

自立運用小規模ネットワークの構築に関する 研究

平成 15 年度 博士論文

東京工業大学 大学院情報理工学研究科 数理・計算科学専攻
木本雅彦

2004 年 3 月 31 日

概要

本研究では、近年需要が増加している家庭やSOHOなどの末端の小規模ネットワークを研究対象とする。これを自立運用ネットワーク (ISNA) と呼称し、ISNA の管理上の問題を低コストで解決することを目的とする。

最初に、この背景に基づいた問題提起の詳細を述べる。ISNA ではサーバと同等の機能を稼働させるため、その管理運用にはサーバと同様の条件が求められる。その結果として発生する問題を指摘し、その問題を管理コストがかけられないISNA で実現するには新しい管理技術が必要なことを指摘する。更にその技術の要件を明らかにし、ISNA に適した管理項目の力点の配分を述べる。

次に PICKLES と POPS という、統一環境を容易に保守する機構について述べる。これらの機構により、分散した多数のホストでの環境の構築や更新作業と故障時の復旧作業を容易にした。

次に小型アクセスルータ上での IPv6 プロトコルスタックの設計と実装について述べる。この実装により、ISNA に対して End to End の通信基盤を提供した。

続いて、ネットワーク可聴化システムについて述べる。このシステムにより、ネットワーク管理作業における、音声インタフェースの有効利用の可能性を示した。

最後にこれらの技術の有効性について議論する。提案した ISNA の管理モデルと、それに基づいて開発した要素技術を用いることにより、技術的、人的、金銭的資源に制限があるという ISNA の条件下でも、安全で安心なネットワークを容易に構築できることを示す。

目次

第1章	はじめに	13
1.1	研究の背景	13
1.2	自立運用ネットワークの管理モデル	14
1.3	本研究の成果	15
1.4	論文の構成	16
第2章	自立運用ネットワークの管理モデル	17
2.1	小規模なネットワークの重要性	17
2.1.1	インターネットの需要の増加	17
2.1.2	インターネットの利用形態の多様化	19
2.1.3	小規模なネットワークが抱える問題	20
2.2	自立運用ネットワークの提案	21
2.2.1	自立運用ネットワーク (ISNA) の定義	21
2.2.2	ISNA の管理に求められる要件	23
2.3	ネットワーク管理手法の比較	24
2.3.1	既存のネットワークの管理モデル	24
2.3.2	既存のネットワーク管理モデルの適用事例	27
2.4	ISNA の管理モデル	30
2.4.1	ISNA の管理モデルの提案	30
2.4.2	ISNA を実現する技術的解決方法	31
2.4.3	ISNA を支える要素技術の開発	32
第3章	自立運用ネットワークの管理を支援する統一環境の構築	35
3.1	自立運用ネットワークの管理のための統一環境の有用性	35
3.2	PICKLES SYSTEM	37
3.2.1	PICKLES の概要	37
3.2.2	PICKLES SYSTEM の管理モデル	38
3.2.3	PICKLES SYSTEM の実装	40
3.2.4	PICKLES SYSTEM の運用事例	43
3.2.5	PICKLES SYSTEM に関する資料	45
3.3	共通の利用者環境の提供	45
3.3.1	FreeBSD におけるアプリケーションの導入	45
3.3.2	POPS(パッケージ集) の設計	46
3.3.3	POPS の実装	49
3.4	情報の明確な分類と分離	50

3.4.1	情報の分離の方針	50
3.4.2	情報の分離の実現	51
3.5	評価と考察	53
3.5.1	他の機構との比較	53
3.5.2	有用性の評価と考察	55
3.5.3	技術的考察	55
第4章	小規模ネットワークの構築に適した IPv6 対応 ISDN ルータの実装と評価	59
4.1	IPv6 ISDN ルータの開発経緯	59
4.1.1	SOHO での対外接続	59
4.1.2	ISDN を用いた接続方法	60
4.2	ISDN ルータ用 IPv6 プロトコルスタックの設計と実装	61
4.2.1	設計	61
4.2.2	実装	62
4.3	IPv6 プロトコルスタックの評価	64
4.3.1	ICMP 処理の速度評価	65
4.3.2	安定性の評価	65
4.3.3	通信速度の計測と評価	66
4.3.4	混在環境での速度計測	68
4.3.5	相互接続試験	68
4.4	IPv6 対応 ISDN ルータの実装に関する考察	69
4.4.1	安定性の考察	69
4.4.2	性能評価の考察	69
4.4.3	現在の技術に与える効果	71
第5章	ネットワークトラフィック可聴化システムの設計と実装	73
5.1	ネットワーク監視手法の比較	73
5.1.1	管理作業の分析	73
5.1.2	監視方法の比較	74
5.1.3	音声を用いたネットワークの監視	75
5.1.4	研究の目標	76
5.2	関連研究	76
5.3	stetho システム	77
5.3.1	stetho システムの概要	77
5.3.2	stetho システムの設計	78
5.3.3	stetho システムの実装	78
5.4	評価実験	81
5.4.1	実験目的	81
5.4.2	実験 A	82
5.4.3	実験 B	85
5.5	考察	88

5.5.1	音楽的考察	88
5.5.2	評価実験結果の考察	89
5.6	今後の課題	92
第6章	本研究がもたらした効果と ISNA 管理モデルの妥当性	93
6.1	要素技術が現在の技術に与えた効果	93
6.1.1	統一環境が与えた効果	93
6.1.2	IPv6 対応 ISDN ルータが与えた効果	95
6.1.3	ネットワーク可聴化システムが与えた効果	95
6.2	ISNA の管理モデルと管理技術の考察	96
6.2.1	実現できた要件	96
6.2.2	ISNA の管理技術の今後の課題	97
6.2.3	ISNA のセキュリティ	98
6.2.4	ISNA の管理モデルの妥当性	99
6.3	今後の展開	102
6.4	今後の社会に与える効果	103
第7章	結論	107
付録A	PICKLES SYSTEM 管理の手引き	119
A.1	PICKLES SYSTEM の導入	119
A.1.1	インストールの概要	119
A.2	基本的な設定	119
A.2.1	configuser の使いかた	119
A.2.2	電子メールの設定	121
A.2.3	ネットワークの設定	121
A.2.4	X-WindowSystem の設定	122
A.2.5	WWW の設定	123
A.3	利用者管理	123
A.3.1	利用者の新規登録/削除	123
A.3.2	標準環境 (設定ファイル等の雛型)	125
A.3.3	利用者情報が格納されているファイル群	125
A.4	ディスクの管理運用	126
A.4.1	ディスク管理の方針	126
A.4.2	ハードディスク構成	126
A.4.3	ハードディスク構成の種類	126
A.5	各種サーバの設定	128
A.5.1	WWW サーバ	128
A.5.2	proxy サーバ	129
A.5.3	IP ルータ	130
A.5.4	DHCP サーバ	130
A.6	アプリケーションの保守	131

A.6.1	新しいアプリケーションのインストール	131
A.7	バージョンアップ	131
A.7.1	バージョン情報の参照	131
A.7.2	システムディスクのバージョンアップ	131
A.7.3	ユーザディスクのバージョンアップ	131
A.7.4	syncuser スクリプト	131
A.8	カーネルの再構築	132
A.8.1	再構築の手順	132
A.8.2	カーネルの再構築に関してよくある質問	133
付録 B	PICKLES FAQ	135
B.1	PICKLES 概要	135
B.2	インストール	137
B.3	ハードウェアコンパチビリティ	139
B.4	システム管理	140
B.5	ユーザアプリケーション	141
B.6	ウインドウシステム	142
B.7	サーバ構築	143
B.8	PICKLES の入手について	144
付録 C	ぷらっとホーム製 PICKLES 対応 PC 仕様書	145
付録 D	PICKLES 対応 PC 仕様書 (1996 年版)	151

目次

2.1	インターネットの人口普及状況	18
2.2	ブロードバンド・アクセスの加入数の推移	18
2.3	世帯・企業・事業所でのインターネット普及率	19
2.4	TMN のネットワーク管理アーキテクチャ	25
2.5	開発/構築した要素技術	33
3.1	カード型端末	38
3.2	ハードディスクモジュール	39
3.3	PICKLES 端末試作機	41
3.4	ディスクの構成	43
3.5	ファイルシステムの構成	52
4.1	ISDN ブリッジを用いたネットワーク構成	60
4.2	IPv6 対応 ISDN ルータを用いたネットワーク構成	61
4.3	WS-One のタスク構造	63
4.4	1999 年 9 月 6 日の結果	66
4.5	1999 年 9 月 7 日午前の結果	66
4.6	1999 年 9 月 7 日午後の結果	67
4.7	1999 年 9 月 8 日午前の結果	67
4.8	1999 年 9 月 8 日午後の結果	68
4.9	実験ネットワーク	69
4.10	RouterX の設定	70
5.1	stetho システムの設計	78
5.2	設定ファイル記述例	80
5.3	stetho 内部データ構造	82
5.4	実験結果グラフ	84
5.5	実験結果グラフ (変換後)	84
5.6	実験システム B	86
5.7	実験 B 実験回数に対する反応時間	87
5.8	実験 B 実験回数に対するロスト率	87
5.9	実験 B 発生速度に対する反応時間	88
5.10	実験 B 発生速度に対するロスト率	89
5.11	実験 B 発生速度に対するボタンごとのロスト率 (exp1)	90
5.12	実験 B 発生速度に対するボタンごとのロスト率 (exp3)	91

表 目 次

3.1	モジュールごとの情報の内容	40
3.2	アプリケーションの利用傾向	47
3.3	情報の分類	51
3.4	クラスタリングソフトウェアの分類	54
4.1	ttcp による転送速度 (Kbps)	67
4.2	IPv4, IPv6 混在環境での転送速度 (Kbps)	68
5.1	実験系のホスト	82

第1章 はじめに

1.1 研究の背景

近年のインターネットの普及度の変化については、量の変化と質の変化という二つの側面から論じることができる。量の変化については、利用率の増加、特に小規模ネットワークの利用率が増加している。政府が発行した資料 [69] によれば、平成14年末の時点で既に98%以上の企業がインターネットを利用しているのに対して、家庭や事業所などの小規模組織では80%前後である。しかし後者はこの数年で著しい増加を見せており、このような小規模な組織でのネットワークの需要が増加していると言える。質の変化については、利用形態の多様化が挙げられる。アクセス網の高速化、多様化、低価格化により、様々な手段でインターネットに常時接続することが可能になった。更にこのような通信インフラを利用して、P2Pアプリケーションやインターネット電話など、さまざまな使いかたを利用者は求めるようになってきている。これらのアプリケーションの多くは、End to End、すなわち末端のホスト同士が直接通信するモデルを用いているという特徴がある。加えて、今後は外部から家庭などにアクセスするという使いかたが想定できる。例えば、家庭のHDDビデオレコーダにアクセスして予約設定や録画映像の視聴が可能な製品が、既に登場している。このようなアプリケーションは機能的にサーバと同等である。

すなわち、今後は家庭などの末端のネットワークの利用が更に増加すると同時に、それらのネットワークでサーバ機能を稼働させる使いかたが増加すると予測できる。サーバ機能を動作させることにより、「複雑化するソフトウェア構成・通信方式への対応」「サービスの安定稼働に対する要求の増大への対応」「十分なセキュリティの確保」などの問題が生じる。サーバ機能は安定して動作するように適切な管理がなされる必要があるし、End to Endの通信が容易かつ安全に行えるような通信基盤を容易する必要がある。当然、セキュリティの確保も必要になる。これらの問題は、従来からサーバの運用では発生していた問題であるが、従来は人的金銭的労力を投入することで解決していた。

従来のネットワーク管理モデルは、大規模なネットワークを対象としたものが主である。代表的な物にITU-T [7] によるTMN[33] やOSI管理モデルなどが挙げられるが、いずれも大規模な基幹網を対象としている。これらのモデルを実装したシステムは多数存在するが、システム自体が高価であったり、個別の作業に専任の作業者を従事させることを前提しているため、ISNAでそのまま利用することはできない。ISNAでは技術的、人的、金銭的資源に制限があるからである。

かといって、現状の個人利用者のように適切な管理技術を用いず、セキュリティ問題などの不具合が発生したら事後対策する方式では安全なネットワークを構築で

きない。これはウイルスなどの被害件数からも明らかである。技術的、人的、金銭的資源に制限がある条件のもとで、安全かつ安心なネットワークを容易に構築する技術を確立する必要がある。

これは将来の課題のように見えるが、現在の状況の中で同じ問題を抱えるものを探すと、大学の研究室や SOHO のように一通りのサーバを独自で稼働させている小規模なネットワークが挙げられる。本論文ではこれを、自立運用ネットワーク (ISNA:Independent Small scale Network Architecture) と呼称する。ISNA の規模は利用者が一名から十数名程度の、単一から数個程度のセグメントから構成されるネットワークであり、DNS を始めとしたサービスを一通り自立して運用しているという特徴を持つ。このような ISNA の運用技術を確立することにより、現在の小規模ネットワークの管理効率を向上させると同時に、将来の末端のネットワーク管理を容易かつ安全に実現できるようになる。

1.2 自立運用ネットワークの管理モデル

本論文では、TMN モデルを参考にした ISNA の管理モデルを提案する。TMN モデルの管理項目のうち、ISNA では「構成管理」「障害管理」「性能管理」「機密管理」を実現する必要がある。しかしその特性から、すべてに均等に力を入れる必要はなく、力を入れるべき内容もコストを重視する必要があるという特徴がある。

構成管理と障害管理については、ISNA の特性を考慮する必要がある。機器構成は頻繁に変更される可能性があるし、障害に備えて二重系を用意するといった高コストの予防策は利用できない。機密管理については、ネットワークの規模に関わらず十分な水準のセキュリティを確保する必要がある。一方で性能管理については、利用者が求めるアプリケーションを利用できる環境を提供する必要があるが、その水準は一定のものが満たさせていけば良いと考えられる。したがって、性能管理は若干力の配分を落とし、他の三つの管理項目には注力すべきである。

以上のことから、ISNA の管理モデルを実現する技術の要件は以下になる。

- ソフトウェア構成を適切に管理できること。
- 低コストな監視と障害復旧が可能なこと。
- 安全で安心であること。
- End to End の通信が可能であること。

著者は ISNA の構築と運用を実現するために、さまざまな要素技術を検討し開発した。

多数の端末に統一した環境を容易に導入できるようにしたオペレーティングシステムパッケージを開発した。これは ISNA の管理モデルにおける、構成管理と障害管理にあたる。また、End to End の通信を行なうアプリケーションを快適かつ安全に利用するためには、フラットなアドレス空間のもとでのシームレスな通信が必要になる。このためには、小規模ネットワークであっても次世代インターネットプロ

トコル (IPv6) への対応が必須になる。著者が小規模ネットワークへの IPv6 の導入を検討し始めた時点では IPv6 に対応したアクセスルータが市場に存在しなかった。この問題を解決するため、SOHO 向けアクセスルータの IPv6 プロトコルスタックを実装した。加えて、障害管理のための監視システムとして、トラフィック可聴化システムを開発した。

1.3 本研究の成果

本研究では、まず ISNA の重要性を示し、その管理モデルが既存のものでは不適切であることを示した。次の既存の管理モデルを ISNA に適用した場合の問題点を指摘し、そこから ISNA に適した管理項目の力点の配分を提案した。また、ISNA の管理モデルを実現する技術として、以下の成果を実現した。

- PICKLES と POPS という、統一環境を容易に保守する機構の開発を行ない、構成管理や障害管理作業の大幅な簡易化を実現した。
- 小型ルータへの IPv6 プロトコルスタックの実装を行ない、末端の組織への End to End の通信基盤を提供した。
- ネットワークトラフィック可聴化システムの実装と評価を行ない、音声を用いることによりネットワークの監視作業の低コスト化を実現できることを示した。

この研究で開発した要素技術によって、以下の効果がもたらされた。

統一環境の実装により、ソフトウェア構成を一元管理し、構成情報の把握を容易にした。同時にシステムの導入更新作業や、障害発生時の代替機能への切り替え作業を、技術力の低い管理者でも従事可能にした。これらのことから、管理コストの大幅な軽減を実現したとともに、すべての機器に同じセキュリティ水準を保証できる。この技術は、国内論文誌に採録されて評価されている。またこの技術を元にしたシステムは、著者らが日常的に用いているサーバで現在も継続して運用されているだけでなく、通信総合研究所が開発した DDoS シミュレータを構成する百台の計算機に導入されている。また内部で用いるだけでなく、統一環境をまとめた CD-ROM を配布し、広く利用されている。IPv6 対応の SOHO ルータの開発については国際会議と国内論文誌において評価されると同時に、低価格帯のアクセスルータとしては世界唯一の IPv6 対応製品として出荷されている。ネットワーク可聴化システムにより、人的資源が乏しい ISNA でも他の作業と並行してネットワークの監視作業が行なえるなどの、障害管理の低コスト化を実現した。このシステムは国際会議に採録されて評価されている。

これらの要素技術を用いることで、ISNA の管理は容易で安全かつ安心なものになる。そのことを、ISNA のモデルケースである著者らの研究室ネットワークの運用経験から示した。

達成できていない項目については、開発した当時の技術的制約を受けていた部分もあるし、結果的に後手に回った部分もある。しかし ISNA の管理モデルを確立し

たことによって、達成できた部分と達成できていない部分とが明確になり、今後の展開を打ち出すことができる。

構成管理については、移動する機器の管理が実現していない。これについては、Mobile IP などの移動体通信技術や、RF-ID タグなどの技術を今後導入できるであろう。障害管理については、動作ログなどを解析することによって、今後の障害予測に活用できる可能性がある。機密管理については、著者らが構築した ISNA ではセキュリティポリシーとその実装がほぼ同一であったが、本来はポリシーを定めた後にそれを実装するという手順である。むしろ実装からポリシーを生成するという方針もあるが、いずれにせよ明文化したポリシーの策定技術については検討する必要がある。性能管理については、利用者が求めるサービスを提供できるか否かが論点になる。End to End のシームレスな通信基盤として IPv6 のアクセスルータを開発したが、実際にその上でどのようなサービスを展開し、利便性とセキュリティをどのように共存させるかなどについては、更に検討する余地がある。

1.4 論文の構成

2章では ISNA の定義と、その管理モデルについて述べる。資料をもとに小規模なネットワークに対する需要が著しく伸びていることと、それをとりまく通信インフラの変遷を述べ、大規模ネットワークと小規模なネットワークの運用技術の差異について述べる。それを基に、小規模なネットワークの特徴を抽出した ISNA の管理モデルについて述べる。また、管理モデルに基づいて著者が開発した様々な要素技術を列挙する。

3章、4章、5章では、要素技術のうちの自立運用ネットワーク向け統一環境の実装と、IPv6 対応 ISDN ルータの実装、ネットワークトラフィック可聴化システムの実装について述べる。

最後の6章にて、これらの要素技術が現在の技術に与える効果と今後の課題、ISNA の管理モデルの妥当性について述べる。

第2章 自立運用ネットワークの管理モデル

本論文は、ネットワーク構築とその運用を支援する技術を主題とする。特に焦点を置く対象は、小規模なネットワークであることと、そのネットワークが自立して運用される場合の問題解決である。

本章では小規模なネットワークの重要性と、その管理手法に新しいモデルを提案する必要があることを述べつつ、解決策を提示する。

2.1 小規模なネットワークの重要性

2.1.1 インターネットの需要の増加

インターネットの利用者が年々増加していることは、改めて言うまでもなく明らかである。通信インフラストラクチャとしても情報交換や情報収集のメディアとしても、インターネットは極めて身近なものになり、多くの人が日々インターネットを利用している。まず総務省が発行している情報通信白書 [69] を概観しつつ、現在のインターネットの普及状況について述べることにする。

平成 14 年末における我が国のインターネット利用人口は 6,942 万人 (対前年比 24.1 % 増) と推計され、1 年間で 1,349 万人増加している。人口普及率は 54.5 % と始めて半数を超え、国民の 2 人に 1 人はインターネットを利用している状況になった (図 2.1)。特に平成 11 年度以降のブロードバンドの普及に注目すると、加入数の推移は図 2.2 のようになっている (既出の情報通信白書による)。

ここで注目すべきは、世帯・企業・事業所でのインターネット普及率 (図 2.3) である。企業普及率は平成 14 年末で 98.4 % であり、ほぼ全ての企業がインターネットを利用している。世帯普及率と事業所普及率の二つは、ほぼ同じカーブ描いて上昇している。平成 9 年では世帯普及率が 6.4 %、事業所普及率が 12.3 % であったのに比べ、平成 14 年ではそれぞれ 81.4 % と 79.1 % になっている。つまりこの 4 年間で家庭と事業所という小規模な組織が顕著な増加を見せている。

これらの資料から以下の点が導かれる。

- 一般利用者向けのアクセス網の高速化と多様化が進んでいること。
- この数年では、家庭などの小規模なネットワークからの利用者が特に増加していること。

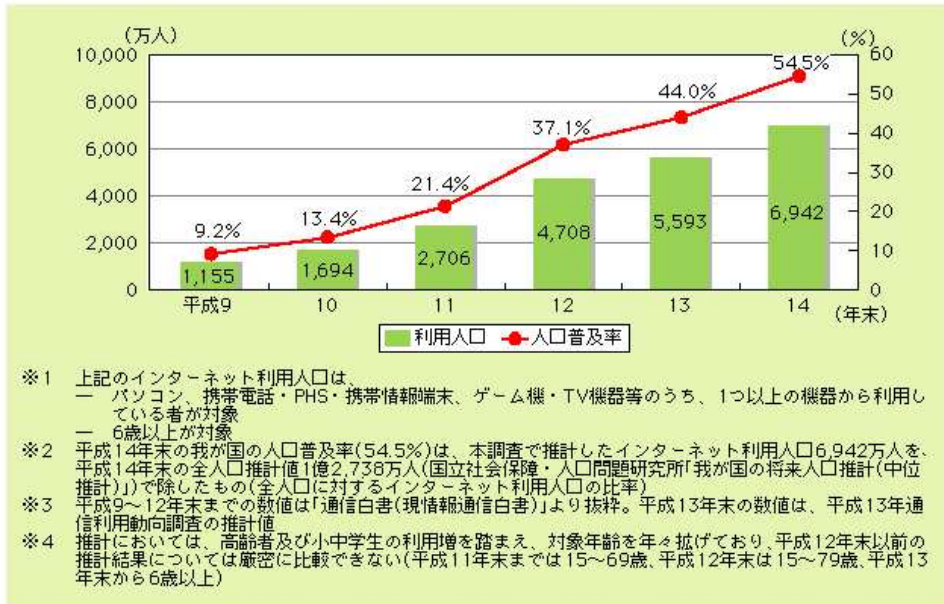


図 2.1: インターネットの人口普及状況

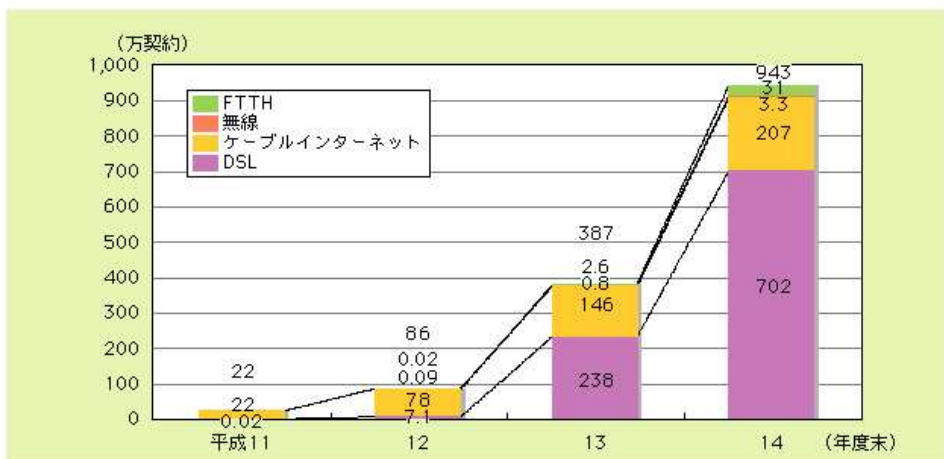
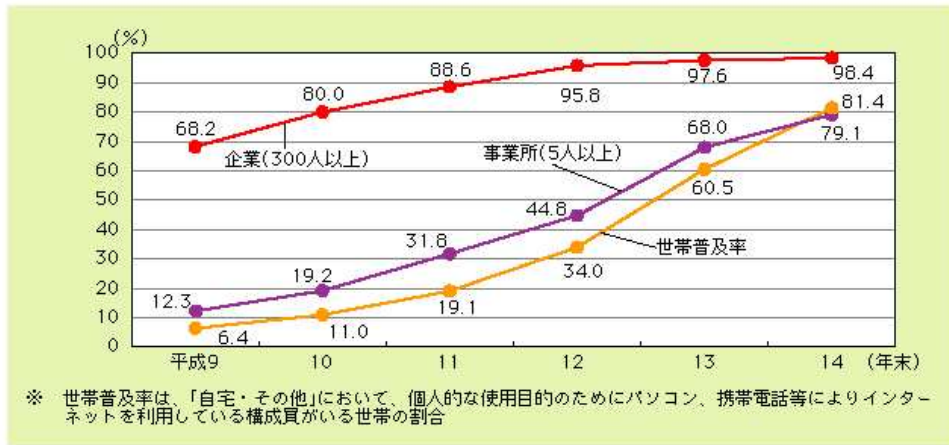


図 2.2: ブロードバンド・アクセスの加入数の推移



図表①、② (出典)総務省「通信利用動向調査」

図 2.3: 世帯・企業・事業所でのインターネット普及率

2.1.2 インターネットの利用形態の多様化

振り返るに、著者が小規模ネットワークの運用というテーマで研究を行なった1995年以降は、小規模組織が利用できる通信インフラが劇的に変化した時代であった[60][52][22]。大きな流れとして、NTTの回線を用いたダイヤルアップを利用するか高価な専用線以外の通信手段がなかった時代から、CATV、FTTH、ADSLが登場して多様な選択肢を選べるようになった。これと並行して起こった価格競争により、通信費用は年々低価格になっている。

