報提示・提供ではメモリにバイナリツリーで管理されているホスト情報をクライアントアプリケーションから，要求に応じ

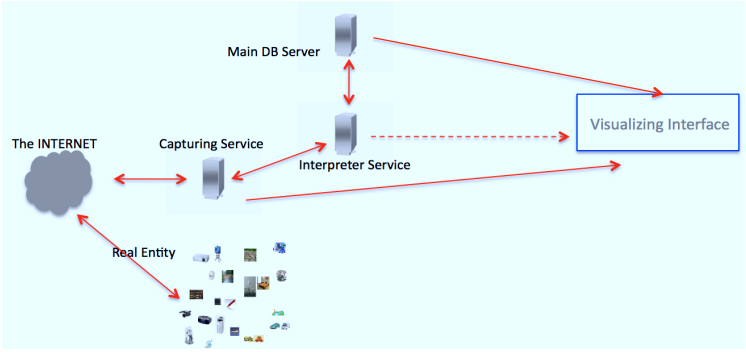


図 6.1 実際に実装したシステムのアーキテクチャ

てホスト情報を送信する．送信する際には 5.1.2 で設計したフォーマットにしたがって，パケットを生成して UDP でクライアントアプリケーションであるホスト情報可視化インターフェースにホスト情報が送信される．ここまでの処理がホスト情報提示・提供である．クライアントアプリケーションはリクエストをインタープリターサービスにパケットを投げて，ホスト情報を受信したあと，各ホスト情報を一覧で表示していく．以上が実装したシステムのアーキテクチャである．

6.3. ホスト発見機構

ホスト発見では，パケットキャプチャ環境の構築とパケットモニタリングの実装を行う．まず，パケットキャプチャ環境の構築について示す．ネットワーク全体のパケットが流れる，L2 スイッチのトラヒックが統合されるポートをモニタリングすることで，あるネットワーク上のすべてのパケットを監視する．統合されるポートとは，例えば vlan などでネットワークが仮想的に分けられている設定の場合などに vlan がすべて統合されるポートを指す．監視対象は統合されるポートであるが，ネットワークトラヒックの監視を行う手法として，ポートミラーリングを使用する．ポートミラーリングは，あるポートに出入りするネットワークトラヒックをコピーしてそのトラヒックを指定したポートへ流す技術である．この手法を使ってネットワークトラヒックをキャプチャサービスのサーバの NIC へ流

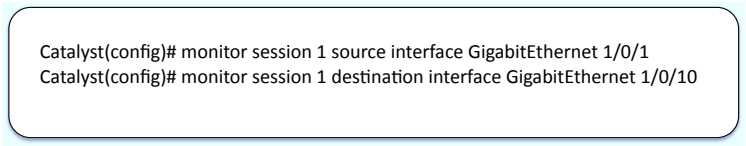


図 6.2 ポートミラーリングの設定

す．そして，キャプチャサービスはそのネットワークトラヒックを監視する．

実際の L2 スイッチの設定を示す．図 6.2 がパケットキャプチャを行う際の L2 スイッチの設定である．モニタリングしたいポート（source interface）を指定し，セッション番号を割り当てる．そして同様のセッション番号トラヒックをコピーして流すポート（destination interface）に割り当てる．そして，パケットキャプチャリングサービス用のマシンの NIC につないで設置は終了である．

次に，PCAP ライブラリーを用いたパケットキャプチャの実装を行う．はじめに，パケットのフィルターをかけるところまでを示す．図 6.3 はパケットキャプチャする前段階である．header に PCAP を呼び出す関数を宣言する．引数の dev は NIC のデバイスを指定．その後，コンパイルをして，フィルターをかける．フィルターをかけるのは，`pcap_loop` 関数を使用する (図 6.4)．`pcap_loop` は無限ループ関数である．パケットを取得するたびに，`pcap_loop` の callback 関数が呼び出され，パケットの処理をすることができるようになっている．ここでの callback は図 6.4 の下にある `packet_handler` である．実際にパケットのデータとして パケットが格納される変数は callback 関数の `pkt_data` にあたる．実際の情報取得であるホスト分析はこの，callback 関数内にて行われるのである．

6.4. ホスト分析機構

ホスト分析の実装を述べる．ホスト分析では，実際に各 IP アドレス設定のプロトコルのパケットが順次，ネットワークトラヒックの中に現れたときに，ホスト発見機構で実際にパケットを取得し，取得した時に，必要なホスト情報を取得する．取得したデータはベンダー情報を取得後，バイナリツリーに格納される．

```
handle = pcap_open_live(dev, BUFSIZ, 1, timeval, errbuf);
if(handle == NULL){
        fprintf(stderr, "Couldn't open device %s:\n", dev, errbuf);
        return(2);
}

if(pcap_compile(handle, &fp, filter_exp, 0, net) == -1)
{
        fprintf(stderr, "Couldn't parse filter %s: %s\n", filter_exp,
        pcap_geterr(handle));
        return(2);
}
```

図 6.3 PCAP の使用

```
pcap_loop( handle, -1, packet_handler, NULL);
void packet_handler(u_char *param, const struct pcap_pkthdr *header,
 const u_char *pkt_data);
```

図 6.4 pcap_loop 関数

以下，順に実装を行う．

6.4.1 パケットキャプチャモジュール

5.1.2 章で設計したように，各 4 つの IP アドレスの設定方法用のパケットキャプチャモジュールの実装を行う．図 6.5 で実際に各アドレス設定用のプロトコルのパケットからホスト情報を取得するモジュールを示す．

`getDHCP()` が DHCPv4 パケットからホスト情報を取得するモジュールである．次いで上から順に，`getPPP()` が PPP を取得するモジュールである．`getStateful()` が DCHPv6 のパケットを取得する．`getStateless()` は RA を取得する．これらのモジュールによって必要なホスト情報が取得される．各モジュールの返り値は構造体である．それぞれ，IP アドレスや MAC アドレスといったものを返す．それぞれ，IPv4 か IPv6 で区別される．

6.4.2 DHCP パケット取得モジュール

DHCP パケットキャプチャモジュールの実装を示す．まず，必要情報の取得に関して実装するために論理を組む．RFC2131[17] を参照して 5.1.2 章で述べた DHCP トランザクションを図示化（図 6.6）する．赤の円で囲った Request と Ack のパケットを取得すれば，すべての必要情報をもれなく取得することができる．あら

```
#include <pcap.h>

Analyzev4 getDHCP(const u_char *pktdata);

Analyzev4 getPPP(const u_char *pktdata);

Analyzev6 getStateful(const u_char *pktdata);

Analyzev6 getStateless(const u_char *pktdata);
```