CATVやフレッツISDNが登場する前は時間課金による価格体系であり、また当時の専用線サービスはもっとも低価格なものでも月額30,000円程度であったため、家庭やSOHOでは間欠接続を使わざるを得なかった。その後固定料金サービス(当初のテレホーダイは時間限定であったが)が始まり、ADSLの普及により常時接続が当たり前になった。また、家庭内に複数のネットワーク機器が置かれることも当たり前に行なわれるようになったため、家庭内にLANを構築し、インターネットへの接続にルータを設置するようになった。通信白書などには現れていないが、この点には十分注目しなければならない。当初はアナログモデム経由で1台のコンピュータだけを間欠接続していた家庭やSOHOが、ルータを経由した常時接続体制に変わり、この結果常に繋がっているインターネットの一部となったのである。

そこで利用するサービスも多様化している。初期の家庭での利用者は、WWWと電子メールが主な利用目的であった。両者は同時にインターネットを爆発的に普及させたキラーアプリケーションでもある。しかし昨今では、P2P(Peer To Peer)と呼ばれる末端の端末同士が直接通信するモデルのアプリケーションや、ネットワークゲーム、Instant Message、VoIP(Voice over IP)など多様なアプリケーションを利用できることが求められている。これらのアプリケーションの多くが持つ特徴として、End to End、すなわち末端のホスト同士が直接通信することが求められている点が挙げられる。また現在研究がすすめられているGrid Computing[71]のモデルも、理想的にはすべてのノード同士でシームレスな通信が行なえる環境を前提とし

ている。更に、今後は外部から家庭などにアクセスするという使いかたが想定できる。例えば家庭の HDD ビデオレコーダにアクセスして予約設定や録画映像の視聴が可能な製品が既に登場している。留意すべき点は、これらのアプリケーションを利用することは、サーバを稼働させることと機能的に等しいことである。

今後は家庭などの末端のネットワークの利用が更に増加すると同時に、それらのネットワークでサーバや同等の機能のものを運用するという使いかたが増加すると予測できる。このような環境では、IPv6 を前提としたフラットなアドレス空間が導入され、End to End の通信が行なわれる。その上で、セキュリティモデルや管理モデルを確立する必要がある。

2.1.3 小規模なネットワークが抱える問題

上記のような背景のもとで、以下のような問題が発生する。

- 複雑化するソフトウェア構成と通信方式に対応する必要がある。
- サービスの安定稼働に対する要求に対応する必要がある。
- セキュリティの確保が必要である。

順に詳細を述べる。

複雑化するソフトウェア構成と通信方式に対応する必要がある まず、End to End の通信が可能な通信基盤を用意しなければならない。なぜならば、現在そして今後求められるアプリケーションは、End To End の通信を要求するものが増加することが予測できるからである。現在の IPv4 のインターネットでは、末端のネットワークは接続点に NAT ルータを設置し、組織内 LAN はプライベートアドレスを用いている場合がほとんどである。この場合、外部のホストが内部ネットワークのホストに接続しようとしたら、NAT ルータに転送設定を行なうか、UPnP などの技術を用いる必要があるが、いずれも暫定的技術であり、本質的な解決方法ではない。フラットなアドレス空間で、シームレスな通信が行なえるような環境を前提とし、その上で更に安全性の確保の問題を考えるべきである。

サービスの安定稼働に対する要求 サーバとして稼働させるサービスは、安定して動作させなければならない。小規模なネットワークでも、適切にネットワークとサービスを管理する必要がある。

管理の定義については後述するが、ここではいくつか具体例を挙げておく。まず稼働させているサービスやそのアプリケーションのバージョンなどを適切に把握し、適切な構成を保たなければならない。従来の家庭や SOHO などの小規模なネットワークでは、導入されているアプリケーションやサービスの種類やバージョンなどは、ホストの管理者、利用者ごとに異なる事例が多い。このような状況では、例えば特定のバージョンの特定のアプリケーションに不具合

が発生した場合、どのホストに対して対策を施すべきなのかを調査するところから始めなければならない。

また、サービスが安定して動作するように、その挙動やネットワークの状態を監視し、障害が発生したら随時適切な対処をする必要もある。

セキュリティの確保 サーバを運用する以上、それを安全に運用しなければならない。しかし、サーバの安全性を維持するためには、通常大きなコストがかかる。

「日本のWEB改竄状況」¹の2002年11月から2003年10月までのデータを集計すると、合計262件のうちWindows系列のOSが127件で最多であるものの、Linuxが73件、その他UNIX系OSが49件となっている。Windows Updateなどで対策ができるはずのWindows系列のOSで脆弱性が放置されているのは論外としても、現在サーバを稼働させることが多いUNIXについても、脆弱性の対策が完全になされているわけではないと言える。

対策すべき脆弱性の数については、FreeBSDのセキュリティアドバイザリ²の数を参考にする。これによると、2002年一年間で報告された脆弱性の件数は50件である。これに要するコストを、通信総合研究所が外部に委託している保守業務の工数のデータから試算する。この業務では一つのアプリケーション(例えばOpenSSH)を最新版に更新する作業として、一つのホストについて約1時間と概算している。この時間は管理するホストの数に比例するが、1時間というのは熟練した管理者が作業した時間であって、未熟な管理者であれば現実的には更に時間を要するかもしれない。また、上記のセキュリティアドバイザリにはユーザが追加するアプリケーションの脆弱性は含まれていないものもある。例えば著名なWebサーバであるapacheには2002年分の脆弱性だけで5件が修正されてリリースされている³。

仮に20台のホストがあったとすると、一年間に55件の脆弱性対処作業を行なう必要があり、その時間は1,100時間になる。この作業を完全に業者に委託した場合、1人時間1万円という相場からすると年間1,100万円になる。これは小規模な組織にとっては大きな負担になる。

2.2 自立運用ネットワークの提案

2.2.1 自立運用ネットワーク (ISNA) の定義

ここまで述べた内容を整理する。現在、小規模な末端のネットワークの需要が増加していて、そこではサーバと同等な機能を稼働させている。そういった末端のネットワークでは様々な管理上の問題が発生するが、現在はこれを解決するために大きなコストを費している。このため、小規模で独立したネットワークの管理モデルを

¹<http://www.fearoot.com/>

²<http://www.freebsd.org/security/index.html>

³<http://www.apacheweek.com/features/security-13>

確立し、容易かつ安全で安心なネットワーク管理技術を提供することが求められている。

これは将来の課題のように思えるが、現在の状況の中で同じ問題を抱えるものを探すと、大学の研究室やSOHOなどのように一通りのサーバを独自で稼働させているネットワークが挙げられる。このようなネットワークの管理技術を現時点で議論することが、今後のネットワーク管理技術の発展につながることになる。

それでは、小規模で独立したネットワークとは具体的に何を指すのか。

利用者の規模としては一人から十数人程度の構成員を想定する。このような組織は、全体や一部が移動して稼働する可能性が高く、また他の組織の下を移動する場合もある。

ネットワークの規模としては、一つから数個程度のセグメントからなり、対外接続には民生用の手段を用いることが多い。民生用接続については、日々新しい技術が提供されているため、速度などの条件をつけることは難しいが、敷設が容易であることと低コストである必要がある。また、このネットワークは独立しており、内部でメールサーバやWWWサーバやDNSサーバなどの一通りのサービスを独自に稼働させている。特にDNSを稼働し、独自の名前空間を持つことは重要である。

また大規模ネットワークと小規模ネットワークの比較について述べた文献 [64] では、小規模ネットワークの特徴として以下を挙げている。

- 熟練した管理者が不足している。
- 利用者と管理者が明確には分離されていない。
- 潤沢な資金を確保できない場合がある。

つまり、「人的資源」「技術的資源」「物理的資源」において、大規模なネットワークと小規模なネットワークの運営には差異があり、全体としては小規模なネットワークのほうがこれらの制約が厳しい。

このような特徴を持つネットワークは小規模ながら自立して稼働することから、以降では自立運用ネットワークと称することにする。また略称として、ISNA(Independent Small scale Network Architecture) を用いることとする。

先に挙げた自前のサーバを運用しているSOHOや大学の研究室ネットワークは、ISNAの現状での具体的な事例である。このうち研究室ネットワークについては、一見専門的技術を持った集団だと思われるかもしれないが、必ずしもそうではない。理由は二つあり、すべての研究室に計算機ネットワークの専門家がいるわけではない点と、計算機ネットワークを専門とする研究室であっても未熟な新米管理者が管理にあたる例はあるという点である。しかしこういった研究室では、自分たちで独自のサービスを運用しようというモチベーションはあり、完全ではないながらもそれなりの技術力は持っている想定できる。

本研究が対象とするのは、そういったモチベーションは持つものの完全な管理を行なう技術は持たない構成員から成るネットワークの管理であり、そういったネットワークの管理の負担をいかに下げるかということを目指にする。

ここで、「自律」ではなく「自立」という言葉を用いた。三省堂 大辞林 第二版によると、それぞれの言葉の辞書的意味は以下のようになっている。

自立 他の助けや支配なしに自分一人の力で物事を行うこと。ひとりだち。独立。
「親もとを離れて自立する」

自律 他からの支配や助力を受けず、自分の行動を自分の立てた規律に従って正しく規制すること。「学問の自律性」

どちらも他者からの助けや支配を受けないという点では同じであるが、「自律」が自分自身を制御する点に重きを置いているのと比較して、「自立」は他者と離れて存在することに重きを置いている。ちなみに、精神分析や心理学の分野では、親など保護者からの独立や自立を「分離 (Separation)」と呼んでいる。これは意味的に「自律」ではなく、あくまで「自立」である。

著者が提案するネットワークモデルは、End to Endの通信ができ、自分の身は自分で守るネットワークであり、固定された他のサーバなどに依存せずに稼働できるものである。このようなネットワークは、運用面で移動(ここでの移動は引越しなどの中長期単位の移動を指す)が容易なネットワークでもある。同時にこのモデルの適用範囲を小さくしていくと、ホスト単位になり、ホスト単位での運用までも階層的に支援しようという試みになる。このモデルはすなわち、特定の母体などの他者から独立し分離することが容易なネットワークやホストの管理モデルであり、意味的に「自立」が適切である。

2.2.2 ISNA の管理に求められる要件

ISNA の管理には以下の要件が求められる。

ソフトウェア構成を適切に管理できること ISNA では以下が大切である。

- 故障時の原因究明を容易に行なえること。
- 適切な箇所に適切なネットワークを配置できること。
- 各ホストに存在するセキュリティホールを正確に把握できること。

この目的のためには、ネットワーク構成や構成機器、各ホストのハードウェア構成や、OS とアプリケーションのバージョンなどを的確に把握することが必要である。

機器ごとに構成にばらつきがあると、個別の対処を繰り返さなければならず、構成情報の管理に大きなコストがかかる。この管理作業を容易に行なえるようにしなければならない。ISNA では機器の購入コストも押える必要があるため、汎用製品を用い、汎用のシステムを用いるという前提で、管理コストを下げる方法を考える必要がある。特に稼働しているソフトウェア構成を適切に把握できていないと、セキュリティ対策が繁雑になる。

低コストな監視と障害復旧が可能なこと 可用性は、平均故障間隔 (MTBF) と平均復旧時間 (MTTR) から求められる。可用性を上げるためには MTBF を長くし MTTR を短くする必要がある。ISNA では金銭的コストの制約があり民生用機

器を用いなければならないため、専用に作られたシステムと比較して MTBF が短くならざるを得ない。その中で可用性を確保するためには MTTR を短くする必要があるが、同じ理由で冗長系を用意するなどの対策がとれない。また構成員の高度な技術力が期待できないため、個別に迅速な対応を期待することもできない。

低コストかつ容易な操作で障害復旧ができる方法を考える必要がある。

安全で安心であること 適切な水準のセキュリティが保たれると同時に、管理者と利用者がそのことを十分に理解し、安心して使えるネットワークでなければならない。これは前述のセキュリティ対策の現状からも明らかである。末端のネットワークであっても外部にサーバ機能を公開すれば、無差別に攻撃の対象になりうる。こういった攻撃の被害にあうのは、実際には脆弱性がそのままにされているような放置サーバであり、最低限 (例えば半年前の水準など) のセキュリティレベルが保たれていれば防げた事例が多い。

したがって、「適切な水準のセキュリティ」を低コストで保つ方法を考える必要がある。

End to End の通信が可能であること ISNA は、利用者が求める多様な要求を満たせなければならない。NAT などによる理不尽な制限を与えるべきではなく、利用者にとって必要十分なサービスを提供した上で、安全性などを考慮したネットワークシステムを構築すべきである。そのためにはシームレスな通信が可能なネットワークインフラストラクチャを構成する必要がある。

2.3 ネットワーク管理手法の比較

2.3.1 既存のネットワークの管理モデル

そもそもネットワーク管理とは何を指すのか。

本論文の 5 章のように「監視」「判断」「対処」といった作業のサイクルとする考え方もあるし、文献 [96] のように「分析」を中心とした「情報調査」「報告」「対策」の関係として図式化することもできる。本論文では文献 [23] に従って、「ネットワークの活動と資源についての監視、会計、制御」であるものとする。しかしこの定義では具体的な作業項目については言及していない。これを定めているのが TMN や OSI 管理モデルの、管理項目である。

本論文では TMN を参照モデルとして、これを基にして議論をすすめることとする。ここで取り上げる TMN の要求項目は、基本的には OSI 管理モデルも踏襲しており、現在極めて一般的な方法論である。

TMN (Telecommunications Management Network) [33] は ITU-T [7] の SG4 によって作成された M シリーズ勧告にて規定されている。TMN は管理システムの機能、インタフェース、参照点を規定した汎用的なモデルであり、オブジェクト指向技術を前提としている。管理項目としては、OSI で定められた「構成管理」「障害管理」

「性能管理」「機密管理」「課金管理」を採用している。これに加えて「計画」「作業工程管理」「物品管理」を補足する項目として挙げている。またネットワーク管理のアーキテクチャを、ビジネス管理レイヤー (BML)、サービス管理レイヤー (SML)、ネットワーク管理レイヤー (NML)、ネットワーク機器管理レイヤー (EML) の4つのレイヤーに分けている (図 2.4)。

TMNでは異なるネットワークを相互接続しつつ管理運用を行なうことを想定し、異なる機能ブロック間のメッセージプロトコルを定義している。しかし本論文で扱うネットワークは基本的にTCP/IPネットワークのみであることから、本節ではTMNのうち管理項目と管理レイヤーについてのみ注目する。

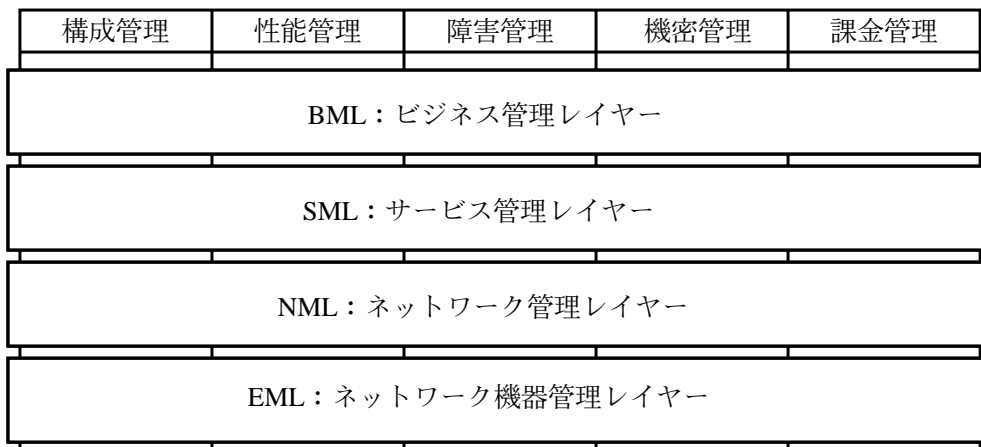


図 2.4: TMN のネットワーク管理アーキテクチャ

構成管理 ネットワークは動的な環境を実現しなければならない。システム内での変化は常に発生し、ルータや端末などのネットワーク機器の相互の関係は常に変化する。柔軟な拡張性があるネットワークは、比較的容易に機器を増設したり撤去することができる。誤った増設や、安易な拡張は、予期せぬ不具合を引き起こすことがある。

構成管理を適切におこなうことにより、適切な増設や撤去、効果的な拡張を実現できる。

構成管理は以下の作業から成る。

- 最新構成情報の収集と把握
- 構成情報の変更履歴の管理
- ネットワーク装置の配置場所の把握
- 設備管理情報 (所有者, 管理者, 機器名称, OS のバージョンなど) の把握

管理する対象は主に以下である。

- ネットワーク装置 (ルータ, スイッチ, 端末など)

- 対外接続
- ケーブル

障害管理 障害管理作業としては、主に以下の作業を行なう。

障害検出 監視またはログレポートなどによって、障害を検出する。

障害診断 構成要素に対して、障害に関する診断を実行する。その手段は、1) 誤りの再現、2) 誤りの分析、3) 報告の受信である。

障害修復 障害修復は構成管理などの他の機能を使って行なわれる。

加えて以下の作業を行なう。

- 障害直後対応 (二次障害防止・拡大防止)
- 障害運転時対応 (代替運転・縮退運転)
- 障害予測・予防保守
- 定期診断・ログ調査

性能管理 ネットワークの性能を向上しようとした時は、適切な機能拡張が必要である。性能管理を実施していれば、改善すべき点が明らかになる。

- 性能情報の収集と蓄積
- 蓄積済み性能情報、リアルタイム性能情報の解析
- 性能ボトルネックの検出と改善
- ネットワークの拡張および縮小計画の立案と実施

機密管理 機密管理は、不正アクセスを防止し内部のデータや環境を守るための管理項目である。インターネットに接続することは、外部のデータにアクセスできる反面、内部のデータを外部からの攻撃にさらす事になる。ネットワークセキュリティを確保するためには、パスワードなどの認証によるアクセス制限や、ファイアウォールによる保護などが必要になる。加えて、組織内のセキュリティポリシーを確立して実施する事が重要である。

OSI ネットワーク管理では14のサービスと8の機構を定義している。前者では、保護すべき対象のサービスを定義し、後者では保護する手段としての機構を定義している。

課金管理 課金管理はネットワークの利用に対してコストを課す管理項目である。部門単位での利用状況に応じた課金を行なうなどの対応を行なう。課金の単位、課金ログ、記録、発信者または受信者に応じた課金などを行なう。

以下では各管理レイヤーについて簡単に述べる [67]。

BML(ビジネス管理レイヤー) ユーザ教育、上位の管理計画や資金計画、意志決定など。

SML(サービス管理レイヤー) 電子メールや WWW などのサービスの管理。

NML(ネットワーク管理レイヤー) 論理的ネットワーク構成や結線の管理。

EML(ネットワーク機器管理レイヤー) 構成機器の管理。

2.3.2 既存のネットワーク管理モデルの適用事例

本節では TMN の適用事例を示しつつ、それを ISNA に適用した際の問題点について述べる。そもそも TMN は電話網や ISP の基幹網などの管理に用いられているものであり、それをそのまま ISNA に適用することは不可能である。また、通常 TMN 準拠をうたった管理システムは、ソフトウェアだけでも数百万～数千万円の価格になることから、ISNA でこのような統合システムを導入することは現実的には不可能である。しかし ISNA の管理方針を考えるにあたり、留意すべき事項の参考にはできる。その過程での取捨選択の考察を本節で述べる。

ネットワーク管理の事例とはソフトウェアの仕様だけではなく、インテグレーションを含めたサービスにも言及する必要がある。しかし事例の詳細が網羅的に外部に解説されることはあまりないため、本節ではソフトウェアが想定してる規模などから利用形態を推測しつつ議論する。

TMN の管理レイヤーのうち、BML は利用者教育などであり、重要な管理レイヤーではあるものの、これはシステムをどのように利用するかといった内容になり比較が困難であるため本節での議論からは除外する。

さて、TMN に準拠したネットワーク管理システムを提供しているベンダーは数多くあるが、Lucent 社と Tercordia(Bellcore) 社が 2 大企業である⁴。以下では、Lucent 社の製品群をもとに議論する。

Lucent 社のネットワーク管理製品は VitalSuite を中心とした複数の製品から構成されている⁵。このため、各管理項目と管理レイヤーごとに、対応する製品について述べ、ISNA に適用することができるか否か、変更するのであればどのような変更が必要かについて議論する。

構成管理

SML SML の構成管理の該当する製品は Lucent からは提供されていない。

この項目は保守業務などでカバーされることになるが、ISNA の場合はこの項目をいかに容易に行なえるかが重要になる。

NML VitalQIP を用いることでホストに対する IP アドレスの割り当てなどを管理できる。VitalQIP は中～大規模ネットワークに対応できるように設計されている。

⁴<http://www.prologue.co.jp/s9906.htm>

⁵<http://www.lucent.co.jp/jp/products/software/index.html>

ISNA の場合は小規模 (数台 ~ 十数台) 程度の IP アドレスの管理ができればよい。低価格の機材を使って構築することのほうが重要である。

EML Navis AQueView EMS によってエレメントレベルとネットワークレベルの構成管理が可能になる。これは HP OpenView フレームワーク内で動作する。SNMP および Java を使ってリモートにあるエレメントの情報を管理できる

ISNA の場合、低価格のネットワーク機器は必ずしも SNMP などに対応していない場合もあり、人手の管理も交えた上でいかに作業を簡易化するかを考える必要がある。

障害管理

SML VitalSuite に含まれる VitalHelp を用いることで、ネットワークとアプリケーション・パフォーマンスに関する障害を検出できる。管理者は遠隔からトラブルシューティングが行える。

サービスの停止は ISNA にとっても深刻な問題である。しかし常時監視している管理者を用意できるとは限らないし、管理対象は少数 (数台 ~ 十数台) なので、管理者が他の作業を兼任しながらでも監視できるような方法が必要になる。

NML Navis NFM (Network Fault Manager) software を用いることで、ネットワークの監視と制御を集中化できる。警告機能、ネットワーク・ステータス表示、ネットワーク・エレメントのリモートアクセス機能を持ち、問題の迅速な特定と修正が可能である。

SML と同じく、ISNA では監視作業に専任のスタッフを用意できない。

EML Navis AQueView EMS によってエレメントレベルとネットワークレベルの障害管理が可能になる。SNMP および Java を使ってリモートにあるエレメントの障害情報を監視できる。

NML と同じく、ISNA では監視作業に専任のスタッフを用意できない。監視システムは確かに障害の早期発見に役立つかもしれないが、その後の対処は別のタスクとされている。大規模なネットワーク管理においては、対処は監視とは別の部隊が行なうのは当然であろう。しかし ISNA では両者の作業は密接に関係している。なぜならそれぞれに人員を配置するという人的コストをかけられないからである。このため、障害復旧の手順までを含めて考える必要があるが、代替系を常時用意しておくコストなどかけられないという問題がある。

性能管理

SML VitalSuite を用いることで、SLA(Service Level Agreement) に基づいたアプリケーションの性能監視を統合されたインタフェースで実現できる。これによりアプリケーションの性能の最適化をはかれる。

ISNA ではリアルタイムで性能監視をするような機構は必要ない。

NML VitalSuite を用いることで、SLA に基づいたネットワークの性能監視を統合されたインタフェースで実現できる。これによりネットワーク性能の最適化をはかれる。また、VitalQIP を用いることで、IP アドレス単位でのネットワーク性能の管理が実現できる。

ISNA では IP アドレス単位で性能監視をするような機構は必要ない。

EML VitalSuite を用いることで、ネットワーク機器の性能監視が実現できる。監視対象の機器としては、複数のベンダーのルータが利用でき、回線としては Ethernet, ATM などが利用できる。

ISNA ではネットワークインフラに対する性能要件は緩い。たとえ速度が低下したとしても、最低でも通信可能な状態を維持できていることが重要である。

機密管理

SML VPN FireWall, Lucent Security Management Server, IPSec Client などを用いたセキュリティソリューションを提供する。

ISNA でもセキュリティ管理は必要であるが、専用機材などのコストはかけられないので、同等の機能を低コストに実現する必要がある。

NML VPN FireWall, Lucent Security Management Server, IPSec Client などを用いたセキュリティ技術を提供する。また、VitalQIP を用いることで、接続する機器の IP アドレスの認証、制限などを実現する。

ISNA でもセキュリティ管理は必要であるが、専用機材などのコストはかけられないので、同等の機能を低コストに実現する必要がある。

EML Navis Radius を用いることで、ユーザはいくつかの単純なフォームを埋めるだけで、NavisRadius をセットアップし認証を受けることができる。Navis Radius は、Java ベースシステムであり、GUI を用いてユーザ・アカウントの管理とメンテナンス全体に対応できる。

ISNA でもセキュリティ管理は必要であるが、専用機材などのコストはかけられないので、同等の機能を低コストに実現する必要がある。

課金管理

SML VitalQIP を用いることで IP アドレス単位、すなわちホスト単位で利用したサービスにたいして課金管理が行なえる。また Navis Radius で認証後、利用するサービスの課金管理が行なえる。

ISNA ではホスト単位の課金管理は必要ない。従来 UNIX で行なわれてきたアカウント管理機能なども、ISNA では必要ない。

NML VitalQIP を用いることで、ネットワークサービス自体に、通信量などに応じた課金管理を行なえる。

ISNA では通信量に応じた課金管理は必要ない。

EML NaviRadius を用いることで、無線 LAN 機器などの通信機器の認証も統合して行なえる。

ISNA では無線 LAN の認証は必要だが、課金管理は必要ない。

2.4 ISNA の管理モデル

2.4.1 ISNA の管理モデルの提案

ISNA の管理モデルは、大規模ネットワークの管理モデルをそのまま適用することは出来ないが、それらを参考にすることはできる。以下では TMN の管理項目と管理レイヤーを参考にし、ISNA に適した力点の配分を行なう。

構成管理 TMN に準じた構成情報の把握、履歴管理、管理情報の把握などを行なう。その際に、頻繁に変化する構成情報の収集と把握を、低コストで行なうための省力化を行なう。迅速かつ容易な設備管理情報の記録を行なう。構成管理には特に力を入れる必要がある。

障害管理 TMN に準じた障害検出、診断、修復を行なう。その際、代替運転などは低コストに実現する。低コストな障害管理の実現には力を入れる必要がある。障害予測と予防は、コストとのバランスを考えつつ、おおまかなものに留める。

性能管理 必要十分な性能の実現を目指す。必ずしも、常に最新最高水準の性能である必要はなく、コストとのバランスを考えた上で適切な性能を導き出すことが重要である。性能情報の解析なども、コストとのバランスを考えつつ、おおまかなものに留める。パフォーマンスだけでなく、利用者が求めるサービスを適切に提供することも必要である。そのための通信インフラの整備なども含まれる。

機密管理 規模に関係なくセキュリティの確保は実現されなければならないため、機密管理は力を入れる必要がある。独自のセキュリティポリシーを策定するとともに、それを構成員に周知させ運用することが肝要である。

課金管理 細かな課金管理は必要ない。よって、この項目は ISNA では優先度が低く、管理項目からは除外する。

また、管理レイヤーごとの ISNA での特徴を以下に列挙する。

BML 組織が小さいので、隅々まで目を行き届かせることが可能である。しかし強制力がないので、利用者のモチベーションの喚起が重要になるであろう。

SML 稼働するサービスは規模に依存せず変わらない。しかし規模が小さいため、要求性能は低い。そのためサービスの実装技術については小規模で解決できる。ただし後述するようにサービスには優先順位をつける必要があり、特に DNS の運用が重要になる。

NML 少数の管理者で全体像を把握できる。基幹部と支線という構造があったとしても、同じ技術(例えば Ethernet) が用いられている。

EML 全体像を把握できるが、同時に管理機材を勝手に移動される可能性がある。民生用機器を用いることになり、ベンダを統一できず、また統合管理ツールのようなものも利用できない。しかし、後述するような統一環境を導入することは管理に有効であるため、その開発が必要になるであろう。

既に述べたように、ネットワーク機器構成の管理と同時にセキュリティホールの把握なども行なう必要があることから、機器管理とサービス管理は密接に関係したものになる。

2.4.2 ISNA を実現する技術的解決方法

前節で述べた力点の配分の目標の下で、2.2.2 節で述べた ISNA の要件について、それぞれ実現するための技術的解決方法を以下に述べる。

ソフトウェア構成の適切な管理と低コストな障害復旧 構成情報は OS やソフトウェア構成を含めて一元管理できなければならない。そのためには、構成を統一することによって構成管理を省力化すると同時に、管理情報として保存すべきものを明確に分離してその記録と共有を促す。また、構成を統一することによって、障害発生時の迅速かつ低コストな復旧を実現する。障害発生時の代替運転を行なう場合でも、余計な代替機を常に用意しておく必要なく、機材の「使いまわし」で代替運転を行なう方法を採用する。また代替機への切替え作業は迅速かつ容易に、専門の技術者でなくても行なえるようにする。

著者がこの問題に取り組んだ 1995 年当時は、NFS を使ってワークステーション間での環境を統一する方法が一般的であった。また、当時 NC[11] や NetPC[10] などのシン・クライアントと呼ばれる端末が登場していたが、その後の状況を見ても普及したとは言えない。また、SunRay[39] のような X 端末の発展形とも言える端末も現れたが、限られた用途にしか用いられていない。結果的に現在、ノート PC にしろデスクトップ PC にしろ、OS を問わず独立して各個人単位で管理される状態で利用されている事例がほとんどである。この理由として、管理コストが分散される個人ごとの管理の方が、目に見える形で管理コストが増加する集中管理よりも、一見低コストに見えるからではないかと思われる。このため構成情報が統一されておらず、利用者は個々に独立してホストを管理するコストを負担している。

この問題を解決するために、ネットワークに依存しない離散したシステムのための、OS やソフトウェア構成を含めた統一環境を構築する。その導入や更新の作業は技術力の低い管理者であっても、容易に行なえるようにする。

安全で安心であること ISNA はセキュリティが確保され、安全かつ安心でなければならない。これは、技術的に十分なセキュリティレベルが保たれていると同時に、利用者がセキュリティについて十分な理解を持ち、組織独自のセキュリティポリシー

が適切に策定され運用されていることを意味する。上記の統一環境の安全性を保つことにより、そのシステムの利用者全てに対して同じレベルの安全性を提供する。

ネットワークのセキュリティ対策については、組織外部からの攻撃と内部からの攻撃の両方を念頭に置かなければならない。

End to Endの通信が可能であること 現在のインターネットの主流プロトコルであるIPv4は、1) アドレス空間の不足、2) 経路制御表の増大、3) セキュリティなどの問題を抱えており、これらの問題に対処するためにNATやCIDRといった技術が使われている。しかし、これらの技術はIPv4が抱える問題の根本的解決とはなり得ない。たとえば一部の発展途上国では、バックボーンにプライベートアドレスを割り当てざるを得ない状況になっている。既に十分なIPv4アドレスを確保している国ではなく、新たにインターネットに接続しようとしている途上国において、IPv4のアドレス不足の問題は特に深刻なものになっている。これらの問題を解決するために、IETF⁶にてIPv6[29]の仕様が策定され、既に実用段階に入っている。移動体通信方式であるモバイルIPやモバイルネットワークなども、今後はIPv6を前提として進んでいくものと考えられている。

ISNAでは利用者の多様な要求に応えるためには、End to Endのシームレスな通信の提供が必要であり、IPv6の導入は必須である。IPv6の導入によりアドレスやルータの自動設定機能を利用できるため、導入設定作業が軽減できる。十分な数の管理者を想定できないISNAでは、これも利点の一つであると言える。

IPv6の導入を実現するためには、移行期間も考えると、ネットワーク内のホストのIPv4/IPv6デュアルスタック化、ルータのIPv6対応が技術的には必要になる。

2.4.3 ISNAを支える要素技術の開発

著者はこれまでの研究で、ISNAの運用の支援を目的として、様々な要素技術を開発してきた。これらの要素技術をTMNの管理モデルを参考にしてISNAに適したものに修正した図に当てはめると、図2.5のような関係になる。前述のようにサービス管理レイヤーとネットワーク機器管理レイヤーは、ISNAのための統一環境を実現するにあたり関係が深いので隣接させる。各要素技術は管理レイヤーをまたがっているため、これを整理するため便宜上「通信基盤」「統一環境の構築」「アプリケーション」「情報共有」とする。

通信基盤の整備として、IPv6対応ルータのプロトコルスタックの開発を行なった[95]。これはルータという構成機器の開発であると同時に、IPv6というEnd to End通信に必須な通信インフラを提供するという点で、ISNAにおける構成管理と性能管理に位置する。このルータは、SOHOなどを対象とした低価格なものであるため、小規模組織に対してIPv6による接続性を提供する手がかりとなり得る。この研究開発を開始した当時は、家庭からインターネット接続をする最も手頃(速度と価格のバランスが取れている)な手段がISDN(NTTのINSネット64サービス)によるものであった。ところがISDNに対応したIPv6ルータは市場には存在していなかった

⁶<http://www.ietf.org>

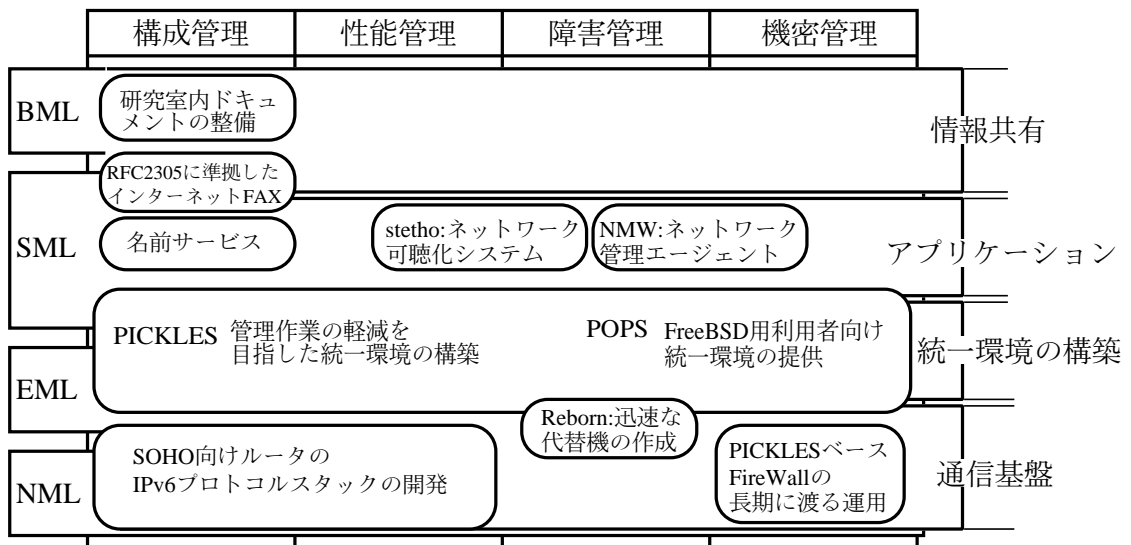


図 2.5: 開発/構築した要素技術

ため、新規に開発することになった。IPv6の導入により最も恩恵をうけるのは、家庭などの小規模のネットワークであり、これらへの接続性の提供が急務であると考えられたためである。この研究開発はヤマハ株式会社との共同研究として行なわれ、その成果は実際の製品に投入され、結果として世界でも最も早い時期に家庭向けの価格帯のルータにIPv6スタックを提供できた。同社からは当時はISDN対応のものであったが、現在はADSLにも対応したブロードバンドルータが発売されており、多くのプロバイダのIPv6接続サービスで推奨ルータとして取り上げられている。

また通信基盤の整備における機密管理としては、著者らが開発したプラットフォームを用いたファイアウォールを、研究室ネットワークにおいて長年に渡って運用してきたことが該当する。

統一環境の構築としては、PICKLES [85] [88] [91] とPOPS [80] [93] [94] という二つの成果を挙げる。PICKLESは端末のハードウェア構成からOS、アプリケーションまでを統一した環境で揃えるものであり、利用者に対する環境の提供だけでなく、管理作業を容易にすることも目的としている。POPSは、OSのパッケージングにおいてPICKLESが目指した目標のうち、統一した利用者環境をFreeBSD上で実現するものである。両者はISNAのすべての管理項目を満たす技術である。

RebornはPICKLESの上に構築されたシステムで、故障時の代替運転を迅速に行なうものである[63]。機能の異なる複数のサーバ間で、個々のホスト依存の情報を相互にバックアップしあうことで、どれかのサーバが故障した際に他のホストで再設定コマンドを一つ実行して再起動するだけで、故障したホストの機能を代替できる。これはPICKLESによりホスト固有情報が明確に分離されたことにより、容易かつ確実に実装できた機能である。

アプリケーションのレイヤーには、いくつかのサービスが挙げられる。まず先の要件として述べた、名前サービスの運用がその一つである。またネットワーク管理

エージェントである NMW[74] [75] を用いた、障害の検知とセキュリティ管理が挙げられる。これはノード間を移動するエージェントを用いて、遠隔地のホストの管理作業や、ホストを巡回しての監視作業などを行なうものである。

更に、ネットワークトラフィック可聴化システムである stetho[65] [66] [92] [35] と、RFC2305 に準拠したインターネット FAX の設計と実装 [82] の二つの成果を挙げる。stetho はネットワークトラフィックを音声に変換して出力するものであり、ネットワーク管理者の監視作業の支援を目的としている。音楽的にも不快でない音作りをめざしているという特徴がある。ISNA においては、監視システムとして障害管理と性能管理の一部を担う。WIDE/IFAX は RFC2305 に準拠したソフトウェアインターネット FAX である。IETF における標準化の過程においては、オープンソースの実装の存在が有力な後押しになる場合がある。著者の開発した WIDE/IFAX は UNIX 上で動作する唯一のオープンソースインターネット FAX であり、国際的な相互接続性試験に参加して他の実装との接続性を確認した。インターネット FAX は紙メディアと電子メディアのゲートウェイととらえることができ、SOHO での業務を電子化する手助けとなる。これは構成情報の流通の円滑化を実現するだけでなく、非常時のインターネット以外のメディアを用いた情報交換手段 [32] ということで、構成管理と障害管理の二つの役割を担う。

最後に情報共有についてであるが、これは利用者教育なども含む。著者らの取り組みの中では、利用者と管理者は極めて近い距離にあり、利用者教育は日々の管理者の教育と密接に関係している。このため図中では研究室内ドキュメントの管理についてのみ言及している。

これらの要素技術を用いて、実際の ISNA を構築した。その事例としては著者らの研究室ネットワークの構築と運用 [89] や、被災者支援システムの構築と運用 [48] [58] [83] [84] [81] が挙げられる。

本論文ではこれらの要素技術のうち、「自立運用ネットワークの管理を支援する統一環境の構築」、「SOHO 向けルータの IPv6 プロトコルスタックの開発」と「ネットワーク可聴化システムの開発」について、以降の章で順に述べる。

第3章 自立運用ネットワークの管理を支援する統一環境の構築

本章では、自立運用ネットワークを支える運用技術のうち、安全性が保証され構成管理がなされた統一環境を簡単に大量に作成する技術について述べる。本章で述べる内容は、国内論文誌に採録され [94] 評価を受けている。

3.1 自立運用ネットワークの管理のための統一環境の有 用性

小規模なネットワークでは、管理コストを抑える必要性などの理由から、適切な機器構成管理を行なう必要がある。これは OS などのソフトウェアの構成管理も含み、用いられているソフトウェアのバージョンなどを適切に管理することで、容易かつ迅速な故障時の代替機材の用意や、セキュリティ管理が行なえる。このためには、サーバから利用者端末までを含めた統一環境を構築するのが望ましく、またこのような統一環境には以下の要件が求められる。

- A 初心者向けに参照となる環境を提供できること。
- B 多数のホストに対して同じ環境をインストールする際の工程を容易にできること。
- C それなりの強度のセキュリティを確保できること。

A については、UNIX では口頭伝承の文化が強く、昔は大学や会社で先人が作りあげた計算機環境に触れることで初心者は勉強したものだが、昨今ではそういった環境が周囲にない利用者も多い。このような利用者に「これだけ揃っていればとりあえず利用できる」という参照となる環境を提供する必要がある。

B については、多数のホストに同じ環境を導入する工程を容易にする必要がある。すべてのホストがネットワークで密に連結されている場合は、ネットワーク経由での同時インストールやディレクトリ自体をネットワークで共有する方法がある。しかし複数箇所に散在している機器やノートパソコンのように、散逸的に導入作業が発生する場合は、例えばインストールする内容を CD-ROM にまとめておくというようにネットワークに依存しない方法が適している。

C のセキュリティの確保のためには、理想的には全てのアプリケーションが常に最新のものに追従されているべきであるが、現実的には全く保守されていないいわゆる放置サーバが多数存在する。2001 年初頭に連続して発生した官公庁の WEB サー

バの乗っ取り事件の例のように、多くの場合に攻撃の被害に遭うのはこういった放置サーバであり、それなりのセキュリティが確保されてさえいれば回避できる問題である。理想的に完全ではないものの「それなりの安全性」が保証された環境を提供する必要がある。無論、統一環境にセキュリティホールが発見された場合は、これを用いているすべてのホストにその脆弱性が当てはまることになる。しかし、散逸の環境の安全性の検査を個別に行なうことと比較すれば、統一環境の安全性のみを検査すれば全てのホストの安全性を把握できることと、脆弱性を修正した環境を配布することで一斉に対策ができるという利点から、著者らはこの方針を採用する。

更に付け加えると、CPUの高速化とメモリやディスクの低価格化と大容量化によって、組み込み機器にもUNIXが用いられるようになってきている。サーバ機器として使われるものもあり、組み込み機器であっても安全性を考慮しなければならない。今後もディスクの大容量化などが進むとすると、組み込み機器にデスクトップと同じ環境を導入してもコストの面で不利にならない状況になるかもしれない。むしろ、統一環境によって得られる安全性や管理コストの軽減の方が有益になる可能性も想定しておくべきである。

さて、上記に挙げた要件を満たすための技術的解決方法を整理すると、以下になる。

1. 共通の利用者環境の提供。
2. ホストに依存した情報と全ホストで共有できる情報との明確な分離。

後者について補足すると、全ホストで共有できる情報は、すなわち導入時のみに変更される内容である。この領域は動作中に書き換えられないことがないため、明確に分離することでROMなどの書き込み禁止メディアにシステム領域を置く組み込み用途に適している。またOSの更新作業の際に書き換える領域を明確にできるため、安全で安心な更新作業が可能になる。この領域は変更されないことから、ネットワーク攻撃によるバイナリの改竄の検出が行ないやすいというセキュリティ上の利点もある。

上記の2つの解決方法を実現するものとして、著者らは1995年からPICKLES SYSTEMを開発してきた[85]。これは現在ではLinux¹で言うところディストリビューションに相当するという表現が端的であり、BSD/OS²をベースに開発していた。PICKLES SYSTEMは著者らの研究室のサーバおよびクライアントとして長期に渡り利用されているばかりではなく、被災者支援情報システム[57]を始めとして著者周辺の研究プラットフォームとして広く用いられている。

Linuxにおけるディストリビューションは確かにインストールして最低限の環境は導入されるが、実用になるものにするためには利用者各自が相当の数のアプリケーションを導入しなければならない。また多くのディストリビューションが乱立しているため、Linuxと一言で言ってもそれだけでは環境を指すことにならない。

BSD系のPC-UNIXはネットワーク性能が高いことや、KAME³による先進的なIPv6の実装をもつことや、NetBSD⁴では多数のアーキテクチャに対応している組

¹<http://www.linux.org>

²<http://www.bsdi.com>

³<http://www.kame.net>

⁴<http://www.netbsd.org>

み込み用途としても用いられているといった特徴を持ち、PC-UNIX の中でも重要な勢力となっている。著者らはこういった BSD 系列の OS の特徴から、BSD 系 OS が統一環境の基盤に適していると考えた。ここで、PICKLES の技術を広く普及させるためには、そのベースとして用いる OS には商用の BSD/OS ではなくフリーの OS を用いる必要がある。新しいベース OS として BSD 系列の OS を比較検討した結果、最も利用者が多いと思われる FreeBSD⁵ を採用することとした。

次節ではまず PICKLES SYSTEM の概要について述べる。次々節以降では、FreeBSD 用でのパッケージ集と導入ポリシーの実装技術について述べる。これは PICKLES SYSTEM で実現した技術をリリースエンジニアリングを簡易にするために二つに分離し、FreeBSD 上で実現したものである。

3.2 PICKLES SYSTEM

3.2.1 PICKLES の概要

PICKLES プロジェクトの当初の目的は情報キオスクを用いたインターネットの普遍的利用環境の構築 [85][91] であった。PICKLES の利用者は、街角に設置された情報キオスクを使うことで、外出時でもインターネットにアクセスすることができる。PICKLES プロジェクトで開発した情報キオスクを「PICKLES 端末」と呼ぶ。

その際利用者は、認証のための個人情報などを IC カードなどの小型の記録媒体に記録し、携帯する。利用時には IC カードを公衆端末に装着することで認証が行なわれ、利用者自身の環境に近い状態で公衆端末を利用できる。その際受け取ったメッセージや収集した情報は IC カードに記録しておく。

PICKLES プロジェクトでは、通常の IC カードではなく情報を閲覧するためのディスプレイと操作のためのキーパッドを備えたカードを用いる。以後このような IC カードを「カード型端末」と呼ぶことにする。このカードを用いることで公衆端末を離れても蓄積した情報を参照することができるという利点があり、PICKLES の大きな特徴のひとつとなっている。

当時の PICKLES プロジェクトではカード型端末として、IBM 社製の ChipCard(図 3.1) を用いていた。著者らは情報キオスクを用いた情報交換を支える機構として、カード型の小型携帯端末を組み合わせた情報システムを開発 [90][86] してきた。特に質問と対応する回答がやりとりさせる形式のメッセージ交換に注目し、この枠組を用いてインターネットを用いたアンケート調査システムのインタフェースを開発した [76]。

ユビキタスコンピューティング [49][50] がひとつの理想の世界であるとする、著者らの情報キオスクとカード型端末との連携による環境構築はそこへの橋渡しをする、現実的な解である。

PICKLES プロジェクトは 1995 年から著者を中心にして勧められ、BSD/OS をベースにした OS である PICKLES SYSTEM を開発した。2000 年からは通信総合研究所非常時通信研究室 (当時) において、FreeBSD をベースにした FreePICKLES

⁵<http://www.freebsd.org>



図 3.1: カード型端末

の開発が勧められた。著者はこの開発にアドバイザーという形で関わった。後述する FreeBSD 用のパッケージ集である POPS は 2001 年から著者が開発を行なっている。これは PICKLES が持つ特徴のうちの、統一利用者環境の提供の部分のみを抜き出して FreeBSD 上に展開したものである。PICKLES の開発経験から、リリースエンジニアリングが大きな負担になることが明らかになった。このため、アプリケーション部分のリリースエンジニアリングと、インテグレーションポリシーとを分離するという方針を採用することにしたものである。

3.2.2 PICKLES SYSTEM の管理モデル

情報キオスクの運用形態では、一台から数台くらいの端末が、それほど高速でない回線によって外部と接続されることが想定できる。よって高速なネットワーク接続を想定した端末管理手法は好ましくない。

公共の端末を管理する上では、特に故障などの障害時に迅速に復旧することが求められる。このためには、故障箇所を迅速に切り分けて交換できるようにするという方法が考えられる。しかしハードウェア部品を交換したために、端末の設定情報がすべて消去されてしまい、端末の管理者が復旧作業を行うようでは効率的でない。同様に、OS を更新したために端末の設定情報が消失するのは好ましくない。

そこで、担当者の責任の範囲を明確にし、管理作業によって互いの責任範囲に影響を及ぼさないような仕組みが必要になる。ハードウェア担当者はハードウェアだけに責任を持ち、OS などを保守する担当者は OS のみに責任を持つ。端末の設定作業担当者は、端末ごとの情報を管理することに責任を持つ。同様に利用者は最低限自分の個人情報だけを管理していれば良いような仕組みになっているべきである。

すなわち、情報キオスクを管理する上では、責任の境界を明確にする必要があるといえる。

以上のことから情報キオスクの管理手法としては、以下の要件を満たすことが必要である。

- 帯域の低い回線でも対応できる
- 故障などのトラブルが発生したときに迅速に復旧できる
- トラブルが発生した際の責任を明確にできる

上記の管理手法を実現するために著者らが採用した方針は、「構成要素を責任の境界によってモジュールに分離し」、「モジュール交換によって保守作業を行う」ものである。まず、情報キオスクの利用者と、それを所有する人、情報キオスクを管理する人、ハードウェアの販売業者とを分離し、それぞれ「利用者」「所有者」「管理者」「販売者」と呼称する。販売者は端末ハードウェアに責任を持つ。最初の3者の責任の範囲は、管理する情報で区分けできる。区分けした情報は、それぞれ異なったモジュールに記録される。

まず利用者が管理する情報は利用者のカード端末に記録される。管理者が管理する情報と所有者が管理する情報は端末のハードディスクに記録される。この二つの情報は分離され、2台の取り外し可能なハードディスクモジュールに別々に記録される。PICKLES 端末では、OS とアプリケーションをディスクモジュールに持ち、バージョンアップ時にはディスクモジュール自体を交換する方法である。ここでのディスクモジュールはメモリディスクなどでも、本質に変わりはないのであるが、現実的に価格や速度、安定性を考慮にいて、ハードディスク (図 3.2) を想定している。



図 3.2: ハードディスクモジュール

端末ごとに異なる情報や、端末動作中に書き換えられる情報は所有者の責任範囲であると考え、これらを一つのディスクモジュールに記録する。残りの OS とアプリケーションの保守は管理者の責任であると考え、これらを一つのディスクモジュールに記録する。前者をユーザディスクと呼び、後者をシステムディスクと呼ぶ。通

責任者	モジュール	内容
利用者	カード端末	個人情報, 認証情報
所有者	システムディスク	OS, アプリケーション
管理者	ユーザディスク	ホストの設定など

表 3.1: モジュールごとの情報の内容

常 PICKLES 端末は一台のシステムディスクと一台のユーザディスクの組で動作する。以上をまとめると表 3.1 になる。

PICKLES 端末ではモジュールの交換によって端末の保守作業を実現する。この「モジュール交換方式」では、ネットワークに依存せずに OS などの更新が可能である。端末が故障してネットワークを利用できないときの保守作業も、同様の手段で実施できる。また OS とアプリケーションを併せてバージョン管理することで、同じ版のディスクモジュールであればどの端末でも完全に同じ環境にできる。

PICKLES 端末では、端末の導入作業や OS の更新はディスクモジュールを入れ換えることで迅速に行える。その手順は次のようになる。

導入時の手順： 所有者はまず PICKLES 端末仕様書 [79] に基づいて販売者から端末ハードウェアを購入する。同時に管理者に対して端末の設定情報を書式に基づいて提出し、ユーザディスクの作成を申請する。すると管理者からシステムディスクと、申請した情報に沿ったユーザディスクが送られる。この2台のディスクを販売者から購入した端末に装着し、端末を起動する。この導入作業で所有者はネットワークの機器の設定を行なう必要はない。

更新時の手順： OS の更新作業時は、管理者から送られてきた最新版のシステムディスクと、旧版のシステムディスクを入れ替えるだけである。ユーザディスクは入れ換える必要はない。

故障時の対処： 端末ハードウェアが故障した場合は、2台のディスクを取り外し、販売者から送られて来た代替機に装着するだけで迅速に復旧することができる。ハードウェアの修理を行なう場合でもディスクは取り外して保管できるため、従来のように修理から戻ったディスクの内容が消去されていたといった事故はない。システムディスクが故障した場合はシステムディスクだけを交換すれば復旧できる。また所有者の情報がユーザディスクというかたちで明確に区分けされているため、バックアップ作業は容易に行える。その量もシステム全体のバックアップと比べると小さい。

3.2.3 PICKLES SYSTEM の実装

PICKLES SYSTEM は PICKLES 端末 (図 3.3) で用いている OS であり、現時点では BSD/OS 3.1 を基に機能を追加したものを用いている。PICKLES 端末では OS や