図 6.5 各プロトコルのアナライズ用モジュール

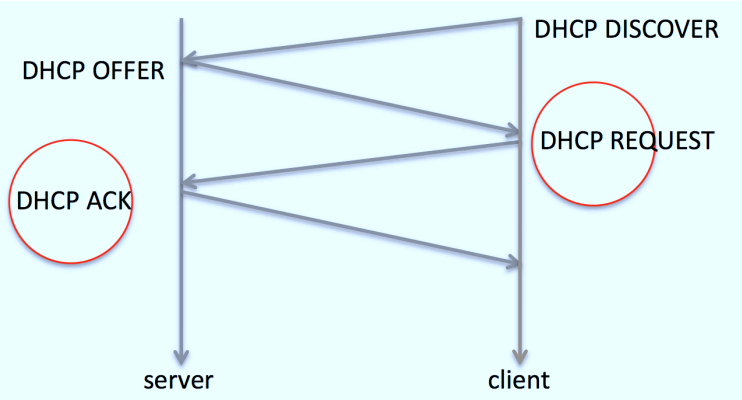


図 6.6 DHCPv4 トランザクション

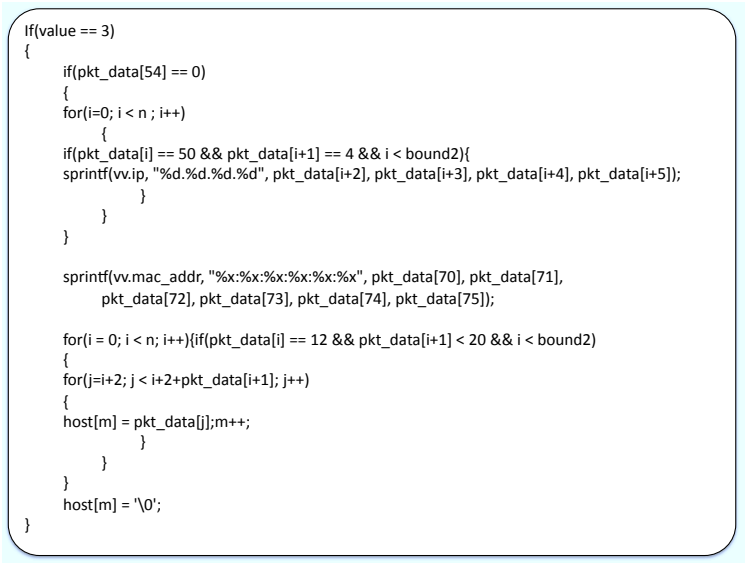
```
If(ntohs(udp->th_dport) == DHCPSERVER || ntohs(udp->th_dport) == DHCPCLIENT)
{
    getDHCP(pktdata);
}
```

図 6.7 pcap_loop の実装

かじめ接続されているホストもまたリースタイムが切れると，Request を送信し，サーバ側が Ack を送信し，再度アドレスの設定を行う．したがって，Request と Ack を取得することによってホストの発見を行うことができる．

続いて，`pcap_loop` 内および，getDHCP モジュールの実装を行う．DHCP パケットは UDP を使用しているので，UDP ヘッダのポート番号を調べ，DHCP パケットかどうかを判断する．UDP ヘッダ情報を取得するための構造体のうち送信先ポート番号を参照して，ポート番号を調べる．DHCP のポート番号はサーバが 67 で，クライアント 68 なので，それを参考に実装すると，図 6.7 になる．DHCP パケットがきた時に，DHCP パケットから情報取得するモジュールを呼び出すのである．

次に，DHCPv4 のモジュールの実装を行う．DHCPv4 モジュールにパケットのデータを引数で渡す．モジュールがはじめに行うことは，DHCP パケットのタイプである．取得対象の DHCP メッセージタイプは Request と Ack なので，そ



```
If(value == 3)
{
        if(pkt_data[54] == 0)
        {
        for(i=0; i < n ; i++)
                {
        if(pkt_data[i] == 50 && pkt_data[i+1] == 4 && i < bound2){
        sprintf(vv.ip, "%d.%d.%d.%d", pkt_data[i+2], pkt_data[i+3], pkt_data[i+4], pkt_data[i+5]);
                        }
                }
        }

        sprintf(vv.mac_addr, "%x:%x:%x:%x:%x:%x", pkt_data[70], pkt_data[71],
                pkt_data[72], pkt_data[73], pkt_data[74], pkt_data[75]);

        for(i = 0; i < n; i++){if(pkt_data[i] == 12 && pkt_data[i+1] < 20 && i < bound2)
        {
        for(j=i+2; j < i+2+pkt_data[i+1]; j++)
        {
        host[m] = pkt_data[j];m++;
                        }
                }
        }
        host[m] = '\0';
}
```

図 6.8 DHCP モジュールの実装

れぞれメッセージタイプが 3 と 5 である．メッセージタイプの値はメッセージフラグの 53 と Length の 1 の後に格納されているので，Length の次のバイト値を取得する．Request の時は IP アドレスと MAC アドレスが取得できる．ホスト情報もまた取得できるので取得する．Ack メッセージではサブネットが取得できる．5.1.4 章で設計したように，オプションとして取得しておく．IP アドレスのメッセージフラグは 50 で Length の 4 の後に格納されているのでその次から取得する．MAC アドレスはパケットの 70 バイト目から格納されているのでそこから 6 バイト分を格納して取得する．ホスト情報は Ack パケットで，メッセージタイプは 1 で Length が 4 であるその次からサブネットの値なので，4 バイト分取得する．以下，図 6.8 実装を示す．vv は取得データを一時保存する構造体である．変数 host もホスト名取得後 vv 構造体に格納される．**Request** の実装を示したが，Ack でサブネットを取得するときも，同様の処理で取得した．

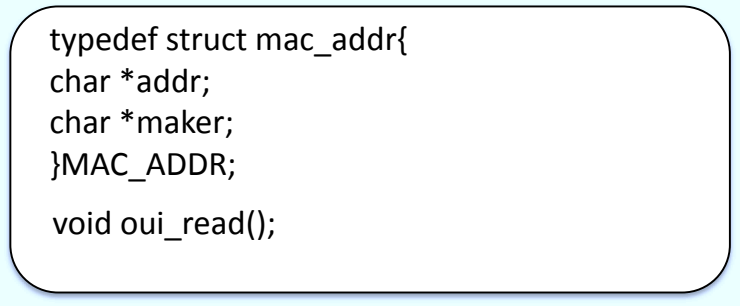


図 6.9 OUI データベースとの照合

6.4.3 機器ベンダー情報取得

ベンダー情報取得に関しては以下のように実装した．まず，OUI からベンダー情報が喝采されている oui.txt ファイルを取得して用意しておく．oui.txt からベンダーと MAC アドレス情報を読み込んでメモリに保持する．そして，6.4.2 で実装したように MAC アドレスを各モジュールで取得した後，メモリ上に保持した OUI のベンダー情報のデータと照合してベンダーを割り出す．図 6.9 に oui.txt を読み込み構造体に格納するモジュールの実装を示す．