必要なアプリケーションを自分のハードディスク内に持つ。同じ版のシステムディスクであればこの内容は同一であるため、常に同じ環境が保証されている。



図 3.3: PICKLES 端末試作機

システムディスクとユーザディスクとの分離は以下に述べる方法で実現した。UNIX ではディレクトリごとにある程度情報が分類されている。そこで、`/etc` 中のホストごとに異なる情報を抜きだして格納した`/etc3` と、可変な情報が含まれる`/var` を異なるパーティションに分離し、ユーザディスクに格納した。また所有者が保持する情報も同じように`/local` としてユーザディスクに格納した。UNIX の`/etc` には `netstart`(BSD/OS でネットワーク関係の初期化を行うスクリプト) に代表されるホスト固有の情報と、`rc` に代表される全てのホストで共通のものが混在している。`/etc` のうちホスト固有のものについてだけ、シンボリック リンクを用いてユーザディスクの内容が参照されるようにした(図 3.4)。同じユーザディスクを使えば、システムディスクを交換しても利用者 から見て違いはない。`/etc3` に置いたファイルは以下である。この実装はすなわち、`/etc` にあるファイル群のうちホスト依存のファイルを抜き出したものと等しい。

```
X11/ XF86Config Xaccel.ini badsect changelist chap_md5_secrets crontab
daily.user dumpdates ftpusers getty/ gettytab group hostconfig hosts
hosts.allow hosts.deny hosts.equiv hosts.lpd idcard inet6d.conf
inetd.conf ipfilter/ licence localtime login.conf mail.rc master.passwd
monthly.user mtools.conf mtree/ mtstailor named.boot named.conf namedb/
netgroup netscripts/ networks ntp.conf ntp.drift openssl.cnf passwd
pccard.conf.local phones phoneshell/ pop.auth.db ppxp/ primes printcap
```

```
protocols proxies pwd.db pxdm/ raddb/ rc.configure/ rc.hardware/ rc.user
remote resolv.conf samba/ services shells skeykeys spwd.db ssh_config
ssh_host_dsa_key ssh_host_dsa_key.pub ssh_host_key ssh_host_key.pub
ssh_host_rsa_key ssh_host_rsa_key.pub ssh_known_hosts ssh_prng_cmds
ssh_random_seed sshd_config sudoers syncuser syslog.conf tcpserver/
tripwire.conf ttys ttys.conf.local vfontcap weekly.user xsetup
```

PICKLES SYSTEM は基本的にハードディスク 2 台から構成されるが、上記のパーティション構成を踏襲していれば、ディスク 1 台の構成も可能である。これはノートブック型で利用する際などに有用である。この場合ディスク単位でのモジュール交換は出来ないが、システムのバージョンアップなどはシステムディスクに相当するパーティションを全て書換えるという方法で容易に行なうことができる。

PICKLES SYSTEM では、アプリケーションなどが参照する設定情報はユーザディスクに分離している。設定ファイルの格納場所はアプリケーションごとにまちまちの場所に記録されていたが、PICKLES SYSTEM ではこれを/etc3 ディレクトリに集約した。このような設定情報は OS がデータベースのかたちで管理し、OS が提供するサービスに設定情報データベースへのアクセスを追加するといったする方法も考えられる。しかし、この方法では既存のアプリケーションにも変更を加える必要がある。PICKLES SYSTEM が採用した方法では既存のアプリケーションには変更を加えずに設定情報を集約できた。

ユーザディスクの内容はシステムディスクのバージョンに対応して少しずつ異なっている。このため、システムディスクを新しいバージョンのものと交換した直後は、ユーザディスクとの間でバージョンの食い違いが発生し、不具合が起こる可能性がある。PICKLES SYSTEM では、起動時にシステムディスクとユーザディスクのバージョンを比較し、食い違いを発見したら自動的にユーザディスクの内容を修正し、不具合が発生しないようにする。この機能を担うのが syncuser スクリプトである。このとき、古いシステムバージョンのシステムディスクと新しいバージョンのユーザディスクという組み合わせでもただしく動作する必要があるため、ユーザディスクの修正手順は以下のように行う。

1. システムディスクとユーザディスクのバージョンを比較する
2. 両者のバージョンが等しい場合は何もしない
3. システムディスクの方が新しい場合は、システムディスクの/etc/syncuser を実行する。このプログラムはシステムディスクより古い任意のバージョンのユーザディスクを、現在のシステムディスクのバージョンで動作するよう更新するプログラムである。
4. ユーザディスクのほうが新しい場合は、ユーザディスクの/etc3/ syncuser を実行する。このプログラムはユーザディスクより古い任意のバージョンのシステムディスクで動作するように、ユーザディスクを変更するプログラムである。

この仕組みにより、システムディスクを更新した際に細かな修正を手動で行う必要がない。また、任意のバージョンのディスクモジュールを組み合わせて使えるため、モジュールの使い回しが容易になる。

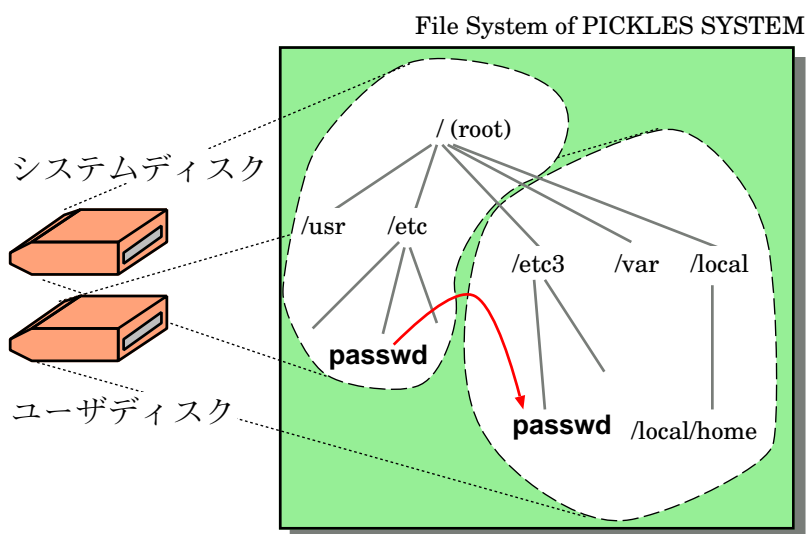


図 3.4: ディスクの構成

3.2.4 PICKLES SYSTEM の運用事例

事例 1: 1996 年 3 月に東京工業大学大学院情報理工学研究科の事務室に PICKLES 端末を設置した。これはかねてより要望のあった事務職員と教官との間の連絡に電子メールを使える環境を実現するためのものである。この時の端末導入手順は最初の事例ということもあり、仕様書に基づいて購入された端末を一旦管理者が預り導入作業と端末の設定作業を行った上で、実際に利用される箇所に設置するという手順を踏んだ。

事例 2: 1996 年中頃より 1999 年にかけて、本学図書館に端末を設置して試験運用する機会を得た [87]。これは電子図書館実現に向けての活動の一部に著者らが協力したものであった。PICKLES 端末を用いた WWW 端末が、大岡山本館に 3 台、長津田分館に 2 台設置された。この事例では端末ハードウェアの購入から設置まで、管理者とは独立した販売者の手によって行われ、管理者はシステムディスクとユーザディスクを作成し装着するという作業のみを行なった。

事例 3: 次の事例として、1999 年まで東工大で活動していた、著者が所属していた大野研究室の当時の内部ネットワークの運用について述べる。この運用の目的は、端末 10 台程度からなる小規模ネットワークでの PICKLES 端末の運用経験を蓄積し、管理作業における PICKLES 端末の有用性を示すことにあった。この規模はイ

ンターネットカフェや図書館などの、町中でインターネットを利用できる場所と同程度と考えられる。

1997年初頭より著者らは、研究室内ネットワークの再設計を行った。その際に利用者端末やルータとして PICKLES 端末に基づいたものを用いた。このネットワークは当時、16名のネットワークの研究者が日常的に利用していた。約15台の利用者端末、3台のノートブック型端末、3台のルータ、メールサーバ、WWWサーバ、バックアップサーバが PICKLES 端末に基づいたシステムで構成されていた。一部のルータはファイアウォールルータとして機能していた。

利用者端末としてはデスクトップ機とノートブック型が存在し、両者を同じ環境に保つことが容易にできた。このため作業環境をノートブック型の端末に移し、作業を行うといった利用方法も容易に行えた。何らかの対外デモンストレーションの時に単体の端末を持ち出して利用することも容易であった。

研究室ネットワークでの運用では、いくつかの PICKLES 端末は mail サーバやルータなど特別な役割を担っている。これらのサービスは端末にトラブルが発生した場合でも、可能な限り短い停止時間ですませたいという要求があった。そこで非常用の端末を一台用意し、重要なサービスを提供している端末の情報を定期的に保存しておく。この非常用端末は、起動時にどの機能を持つサーバ、あるいはルータとして機能するかを指定すると、その機能を持って起動する。この機構を「reborn システム」と呼ぶ。大野研究室での非常用端末は3つの Ethernet インタフェースを持ち、それぞれ3つの Ethernet セグメントに接続している。これと「reborn システム」によって、3つのセグメントを接続するルータと、WWW、メール、ネームサーバのどれかが故障した場合に、その機能を瞬時に代替できる。

以下では運用事例から得られた知見に基づき、PICKLES 情報キオスクの管理手法について考察を行う。

システムのバージョンアップは、ディスクモジュールの交換を行うだけで終了した。PICKLES SYSTEM は1997年度にもととなる BSD/OS をバージョン 2.1 から 3.1 へと移行した。通常このような OS の大幅な更新のときには、新規にインストールしたり、設定ファイルを手動で更新する必要がある。PICKLES SYSTEM は、syncuser スクリプトにより、このような大幅な更新の際でも、ユーザディスクの変更はすべて自動で行うことができた。このため、バージョンアップ作業はディスクモジュールの交換ができる端末で約5分、単一のディスクモジュールにシステムディスクとユーザディスクを記録している端末であっても、システムディスクの内容を書き換えるための約30分ですべて終了できた。

前者の場合はディスクモジュールを交換するだけであり、後者の場合であっても、実際の手による作業は更新用フロッピーディスクから端末を起動するだけなので、最初の5分程度であった。PICKLES 端末はトラブル時の対処などをモジュール交換で行う。したがって、ディスクモジュールに関するトラブルは上記の時間で迅速に対処できるといえる。また、PICKLES SYSTEM はシステムディスクの完全な複製を作成する方法でインストールを行う。このため、CD-ROM などの媒体がなくても、稼働しているシステムを複製することによる新規インストールが容易に行える。このことは緊急に新規インストールを行う必要が発生した場合でも、動作している

端末があればその複製を作成すれば容易に対処できることを意味している。

著者らのグループではネットワーク管理を支援するためのエージェントシステムを開発してきた [97][74]。公衆端末のように多数の箇所に散在して運用されるシステムの管理には、このようなエージェントシステムを用いた手段が有効である。また、実際の運用では複数のネットワークに散在した端末を単一の管理者が管理する場合がある。このような場合同一の管理者の管理範囲を物理的ネットワーク構成に依存しない「管理ドメイン」として扱えると便利である。同時に接続先のネットワークの事情もそれぞれ反映する必要がある。そこで、前述のエージェントシステムに対して、仮想ネットワークを用いた管理ドメインの実現と、ネットワークごとの運用ポリシーを扱えるような拡張を施した [72]。

次に PICKLES 端末の開発過程において得られた知見について述べる。運用事例において自分たちが設計開発した PICKLES 端末を用いて日常的に利用するネットワークインフラストラクチャを構築したことは、次の点で意義があったといえる。開発者だけではなく多くの利用者を得ることで、問題点の洗い出しができたとともに構成員間での知識の共有も実現できた。これは特に大学のような小規模な組織でのシステム開発で、専門の試用スタッフを置けない場合には有効である。事例では、研究室での運用ということもあり、システムディスクの新規作成や更新が頻発する。これを簡便に行うためのインストーラなどの開発が、開発者の手を離れて行われた。複数の開発者の共同作業を支援するために、独自の作業リスト管理システムも開発された。これは電子メールで行われている障害報告や作業状況の報告を自動的に集約し、HTML 文書化するものである。作業リスト管理システムによって、現在未解決の問題とその担当者が一目で把握できるようになった。

3.2.5 PICKLES SYSTEM に関する資料

PICKLES SYSTEM の管理の手引を付録 A に、よくある質問と回答集を付録 B に記した。また、PICKLES SYSTEM に対応した仕様のコンピュータを秋葉原の販売店で購入できる体制を確立した。その際に用いたハードウェアと設定の仕様書を付録 C に記した。また、1996 年時点でのハードウェア仕様書を付録 D に記した。この仕様書は 1996 年という古いものではあるが、FAQ 集などで Hardware Compatibility List(HCL) についての記述が登場するものの、HCL ドキュメントは作成されていないため、情報を補完する目的のために歴史的資料として添付した。付録 C と付録 D を合わせることで、おおよそのハードウェア要求仕様を把握できる。

3.3 共通の利用者環境の提供

3.3.1 FreeBSD におけるアプリケーションの導入

最初に FreeBSD におけるアプリケーションのインストール方法について概観しておくことにする。ここで述べるのは、ソースファイルを入手してドキュメントに従ってコンパイルしてインストールといった作業ではなく、それを簡易化するための機

構についてである。

Linux ではRPM[26] やAPT[46] といったバイナリ形式でのアプリケーションの配布と導入支援機構が用意されている。これに対してFreeBSDではportsとpackagesという2つの方法がある。portsはソースファイルをネットワーク経由で入手して必要なパッチを当ててコンパイルしてインストールする。コンパイルオプションの選択やパッチ当てなどの作業を自動化したものである。packagesはportsによってコンパイルされてインストールされたバイナリをアーカイブにまとめたものである。packageはpkg_addというコマンドでインストールできる。アーカイブ内にそのプログラムが必要とするライブラリなどについての情報も格納されているため、pkg_addを実行すると依存している他のpackageも同時にインストールされる。portsとpackagesは基本的に1対1に対応しており、portからはmake packageを実行することでpackageを作成できる。

portsの欠点はコンパイルできない状態が発生しうることである。これにはportsが対応するバージョンが古すぎて配布サイトからソースコードが消滅してしまったり、同じファイル名のソースコードなのに内容が変わってしまったりパッチが正常に適用できなかつたりといった原因がある。対してpackageはコンパイル済のバイナリ配布なので、インストールに失敗することはない。しかし、依存するライブラリやアプリケーションのバージョンのチェックが厳しく、本来問題なく動作する程度の差異しかないにも関わらず、依存するpackageの更新も要求されることがある。逆に依存関係が整理されているpackageの集合であれば、問題なくインストールできることになるため、次節で述べるPOPSではpackagesのみを用いて共通の利用者環境を構築する。

3.3.2 POPS(パッケージ集)の設計

FreeBSDでは、必要な多数のアプリケーションを管理者が手動でインストールする方法が一般的である。この作業は初心者にとってばかりではなく熟練者にとっても煩わしい作業である。前者については最低限の作業を可能にするためにどのアプリケーションをインストールすれば良いかという知識が欠如している。後者は逆に作業に慣れてしまい、何度も同じ作業を行なうことが苦痛である。この問題を解決するために、FreeBSD上での共通の利用者環境として、パッケージ集であるPOPSを開発した[80][93]。POPSをインストールするだけで一通りの利用可能な環境が構築できる。

導入するアプリケーションの選定作業は、Linuxのディストリビューションでも行なわれていることではあるが、多くの場合実用に供するために必要なアプリケーションを自分で追加しなければならない。POPSではそれだけで実用に供する環境の提供を目的としている。そのためには、利用者がどのようなアプリケーションを必要としているかを調査する必要があり、利用者の要求に随時追従するためには、利用者が行なったアプリケーションの追加と削除を調査する必要がある。理想的にはできる限り多くの利用者から利用頻度を収集し、優先順位を付けて収録するアプリケーションを決定するとともに、この作業を簡略化する支援系を開発するべきで

利用者数	アプリケーションの数
1	534
2	175
3	85
4	50
5	31
6	19
7	15
8	11
9	12
10	9
11	5

表 3.2: アプリケーションの利用傾向

ある。しかしまずは、著者の近隣の利用者の範囲で利用頻度の情報を集めることにした。

FreeBSD ではディストリビューションのような環境を作ることを念頭におかれていないため、上記を含めた以下のような問題を解決しなければならない。

1. 使える環境にするためのアプリケーションを選定する必要があること。
2. アプリケーション同士の依存関係があること。
3. FreeBSD 本体のリリースエンジニアリングとの関係。
4. バイナリパッケージの信頼性を確保しなければならないこと。

1. については実用的な環境を提供するためには、実際に FreeBSD で日常業務を行っている利用者の環境を参考にすることが適当である。そこで著者らの近隣の研究者のうち、FreeBSD を日常業務に利用している利用者が使っているアプリケーションのリストを収集した。11 名のユーザから集めたリストから、バージョン番号を取り除いてアプリケーション名だけに着目して集計した結果、表 3.2 のようになった。1 人が使っているアプリケーションの数は、最多で 384、最少で 57、平均が 201 であった。重複がそれほど多くないのは、アプリケーションとしては同じものであるのに ports の構成の変更と同時に名称が変わったものがあるためである。一旦インストールしたアプリケーションを更新せずに使う場合が多いのと、バージョンの更新の際に古いものが中途半端に残ったままになるという 2 つの原因がある。言い替えば、FreeBSD の通常の手順で環境を更新していくと、次第に「汚れた」環境になっていく可能性がある。

参考のために、利用者数が多かったアプリケーションを列挙しておく。

利用者数 11 :

xpm, tiff, png, jpeg, gettext

利用者数 10 :

m4, linux_base, ja-vflib, ja-nkf, ja-kterm, ja-kaname12, ja-a2ps,
glib, freetype

利用者数 9 :

zip, wget, unzip, libtool, ja-netscape-fonts, ja-naga10, ja-kappa20,
ja-elisa8x8, gtk, gmake, autoconf, Mesa

利用者数 8 :

sudo, ruby, popt, netscape-wrapper, netscape-remote, netpbm,
ja-vfghostscript, ja-groff, ja-Canna, esound, bzip2

利用者数 7 :

xv, wmicons, tgif-nls, psutils-a4, libungif, libaudiofile, lha,
ja-tgif, ja-ptex-common, ja-man, ja-less, gd, automake, Xaw3d, ORBit

利用者数 6 :

viewfax, mpeg_lib, libxml, libproplist, libmng, lcms, ja-xv, ja-samba,
ja-man-doc, ja-makejvf-fkr, ja-dvipsk-vflib, imlib, gdk-pixbuf, faces,
emacs, Boehm-gc, acroread-commfont, aalib, XFree86-aoutlibs

利用者数 5 :

windowmaker-i18n, tcl, rsync, qt, qmail, pkgconfig, p5-type1inst, nmap,
mule-common, lynx, libwww, libhttp, ja-truetypefonts, ja-ptex-euc,
ja-mtools, ja-mozilla-jlp, ja-mh, ja-magicpoint, ja-kininput2-canna,
ja-kakasi, ja-acroread-jpnfont, ispell, im, gv, guile, gimp, gdbm,
cvsup, bison, bash, XFree86

2. については、既に述べたように packages では依存する package のバージョンが厳密に決められている。先に集めたりリストを整理して、最新のバージョンの情報を集めたところ、追加してインストールしなければならない package がいくつかあった。また、依存しているものと異なるバージョンのアプリケーションであっても実質的に問題なく動作するような経験的知識を用いる必要がある場合もあり、これらの整理は手作業で行なった。

3. については、FreeBSD のリリースエンジニアリングでは、本体のリリースと ports の安定性との間に厳格な関連性がないため、本体リリース時にすべての ports が完全にコンパイルできるとは限らないという問題がある。更に言うと、すべての ports が完全である瞬間が存在するという保証もないため、配布状態の ports から共通環境のパッケージ集を構築できるという保証がない。とは言え、本体リリースのしばらく前から ports freeze という形で ports への大幅な変更が禁止される期間が設けられていて、多くの ports はこの期日に間に合うように更新されることから、OS 本体のリリース時の ports が最も安定している可能性は高い。また、リリース時に FreeBSD の配布サイトでまとめて packages が作成されるため、パッケージ集の元になるファイルを得やすいという利点もある。従って、POPS では基本的にリリース時の ports ツリーから生成される package を集めるという方針を取ることにした。

4. については、バイナリパッケージを配布する際には、常にトロイの木馬の混入

の危険性を考慮しなければならない。これを回避するためには、配布する package が改竄されていないことを確認できるような仕組みを用意する必要がある。そこで、配布するバイナリパッケージのチェックサムを用いて、改竄されていないことを調べる方式を導入することとした。

3.3.3 POPS の実装

POPS は package の集合とインストールスクリプトから構成されている。これを 1 つの CD-ROM イメージの形式にまとめて配布しており、その大きさは約 600MB である。FreeBSD 4.8-RELEASE に対応した POPS には 750 個の package が含まれている。これは先に 11 人から集めたリストを元に、同じアプリケーションの最新版だけを選び出し、重複する機能を持つものは最も一般的なものを選び出すといった作業を行なった結果である。この選定作業は以下の方針に基づいて行なった。

- 11 人の利用者の日常作業に不足なく利用できること。
- 将来 JDK などを加えた場合でも、1 枚の CD-ROM に収まること。

現在 FreeBSD では全体で約 8900 個のアプリケーションが ports 形式で提供されており、package で提供すると合計で約 9GB の大きさになる。POPS のアプリケーション数と容量は、実用性と CD-ROM 1 枚でインストールできるという利便性との両面を考慮した結果である。

POPS をインストールするためには、対応したバージョンの FreeBSD をはじめにインストールしておく必要がある。FreeBSD 4.8-RELEASE に対応した POPS では、/usr には 3GB 以上を割り当てる。あとは POPS のインストールスクリプトを実行するだけで、すべてのインストール作業が終了する。インストールには、Celeron 500MHz と 24 倍速 CD-ROM の組み合わせのシステムで約 1.5 時間程要する。

package の依存関係を調査した結果、以下の 3 種類に分類し、段階的にインストールを行なう必要があった。そこで 3 種類の package のリストに基づいて、順番にインストールを実行するスクリプトを作成した。

1. 通常通り pkg_add を実行すれば良いもの。
2. 他のパッケージを上書きするために pkg_add の順番に留意する必要があるもの。
3. pkg_add を -f オプション (強制インストール) 付きで実行するもの

2. については、例えば foo というライブラリと、それを日本語化した ja-foo というライブラリのパッケージが存在した場合、ja-foo は foo を上書きするので ja-foo だけがインストールされていれば foo を利用するアプリケーションも動作するが、依存関係の情報から foo がインストールされていることも要求されていることがある。この場合、最初に foo をインストールしてから、ja-foo をインストールする必要があるので分離した。

FreeBSD 4.8 用の POPS において 2. と 3. に該当するパッケージを、以下に列挙する。

phase 2:

ja-gal-0.23.tgz, ja-gnomelibs-1.4.2_1.tgz, ja-xv-3.10a_3.tgz

phase 3:

auctex-11.13.tgz, ja-VTPSfont-1.3.tgz, ja-addttfont-1.11.tgz,
ja-latex2html-2002.2.1j2.0.tgz, mgp-mode.el-1.30.tgz,
w3-4.0.p46.tgz, i18ntools-1.0.tgz

インストール作業中は、`pkg_add` コマンドが異常終了しても、無視して処理を継続する。これは `pkg_add` コマンドの終了ステータスが、「依存しているパッケージファイルが存在しない場合」「既に `pkg_add` されている場合」「パッケージファイルが壊れている場合」を区別しないという問題に対処するためである。次に POPS の環境に適したドットファイルのサンプルを、`/usr/share/skel/`以下にコピーする。

更に、ダウンロードしたパッケージが改竄されていないことを確認するためのスクリプトを用意した。このスクリプトは手元のパッケージファイルの MD5 [44] と、POPS のマスターサイトにある個々のパッケージの MD5 とを比較するものである。

3.4 情報の明確な分類と分離

3.4.1 情報の分離の方針

既に述べたように、ハードディスク上の情報は2つに分類でき、著者らは PICKLES SYSTEM においてこれを明確に分離した。ホストに依存した情報と、すべてのホストで共有できる情報である。後者はシステム動作中は書き変わらないはずなので、理想的には読み込み専用ファイルとして設定できるはずである。

一般的な UNIX ではこの2つの情報が、ディレクトリごとにおおまかに分類されている。しかし、例えば設定ファイルのほとんどは `/etc` に置かれているが一部は `/usr/local/etc` 以下や `/usr/X11R6/lib/X11` 以下に置かれているというように、完全には分離されていない。ホスト間で共有できる部分を `/usr/local/`以下に集約して NFS で共有する管理手法は広く行なわれているが、この情報を完全に分離することはそれほど容易ではない。

例えば `/etc` 以下に置かれているファイルのすべてがホストに依存した設定ファイルというわけではない。`/etc/`にはポート番号とサービス名の対応表 (`/etc/services`) なども含まれており、2種類の情報の両者が混在している。その他については、`/var`はその名称が示す通り変更されうる情報が格納されており、この下にはホスト固有情報が格納されている。ホームディレクトリなどのその他のホスト固有の情報については、`/local/`の下に集約する方針を採用した。

PICKLES SYSTEM では2種類の情報を明確に分類し分離した。ホスト共有の情報は「システムディスク」に格納し、ホストごとに依存した情報は「ユーザディスク」に格納する。両者は通常2台のディスクに分けて格納するが、ノートPC などの場合には1台のディスクにパーティションを分けて格納することも可能である。上記を整理すると 3.3 になる。

Directory	含まれる内容	分類
/	root partition	システムディスク
/usr	アプリケーション	システムディスク
/etc の一部	設定ファイル	ユーザディスク
/var	可変ファイル	ユーザディスク
/local	ホームなど	ユーザディスク

表 3.3: 情報の分類

3.4.2 情報の分離の実現

以下では2種類の情報の分離方法について述べる。説明を簡易にするため、前述のシステムディスクとユーザディスクという用語を用いることにする。

情報の分離を実現するためには、以下の2点が問題になる。

- /etc 以下の一部だけをどのようにユーザディスクに格納するか。
- ユーザディスクのパーティション情報をどのように発見し処理するか。

前者の点については、union ファイルシステムを用いて解決する。他の解決方法として、BSD/OS 版の PICKLES SYSTEM では一部のファイルだけを別パーティションに移し、/etc からはシンボリックリンクを張って参照するという方法を採用していた。しかし、この方法では設定ファイルの編集を行なって誤ってシンボリックリンクを破壊してしまうことがあり、また sudo などのように設定ファイルが通常ファイルでない動作しないアプリケーションも存在した。union ファイルシステムを用いれば、見た目上は/etc に通常のファイルが置かれているのと変わらないため、このような問題が発生しない。

以下では分離作業の実際を述べる。まず最初にすべての設定ファイルを/etc 以下に集約する。例えば/usr/local/etc は/etc/usr.local に移動し、/usr/local/etc からシンボリックリンクを張る。次に/u/etc というディレクトリを作成し、このディレクトリを/etc に union マウントする。union ファイルシステムは、あるディレクトリを他のディレクトリの上に「かぶせる」ことができるファイルシステムであり、下のディレクトリのファイルを編集するとまずそのコピーが上のディレクトリに作成され変更後のものが記録される。/u は独立したユーザディスクのパーティションに格納する。これによって、/etc 以下のファイルのうち、修正が加えられたものだけがユーザディスクに格納されることになる (3.5)。

次の課題はユーザディスクの発見方法である。既に述べたように、ユーザディスクはシステムディスクと同じドライブに格納される場合もあるし、異なるドライブに格納されることもある。このため、起動時にユーザディスクが存在するドライブとそのパーティション構成を検出しなければならない。ユーザディスクのパーティション情報はユーザディスク固有の情報であるため、必然的にユーザディスクに置かれる。また起動時に参照される情報 (FreeBSD の場合は/etc/rc.conf に起動時の挙

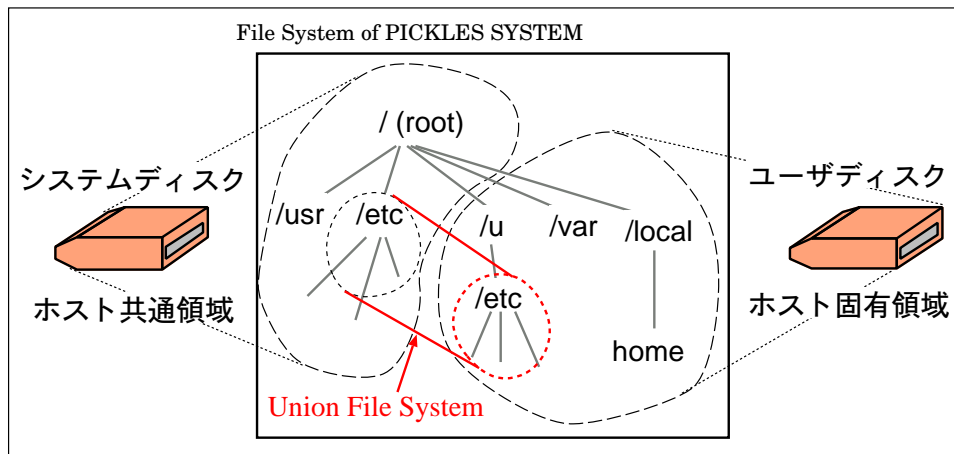


図 3.5: ファイルシステムの構成

動を記述する) もユーザディスクに格納される。これらの情報を適切に反映させるために、起動スクリプトに以下の修正を加える必要があった。

- 候補となるパーティションを検索して、`/u` にマウントすべきパーティションを発見する。
- `/u/etc/` にある `fstab.userdisk` の情報を反映させる。このファイルには、ユーザディスクのパーティション情報が書かれている。
- `/u/etc` を `/etc` にマウントした後に、改めて `/etc/rc.conf` の情報を起動スクリプトの変数に反映させる。

`/u` の候補となるパーティションのリストは、`/etc/userdiskcandidates` というファイルに列挙されている。 `checkuserdisk` というコマンドを作成し、このファイルに書かれているパーティションを検査して `userdisk` というファイルを発見したらそのパーティションを `/u` にマウントするようにした。この状態で、ユーザディスクのどのパーティションをどこにマウントすべきかという情報は `/u/etc/fstab.userdisk` に書かれている。

`/etc/fstab` は、 `libc` に中に含まれる `getfsent()` や `getfsfile()` などの関数を経由してアクセスされる。これらの関数を変更し、 `/etc/fstab` と `/etc/fstab.userdisk` という両方のファイルの内容を反映するようにした。またシステムディスク側の `/etc/fstab.userdisk` から `/u/etc/fstab.userdisk` へのシンボリックリンクを作成した。この結果、 `/u/etc` が `/etc` に union マウントされていない状態でも、2つの `fstab` を読みこめるようになった。どのディレクトリを `/etc` に union マウントするかはユーザディスク側に記述する内容である。上記の方法により、 `mount` コマンドでシステムディスク側とユーザディスク側の `fstab` を一括して扱えるため、例えば `mount /etc` を実行するとユーザディスク側の情報に従って適切なパーティションが `/etc` に union マウントされることになる。

次に/etc/rc.confを再度読みこむ。rc.confには起動時の挙動を決定するための変数の値の設定などが記述されており、このファイルは通常/etc/rcの先頭で読み込まれる。しかしこの時点ではユーザディスクがマウントされていないため、ユーザディスクがマウントされた時点で、改めてその内容を反映させるために再読み込みを行なう必要がある。続いて/etc/rcの残りの起動手順を続行する。

3.5 評価と考察

3.5.1 他の機構との比較

多数の計算機に同一の環境を導入する機構としては、いわゆるクラスタリングソフトウェアが挙げられる。しかし、多くのクラスタリングソフトウェアは2000年以降に開発が始められた。例えば後述するOSCARの開発が始まったのは2001年初頭である。これに対してPICKLESの開発は1995年から開始されており、開発初期にクラスタリングソフトウェアとの比較を行なう余地はなかった。

当時、同様の目的のために用いられていた方法としては、/usr/local/などの共通領域をNFSで共有する方法であった。ディスクレスワークステーションは技術的にはこの方式の発展形と言える。この方式だと、最初に通常のOSのインストールを行なった後にNFSを設定するという二段階の作業を行なう必要がある。その作業を簡易化するための技術も開発された。SunのJumpStart[47]はネットワーク経由でOSの導入が可能であるが、OSベンダが用意した標準の環境しか導入されず、導入後にパッチ当てなどの作業を別途行なう必要があるという問題がある。北陸先端大学院大学のFRONTIER[25]では新規購入したワークステーションをネットワークに接続し、ネットワークブートするだけで、ネットワーク関係の設定を含めて初期導入と初期設定を一括して行なう機構を開発した。しかしこれらの方法はネットワークに強く依存している。PICKLESを開発し始めた当時は、UNIXのプラットフォームがワークステーションからPCへと広がり始めた時期で、ノートPCや構成員の自宅など、高速で密なネットワーク接続を前提とできない環境でも共通の環境で使えることを目標としており、目標とする点が異なっていた。PICKLESにおいてユーザディスクを分離する作業は、NFS環境でも共通する技術であるが、PICKLESではそれまであいまいにしか分離されていなかった情報を選別し、分離する必要があるファイルを厳選する作業を行なったという優位性がある。

その後、2000年以降になり、クラスタリングの技術が現れた。

OSCAR[42]はクラスタの構築だけでなく管理も行なうソフトウェアである。インストール作業は、System Installer, System Imager, System Configuratorを用いたイメージのコピーによって行なわれる。

NPACI Rocks[40]もクラスタの構築を行なうソフトウェアであるが、導入はCD-ROM経由で行なわれる。XMLでノードの設定を記述し、それをもとにRedHat KickStart[14]のCD-ROMイメージを作成する。個々のノードでは、そのCD-ROMを用いてインストール作業を行なう。

LUCIE[61]は大規模クラスタの高速セットアップツールである。インストールメ

	通常インストール型	クラスタリング特化型
イメージコピー型	BSD/OS PICKLES	DCAST OSCAR
パッケージ型	NPACI Rocks	LUCIE

表 3.4: クラスタリングソフトウェアの分類

ディアを必要とせず、ノードの設定や必要なアプリケーションパッケージを記述しておき、それに基づいてネットワーク経由でパッケージの配布や設定作業を行なう。Dolly+[2] を用いてネットワーク転送の高速化を実現しているという特徴がある。

DCAST[62] では、マスターとなるイメージを作成し、その複製をネットワーク経由で配布することによってクラスタを構築する。Grub を用いることによってインストールメディアを必要としないという特徴がある。

これらを整理すると、既存のインストール手段を改良した方式とクラスタリングに特化した方式に分類できる。また、ディスクイメージコピー型と、パッケージ配布型とも分類できる。この分類に沿うと、BSD/OS の PICKLES SYSTEM は、既存のインストール手段を改良した方式であり、ディスクイメージコピー型になる。この分類を表にすると、表 3.4 になる。

以上のようにクラスタリングソフトウェアを分類したが、繰り返し述べているように、PICKLES はネットワーク結合に依存しないことを前提としているため、クラスタリングソフトウェアとはそもそも目標点が異なる。ただしこれも前述の繰り返しになるが、PICKLES でユーザディスクに格納する情報のうちの設定情報は、クラスタリングソフトウェアにおいても分離した上で、インストール後に個別に書き換えるという方式をとっており、この情報の分離作業を早期に行ないインテグレーションポリシーとして確立した点は、PICKLES に一日の長があると言えるだろう。

さて、表 3.4 の分類で、イメージコピー型とパッケージ型とに分類した。BSD/OS の PICKLES は前者になり、POPS は後者になるだろう。この両者の違いについて以下で議論する。

パッケージ型のクラスタリングソフトウェアでは、その利点として、更新時のデータ転送の少なさや、マスターになるイメージの作りこみのコストが必要ないことを挙げている。しかし、実際に運用してみると、パッケージの削除と更新作業の過程で余計なファイルが残る場合があり、それが動作に不具合をきたす可能性がある。また、パッケージを指定するだけで問題なくインストールして実行できる環境が構築されれば問題はないのだが、実際は一旦ライブ環境を構築し、その上で諸々の動作確認作業などを行なう必要があり、結局マスターのイメージを作る作業と同等のコストが必要になる。パッケージ型はその完成度が、パッケージシステムの完成度に依存しているという問題があり、両者を比較した場合どちらが優れているとは言いがたい。

加えて言えば、PC クラスタリングソフトウェアのほとんどは Linux を用いており、BSD 系列を用いるという著者の方針には適用できない。この点については、後の章で考察する。

3.5.2 有用性の評価と考察

インストール作業の簡易化については、著者の1人がFreeBSDと必要と思われるpackageを手動で選択してインストールした時にはおよそ4時間を必要としたが、POPSを用いる場合は合計で2時間程度で完了した。またPOPSの場合はインストールスクリプトを最初に行わせて待つだけであり、1つ1つのパッケージをメニューで選択するといった作業を行なう必要がない。

システムディスクとユーザディスクを分離するという設計により、理想的には/usrが格納されるパーティションを読み込み専用属性で利用できることになり、これは組み込み用途などで有効である。この設計の有用性はPICKLESの運用実績からも確認されている[91]。システムディスクとユーザディスクを2台の容易に脱着可能なハードディスクモジュールに格納したPICKLESでは、システムディスク側の故障対策やバージョンアップがディスクモジュールの物理的交換だけで可能である。ノートPCなどのように1台のディスクにシステムディスクとユーザディスクの両方を格納した場合でも、システムディスクのパーティションが分離されているため、システムの更新作業で変更が加わる領域を制限できる。このため、安全かつ安心な更新作業が実現できる。

また、FreeBSD版のPICKLESであるFreePICKLESは通信総合研究所非常時通信グループが開発したDDoSシミュレータ[70]でも採用されている。これは分散DoS攻撃をシミュレートするために100台の攻撃用計算機を用いるもので、この100台の計算機にFreePICKLESが用いられている。FreePICKLESではネットワーク経由でシステムディスクを更新する仕組みも容易されており、この100台の計算機へのOSのインストールはネットワーク経由で行なわれた。

POPSは実用的な環境の提供を目指してはいるが、誰にとっても完璧な環境を1つのパッケージ集で提供するのは不可能である。特殊な要求に応じることも含めて考えると、POPSをベースにした環境のカスタマイズを行なう必要がある。例えば以下のような支援系の開発を考えている。

- アプリケーションを機能ごとにグループ分けする。
- 機能の組合せを選択すると、依存関係も含めて必要なpackageを収集し、POPS同様のインストール環境を作成する。

この方法にはPOPSでpackageを3種類に分類したような経験的知識を加味する必要があり、portsで提供されている全アプリケーションについて、そのような知識を整理する必要があるという課題がある。

3.5.3 技術的考察

メタパッケージとしての実装の可能性

POPSについては、理想的には他のpackageの依存関係の頂点となるメタパッケージとして実装するのが望ましい。しかしPOPSでインストール手順を3段階に分離

せざるを得なかった理由から、現状で単一のメタパッケージとして実装するのは難しい。

単体のメタパッケージとするためには、依存関係の矛盾を無くし、日本語化されたライブラリなどを元のライブラリに依存するようにする必要があるだろう。または、POPSと同じように3段階に分けてメタパッケージ化する方法も考えられる。

portupgrade との関係

portupgrade はインストールされているパッケージを ports や packages を使って最新版に更新するものであり、依存関係の追跡なども自動的に行なう。通常の FreeBSD の使いかたでは、ports を最新にした上で portupgrade を使ってアプリケーションを最新に更新するという方法が一般的である。

しかしこの方法だけでは、例えば ghostscript が日本語版 5.50 からバージョン 6 に更新された時のように、ports の構造が大幅に変わってしまうものなどには対応できない。

この問題を解決するためには、定期的に全体の入れ換えを行なう必要があり、POPS を用いることでこの作業を容易に行なうことができる。その上でアプリケーションを常に最新版に保ちたい場合は、portupgrade を用いて更新するという方法が効果的であると言える。

収録アプリケーションの選定

最初の版の POPS は、11 人の FreeBSD の利用者から集めたパッケージのリストをもとに収録するアプリケーションを決定した。その際の導入するアプリケーションの整理と選択は手作業で行なわれ、複雑なものであった。その後の版では少しずつの修正に留まっているが、今後も POPS に含めるべき適切なアプリケーションの選別手順については検討の余地がある。

POPS の設計で述べたように、できるだけ多くの利用者からアプリケーションの利用頻度の情報を集めるべきである。使われていないアプリケーションについては、package から展開されるファイルの最終アクセス時間の情報を元に判別できる。この情報収集と収録パッケージの決定作業の支援系の開発が急務である。

安全性の確保

今後 POPS がミラーサイトなどでも配布される可能性を考えると、バイナリ配布の際のトロイの木馬の危険性を考慮する必要がある。POPS については、package の MD5 を配付サイトで管理する物と比較する方法を採用したが、安全性の確保については引き続き検討しなければならない。

現在の POPS では、MD5 のファイルはベースになる FreeBSD のバージョンごとのディレクトリに package のファイル名に.md5 という拡張子をつけて格納している。しかし、package のバージョンが同じでファイル名が同じであっても、異なる内容

の package が生成される場合がある。package 内には依存している他の package の情報が含まれるため、依存先のバージョンが変われば依存元の内容も変化する。更に、make package という通常の方法と、portupgrade で package を生成した場合とでは、依存しているアプリケーションのバージョン情報の扱いが異なるため、同じ ports ツリーから生成しても異なる内容の package が出来上がる。言うまでもなく、使うコンパイラによっても異なるバイナリが出来上がるであろう。

つまり FreeBSD 自体には、バイナリパッケージの一貫性を保証する方法が存在しないため、Debian などの Linux におけるディストリビュータのようなバイナリパッケージを保証する母体のような活動までも視野に入れる必要があると考えている。

起動スクリプトに対する変更

本文中で述べた FreeBSD4.8-RELEASE に対する変更点だけでなく、NetBSD や FreeBSD-current に導入されている rcorder という機構 [38] での実装を考えなければならない。rcorder は個々の起動スクリプトの実行順序を最初に決定して順次実行するため、ユーザディスクをマウントしてからその内容を反映させることができない。rcorder を 2 段階に分けて実行するなどの方法が必要になる。

第4章 小規模ネットワークの構築に適したIPv6対応ISDNルータの実装と評価

インターネットの普及が進むにつれ、IPv4[43]の欠点が明らかになってきた。IPv4が抱える問題を根本から解決するために、IETFでIPv6[29]の仕様が策定された。IPv6は既に世界規模のバックボーンが完成し、これを用いたさまざまな研究が行われている。著者は小規模ネットワークへのIPv6の導入を推し進めるためにはアクセスルータの開発が必須であると考え、SOHO向けルータ上でのIPv6プロトコルスタックの開発を行なった。

本章で述べる内容は、図2.5における、通信基盤の整備に当たる。また、この内容は国際会議[36]と国内論文誌[95]に採録され評価を受けている。また、開発当初はISDNの通信速度を念頭に置いたものであったが、このプロトコルスタックを発展させたものが、現在ではADSL対応のブロードバンドルータとして商品化されており、家庭用製品では唯一のIPv6対応ルータとして評価を受けている。

4.1 IPv6 ISDN ルータの開発経緯

本章では、IPv6に対応したISDNルータの開発の動機となった背景に関して、対外接続方法の比較と著者の研究室でのIPv6ネットワークの構築の二点について述べる。

4.1.1 SOHOでの対外接続

インターネットへの接続手段については、1999年以降劇的な変化が起こっている。1998年の段階では128Kbpsよりも高速な帯域を得るためには専用線サービスしか存在しなかったが、2001年以降はADSLの普及率が上昇をたどり、10Mbpsや100Mbpsの光ファイバーサービスも現実のものになっている。このため対外接続手段の議論を行う際には、その時期を明記して論じるものとする。

著者らがIPv6での対外接続を検討しはじめた1997年当時、SOHO環境からインターネットに接続する際、日本ではISDNやアナログモデムなどの手段を利用していた。この場合の通信速度は128Kbpsか、それ以下であることがほとんどであった。これ以上の帯域の専用線という選択肢もあったが、SOHOには適当とは言えなかった。以下では当時のコストの比較からその理由を述べる。

128Kbps の専用線と ISDN¹を比較すると、128Kbps の専用線は月に 112,000 円かかるのに対し、ISDN は一分あたり約 8 円である。それゆえ、一日 8 時間未満の接続であれば、ISDN の方が低コストになる。当時は Flets' ISDN サービスが提供されていなかったため、ISDN は必然的に従量制の課金になったが、それでも昼間の一部の時間だけ利用するのであれば専用線よりも低価格であった。したがってこのような条件下では、ISDN が当時 SOHO から外部ネットワークへの接続に最も適していた。また、行事用のネットワークを暫定的に構築するような短期間だけの運用の場合も、導入コストが低いことから ISDN のほうが低コストになる。

4.1.2 ISDN を用いた接続方法

著者らの研究室では、1997 年の時点で IPv6 バックボーンへの接続を試みた。この当時、市場には IPv6 に対応した ISDN ルータは存在しなかった。そこで、著者は IPv6 ネットワークへの接続に ISDN ブリッジを使う方式を採用した。この場合のネットワーク構成を図 4.1 に示す。

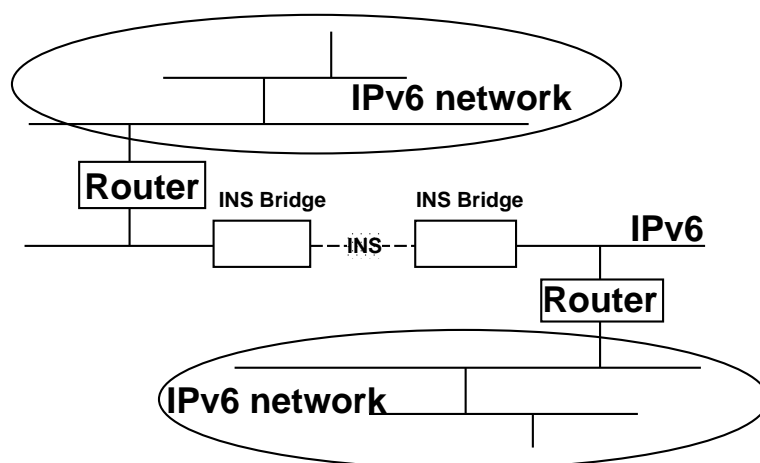


図 4.1: ISDN ブリッジを用いたネットワーク構成

このネットワークの運用経験からは、以下の問題点が明らかになった。

- ブリッジのためのネットワークセグメントとルータが必要である。
- ルータ間の ND(Neighbor Discovery) などのブロードキャストパケットにより、不要な発呼が発生する。
- 上記の不要なパケットにより帯域が浪費される。

前述の問題は、IPv4 でブリッジ接続を行った場合にも発生する問題であるが、IPv6 では対応する ISDN ルータが存在していないために、この問題を回避することがで

¹INS ネット 64 は NTT が提供する ISDN サービスの名称であるが、本論文ではこれ以降両者を同じものとして ISDN と統一表記する

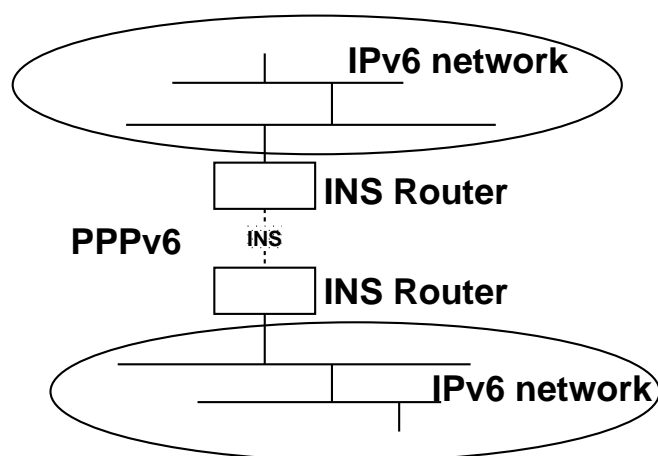


図 4.2: IPv6 対応 ISDN ルータを用いたネットワーク構成

きない。この経験より、著者は SOHO から IPv6 ネットワークに接続するためには、IPv6 対応の ISDN ルータが必要不可欠であると考えた。ISDN ルータを用いた場合、ネットワーク構成は図 4.2 のようになる。これにより、以下の利点が生ずる。

- 外部ネットワークに接続する際、ISDN ルータ以外の機材を必要としない。
- アドレスを動的に割り当てられる。
- 不必要なパケットによる発呼や帯域の消費が起こらない。

4.2 ISDN ルータ用 IPv6 プロトコルスタックの設計と実装

4.2.1 設計

前節に述べた運用経験から、著者は 1997 年から 1998 年にかけて以下の結論を得た。

- 各ホストでの IPv4 と IPv6 のデュアルスタック対応は十分な水準に達し、実用可能な状態にある。
- 基幹ネットワーク向けの IPv6 対応ルータも登場している。
- しかし SOHO 用として適当と思われる、ISDN 回線を用いて IPv6 で対外接続をする環境は整っていない。

そこで、この経験を背景として、著者は 1998 年より IPv6 対応 ISDN ルータの開発を行なった [36] [95]。この開発プロジェクトは東京工業大学大学院 情報理工学研究科 大野研究室とヤマハ株式会社の共同研究として行われ、この IPv6 プロトコルスタックの開発コードは「WS-One」とした。WS-One はヤマハ製の家庭用 ISDN ルー

タを動作プラットフォームとしている。この共同研究は守秘義務契約が締結されたもとで行なわれたため、実装の技術的細部については論文中では触れる事ができない。本論文ではヤマハ株式会社と協議の上公開可能と判断された内容と、公開された実装から外挿可能な内容の範囲で議論を行なうことにする。

IJ の SEIL-T1 や松下電送システムの SJ-6 などの IPv6 対応を標榜している小型ルータの多くが、内部的には FreeBSD, NetBSD, Linux などの UNIX を採用しているのに対して、WS-One はタスクベースの組込み用途向け RTOS(リアルタイムオペレーティングシステム) 上で実装された独自実装のプロトコルスタックである。

WS-One を開発するにあたり、以下の技術的課題があった。

- 少ない計算機資源で動作させなければならない。
当初ターゲットにしたハードウェアは、CPU i386 33MHz, RAM 4MB, ROM 1MB であった。
- 組込み用 OS という制限がある。
汎用的なプロトコルスタックの枠組が用意されているわけではない。
- 短時間で動作させる必要がある。
共同研究の契約期間内に動作させる必要があった。
- IPv6 の仕様変更に合わせて、実装を変更しやすいようにしなければならない。
当時はまだ IPv6 の仕様変更される可能性があり、
- IPv6 の仕様自体が持つオーバーヘッドを考慮しなければならない。
IPv6 はルータの処理を軽減されるような仕様になってはいる。しかしアドレス長が IPv4 と比較して 4 倍の 128 ビットになっているため、アドレスの比較などの処理に大幅な変更を加えなければならない。

WS-One は短時間で完成させて動作させる必要があるため、デバッグを容易にする必要があった。また実装を変更しやすいようにするためには、IPv6 の部分だけを切り離せるようにする必要があった。IPv6 の処理が散在しているコードでは修正作業は膨大になる。この二つの条件から、IPv6 の処理を独立したタスクとして分離することにした (図 4.3)。

ただし、この設計はパケットの受信の度にタスク切替えが発生するため、速度が低下する可能性があった。これは少ない計算機資源上で動作させなければならないという制約に反する。しかし、当初の目標性能が ISDN の帯域である 128Kbps とそれほど高速ではないことと、現実的に多くのトラフィックは HTTP や FTP などの大きなサイズのパケットであるため、パケット単位の処理の負荷は相対的に小さくなると予想できたことから、開発効率を優先させて IPv6 タスクを分離する設計を採用した。