6.4.4 バイナリツリーの実装

バイナリツリーに関して述べる．バイナリツリーはメモリ上でホスト情報を管理するために使用される．実際に使用する構造体を示す．図 6.10 が構造体である．IP アドレスをキーに，左右へ振り分けていくことになっている．実際にホスト情報を格納するのは，図 6.11 のように実装した．変数 pp−>ip が現在のノードの IP アドレスの値であり，変数 s が挿入する IP アドレスのデータである．変数 cmp に IP アドレスの値の比較結果を格納して左右へ振り分けていく．最終的に最下層のノードに到達したら IP アドレスの値を挿入するように実装した．データをバイナリツリーへ格納した直後にホストのログ情報を残すために DB へアクセスし，ホスト情報を挿入するようになっている．DB は mysql を使用している．

```
typedef struct binary{
char *ip;
char *subnet;
char *mac_addr;
char *host;
char *vendor_name;
struct binary *left;
struct binary *right;
}BINARY;

BINARY root = {NULL, NULL, NULL, NULL, NULL, &root, &root};
```

図 6.10 ホスト管理用バイナリツリー

6.5. ホスト情報提示・提供の実装

ホスト情報提示・提供では，5.1.4 章で述べたように定義したヘッダフォーマットを実装する．そして，実際に UDP での通信を行うところを実装する．図 6.12 は Provide メッセージ用のヘッダフォーマットである．バイナリツリーを探索してホスト情報を格このヘッダに格納してクライアントへ送信する．

次に，ホスト情報を送信するサーバの実装に関して述べる．ホスト情報を送信するサーバは別のスレッドで待機するように実装した．スレッディングは pthread を使用した．別スレッドで UDP サーバをたて，クライアントからアクセスとメッセージの受信するために待機する機能である．実際の実装は下記の図 6.13 に示す．スレッドを作成する関数が `pthread_create()` 関数である．引数の中に callback 関数があり，その callback 関数が `socket_server()` 関数である．

`socket_server()` 関数内の処理は，通常の UDP ソケットの処理である．`socket()` でソケットを開き，アドレスやポートの設定を行ったら，`bind()` する．`while` ループの中で `recvfrom()` で受信する．クライアント側からアクセスがあれば，接続を許可し **Request** メッセージを受信したら **Provide** メッセージとして，ホスト情報をバイナリツリーから取得して送信していく．

```
while(1){
        cmp = strcmp(pp->ip, s);
        if(cmp == 0){
        }
        else if(cmp > 0)
        {
                if(pp->left == NULL)
                {
                        pp->left = new;
                        break;
                }
                pp = pp->left; //next node
                }
        else{
                if(pp->right == NULL)
                {
                        pp->right = new;
                        break;
                }
                pp = pp->right;
                }
}
```

図 6.11 バイナリツリーへホスト情報格納

```
struct gcsend{

        int totallen;
        unsigned int msgtype;
        char Message;
        int lenip;
        char *ip;
        int lenmac;
        char *mac_addr;
        int vedorlen;
        char *vendor;
        //option
        char *subnet;
        char *hostname;

}GCSEND;
```

図 6.12　ホスト情報送信用ヘッダフォーマットの実装

```
rc = pthread_create(&threads[NUM_THREADS], NULL, socket_server, (void *)NUM_THREADS);

void socket_server(){
        int sock;
        int length;
        struct sockaddr_in addr, from;
        char buf[2048];
        memset(&addr, 0, sizeof(addr));
        addr.sin_family = AF_INET;
        addr.sin_port = htons(port);
        addr.sin_addr.s_addr = htonl(INADDR_ANY);

        if((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0){
                perror(NULL);
                exit(2);
        }

        bind(sock, (struct sockaddr *)&addr, sizeof(addr));
        memset(buf, 0, sizeof(buf));

        while(1){
                recvfrom(sock, buf, sizeof(buf), 0, (struct sockaddr *)&from, &length);
        }
}
```

図 6.13　UDP サーバの実装

図 6.14　ホスト情報可視化アプリケーション

6.6. ホスト情報可視化アプリケーションの実装

クライアントアプリケーションであるホスト情報を可視化するアプリケーションの実装に関して述べる．開発に使用したのは前述したように .NET を使用．C#を使用して WPF で GUI(Graphical User Interface) を開発した．WPF は XAML と呼ばれる宣言型の言語で，C#とのデータのバインディングができ，動的に GUI を構築することができるのが特徴である．今回はこの XAML を使用してホスト情報可視化アプリケーションを実装した．実際に実装したものは，図 6.14 にて示す．ホスト名と MAC アドレスは個人情報にあたるため表示するのを避けた．

左の上のボタンを押すと，アプリケーションがスタートするようになっている．実際の内部の処理はボタンイベントハンドラに UDP クライアントの処理がコーディングされている．サーバに接続したら **Request** メッセージをサーバ側に送信する．送信した後は受信するように待機し，ホスト情報を受信する．受け取ったデータはそれぞれ，ベンダー情報をもとに，ベンダーのロゴイメージを読み込み XAML(Extensible Application Markup Language) とデータをバインディングして情報が表示されるようになっている．図 6.14 に表示されているホスト情報は実際にあるネットワークに設置し，取得した情報を表示させている．取得時間は約 3 時間程度である．

6.7. システムまとめ

本章では，グローバルコンピューティングのインフラストラクチャシステムとして，ホスト情報取得および可視化システムの実装を行った．ホスト発見ではパケットキャプチャ環境の構築，各プロトコルに対する処理のモジュール化を行った．そして，ホスト分析ではそのモジュールの処理を実装した．DHCP では Request と Ack パケットのキャプチャから必要なホスト情報を取得した．取得した情報はバイナリツリーにて管理する機能を実装した．ホスト情報提示・提供では，ホスト情報送信用パケットフォーマットの実装や，各クライアントとのメッセージベースのトランザクション処理を実装した．そして，ホスト情報送信用の UDP サーバの実装を行った．最後にクライアントアプリケーションとしてホスト情報を可視化するアプリケーションの実装を行った．本研究における実装は以上である．

第7章
評　　価

本研究における評価をおこなう．評価を行うものは2点ある．1点目がリアルタイムホスト発見と情報取得の精度である．リアルタイムホスト発見とホスト情報取得の精度とはネットワークにアクセスしてきたホストの検知の成功か失敗かのことを指す．成功の場合はバイナリツリーに格納されることになる．失敗の場合はバイナリツリーに格納されていないことになる．この精度を評価する目的はホスト情報取得の手法として採用したパケットキャプチャが適切であったかを判断するためである．パケットキャプチャはパケットロス・パケットドロップでホストの検知を失敗する可能性を有している．そのためホスト発見手法としてパケットキャプチャがホストをもれなく検知できているかを調査するために精度の評価を行うのである．

2点目がクライアントアプリケーションの応答時間の測定である．サーバーアプリケーションに対して，クライアントのアクセスによる応答時間の評価を行う．応答時間とは，クライアントが **Request** メッセージをサーバへ送信し，今現在ネットワークに接続しているホスト情報をサーバから送信される **Provide** メッセージから取得し描画するまでの時間である．このクライアントの応答時間を評価する理由は，スケーラビリティの評価をする必要がある．グローバルコンピューティングではすべてのネットワークでの運用を想定している．そのため本研究で実装されたシステムが大規模に運用できうるものか評価する必要がある．そのために，スケーラビリティが担保されているか評価を行う．そしてスケーラビリティを評価する上で必要な項目の1つである．ホストの増加がスケーラビリティに影響するかを測ることで，クライアントアプリケーションを評価する．

実際の評価を行う前に，サーバーの機器，クライアントの機器のハードウェア

表 7.1 サーバーおよびクライアントのハードウェア

種類	各種ハードウェア	名前
サーバー	マザーボード	ASUS P6T
サーバー	CPU	Intel corei7
サーバー	メモリ	2.00 GB
サーバー	ハードディスク	Seagate Barracuda7200.12 SATA
サーバー	NIC	RealTek 1000-BASE T
サーバー	OS	FreeBSD 8.0
クライアント)	名前	Compaq 8510w (Hewlett Packerd)
クライアント	CPU	Inter Core 2 Duo
クライアント	メモリ (RAM)	2.00 GB
クライアント	ハードディスク	Serial ATA 7200rpm
クライアント	無線 LAN	インテル Wireless WiFi Link4965AGN
クライアント	OS	Windows 7 (64 bit)

を図 7.1 に記載する.