4.2.2 実装

IPv6 プロトコルスタックを実装するにあたり、IPv4 と比較して異なる点は以下である。

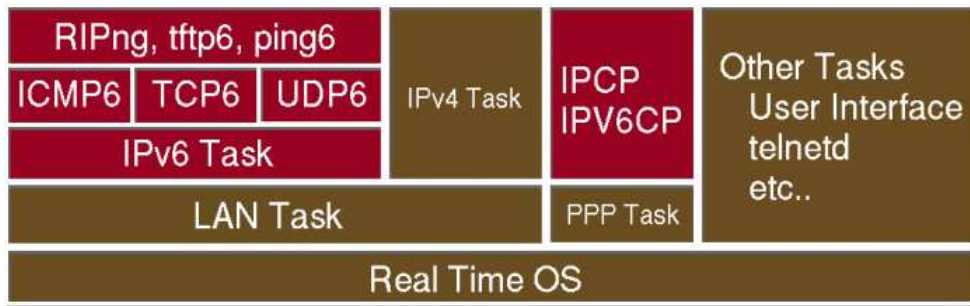


図 4.3: WS-One のタスク構造

アドレス長 アドレス長が IPv4 の 32 ビットから 128 ビットに拡張された。多くのアーキテクチャが 32 ビットレジスタを持つため、IPv4 アドレスは 32 ビット整数の変数として扱われてきたが、この前提がなくなったためにアドレスの受渡しや比較などの処理を全面的に作りなおす必要があった。

マルチキャスト IPv6 では近隣探索やルータ広告にマルチキャストを用いる。このため、マルチキャストの送受信ができることが必須の機能になる。ただしマルチキャストルーティングは必須ではないため、ノードとしての機能、具体的にはマルチキャスト IP アドレスの送受信に Ether マルチキャストに相互変換する処理を追加する必要があった。

一つの I/F に複数のアドレスがつく IPv6 では、リンクローカルアドレス、サイトローカルアドレス、グローバルアドレスというように、一つのインタフェースに複数の IPv6 アドレスが付与される。IPv4 でも複数の IP アドレスを付与して、マルチキャストやバーチャルホストなどを実現していたが、IPv6 では基本機能として必須になる。

第一段階の実装 第一段階では RT80i 上に IPv6 プロトコルスタックを実装した。RT80i の仕様は以下である。

- CPU: i386 互換 33MHz
- RAM: 4MB
- ROM: 1MB

実装期間は 1998 年 8 月～1999 年 3 月であった。第一段階では以下の機能を実装した。

- IPv6 パケットの送受信
- ICMP の処理
- 近隣探索 (Neighbor Advertisement, Neighbor Solicitation)
- ルータ要請 (Router Solicitation)

第二段階の実装 第二段階ではRTA50i上に実装した。RTA50iの仕様は以下である。

- CPU: SH3 80MHz
- RAM: 4MB
- ROM: 1MB

実装期間は1999年4月～2000年3月であった。第二段階では以下の機能を実装した。

- ルーティングプロトコル (RIPv6) の実装
- PPP の実装
- ルータ広告

また、第二段階では他の実装との相互接続性の確認を行なうことも目標とし、TAHIの相互接続性試験に参加した。その結果は後述する。

その後、東工大とヤマハ株式会社との共同研究が終了し、ヤマハ(株)側での拡張が行なわれ、2000年8月の段階で以下の機能の実装が終了している。

- Neighbor Discovery、 Router Advertise などのIPv6の基本機能
- Fragment Header の処理
- ICMP、UDP、TCP (例: ping、tftp、telnet など)
- RIPv6
- PPP(IPV6CP)
- IPsec(AH、ESP、IKE)
- IPv6 パケットフィルタ

IPv6対応ファームウェアはWIDEプロジェクト²のメンバーを対象としたアルファテストを経た後、一般を対象としたベータテストとして公開した。本論文で述べる評価実験は、2000年秋のベータテストの実装に基づいているが、その後2001年6月にはIPv6を正式にサポートしたファームウェアが一般公開され、これ以降のヤマハ製のルータはすべてIPv6に標準対応した状態で出荷されている。

4.3 IPv6 プロトコルスタックの評価

本章では、WS-Oneの運用実験と計測実験について述べ、その評価を行なう。

²<http://www.wide.ad.jp/>

4.3.1 ICMP 処理の速度評価

BSD 系列以外の独自の IPv6 プロトコルスタックとして、組み込み用途にも用いられている Apertus での IPv6 の実装がある [56]。文献 [56] では Apertus 上の IPv6 の簡単な性能評価として ICMP 処理の時間を調べているため、比較のため WS-One の ICMP 処理の速度を計測した。

ping コマンドを用いて RTT を計測した。送信元は FreeBSD 4.8 が稼働している VIA C3 700MHz のホストであり、送信先は WS-One が稼働している RTA50i である。ヘッダを含めて 64 バイトのパケットを約 100 回送受信し、RTT を計測したところ、IPv4 で平均 $971\mu\text{sec}$ 、IPv6 で平均 $1318\mu\text{sec}$ であった。

同じ条件で FreeBSD 4.8 のホスト同士で RTT の平均を計測したところ、IPv4 で $402\mu\text{sec}$ 、IPv6 で $416\mu\text{sec}$ であった。この値から 10BASE-T で 64 バイトの IP パケット送信するのに要する約 $16\mu\text{sec}$ を引いて半分にすれば、FreeBSD 側で ICMP の送受信に要しているおおよその時間が導かれ、IPv4 で $193\mu\text{sec}$ 、IPv6 で $200\mu\text{sec}$ になる。

WS-One との通信に要した RTT から FreeBSD 側の処理の時間を引くと、WS-One での ICMP の送受信処理に要している時間は、IPv4 で $778\mu\text{sec}$ 、IPv6 で $1118\mu\text{sec}$ になる。

文献 [56] で述べられている Apertus 上の ICMP 送受信処理の時間は、IPv4 で $360\mu\text{sec}$ (最適化前で $704\mu\text{sec}$)、IPv6 で $550\mu\text{sec}$ である。この時間は i486(66MHz) のホストでの値である。

4.3.2 安定性の評価

WIDE プロジェクト主催のワークショップが 1999 年 9 月 6 日から 1999 年 9 月 9 日にかけて開催された。著者はワークショップ会場ネットワークと外部ネットワーク間の接続に、WS-One を一組提供した。ワークショップには 200 人以上の技術者、研究者が参加した。著者らの実験の目的は、WS-One のパフォーマンスと安定性の評価であった。

会場ネットワークと外部ネットワーク間は、静的経路制御を利用した。最初の実験案は IPv6 のうちのいくつかのサービスのみを WS-One 経由で提供するというものであったが、WS-One の接続性が安定していたため、ワークショップ期間中 IPv6 の全パケットが WS-One 経由で通過した。

実験結果を図 4.4 から図 4.8 に示す。グラフの横軸は時刻を示し、縦軸は ISDN のチャンネル数を示す。

3 日の実験期間中、異常終了を起こした回数は 2 回であった。この原因は、ISDN インタフェースの頻繁な接続と切断であると考えている。現地でのソフトウェアの改修の結果、最終的に WS-One は最大約 12 時間連続稼働し続け、当初期待した安定性を確認できた。

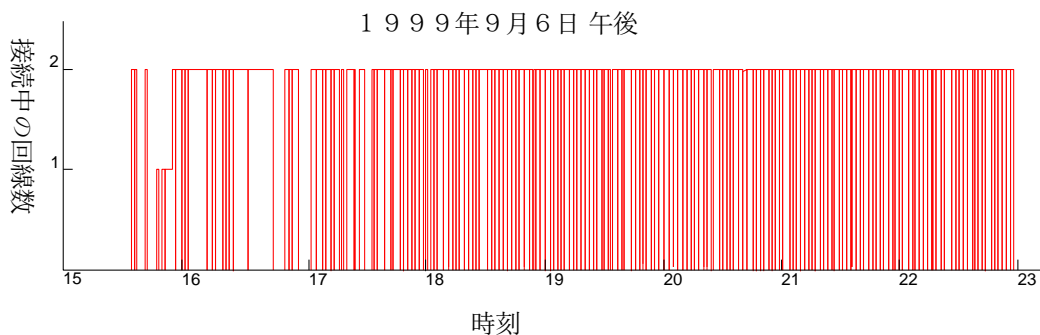


図 4.4: 1999 年 9 月 6 日の結果

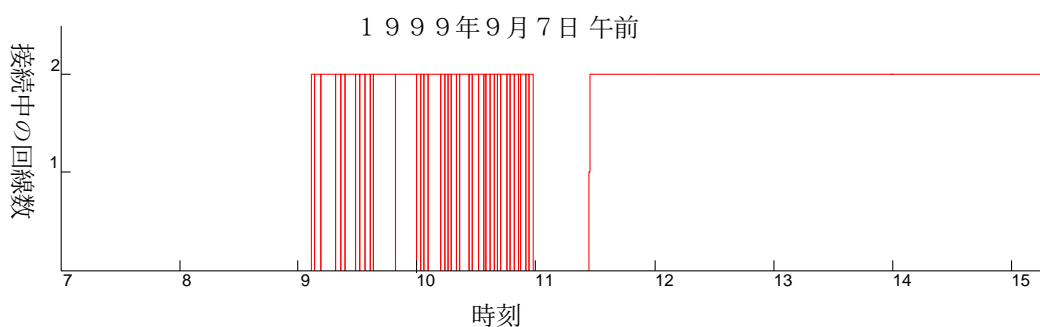


図 4.5: 1999 年 9 月 7 日午前の結果

4.3.3 通信速度の計測と評価

前述の WIDE プロジェクトワークショップの運用実験では、転送量の推移のデータは計測しなかった。そこで、この運用実験と同等の状況を想定して、WS-One 間の転送速度を計測した。この実験では以下の調査を目的としている。

- IPv6 タスクを分離したことによる速度への影響
- IP ヘッダの構造の変更による速度への影響

実験環境を図 4.9 に示す。ホスト A ~ B 間での `ttcp`[17] による TCP の転送速度を計測した。`ttcp` は TCP および UDP の転送速度 (スループット) を計測するために広く用いられているプログラムである。ここでバッファ長は 1024byte、バッファ数は 1024 とした。バッファ長を 1024 にした理由は、パケット長を MTU (Maximum Transfer Unit) である 1500byte に近づけるためである。これにより、ISDN の最大転送速度に近い転送速度が得られるものと予測できる。また、ホスト A, B はそれぞれ BSD/OS 3.1 (KAME-20000214SNAP) を用いた。ハードウェアはそれぞれ、PentiumIII 450MHz と PentiumII 333MHz、Ethernet カードはともに Intel EtherExpress 100 である。`ttcp` の IPv6 対応は独自に行った。

参考として、Router X の WS-One の設定の一部を図 4.10 に示す。Router X, Y は WS-One 同士の場合と、同等の版の IPv4 のみを搭載したファームウェア (Rev.

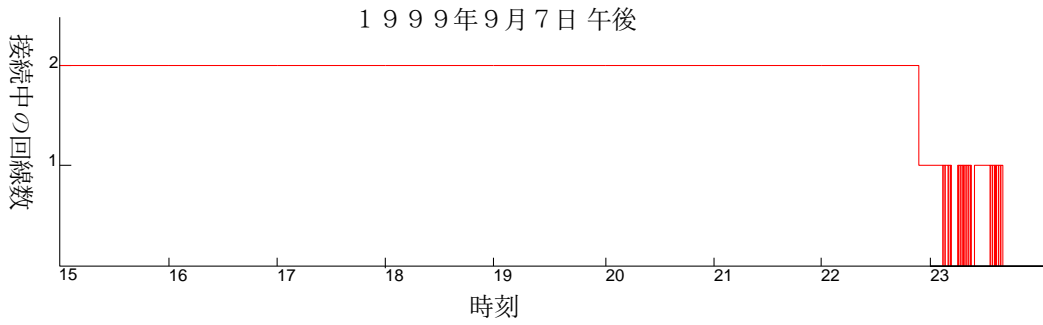


図 4.6: 1999 年 9 月 7 日午後の結果

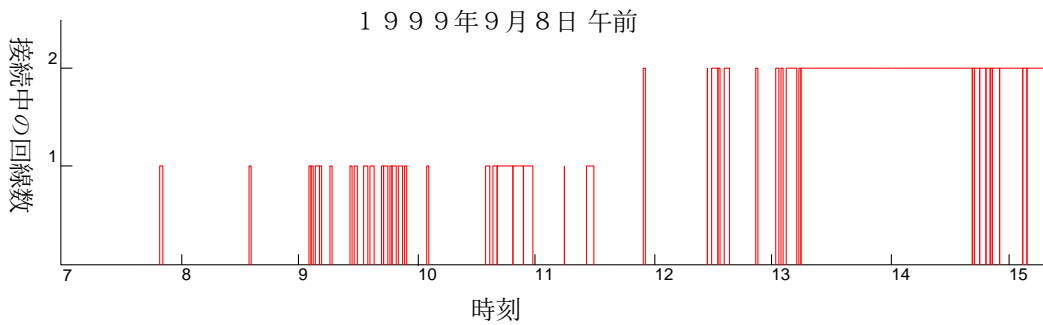


図 4.7: 1999 年 9 月 8 日午前の結果

3.05.30) 同士の場合を計測した。WS-One については、IPv4 と IPv6 での転送速度を計測した。計測は 1B(64Kbps)、2B(128Kbps) それぞれについて 20 回行い、最大値と最小値を除いた 18 回分の値の平均と標準偏差を求めた。

	1B			2B		
	平均	標準偏差	補正後	平均	標準偏差	補正後
IPv4	59.39	5.3×10^{-3}	61.71	117.0	1.1×10^{-2}	121.6
WS-One/v4	59.38	7.1×10^{-3}	61.70	117.0	9.0×10^{-3}	121.6
WS-One/v6	57.97	5.4×10^{-3}	61.37	113.5	2.1×10^{-2}	120.1

表 4.1: tcp による転送速度 (Kbps)

実験はルータ同士の PPP 接続は完了している状態で始めた。したがって、PPP 接続の調停にかかる時間は含まれない。ルータ間は、ワークショップの運用環境と同等にするために、NTT の ISDN 公衆回線を経由した。tcp はペイロードの転送速度のみを計測する。IPv4 と IPv6 では IP ヘッダ長が異なるため、ルータ間の転送速度を求めるためには補正を行う必要がある。この補正では、一つのバッファが一つのパケットとして送信されるものとし、全送信バイト数に、TCP,IP ヘッダ長 (IPv4 では $20 + 20 = 40$ byte, IPv6 では $40 + 20 = 60$ byte) をバッファ数分加算したものを、転送時間で除算した (結果表の補正後の項目)。ヘッダやペイロードなどの圧縮

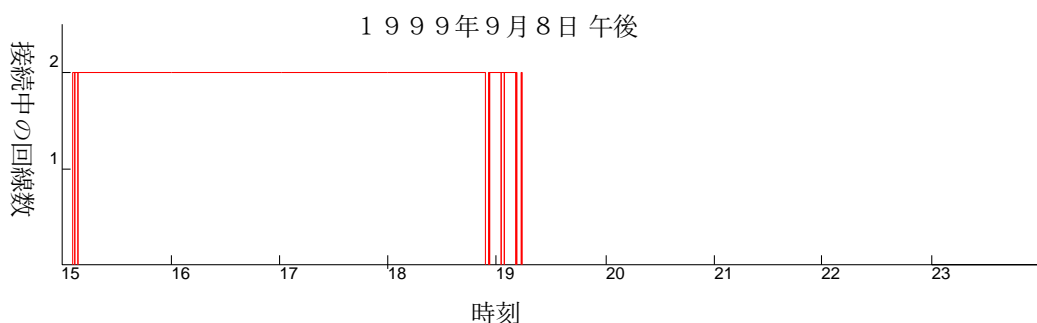


図 4.8: 1999 年 9 月 8 日午後の結果

は行っていない。実験結果を表 4.1 に示す。なお結果については次節で考察する。

4.3.4 混在環境での速度計測

IPv4 から IPv6 過渡期では、両方のパケットが混在する環境が想定される。そこで、この環境を想定した WS-One の転送能力の測定を行った。実験環境は前述の実験とほぼ同等である。ただし、公衆回線の代わりに ISDN 擬似交換器を用い、両端のホストの性能は前述でのホスト B と同じものを用いた。実験では 2B で接続が確立された状態で、同じく `ttcp` による速度測定を行った。IPv4 と IPv6 の `ttcp` を同時に実行し、20 回の計測のうち最大最小を除いた 18 回分の平均と標準偏差を求めた。結果を表 4.2 に示す。

	2B	
	平均	標準偏差
WS-One/v4	59.38	2.6×10^{-1}
WS-One/v6	57.14	2.3×10^{-1}

表 4.2: IPv4, IPv6 混在環境での転送速度 (Kbps)

4.3.5 相互接続試験

実装としての品質を向上させ普及を図るためには、相互接続性試験と普及に向けての活動が必要不可欠である。著者は 1999 年 9 月に行われた TAHI³相互接続性試験イベントに参加した。この実験には 15 組織 18 種類の実装が参加した。WS-One に関しては、実験ネットワーク内に ISDN 擬似交換器を介して接続し、IPv6 の基本機能について、他社との相互接続性を確認した。TAHI プロジェクトは試験プログラムを公開しており、これの非公式な版には WS-One への対応コードが含まれている。

³<http://www.tahi.org/>

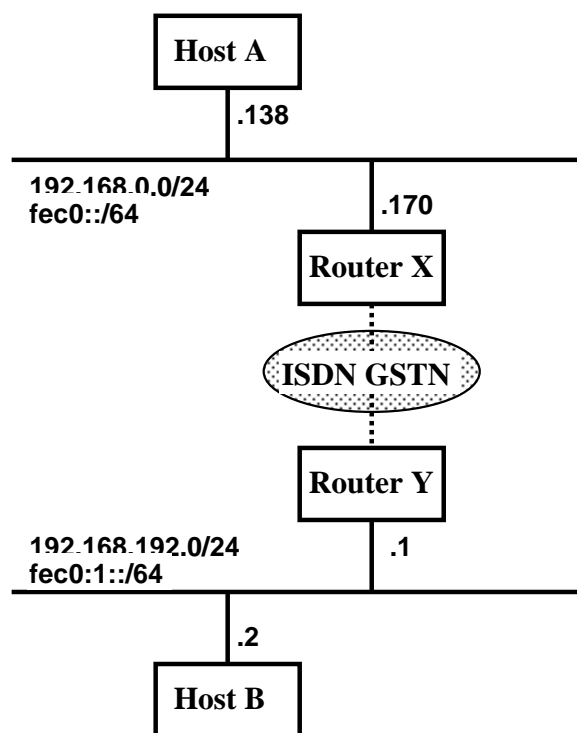


図 4.9: 実験ネットワーク

4.4 IPv6 対応 ISDN ルータの実装に関する考察

4.4.1 安定性の考察

WIDE プロジェクトのワークショップで行なった実験結果では、一日あたりの平均累計接続時間は約 10 時間、最大連続接続時間は約 12 時間であった。このような仮設的ネットワークを構築するには、導入にかかる費用を含めると専用線より ISDN の方が低コストである。このことから、短期間のインターネット接続性が必要な場合、ISDN 接続が適している。また、ダイヤルアップ環境では、連続して 12 時間以上接続することはまれであり、初期のファームウェアの安定性としては十分であったといえる。

4.4.2 性能評価の考察

最初に ICMP の処理速度について考察する。Apertos 上のスタックとの比較では、WS-One のほうが IPv4、IPv6 とともに遅いが、動作ハードウェアが異なるため単純な比較は難しい。そこで IPv4 と IPv6 との比較を行なうとどちらも IPv6 の ICMP 処理のほうが時間がかかっており、Apertos では処理時間の差は $190\mu\text{sec}$ で、WS-One では $340\mu\text{sec}$ である。CPU やメモリ転送能力を平均したものが同程度だと仮定すると、WS-One の IPv6 独自の処理、すなわちタスク切替えとタスク間通信に $150\mu\text{sec}$ かかっていることになる。

```

ip lan address 192.168.0.170/24
ip route 192.168.192.0/24 gateway pp 1
ipv6 route fec0:1::/64 gateway pp 1
ipv6 prefix 1 fec0::/64
ipv6 lan address fec0::2a0:deff:fe06:99b5/64
ipv6 lan rtadv send 1
pp select 1
isdn remote address call XXXXXXXX
isdn remote address arrive XXXXXXXXXX
isdn disconnect time off
ppp ccp type none
pp enable 1

```

図 4.10: RouterX の設定

多くのTCPのパケットがMTUの大きさに近いとすると、MTUの大きさのパケットに対してこの時間がどの程度の負荷になるかを考察する必要がある。EthernetのMTUである1500Byteを送信するためには、回線速度が128Kbpsの場合は $91553\mu\text{sec}$ を要し、1Mbpsの場合は $11444\mu\text{sec}$ を要し、10Mbpsの場合は $1144\mu\text{sec}$ を要する。 $150\mu\text{sec}$ はそれぞれに対して、0.2%、1.3%、13%のオーバーヘッドになる。IPv6タスク処理の時間はCPU能力をあげれば簡単に改善できる点ではあるが、現在のハードウェアをそのまま使ったとしても数Mbpsくらいまでであれば許容できるオーバーヘッドであると推測できる。

転送速度については、計測結果から以下の点が明らかになった。まず、WS-OneのIPv4での転送速度は、v4のみのファームウェアと同等である。また2Bの時の速度は、IPv4、IPv6ともに1Bのほぼ2倍の値になっている。このことから、IPv6タスクは実装の他の部分の性能に影響を与えていないと言える。WS-OneのIPv6、IPv4の転送速度を比較すると、IPv6のほうが僅かに遅い。計測回数が18回であるので誤差の範囲とも考えられるが、標準偏差からすると、おそらくIPv6タスクの分離による速度低下であると考えられる。しかしWS-Oneでの1Bでの結果を例にとると、平均の差が約2.5%であるのに対し、補正後の差は約0.5%である。このことから、64Kbps程度の速度の場合は、内部処理よりもIPv6のヘッダ長の増分による速度低下の方が影響が大きく、ヘッダ圧縮などの導入が効果的であると考えられる。

またIPv6とIPv4の混在環境での性能評価については、それぞれのプロトコルにほぼ均等の転送能力が割り当てられていることが示せた。これに関しては、WS-Oneの実装ではIPv4もIPv6のパケットも区別せずに到着順に転送処理を行っているため、ルータでの処理で差が出ないのは妥当な結果だといえる。

なお、WS-Oneを元に開発された製品版のプロトコルスタックは著者らの手によるものではないが、2003年7月の時点で数十Mbps程度の速度に対応した製品がヤ

マハ株式会社から製品化されている点を参考として述べておく。

4.4.3 現在の技術に与える効果

1998年の時点では、処理能力の小さな当時の組み込み用ハードウェア上でIPv6プロトコルスタックを動作させること自体が十分意義のあるものであった。そもそもの開発目的が、「動かす」ことにあったため、研究目的は達成できたと言える。

とはいえ、ハードウェアの能力は年々進歩しているため、処理能力の小さなハードウェアで動作させることに意味はあるのかという疑問もあるかもしれない。そこで、当時の技術が2003年の現在にどのように生きるかについて考察する。

TACA⁴では、超小型の組み込み機器(LNCA)用IPv6プロトコルスタックの仕様について議論している[68]。そこで挙げているセンサのハードウェア性能としては、CPU 40MHzのマイコン、メモリ1MB程度というスペックが挙げられている。これは当初WS-Oneが動作していたハードウェアとほぼ同程度であり、このスペックでIPv6スタックが動作したという実績を示している点は重要である。

同時にこの程度のスペックでルータとして機能できることを早期に示した意義は大きい。なぜなら、超小型組み込み機器をウェアラブルコンピューティングに用いる場合、各ノードがアドホックネットワークを構成し、対外接続用ルータを経由して外部へ接続するからである。アドホックネットワークでは、すべてのノードがパケットの転送を行なうルータとしての機能を持つ必要がある。この時の要求性能としては、例えば対外接続が3G(144Kbps～2Mbps)で内部の接続がBluetooth(1Mbps)程度のものが求められると思われる。WS-Oneの後継スタックではRTA54i(SH3 80MHz)で数Mbps程度の速度を実現しており、この要求に十分応えられる。

一つのプロトコルスタックを開発するにあたっては、実装を成熟させる期間も含めてそれなりの時間を要する。WS-Oneは早期からIPv6対応を実現していたため、TACAで提唱されているような新しい利用形態が現れた場合でも、成熟した実装を迅速に提供できる。

⁴<http://www.taca.jp>

第5章 ネットワークトラフィック可聴化システムの設計と実装

著者が所属する研究室で1995年より開発してある stetho[65] は、ネットワークトラフィックをパケット単位で音声(以下では音、言語、音楽などを含めた意味で音声という言葉を用いる)に変換するシステムである。通常ネットワークシステムの稼働状況の監視には、文字やグラフなどを用いた情報の可視化の手法が用いられる。これに対して stetho はトラフィック情報の音声による表現すなわち可聴化を行なうものである。

本節では、stetho の設計と実装および音声を管理情報として意味付けできることを検証するための評価実験について述べる。

本節で述べる内容は、図 2.5 のうち、ISNA の性能管理と障害管理を支援するアプリケーションにあたる。また、この内容は国際会議に採録され [35] 評価を受けている。

5.1 ネットワーク監視手法の比較

5.1.1 管理作業の分析

ネットワークや計算機システムの管理作業は、以下の作業の流れから構成される。

1. 監視
2. 判断
3. 対処

この流れを延々と繰り返すことになる。このうち、監視作業は管理作業の中で大きな位置付けであると同時に、管理者に大きな負担がかかる。適切な判断を行なうためには監視した情報を適切に取舍選択する必要があるが、これには一般的に知識と経験が必要なためである。本研究では監視作業の支援を対象とし、最初に監視作業の特徴を分析するところから始めることにする。

まず監視対象となる情報の傾向を分析する。これらの情報は時々刻々と変化し続ける。これはネットワークの利用状況が常に変化し続けることと、ネットワークの構成が変化し得ることが原因である。次の傾向として、種類が多いことが挙げられる。ネットワークは様々なシステムから利用され、その上で様々なサービスが稼働しているため管理者は多くの種類の情報を扱う必要がある。更に、特に近年ネット