7.1. ホスト発見精度の評価

ホスト発見精度の評価として，DHCP ログとバイナリツリーで管理しているホスト情報の照合を行う．DHCP ログとバイナリツリーで管理しているホスト情報との一致で評価する目的に関して述べる．4.2 での議論からパケットキャプチャ手法を採用した．パケットキャプチャ手法の問題点はパケットロス・パケットドロップであった．パケットロス・パケットドロップが発生した場合にはホスト情報取得の各プロトコルのパケットを取得できずホストの検知に失敗する可能性がある．パケットロス・パケットドロップが顕著な場合は，パケットキャプチャ手法は適していないことになる．そこで，パケットキャプチャ手法が適しているかパケットロス・パケットドロップのためにホスト検知を失敗していないか実際に

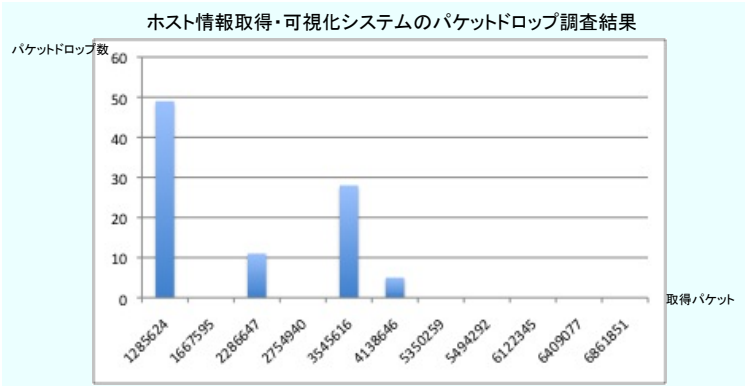


図 7.1 ホスト情報取得・可視化システムのパケットドロップ調査結果

パケットドロップ数を調査し，また DHCP ログと差分をとりホストの一致で評価を行う．

評価用実験実施の詳細に関して述べる．パケットドロップの調査は実装でアプリケーションの終了と同時に書き出すよう PCAP ライブラリの `pcap_stats()` 関数を使用してパケットドロップ数を調査を行う．バイナリツリーに管理されているホスト情報をテキストファイルに書き出し，そのデータと DHCP ログファイルの照合を行う．DHCP ログは各メッセージが送信する度にログファイルに書き出されている．ホスト情報が取得できているかどうかを判断するために IP アドレス，MAC アドレスやホスト情報の照合を行う．

パケットドロップ数の調査と，DHCP ログとの照合後はパケットキャプチャ手法がホスト発見手法として適切であったか議論する．その際には 4.2 で議論した手法も再度検討し評価後の結論とする．

まずパケットドロップの調査を行う．実際に本研究で実装したホスト発見・ホスト情報取得システムにてパケットキャプチャを行い，パケットドロップ数を調査した．ネットワークの構成は GigabitEthernet にて構成される．実験内容は 15 分間アプリケーションを動かし，ランダムにクライアントアプリケーションからホスト情報取得の **Request** メッセージを送り，**Provide** メッセージを送信する処理を行った．15 分の後，ホスト検知のシステムを終了させ，最後にパケットの総数とパケットドロップした数を出力させ，データを収集した．

図 7.1 は収集したデータを微量ながら発生している．原因の一つとして考えられるのが，クライアントアプリケーションを行う処理の割り込みが発生したときである．クライアント側から **Request** メッセージを受信すると別スレッドの socket サーバがそのメッセージを受信して，**Provide** メッセージを送信する．その処理の間にバッファメモリからパケットが落ちる可能性が考えられる．また，FreeBSD の別の割り込み処理が発生しその処理をしている間にバッファメモリからパケットが落ちた可能性もある．しかし，もっともパケットドロップしたのがわずか 49 であった．これは取得パケット数が 1285624 であることから微量のパケットを落としたに過ぎない．しかし，この数値は実際にポートミラーできた量である．L2 スイッチでは，SPAN(スイッチド ポート アナライザ) のポートベースミラーを行って往復分をミラーしている．GigabitEthernet 環境では 2 ギガ分のトラヒック量となる．したがって，10G 環境でないとすべてのトラヒックを監視することはできない．L2 スイッチ側で破棄されているパケット数もまた計測する必要がある．さらに，BPF のバッファから PCAP のバッファへ入力するときパケットが落ちる可能性もある．今回の実験結果から PCAP へ入力された量のパケットドロップは問題がなかったということが言える．

次に DHCP ログとの照合の評価を行う．実際に実施した時間は 11 月 11 日午後 2 時 36 分から 11 月 11 日午後 5 時 1 分まで行った．システムを設置したネットワークはある大学のネットワークである．ネットワークは Gigabit Ethernet にて構成される．収集したホスト情報はのべ 209 ホストである．この取得したホストを DHCP ログと照合し，各ホスト毎に必要な情報それぞれ収集できているかどうかを判定する．ホスト間の通信で必要な情報として IP アドレス，ホストの種類を知る上でベンダー情報が役に立つが，それは MAC アドレスから取得するよう実装したので，MAC アドレスの 2 種類および，ホスト情報を加えた 3 種類の情報を照合して精度を測定する．

まず，DHCP ログファイルをみる．実装したホスト検知の機構上 Nak が返される場合の IP アドレスを 1 ホストと認識してホスト情報を格納してしまうので，209 から Nak が返される場合の IP アドレスとそのアドレスを要求したホストを抜く．すると実際のホスト数は 185 であった．

表 7.2 DHCP ログファイルと実際に取得したホスト情報の一致

情報の種類	一致したホスト数	不一致のホスト数	結果
IP アドレス	185	0	一致した
MAC アドレス	185	0	一致した
ホスト名	185	0	一致した

DHCP ログとの照合はバイナリツリー内のホスト情報との差分をとることで行った．DHCP ログで差分をとるにあたって，必要な情報を保持しているメッセージが Ack メッセージを送信したときのログ情報である．Ack メッセージの行には IP アドレス，MAC アドレス，ホスト名が記載されている．したがって，DHCP ログファイルから Ack メッセージ部分を抜き出し，バイナリツリー内のホスト情報をテキストファイルへ書き出したファイルとの差分をとった．

結果は，表 7.2 にまとめた．具体的な，IP アドレスやホスト名，MAC アドレスはネットワークの特定やホストの特定に利用できる情報なので割愛する．DHCP ログとバイナリツリー上のキャッシュデータとの差分は一致した．

パケットドロップの実験と DHCP ログとの差分をとる実験の結果から，ポートミラーされた量のトラヒックの中に DHCP のパケットが DHCP ログに記載されている分が含まれていた．さらに，PCAP に入力されたパケットの中にも含まれていた．PCAP に入力された分のパケットも落としていない．したがって，DHCP ログとの差分で一致することができた．前述したようにポートミラーされる量と PCAP への入力間でパケットを落とす可能性があるので，この 2 点も考慮してシステムを組み上げる必要がある．

7.1.1 ホスト発見手法の再検討

今回実装に使用した環境である FreeBSD 環境で検討する，パケットが NIC に届けられるとバッファメモリにパケットが格納される．そのパケットは BPF に渡され，PCAP は BPF のメモリからパケットのデータを取得している．パケッ

トドロップによってパケットを喪失する場合はこのバッファメモリから各メモリにデータを受け渡す際に生じる．バッファメモリは一時的にデータを蓄積するメモリである．一時的にバッファメモリに保存できないデータ量に達したときにパケットドロップは発生する．そして，パケットキャプチャはすべてのパケットを受信する．その中で DHCP などの対象とするパケットを監視して，DHCP などの時にホスト情報取得の処理がなされるようになっている．そのためすべてのパケットを取得し，監視しているのである．

ここでパケットキャプチャをしている際に別の処理が割り込んだ場合は，上記したようにバッファメモリにパケットが蓄積されるがデータ量が多ければパケットドロップが発生することになる．実際に本研究で実装したシステムもクライアントアプリケーションから **Request** メッセージが届き，ホスト情報を送信する処理をしている間にパケットドロップが 49 見られた (図 7.1)．GigabitEthernet 環境で微量ながらパケットドロップが発生するということは，より高負荷な環境ではパケットドロップが増加すると言える．もちろん，パケットキャプチャの性能を向上させる手法もある．例えば，バッファメモリを増やし，バッファメモリの量に合わせて CPU のクロック数を変更するなどである．しかし，10GigabitEthernet 環境のような広帯域ではクライアントから **Request** によって割り込みが発生し，あるいはそれ以外での割り込みが発生した場合にバッファメモリに蓄積されるデータ量は増加する．GigabitEthernet 環境では微量だったかもしれないが，10GigabitEthernet 環境のような高負荷な環境では機器の調整やパケットキャプチャの性能に関わるハードウェアの性能を高いものにする必要がある．このようなことはハードウェア依存が発生する可能性を示している．ホスト検知手法として各プロトコルへの対応やポートミラーリングするだけで設置が可能という適応力の高さで採用したパケットキャプチャ手法はハードウェアや機器の調整といった必要性によって適応力が著しく損なわれることになる．これは問題である．性能の高いハードウェアは高価であり，また機器の調整などは専門的な技術が要求される．したがってシステムの設置が難しくなってしまう．そこで，パケットドロップが増加するような高負荷な環境では，別の手法の利用を検討する必要がある．

パケットドロップが増加するような高負荷な環境ではリアルタイムにホストを

検知するのはアドレス設定を行うサービスとの連携手法が適している．広帯域下でデータ量の多いネットワーク下では例えばDHCPログのデータを参照するあるいは，DHCPがリアルタイムに管理しているメモリ内のホスト情報を参照することは扱うデータ量や処理のハードウェアの負荷が少ない．したがって広帯域下でもホストの検知が容易にできる．しかし4.2で議論したことに加えて，この手法にはいくつか問題点がある．ステートフルアドレッシングの環境ではアドレス設定を行っているサービスとの連携が可能である．なぜなら，サービスを提供するサーバがホストを管理しているからである．管理してるとはホスト情報を保持しているということである．だが，ステートレスアドレッシングの環境ではホスト管理を行っていない．そのためIPv6環境でのRAなどによるアドレス設定手法を採用しているネットワークではアドレス設定のサービスとの連携手法は適していない．

さらに，L2スイッチでポートミラーしたポートに直接計算機を接続できない場合もまた，アドレス設定サービスとの連携手法は適している．直接計算機に接続できない場合とは，L2スイッチとの距離が離れていて接続できない場合などである．L2スイッチを設定している環境によってはポートへ計算機を接続できない場合，RSPANを使用して，別のL2スイッチのポートへポートミラーすることもできる．しかし，RSPANはセッション数の制限がある．また，対応していないL2スイッチもある．加えて，トランクポート経由でネットワークトラヒックを搬送するので，帯域幅を埋める問題がある．GigabitEthernet環境では10Gなどの広帯域なネットワークを必要とする．また，ポートミラーする対象が広帯域ならポートミラーをしミラーしたネットワークトラヒックを搬送することができない．したがって，高負荷な環境でステートフルアドレッシングサービスを使用している環境ではアドレス設定サービスとの連携手法が適している．また，ポートミラーしたポートに直接計算機を接続できない場合もアドレス設定サービスとの連携手法が適している．

7.2. クライアントアプリケーションの応答時間の評価

本章冒頭で述べたようにクライアントの応答時間を評価する．応答時間の定義をする．クライアントアプリケーションがサーバに **Request** メッセージを送信する．**Request** メッセージを受信したサーバ側が **Provide** メッセージでバイナリツリー内に保持しているホスト情報をクライアントへ送信する．ホスト情報を受信したクライアントはそのホスト情報を逐一出力していく処理をする．**Exit** メッセージを受信したら Socket 通信を閉じる．そしてクライアントアプリケーションは受信したホスト情報を可視化する．クライアントが **Request** メッセージを送信し，**Exit** メッセージを受信し，可視化するところまでがクライアントの応答時間とする．

次にクライアントの応答時間の評価を行う目的に関して述べる本章冒頭で述べたように，スケーラビリティの評価を行う必要があるからである．グローバルコンピューティングで実現する世界は，世界中の機器をネットワークを介して，自由に利用できることである．ホスト間を自律的に通信させたり，制御したり，管理したりすることができる世界である．そのため，扱うホスト数は無数にある．IPv4 と IPv6 を組み合わせるとホスト数は実質無限である．そのため，クライアントアプリケーションがどのくらいのホスト数を扱うことができるのか評価する必要がある．ホスト数の増減でスケーラビリティに影響するのか評価を行う．

実際にスケーラビリティの評価を行う方法に関して述べる．クライアントアプリーションがコントロールポイントとホスト情報を取得するトランザクション処理の開始と終了にタイムスタンプを押し，開始時間と終了時間の差分をとって，クライアントの応答時間を取得する．実際に応答時間のデータ収集にはクライアントアプリケーションにボタンを設置し，そのボタンのクリックイベントハンドラ内にホスト情報取得のトランザクション処理と共に，タイムスタンプの処理を埋め込んだ．図 7.2 にそのコードを示す．