ワーク規模の拡大に伴って、常に多量の情報が発生し続けるために、トラフィック情報やサーバが出力するログの分量が多くなる傾向にあるという特徴が挙げられる。

次に監視作業の内容を分析する。監視作業には通常何らかの支援系を用いる。支援系は、情報を入手する機能と、それを管理者に提示する機能から構成される。前者は例えばSNMP[31]によるイベントや状態の入手であったり、libpcap[13]を用いたパケットの取り込み、IDSを用いた侵入検知などである。後者は収集した情報を出力する方法で、個々の入手方法に対応した出力方式であったり、HPのOpenView[77]のような統合管理システムやMRTG[9]のようなグラフに出力するシステムもある。

ここで監視作業の実例をいくつかのシナリオとして列挙する。

- ネットワークに何か異常が発生した場合、とりあえずtcpdumpを実行しトラフィックを観察する。
- ディスクの使用量が9割を過ぎた時点から、その後の変化を観察する。
- IDSの同一のイベントが100件/分を越えたら、そのイベントを監視し始めて変化を観察する。

これらのシナリオから、監視対象の情報の特徴をまとめると以下になる。

- 時間とともに量は質が変化するもの。
- 突発的に発生するもの。
- 上記の両者を複合したもの。

したがって、監視作業を容易にするためには、上記の特徴を持つ情報を監視しやすい方法で提示する支援系を開発すれば良いことになる。

5.1.2 監視方法の比較

既に述べたように、監視作業の支援系は情報の入手機能と提示機能に分離できる。提示機能としては様々な方法が提案されている。MRTG[9]、TTT[18]、Ethereal[4]などはトラフィック情報を図示するものであり、Analog[1]のように各種ログ情報を整理して提示するものもある。Etherape[3]は、トラフィックの量などをネットワークで図示するツールである。これらは視覚情報として整理した情報を提示する。

著者は聴覚を用いた監視方式に着目する。

視覚と聴覚の比較については、既に多くの文献で述べられている。

例えば文献[45]では、人間の聴覚には大別して三つの特徴があるとしている。第一の特徴は聴覚が常に開かれているということである。目を通しての視覚を利用した入力は、観測者が観測対象として注目したものにのみ開かれているため、観測者は視野に入らない事象に気付くことができない。それに対して、耳を通じた聴覚による入力は常に全方位に対して開かれているので、観測者はあらゆる事象に気付く

ことができる。この事実は観測者が全ての注意を音に対して注ぐ必要がないことも表している。例えば、観測以外の作業にほとんどの意識を向けている場合でも、観測者は警告音に気付くことができる。

第二の特徴は時間と共に変化する事象の把握に、視覚よりも聴覚の方が優れている点である。その根拠としては、音からは動的なイメージを連想しやすいことや、音の間の時間差をリズムとして細かく把握できることなどが挙げられる。

第三の特徴は複数の事象を同時に把握するのに適しているという点である。これは、人間が音の方向に敏感であることや、さまざまな音のパターンを記憶して異なるパターンを識別できることから導かれる。この特徴の例としては、オーケストラによる合奏の中から、特定の楽器の旋律を聴き分けられることが挙げられる。

しかしこれらの特徴を補うような視覚インタフェースを開発することは不可能ではない。第一の特徴については、管理者のほとんどがモニタを見ながら作業しているとすれば、画面上に強制的にアラートを表示させることで強制的に意識に割り込みをかけることはできる。第二の特徴については、折れ線グラフによる情報の出力は時間軸での変化を把握するのに十分である。第三の特徴についても、同時に多数の情報を提示された時に、特定の情報を見分けることと聞き分けることは同程度に難しいと想像できる。

しかし当り前のことであるが、目の見えない人にとっては音声インタフェースは有効な手段であるだろうし、耳の聞こえない人にとっては映像インタフェースが有効であろう。音声と映像とを比較して、どちらの提示手段が優れているという議論は無意味である。

ただ、現在のネットワークの監視作業はほぼ視覚情報のみに頼っており、そこに聴覚情報を介在させることの有効性を議論する必要がある。従って目指すべきは、映像での情報提示よりも優れた手段の提供ではなく、同程度または併用することによって効果が増強される手段の提供である。

そこで本論文では、主に視覚情報に依存しているネットワーク監視作業に聴覚情報を導入することの可能性の議論と、それが視覚情報と同程度の効果を得られるか否かの調査を目標とする。

5.1.3 音声を用いたネットワークの監視

ネットワーク監視作業に音声を導入することにより、例えば以下のような利点が考えられる。

ping コマンドは ICMP REQUEST パケットを送信して ICMP REPLY パケットを受信し、その間の時間を測ることで二つのホスト間のネットワーク的距離を調べるものである。この二つのパケットを異なる音声に割り当てて再生すれば、二つの音の間隔から二つのホスト間の距離を間隔的に知ることができる。ping コマンドの名称になった潜水艦のソナーと類似の感覚である。これは管理作業だけでなく、教育的な効果も望めるであろう。

ネットワークの状態を音声で表現して常に流しておくことにより、ネットワーク管理に精通していない人であっても、定常状態と異常状態の判別が可能であるかも

しれない。トラフィック量を音の変化に反映させるようにすれば、ネットワークの利用率が分かるであろうし、異常なパケットに対応して聞き鳴れない音が鳴れば、専門家でなくても異常に気がつくであろう。

また、音声で監視ができるので、テレホンサービスでこの音声を流すようにしておけば、コンピュータを持ち歩かなくても電話だけで監視作業が可能になる。

これらを発展させると、究極の目標の一つはネットワーク管理者のためのBGMの作成である。これはデパートのBGMに例えて説明できる。例えばデパートのBGMは来店している客が聞いても普通のBGMだが、曲の種類などを用いて従業員に合図を送るという使われかたをすることがある。すなわち、符号を理解している人にとっては音楽を聞いて適切な状況理解と判断が可能である。同時に符号を理解していない来客であっても、「蛍の光」が鳴ると感覚的に店を出ようとする。このような感覚的な判断に加えて、特別に注意喚起を行なう場合は館内アナウンスを出す。このような音声の利用の仕方を、ネットワーク管理にも適用できないかというのが、著者の目標の一つである。

5.1.4 研究の目標

本研究の目的は、ネットワーク監視作業への音声情報の導入の可能性を探ることになる。理想としては、管理者のためのBGM作成であることを前節で述べた。しかし、情報伝達として機能すると同時にBGMとしても心地よい音楽の製作は容易ではない。そこで最初の段階として、既存の監視作業に用いるツールの表示手段を音声出力に置き換えるシステムを開発する。具体的には、トラフィックを監視するための最も基本的なツールである `tcpdump` の出力を音声に置き換えるシステムである。

また、研究の目的は音声インタフェースの開発や改良ではなく、あくまでネットワーク監視支援システムの開発にある。このため、システムの評価は監視支援システムとしての評価を重視する。先の議論で監視作業のシナリオを整理したところ、以下の二点または両者の複合を実現できるかを検証する必要があるという結論になった。

- 時間変化を把握できるか。
- イベントの発生を把握できるか。

そこで、論文中で開発するシステムと `tcpdump` とを比較して同程度の効果をもたらすか否かについて評価する。

5.2 関連研究

情報の提示に音声を用いるものとしては、計算機のインタフェースに音声を用いるというものがある。文献 [59] ではメニュー項目を音に変換し、同時に発音した場合の認識率を調査している。文献 [28] では音声版のアイコンである“Earcon”を提案し、どのような音を割り当てるのが聞き取りやすいかを提示している。

また計算機で扱うデータを音声にして、目的の情報を捜し出すという行為は一種のデータマイニングであり、実際データマイニングの分野などで音声インタフェースを用いる試みが行なわれている。文献 [78] では並列プログラムの聴覚化が行なわれており、並列プログラム動作時の計算機の状態やシステムコールなどのイベントを音声に割り当てることでプログラムの動作を把握できるか否かを調べている。音声から把握した結果と実際のソースコードとを比較し、プログラミングの知識があれば動作をイメージできることを示している。文献 [41] では時系列データの音声表現の有効性について調査している。二次元グラフと三次元グラフで表現されるデータそれぞれを音声表現に割り当てて、被験者が値の変化を把握できるかを調査している。この実験では、二次元グラフについてはおおよそ把握できるものの、三次元グラフについては把握できるとは言いがたいという結果が出ている。

本研究での目的と同じように、ネットワーク監視作業の支援ツールとして音声を用いている事例として、WebMelody と Peep の二つを挙げる。

WebMelody[27] は、WWW サーバのログを音声に変換する。ログの内容に応じて、SMF を再生する。聞いていて疲れな音を生成するように工夫しているが、システム監視ツールとして見た場合監視対象が限定されている。また、本研究で開発した stetho ほど多様な音源に対応していないという差異がある。

Peep[24] は、観測部と発音部が分離され、ネットワーク経由で通信する。ログに対するパターンマッチや、CPU 負荷などの条件に応じて音をだせる。stetho でこのような機能は外部モジュールに依存する形になる。Peep は MIDI イベントではなく PCM 音を再生するため、動的に変化する情報を音声として表現しにくいといった欠点がある。

Algorithmic Composition と呼ばれる分野では、フラクタル集合 [6][5] や円周率 [21] などの数列を音楽に変換する試みなどが行なわれている。近年の研究でネットワークトラフィックは大局的には自己相似性を持つことが知られており [37]、本研究での試みは Algorithmic Composition の典型例の 1 つであると言える。しかし本研究での stetho が発生する音声は単なる音の羅列ではなく、管理情報に対応した意味づけができるという点で、上記のような Algorithmic Composition とは異なる。

5.3 stetho システム

5.3.1 stetho システムの概要

著者らの研究室では、ネットワークトラフィックを可聴化する stetho システム (以下 stetho) を 1995 年より開発している [65]。stetho の目的は、従来 tcpdump を用いていたトラフィックの監視作業を、音声を用いて行なおうというものである。tcpdump は一つのパケットについて一行の情報を出力する (例外もある)。そこで stetho では tcpdump の出力を読みこみ、行単位で正規表現によるパターンマッチを行ない、適合した行に対応する MIDI イベント列を発生する。

しかし tcpdump からの入力を読み取るのみでは、遠隔のネットワークの監視が出来ないことと、IDS(侵入検知システム) や SNMP エージェントが発生させるイベン

トなどの抽象度の高いイベントを受けられないという問題がある。そこで、ネットワーク経由でさまざまなイベントを受け取る機構を用意した。

5.3.2 stetho システムの設計

stetho の構造を図 5.1 に示す。以下を設計目標とした。

- 多くの MIDI デバイスへ対応する。
 - 音源は stetho 内部には持たず、外部音源に依存する。
 - 音源には演奏情報のみを送ることにより、動的に音高やリズムを変化させるといった処理にも対応できるようにする。
- 拡張ポートによる外部入力を受付機能を追加する。
- さまざまな音楽表現を記述可能にする。
 - 一般的な MIDI シーケンサーと同等の機能を持たせる。

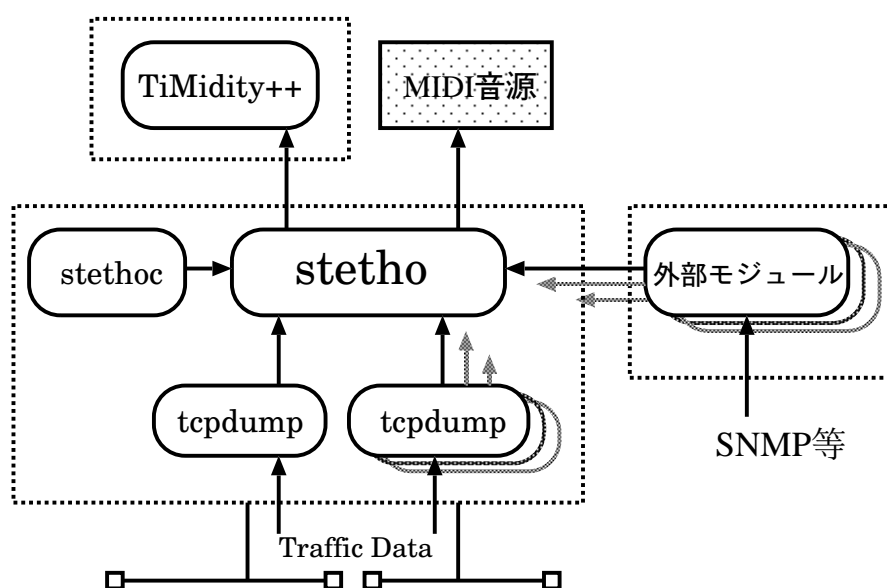


図 5.1: stetho システムの設計

5.3.3 stetho システムの実装

実装は C 言語を用いた。現在のコード量は約 7000 行である。

まず出力デバイスとして、ソフトウェア MIDI 音源である TiMidity++[16] を標準採用とする。TiMidity++ 2以降では、リアルタイムのイベントをネットワーク経由で読み込んで再生できるようになったため、この機能を用いることにした。

また、UNIX 用サウンドドライバとして著名な、OSS[12] のシーケンサ・デバイスへの対応も行なった。これ以外に raw MIDI デバイスと、シリアル MIDI も利用できる。OSS 対応は PC-UNIX を意識したものであるが、RS-232C 入力に対応した MIDI 音源があれば、RS-232C 端子をもつ機器上で stetho を利用できる。この結果として、stetho はほぼすべての UNIX プラットフォームで各々の特長を活かした形で利用可能であると考えている。

stetho を用いたメディアアート作品である NetSound[66] では、音楽家の手によるサンプリング音を用いたことによって作品としての完成を見たが、一般の利用者が stetho を使う場合は一般的な GM 音源などの MIDI 音源を用いることが想定できる。したがって、サンプリング音の編集ではなく設定ファイルの記述の段階で多様な表現が出来ることが望ましい。そこで、音源デバイスの厳密な制御と表現の多様化を目標として設定ファイルの文法を設計した。その際、以下の記述を可能にした。

- 1つのパケットパターンに対して、複数トラック (和音) の音を割り当てられる。
- デバイスごとの初期化シーケンスを記述できる。
- 外部起動するプロセスを tcpdump 以外にも指定できる。
- 連続してイベントが発生した場合に、重ねて発音するか否かを最大発音数を含めて指定できる。

音の発生は入力に対する行単位のパターンマッチという方針を採用した。再設計の過程では、libpcap[13] を用いて tcpdump の機能を stetho 内部に組み込む方向も検討された。これは、tcpdump 内での処理を排除できるとともに、tcpdump の出力が OS ごとに異なるという問題を解決できるという利点がある。しかし文字列に対するパターンマッチを行なうという方針の下で、tcpdump 以外の外部プロセスや後述する拡張ポートの実装が容易であるという理由から、従来通り tcpdump を呼び出すこととした。

設定ファイルの記述例を図 5.2 に示す。

更に外部に観測モジュールを追加するために、TCP での接続を受け付ける機能をつけた。外部のモジュールからの入力も、tcpdump 同様行単位のパターンマッチを行なう。イベントの発生時刻は 1 行分の入力を受け付けた時点の stetho 内部での時刻になる。このため、通常は外部モジュールと stetho の間には、ネットワーク上の遅延やそのゆらぎが少ないことを前提としている。

関連研究として前述した WebMelody[27] は、イベントに応じて SMF (Standard MIDI File) を再生する。stetho ではトラフィック量に応じて音程を変化させるが、SMF を再生するような方式ではこの対応が難しい。何故ならば、ノート・オンの際に音程を変えたらそれに対応するノート・オフを検索して音程を変更させる必要があるからである。そこで stetho では、設定ファイル中で演奏情報を MML (Music Macro Language) で記述し、それを中間コードに変換して蓄積しておく。中間コードは発音時刻と音程、音長の列で構成される。stetho 内部ではイベントのタイミングを起動時からの「時刻」で管理し、イベントの再生は発生時刻とイベントの組を

```

# # から行末まではコメント
inhibit_midi_channel=10
oss_sequencer_device=1
timidity_buffer=1.0
tcpdump_options=-n -e

define midi_initialize # MIDI 初期化イベント
A EXx41,x10,x42,x12,x40,x00,x7f,x00,x41,xf7
end

define 1 # define で音のパターンを定義する。番号を指定。
a CH @2 v110 k110 # 先頭が小文字の場合はトラックの初期化用 MML
b CH @14 v110
A o4c4,110 d4,110 # 先頭が A ~ K の大文字の場合は通常の MML。
B o3e4,110 f4,110
end

when /*\.(80|8080|http)[: ]*/
# when の後に tcpdump の出力に対するパターンを記述。
play 1 # play で再生する音の番号を指定。
mono # mono か poly を指定する。poly の場合は
poly 4 # 同じパケットが複数来たら、その度に音を鳴らす。数字は
# 最大同時発音数。mono の場合は一度に一音しか鳴らさない
# 但し発音中に同じパケットが流れたらもう一度繰り返す。
vshift +10 # パケットの流量に応じて volume を変化。
nshift +1 # パケットの流量に応じて note を変化。
end

```

図 5.2: 設定ファイル記述例

要素とするキューで管理される。正規表現のパターンに対しては再生すべきフレーズと、現在再生中のフレーズの情報に割り当てられ、1/50 秒ごとに以下の処理が行なわれる。

- 時刻の更新。
- 再生中のフレーズを検索し、その時刻に発音すべき音の有無を調べる。
- 発音すべき音があったら、トラフィック量に応じた音程と音量の変化を加味して、ノート・オンとノート・オフのイベントをキューに蓄積する。
- イベントのキューを検索し、現在時刻までに発生すべきイベントを発生させる。

再生中フレーズの追加は、時刻とは非同期に tcpdump から外部モジュールからの入力に対応して行なわれる。図 5.3 に stetho 内部のデータ構造を示す。図中左の struct when は正規表現と発音する音の対応付けの情報を格納する。この構造体には regcomp() 関数で変換後の正規表現と再生すべきフレーズ、および現在再生中のフレーズへのポインタなどが格納されている。struct phrase はフレーズの情報に格納されている。設定ファイル内での宣言時に割り当てられた MIDI チャンネル、初期化シーケンスと演奏シーケンスへのポインタである。演奏シーケンスは MML を変換した形式になっており、発音時刻、ノート番号、音長の列から構成される。struct playing は現在再生中のフレーズの情報に格納された構造体であり、リスト構造になっている。フレーズの再生開始時刻と、演奏シーケンスの中の再生中の箇所を指すポインタから構成される。

再生時には struct playing を検索し、次に発音する演奏シーケンスに基づいて発音と停止のイベントを生成し、再生キューに格納する。図 5.3 中上部の playing queue は再生キューであり、再生時刻と再生イベントからなる線形リストである。

5.4 評価実験

5.4.1 実験目的

stetho はネットワークの監視作業を支援するツールである。従来トラフィックの監視を行なう場合、tcpdump の出力を眺めて経験的に把握するという方法が用いられることが往々にあったが、この作業を音声を用いて把握しやすいものにしてという目的で開発した。したがって、トラフィックの把握しやすさについて tcpdump との比較実験を行なう必要がある。

無論、生の tcpdump の出力に対して何らかの可視化を行なう技術は存在する。しかし stetho はあくまで tcpdump という基礎的な管理ツールの音声出力版という位置付けであり、比較対象としては tcpdump そのものを用いるべきである。このため、本論文では tcpdump との比較のみを行ない、他の可視化ツール等との比較については考察にて述べるものとする。

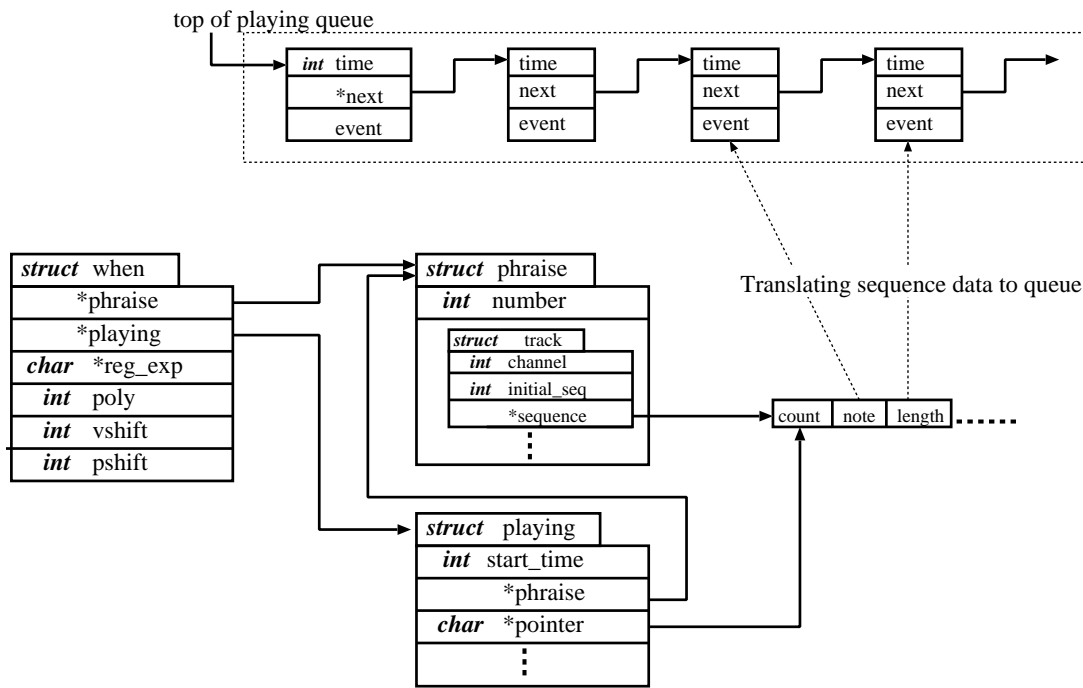


図 5.3: stetho 内部データ構造

5.4.2 実験 A

実験内容

基礎的な実験として、擬似的に発生させたトラフィックを音声に変換したものを被験者に聞いてもらい、被験者がその傾向をどう認識するかを調査した。

実験系では三台のホスト (ホスト A ~ C) を用意した。各ホストの仕様と役割は表 5.1 に示した通りである。全てのホストは 1 つの 10BASE-T ハブに接続されている。このハブはリピータハブであり、ネットワーク上のすべてのトラフィックは stetho が稼動している観測部で観測される。10BASE-T を用いたのは、一般的な対外接続の観測を想定した場合に 100Mbit/sec では高速すぎるという見積もりを行なったためである。

ホスト	CPU	メモリ	OS	役割
Host A	VIA C3 700MHz	256MB	FreeBSD 4.3	WWW サーバ
Host B	Pentium 200MHz	128MB	FreeBSD 4.1	観測部 (stetho 稼動)
Host C	PentiumII 333MHz	128MB	NetBSD 1.5	トラフィック生成部

表 5.1: 実験系のホスト

以下に述べるすべてのプログラムは ruby[53] で記述した。トラフィック生成部 (Host C) では、1KB から 128KB まで (2^n KB) の 8 種類の大きさのデータ複数個を 5 秒から 20 秒までの乱数時間間隔で WWW サーバ (Host A) から転送する。このプロセス

を同時に4つ並行して実行させた。

被験者は Host B から発せられる stetho の音声を聞きながら、5つのボタンが横にならんだパネルを操作する。1から5のボタンはそれぞれトラフィックの流量を意味するものとする。

観測するトラフィックは HTTP のみであり、音声は前述の NetSound で HTTP に割り当てられているものと同じ音を用いた。これは1つのパケットに対して「チャリン」といったベルに似た音が鳴る。また、流量が増えるにしたがって音程が上がっていきようにした。この設定も NetSound と同じものである。

なお、被験者 (A ~ D の4名) はネットワーク管理に精通し、ネットワークトラフィックパターンに関する漠然としてイメージは持っており、stetho および NetSound の音を聞いた経験はあるものの、今回の実験系でのトラフィックパターンを聞くのは始めてであった。

実験では、まずトラフィックパターンを2分間聞いて定常状態、特に最大と最小を記憶した後、観測インタフェースの操作を3分間行なった。

実験結果

実験結果を図 5.4 に示す。結果のグラフの縦軸には tcpdump で同時に観測した HTTP のパケット数 (1秒単位) と、インタフェースでのボタンの遷移を示した。stetho は内部で5秒間のバッファリングを行なった後に発音するため、インタフェースの操作遷移は5秒分ずらした結果をグラフに反映させた。横軸は操作開始である聞き始めて120秒後から300秒後までを記している。

グラフを概観すると、被験者によって操作の傾向が異なることが分かる。まず被験者 C は目立って操作回数が少ないが、後半になると操作の頻度が増している。これは操作に対する慣れが影響しているものと考えられる。被験者 A と被験者 D は、操作の頻度や段階的に値が変化する点などで、弱干類似傾向が見られる。

ピークのずれを明確にするために、結果に対していくつかの変換を行なった。まずそれぞれの被験者の結果を高低の二つに分けた。1,2,3 は低で 4,5 は高である。次に三人以上の被験者が高を示したら高い、それ以外は低をプロットした。トラフィックについては、180パケット/秒以上の箇所と、60パケット/秒のトラフィックが5秒以上続いた箇所を高にプロットした。この変換を加え、ピーク値の対応を矢印で示したのが図 5.5 である。

また、実験後に気がついた点を被験者に任意に書き留めてもらったところ、以下の内容が提示された。

- 「騒がしさ」を流量の基準とした。
- 「騒がしさ」で認識できる流量と、音程とが必ずしも一致しなかった。
- 「騒がしさ」の判別には、約1秒分を頭の中に蓄積して判別していたように思える。
- 無音状態をもって、トラフィック量が1とした。