サーバアプリケーションのデータは恣意的にデータを生成して送信した．ホスト数は 2^{10} から 2^{17} 台のホスト数で実験を行った．恣意的に生成されたデータは実際にバイナリツリーにて格納されるデータと同様で，IP アドレスや MAC アドレス，サブネット，ホスト名が格納される．IP アドレスはバイナリーツリーノー

```
private void button_click(object sender, RoutedEventArgs e)
{
    //開始のスタンプ
    DateTime time = DateTime.Now;
    Console.WriteLine(time.ToString("ss.fffff"));
    //トランザクション処理
    ・・・・・・・・・・
    ・・・・・・・・・・

    //終了スタンプ
    DateTime time2 = DateTime.Now;
    Console.WriteLine(time2.ToString("ss.ffff");

}
```

図 7.2 タイムスタンプのコード一例

ドの右側に格納されるように生成される．今回のクライアントの応答時間の実験ではすべてのノードが右側に格納されるようにした．バイナリツリーの検索処理速度がもっとも遅くなるような場合での実験によって，ホスト数の増加がスケーラビリティに影響するのか評価を行う．

実験の結果は図 7.3 であった．まず，グラフの説明をする．サーバ側でホストを検知するアプリケーションを動かし，図 6.14 のアプリケーションにて **Request** メッセージを送信し，ホスト情報を取得した．両グラフの X 軸はサーバアプリケーションが検知してバイナリツリーで確保しているホストの数を示している．Y 軸はトランザクション処理を終え，ホスト情報を出力するまでの応答時間である．

実験結果では，ホスト数の増加と応答時間の増加はホスト数が 2 倍になると 1.5 倍から 2 倍の増加を示している．この結果から今回実装したクライアントアプリケーションはスケーラビリティに問題がある．ホスト数がさらに 2 倍になった 2^{18} に増加したときにはさらに応答時間が増加することが予想できる．そのため，ホスト数の増加はスケーラビリティに影響を与えており，解決する必要のある問題であることがわかった．

応答時間の増加の原因として考えられることはベンダー情報を表示するために，該当する画像データを検索する処理があげられる．ベンダー名を参照して条件分岐にて該当するベンダーロゴを検索し，画像パスを XAML にバインディングする処理を行っている．この場合は該当する条件一つ一つと照合するため，処理が

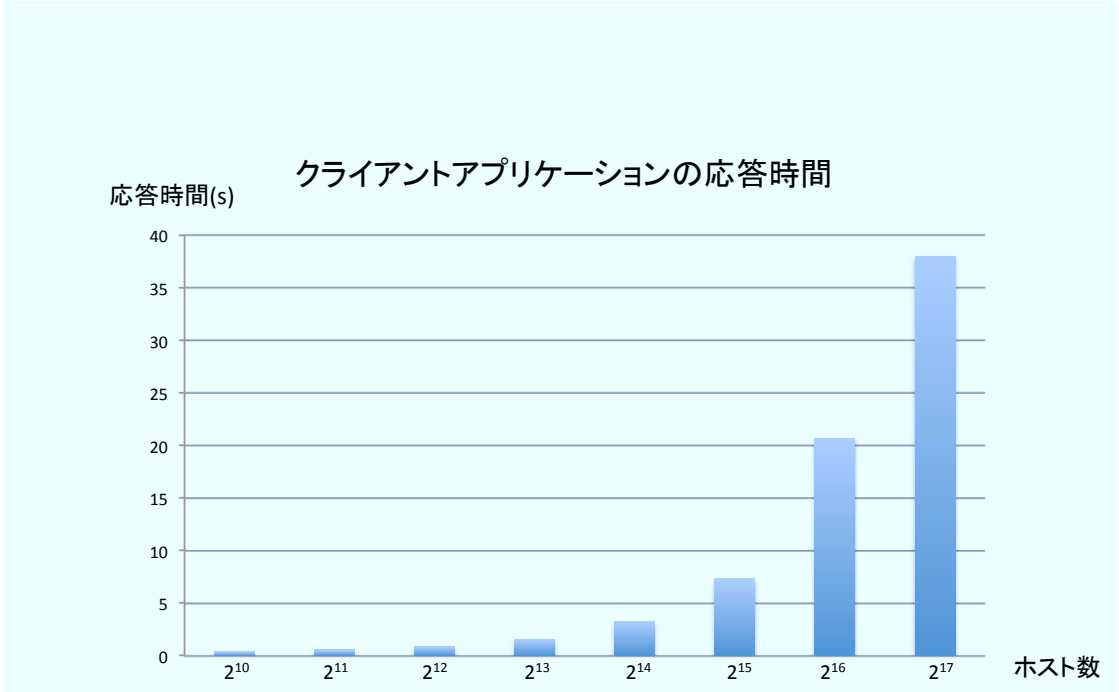


図 7.3 応答時間の実験結果

遅くなる．2^{15} までのホスト数ならば処理速度が顕著に遅くなっていないが，2^{16} では急激に処理が遅くなっている．そのため，MAC アドレスの先頭 3 バイトとベンダー毎の画像パスとを使ってマスクテーブルを作成して検索処理の効率化を行うなどの対策が必要である．

加えて，ハードウェアの性能もスケーラビリティに影響を与えていることもこの結果は示している．ホストの IP アドレスやホスト名，MAC アドレス，サブネットに加えて，ベンダー情報としてのロゴ画像がクライアントアプリケーションには表示される．画像データが約 15K なので，ホスト数が増加するとホスト情報に加えて画像データ量も加算されるためデータ量が増加し，メモリが足りなくなるなどの制限が加わる．したがって，ハードウェアのリソースもスケーラビリティに影響えることとなる．

今回のクライアントアプリケーションの応答時間の評価から，クライアントアプリケーションのスケーラビリティに問題があった．2^{16} 以上のホスト数ではホ

スト情報の処理の方法やリソースの問題で処理速度の性能が著しく低下した．そのため，処理の方法やメモリなどのハードウェアのリソースを増設するなどより多くのホスト扱えるように考慮する必要がある．

7.3. 評価まとめ

本研究において開発したシステムの評価をおこなった．評価方法は２つで，１点がホスト発見精度の評価を行った．２点目はクライアントアプリケーションを使って，実際にサーバ-クライアント間で 5.1.4 で定義したトランザクション処理を行ってホスト情報が表示されるまでの応答時間の評価を行った．ホスト発見精度の評価では，バイナリツリーに格納されているホスト情報と DHCP ログとの差分をとった．IP アドレスと MAC アドレスが一致したことから DHCP ログが保持しているホスト情報と一致した．しかし，ポートミラーしたときの L2 スイッチでのパケット破棄や PCAP への入力まえでのパケットドロップなど，考慮しなければならない問題があった．また，高負荷がかかるネットワークやポートミラーしたポートに計算機を直接接続できない場合などの課題もある．そのため，アドレス設定サービスとの連携手法を採用するなどの検討が必要である．

応答時間の評価ではクライアントアプリケーションがホスト情報を表示するところまでを応答時間とし，実験をおこなった．使用したデータは実際にパケットキャプチャにて取得するデータと同様の形式をとり，バイナリツリーのノードがすべて右側に格納される最も検索処理が遅くなる場合を想定して行った．結果は 2^{16} 以上から著しく応答時間が悪化した．この問題はベンダー情報の画像を検索する処理の方法やメモリなどのハードウェアのリソースに起因する．結果として，大規模なホストを持つネットワークで運用する場合はリソースの増設や，画像を検索する処理を別の方法で行い改善する必要がある．

第8章
今後の課題

本章では本研究の今後の課題に関して議論する．これまでグローバルコンピューティングに基盤システムとして，ホスト情報取得および可視化システムの開発を論じてきた．今後の基盤システムとしての課題として以下があげられる．

1. 複数ネットワークでの運用

2. ノード間の情報収集手法の実装

3. NAT 越えの実装

4. 静的アドレスへの対応

5. アクセス制御

1.複数ネットワークでの運用に関して述べる．現在は，ある大学院のネットワークに設置し，運用を行っている．パケットキャプチャにてホストの発見の手法を採用した．そのために，グローバルコンピューティングを導入するには，各ネットワークへパケットキャプチャサーバを設置する必要がある．家庭用ネットワークから他の大学ネットワークや様々な施設のネットワークへの導入である．グローバルコンピューティングの実現のために，各ネットワークのホストを利用できる

環境を構築しなければならない．そのために，複数のネットワークへ導入し，運用していく必要がある．

2. ノード間の情報収集手法の実装に関して述べる．1. に関連することだが，複数ネットワークで運用した場合に，どのような手法で分散されているホスト情報を収集してくるのかを検討しなければならない．各ネットワークに設置されているパケットキャプチャサーバを 1 ノードとする．すべてのノードより情報を収集することはデータ量増量などで 7 章で評価したクライアントアプリケーションのレスポンスを下げることにつながる．また，ネットワークトラヒックの増量から，帯域の狭いネットワークなどへは負荷をかけることになる．こうしたことを考慮すると，他のネットワークのホストを利用する場合は，適宜該当するホストがアクセスしているネットワークのノードからホスト情報を取得するなどの手法を検討する必要があるからである．さらに，ノードに不具合が出た場合にどのようにバックアップをとるのかなども検討しなければならない．一つのノードの不具合によってそのネットワークのホストを利用することができなくなってしまうからである．よって，ノード間の情報収集手法の実装を課題に挙げた．

3.NAT 越えの実装に関して述べる．各ホストはネットワークによって NAT の配下にある場合がある．その場合はホスト間の自律的な通信を行うためには NAT 越えを可能にする機能が必要である．NAT 越えができなければ，NAT 配下にあるホストを利用することはできない．よって，グローバルコンピューティング実現のために基盤システムの一つとして NAT 越えの機能を持たなければいけない．NAT 越えの手法は STUN や UPnP といった手法があるが，各手法の問題を考慮し，NAT 越えを実現しなければならない．

4. 静的アドレスへの対応に関して述べる．ホスト発見で設計したモジュールは４つの IP アドレス設定手法のモジュールであった．これらはネットワークへアクセスしてきたホストに動的 IP アドレスを設定するプロトコルである．そのため，各ネットワークに固定された IP アドレスを持つホストを発見することはできない．すべてのものを対象としたグローバルコンピューティングにおいて，静的 IP を持つホストもまた対象である．よって，静的 IP を持つホストもまた，動的に発見する機能を有する必要があり課題としてあげた．

5. アクセス制御もまた今後の課題である．利用されては問題になる機器もある．そのため，クライアントアプリケーションに表示させないなどの機構も必要である．また，不正なホストの場合もある．不正をしていることを見抜き，そのホストをグローバルコンピューティングのネットワークにアクセスさせないようにするなどの対策も必要である．

第9章
結　　論

本論文ではグローバルコンピューティングとしてのホスト情報取得および可視化システムの提案を行い，設計，実装を行った．現在では我々身の回りの物がIPネットワークへアクセスされるようになった．こうした背景からより多くのものがIPネットワークへとアクセスされるようになる．我々身の回りの物がIPネットワークへとアクセスされるようになった事柄を1章にて述べた．我々の身の回りの物がIPネットワークにアクセスしている環境で，ホストを管理，制御，あるいはホスト間の自律的な通信を行い，すべてのものをIPネットワークで利用することができる環境であるグローバルコンピューティングの提案を2章で行った．そして，3章ではグローバルコンピューティングに関連する技術に関して現状と課題について議論した．グローバルコンピューティング実現に向けて，各IPアドレス設定手法に対応する基盤システムが必要であることを示した．4章ではグローバルコンピューティングに関連する技術に関して現状と課題をふまえ，グローバルコピューティングとしてのホスト情報取得および可視化システムの提案をおこなった．そして，各IPアドレス設定手法に対応した基盤システムとなるべく，ホストのリアルタイム検知に関してその手法を議論した．そして，アドレス設定用のプロトコルのパケット取得の手法を使用することを導いた．この議論を踏まえて，5章，6章では設計と実装を行った．7章では，グローバルコンピューティングの基盤システムとして実装したシステムが妥当かの評価をおこなった．ホストのリアルタイム検知にはパケットドロップといった問題があるため，そういった問題を回避できているのかなどの評価をおこなった．最後に8章では今後の課題に関して議論した．今後の課題としてグローバルコンピューティング実現に向けた基盤システムとしての課題を挙げた．本研究はグローバルコンピューティング

環境を実現する，基盤システムとしてホスト情報の取得および可視化を機能を持ち合わせたシステムであると言える．

謝　　辞

本研究は NUS(National University of Singapore) と慶應義塾大学大学院メディアデザイン研究科によって，共同設立された研究機関 CUTE センターに支援いただいた研究である．

本研究の主指導教員であり，２年間ご指導いただいた慶應義塾大学大学院メディアデザイン研究科杉浦一徳准教授に感謝いたします．日々成長を感じることができる充実した２年間を過ごすことができました．また，激辛カレーや四川料理をはじめ充実した食生活もおくることができ，３キロ肥やすことができました．

研究が行き詰まったときに幅広い知見から助言していただいた本研究の副指導教員加藤朗教授に感謝いたします.「とろける」２年間を過ごすことができました．

本研究の副査していただきまし古川享教授に感謝いたします．大変豊富な経験談は大変心躍るお話ばかりで，日々ときめきを感じていました．

慶應義塾大学大学院メディアデザイン研究科研究科長稲蔭正彦教授に感謝いたします．私が慶應義塾大学環境情報学部時代からの３年間稲蔭正彦研究室 (imgl) に所属させていただきすばらしい時間過ごすことができました．JST/CREST プロジェクトなど大変貴重な経験をすることができました．

慶應義塾大学大学院メディアデザイン研究科教授砂原秀樹教授に感謝いたします．Live E!プロジェクトのウェザーセンサデータを使用したフリミフラズミの開発研究では，Live E!シンポジウムの機会を与えていただき貴重な経験をすることができました．

慶應義塾大学メディアデザイン研究科植木淳朗講師に感謝いたします．imgl 時代から Surroudings Project に指導していただきました．私の環境情報学部時代，先輩として的確な助言は大学院での研究行っていく上での基盤をつくっていただいたと感じています．また，学部３年時に私が起こした事件によって，私が家な

き子になったときは熱い抱擁によって励ましをしていただきました.「生きててよかった」とおっしゃっていただいた時は涙が止まりませんでした.

慶應義塾大学メディアデザイン研究科徳久悟講師に感謝いたします.JST/CREST プロジェクトにおいてフリミフラズミの研究ではご指導いただきました.imgl の先輩として,研究科の講師として助言をしていただいたことに感謝いたします.

慶應義塾大学メディアデザイン研究科博士課程遠峰隆史氏に感謝いたします.研究領域であるネットワークに関して,的確な助言をいただきましたことを感謝いたします.

慶應義塾大学メディアデザイン研究科博士課程山内正人氏に感謝いたします.研究が行き詰まった際の的確な助言がなければ研究を進めることができませんでした.

グローバルコンピューティング,メディアテレスコープの皆様に感謝いたします.日頃楽しい 2 年間を過ごすことができたのも皆様のおかげです.

ネットワークメディアの皆様に感謝いたします.皆様のおかげですばらしい経験することができました.助言などもいただきました.

National University of Singapore, Mixed Reality Laboratory の皆様に感謝いたします.Cute センタープロジェクトとして研究の支援をしていただきました.また出張の際の調整や,研究物資の購入等,研究を進めることができたのは皆様のおかげです.

最後に両親に感謝いたします.当初は進学を反対した両親ですが修士の間支援していただいたことを深く感謝いたします.

以上をもって本研究の謝辞とさせていただきます.

参 考 文 献

[1] Global Computing Project. http://www.mixedreality.nus.edu.sg/index.php/about-us/research/.

[2] Universal Plug and Play. http://upnp.org/.

[3] Bonjour. http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/NetServices/Introduction.html.

[4] Zero Configuration Networking. http://www.zeroconf.org/.

[5] iCAR Working Group, WIDE PROJECT. http://www.wide.ad.jp/project/wg/iCAR-j.html.

[6] WIDE PROJECT. Wide 報告書, 第 1 3 章. Technical report, WIDE PROJECT, 2001.

[7] Twitter. http://twitter.com/.

[8] Facebook. http://www.facebook.com/.

[9] mixi. http://mixi.jp/.

[10] Google. http://www.google.co.jp/.

[11] Energy conservation and homecare network. http://www.echonet.gr.jp.

[12] Digital Living Netwokr Appliance. http://www.dlna.org/home.

[13] Open services gateway iinitiative alliance. http://www.osgi.org/Main/HomePage.

[14] 稲垣 勝利 戸崎 明宏樋口 正生. Pioneer r&d vol.11 no.2. 家庭内 AV ネットワーク技術「HAVi」の概要, pp. 39–49. Pioneer, 2008.

[15] 阪田史郎. 情報家電ネットワークと通信放送連携. 情報家電ネットワークと通信放送連携-IPTV で実現する家庭内ユビキタス, pp. 7–24,117–175. 電気学会, 2008.

[16] Jini. http://www.jini.org/wiki/Main_Page.

[17] R. Droms. Rfc2131, dynamic host configuration protocol. Technical report, IETF, 1997.

[18] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, Microsoft Corporation, Shivaun Albright, Hewlett-Packard Company. Simple service discovery protocol/1.0 operating without an arbiter. Technical report, IETF, 1999.

[19] S. Cheshire, Apple Computer, B. Aboba, Microsoft Corporation, E. Guttman, Sun Microsystems. Rfc3927, dynamic configuration of ipv4 link-local addresses. Technical report, IETF, 2005.

[20] UPnP Forum. Upnp device architecture 1.1. Technical report, UPnP Forum, 2008.

[21] W. Simpson. Rfc1548, point to point protocol. Technical report, IETF, 1994.

[22] Ip 枯渇タスクフォース. http://www.kokatsu.jp/blog/ipv4/.

[23] UPnP Forum. Upnp device architecture v1.0 annex a - ip version 6 support. Technical report, UPnP Forum, 2002.

[24] S. Thomson, Cisco, T. Narten, IBM, T. Jinmei, Toshiba. Rfc4862, ipv6 stateless address autoconfiguration. Technical report, IETF, 2007.

[25] Ed. R. Droms, Cisco, J. Bound, Hewlett Packard, B. Volz, Ericsson, T. Lemon, Nominum, C. Perkins, Nokia Research Center, M. Carney, Sun Microsystems. Rfc3315, dynamic host configuration protocl ipv6. Technical report, IETF, 2003.

[26] N. Kushalnagar, Intel Corp, G. Montenegro, Microsoft Corporation, C. Schumacher. Rfc4919, ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals. Technical report, IETF, 2007.

[27] Apple Inc. Bonjour overview. Technical report, Apple, 2006.

[28] J. Postel, ISI. Rfc792, internet control message protocol. Technical report, IETF, 1981.

[29] Ping man page. `http://linuxjm.sourceforge.jp/html/netkit/man8/ping.8.html`.

[30] Ryan Blue, Cody Dunne, Adam Fuchs, Kyle King, Aaron Schulman. Visualizing real-time network resource usage. *izSec '08 Proceedings of the 5th international workshop on Visualization for Computer Security*, pp. 119–35, 2008.

[31] Peter Valian and Southwestern University Todd K. Watson. Netreg: An autmated dhcp registration system. Technical report, SysAdmin: the journal for UNIX systems administrators, 2000.

[32] Pcap. `http://www.tcpdump.org/pcap.html`.

[33] D. Levi, Nortel Networks, P. Meyer Secure Computing Corporation, B. Stewart Retired. Rfc3413, simple network management protocol. Technical report, IETF, 2002.

[34] Oui. `http://standards.ieee.org/develop/regauth/oui/public.html`.

[35] Posix threads programming. https://computing.llnl.gov/tutorials/pthreads/.

[36] OSGi Alliance. Osgi service platform core specification. Technical report, OSGi Alliance, 2008.

[37] L. Mamakos, K. Lidl, J. Evarts, Inc. UUNET Technologies, D. Carrel, D. Simone, Inc. RedBack Networks, R. Wheeler, Inc RouterWare. Rfc2516, point to point protocol over ethernet. Technical report, IETF, 1999.

[38] J. Chapman, Inc. D. Coli Cisco Systems, Inc. A. Harvey Cisco Systems, Inc. B. Jensen Cisco Systems, Inc. K. Rowett Cisco Systems. Rfc1841, ppp network control protocol for lan extension. Technical report, IETF, 1995.

[39] G. McGregor, Merit. Rfc1332, the ppp internet protocol control protocol. Technical report, IETF, 1992.

[40] J. Rosenberg, J. Weinberger, dynamicsoft, C. Huitema, Microsoft, R. Mahy, Cisco. Rfc3489, stun - simple traversal of user datagram protocol (udp) through network address translators (nats). Technical report, IETF, 2003.