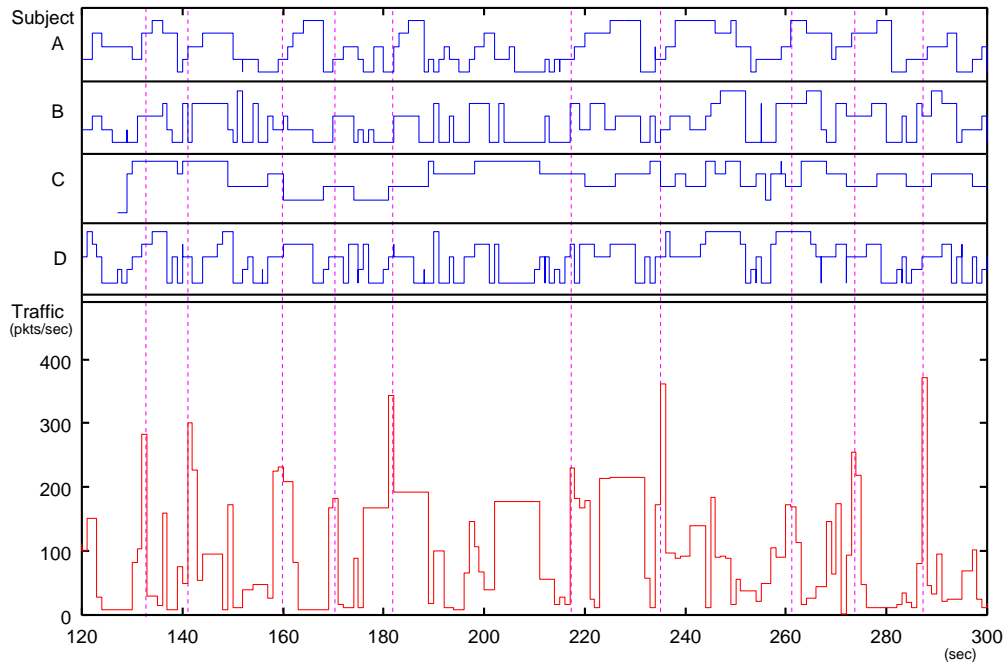


図 5.4: 実験結果グラフ

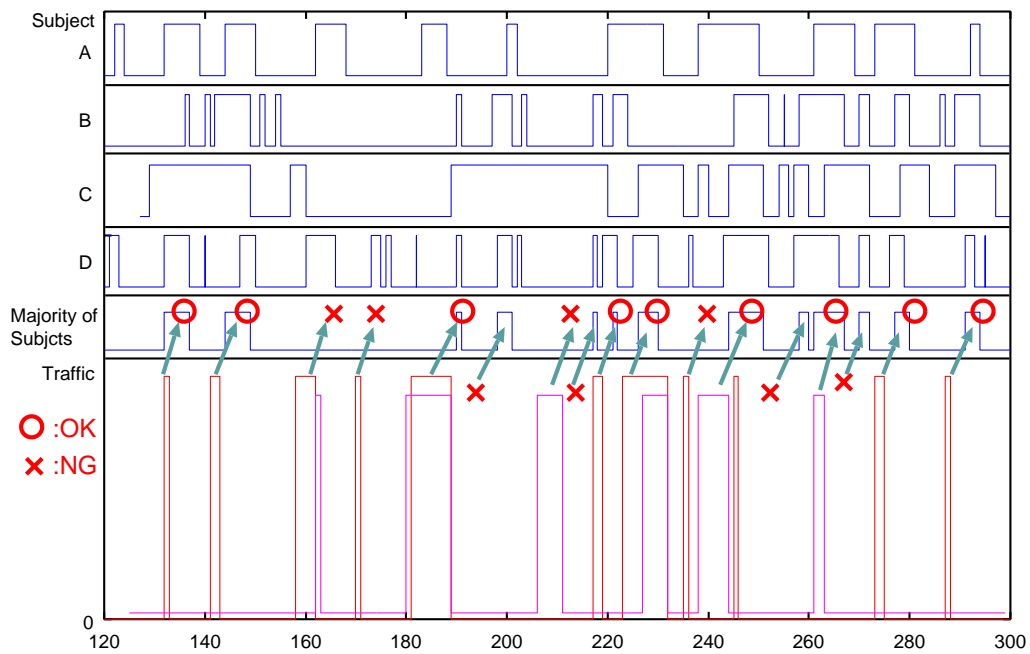


図 5.5: 実験結果グラフ (変換後)

- 断続的に音がなる時は2もしくは3にした。
- 同じ音程、騒がしさでも、長時間続くと「多量」という認識をする傾向にある。
- 音が鳴り出す時は意識にのぼるが、静かになるタイミングはあまり意識にのぼらないように思える。

5.4.3 実験 B

実験内容

tcpdump との比較実験を行なった。ランダムなトラフィックを生成し、tcpdump を模した文字による出力と、stetho による音声の出力を被験者が見聞きし、その中で発生する三種類のパケットを判別できるか否かを調査した。

疑似トラフィックは、毎秒 2 パケットから毎秒 20 パケットまで、10 秒ごとに発生速度を変化させた。ほとんどのパケットは HTTP であるが、1 秒に 1 回程度の割合で SSH、DNS、ICMP のパケットをランダムに発生させた。

実験 1(exp1) では tcpdump を模した出力、実験 2(exp2) では SSH、DNS、ICMP のパケットにだけそれぞれ赤、緑、青の色付けをした tcpdump の出力、実験 3(exp3) では stetho による音声を用いた。stetho の音声については、HTTP にピアノの音色で O3G、SSH にトランペットの音色で O4F O4G、DNS にバイオリンの音色で O5E O5D O5C、ICMP に鐘の音色で O3G を割り当てた (音高についてはそれぞれ MML 表記に従う)。メロディを割り当てたのは、文献 [28] に示されたガイドラインに従ったものである。

実験システムは FLASH ムービーを用いた 5.6。開発環境は FreeBSD 4.8-RELEASE, Ming 0.2a, ruby-ming である。

再生環境は Pentium III 550MHz, FreeBSD 4.8-RELEASE の計算機を用いた。FLASH プレイヤーは Macromedia 純正の Linux 用プラグインを用いた。

被験者はネットワーク管理に精通した管理者一名で、三種類のトラフィックパターンを用意し、それぞれを五回ずつ見聞きし、パケットを認識したら対応するキーボード上のボタンを押した。

実験結果

実験結果については、反応時間とロスト率の二点から評価する。反応時間は、パケットが発生してから、ボタンが押されるまでの時間である。パケットの発生時刻以降最近にある対応するボタンの記録を、押下したものと判断した。ただし、2 秒ボタンが押されなかったものについては、見逃したと判断しロスト数に計上した。ロスト率は、パケット数に対するロスト数の割合である。

まず単一のトラフィックパターンについて、十回連続して実験を行なった。横軸に回数、縦軸に反応時間の平均をとったグラフが図 5.7 である。また、横軸に回数、縦軸にロスト率をとったグラフが図 5.8 である。

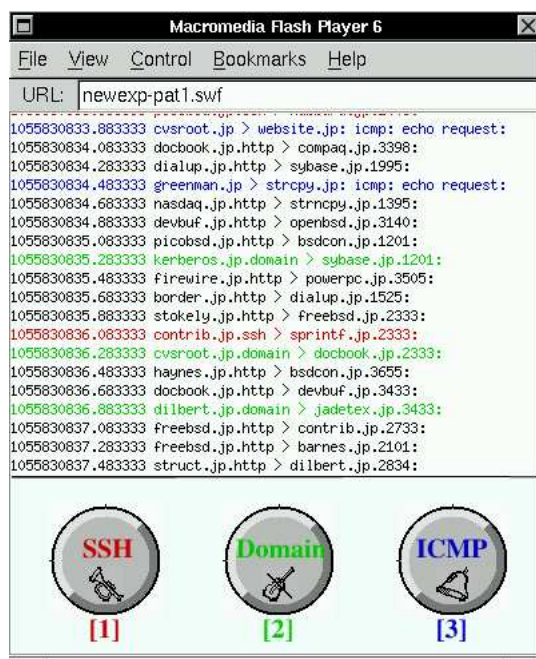


図 5.6: 実験システム B

グラフをみると、特に回数が増えても反応時間が著しく単調減少しているという
ことはなく、学習効果は現れていないことが分かる。

学習効果が見られなかったことから、三種類のトラフィックパターンすべてを合
計して集計することにした。横軸にパケットの発生速度、縦軸に反応時間をとった
ものが図 5.9、縦軸にロスト率をとったものが図 5.10 である。

図 5.9 からは、パケットの発生速度が増すに従って反応時間がわずかだが短縮さ
れていることが見てとれる。これは、次々に発生するイベントに反応するために迅
速に操作を行なった結果と思われる。また実験ごとの比較については、exp1 の反応
時間がもっとも長く、次いで exp2, exp3 となっている。exp2 と exp3 では一部で順
位が入れ替わっている箇所もある。

図 5.10 からは、発生速度が増すに従ってロスト率が増加していることが見て取れ
る。ただし、exp2 についてはほとんどロストが発生していない。exp1 と exp3 の関係
については、前半は exp3 のほうがロスト率が若干高く、後半では exp1 の方が高い。

ここで、exp1 と exp3 のロスト率に注目し、更に細かく結果を見ることにする。
exp1 についてボタンごとのロスト率を示したものが図 5.11、exp3 についてボタンご
とのロスト率を示したものが図 5.12 である。

この結果を見ると、exp1 では Button1 すなわち SSH のロスト率が高く、Button3
すなわち ICMP のロスト率が低い。また exp3 では ICMP のロスト率が相対的に高
いことが分かる。

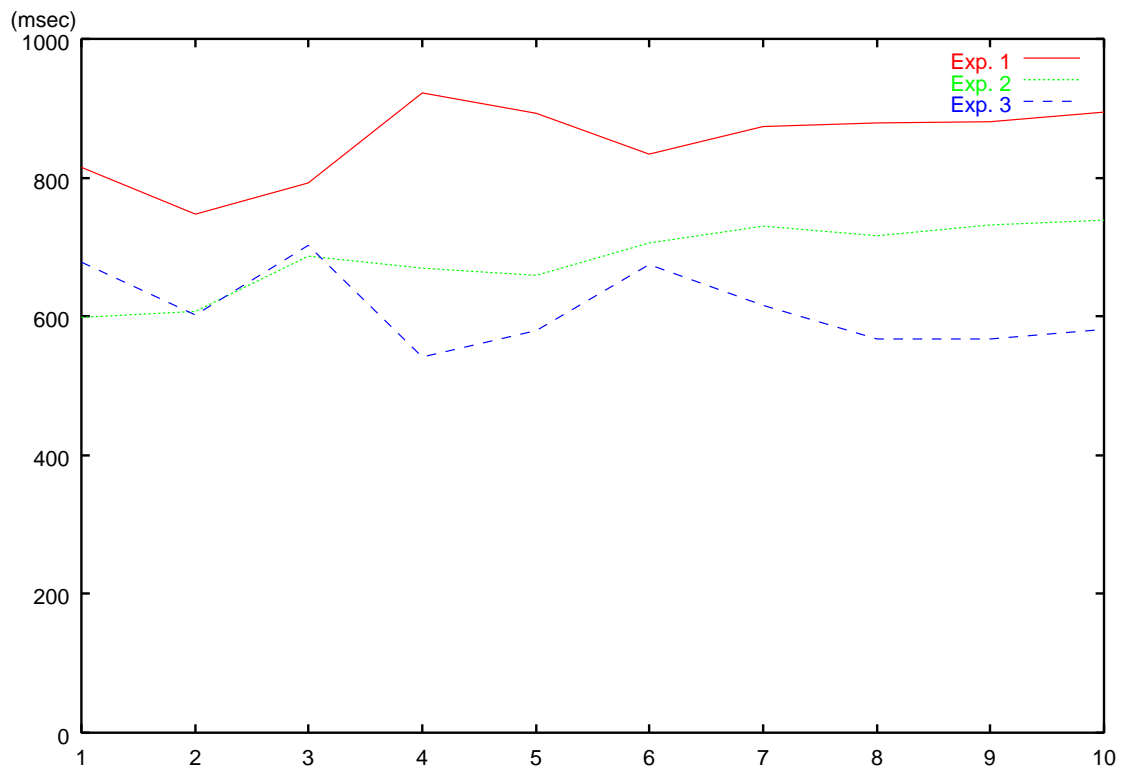


図 5.7: 実験 B 実験回数に対する反応時間

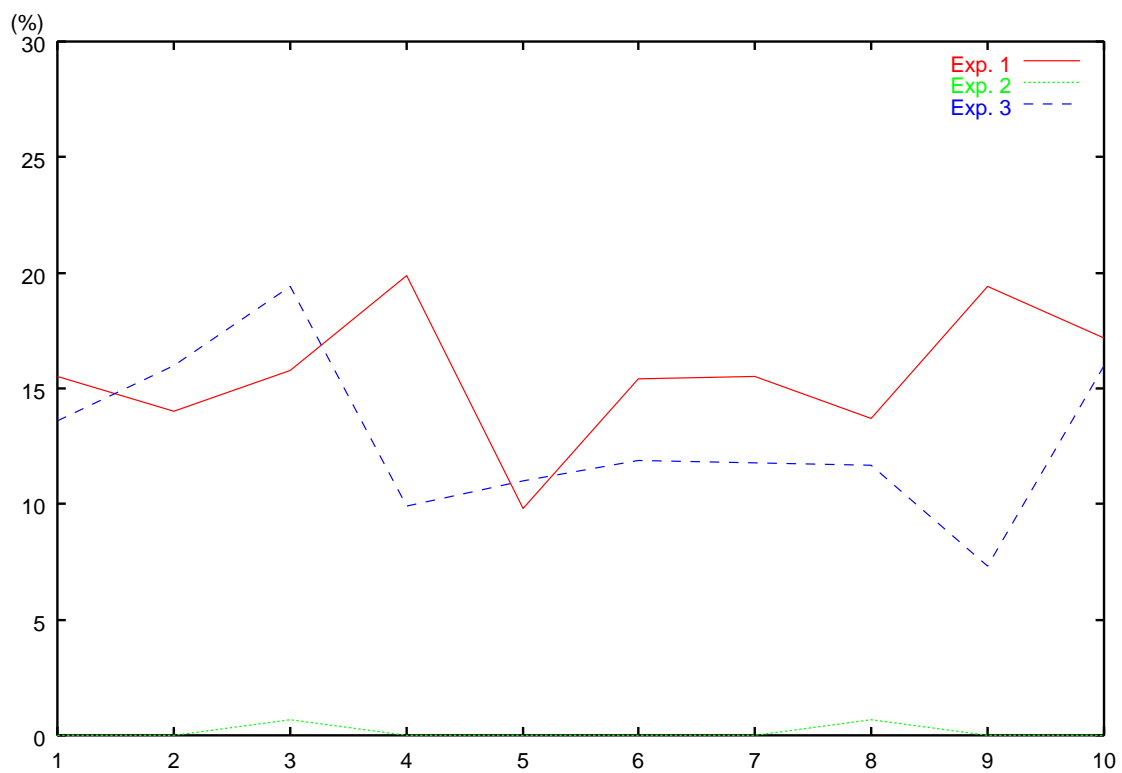


図 5.8: 実験 B 実験回数に対するロスト率

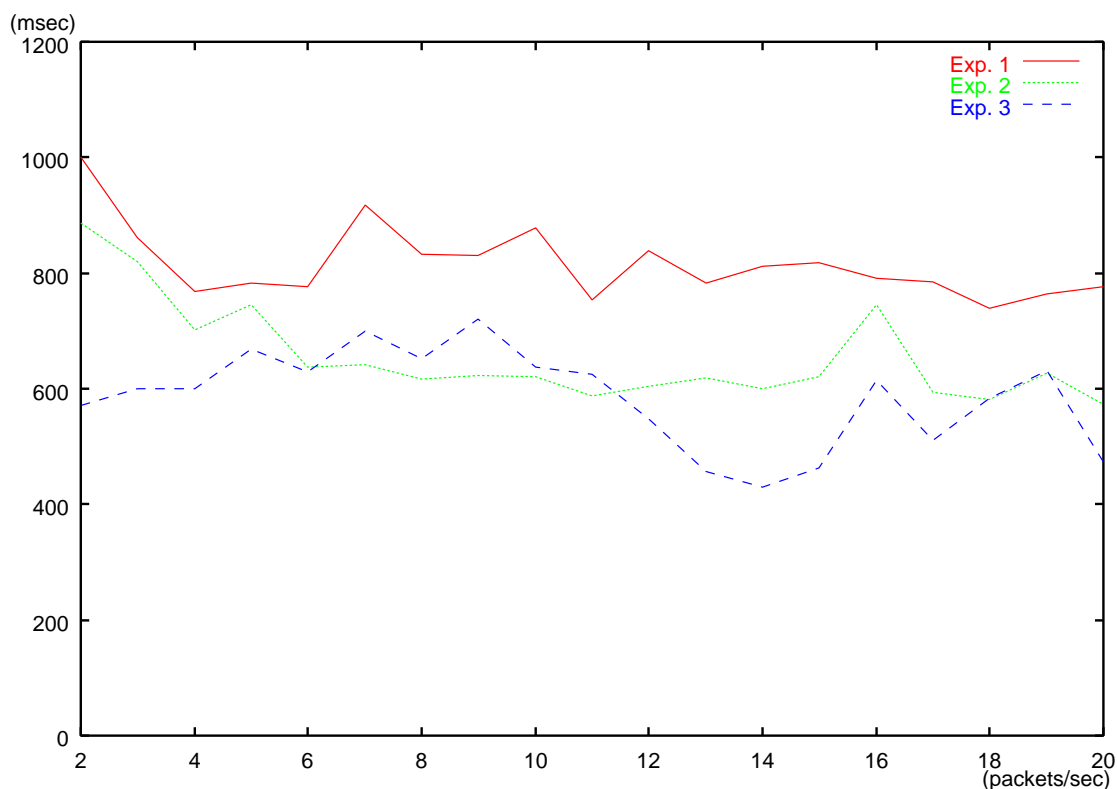


図 5.9: 実験 B 発生速度に対する反応時間

5.5 考察

5.5.1 音楽的考察

音楽的考察については、著者の研究の範囲外とする。

なぜなら、音楽的な優劣の判断は主観に依存する感性的な部分が多く、その評価基準は様々だからである。特に著者がコンピュータ音楽の国際会議である ICMC2002 に参加した感想から、いわゆる前衛音楽というのは何をもち「良い」と判断しているのか極めて理解に苦しんだという経験がある。コンピュータ音楽の専門家の間でも「心地よいとは言えない音楽をどうして作るのか [73]」という議論があるくらいである。無論、その分野の専門家である芸術家の中では、一致した見解もしくは個々の意見があるのかもしれないが、著者はそれについて明るくないし、この研究の目的からしてもこの点を追求することは本論ではない。

ただ、stetho を用いたメディアアート作品である NetSound を含めた Sensorium [15] は Arc Electronica Center ¹主催の Arc Electronica Festival '97 において金賞を受賞し、芸術作品として評価を得ている。また、stetho の論文は前述の ICMC2002 に採録されており、査読の際には stetho の音声もまた評価対象になっていることから、stetho の音声はメディアアート作品としてもそれなりの外部評価は得ていると判断できる。ただし、この時の音声は音楽家との共同作業によって作成されたもの

¹<http://www.aec.at>

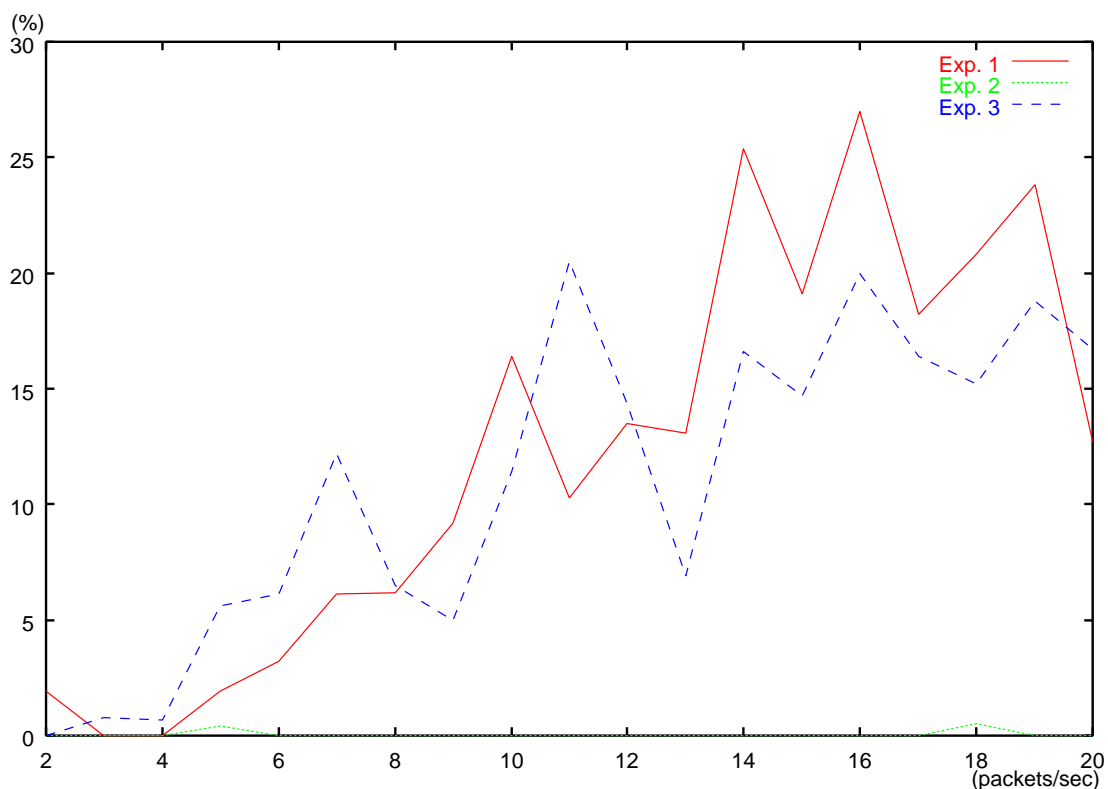


図 5.10: 実験 B 発生速度に対するロス率

であり、芸術作品としての評価を得るためには芸術家の力に依存する部分が大いとも考えられる。

5.5.2 評価実験結果の考察

実験 A

この実験の目的は以下であった。

- 被験者が stetho の音声からトラフィック量を判断できるか否かを調べる。
- 被験者がどのように stetho の音声を認識しているかを調べる。

評価実験の結果から、被験者が stetho の音声を聞いて認知した結果は、ピーク時の判別については遅延があるものの、おおむねトラフィックのピークと一致しているように見える。しかし変換を加えたグラフでの比較結果からすると、一致した箇所が 9 箇所、一致していない箇所が 8 箇所となり、優位に聞き分けられているとは言えない。

音の騒がしさを流量の判断基準とした場合、時間的に過去 1 秒分程度の流量に対する相対的な評価となる傾向がある。音程を流量の判断基準とした場合は、被験者が騒がしさから認識する流量と、stetho が発する音程から認識する流量とが必ずし

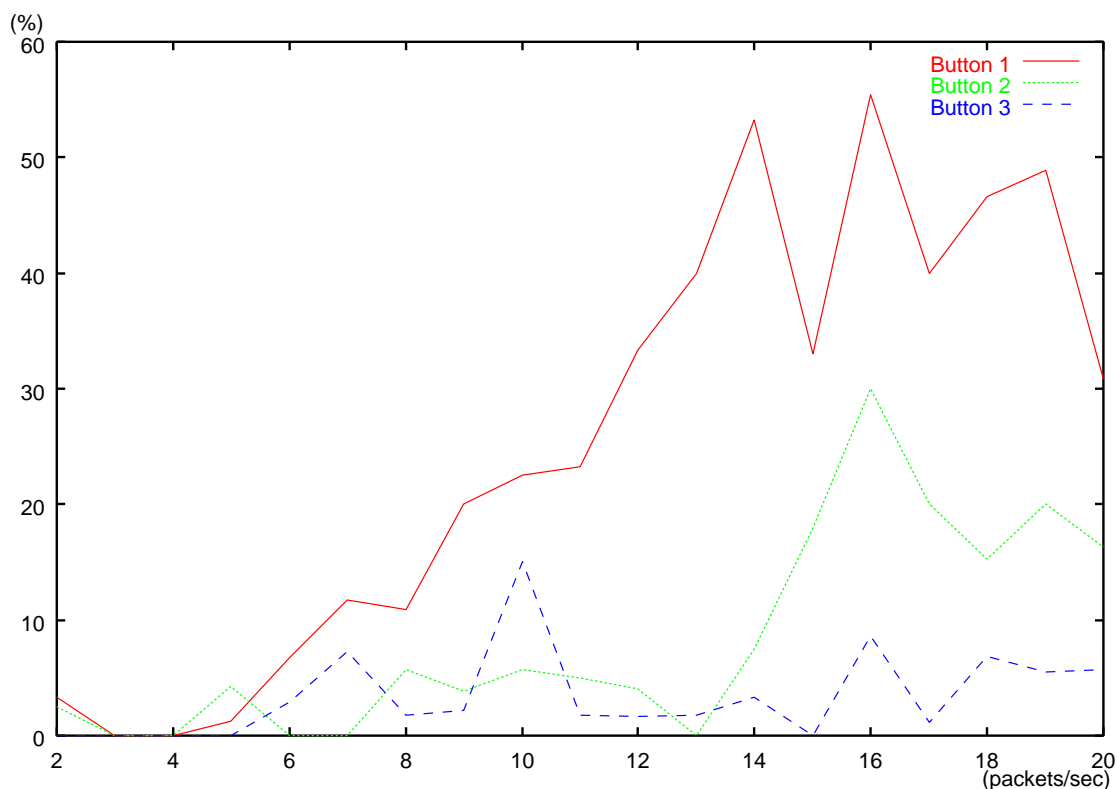


図 5.11: 実験 B 発生速度に対するボタンごとのロスト率 (exp1)

も一致しないという結果になった。音程の変化と騒がしさの変化を同時に発生させるという表現方法は、NetSound の作成の際に作品としての面白さを指向して定めたものであるが、認知という点ではあまり適さないと言える。音程を用いるのであれば、持続系の音色を用いて過去 1 秒分の流量を元にして変化させるという方法が適しているものと考えられる。

また、結果のグラフのうち、ピークが操作と一致している 9 箇所についても、ほとんどの箇所で 1 秒から数秒程度の操作に遅れが出ている。認識してから操作までのずれであろう。変換後のグラフの中でのトラフィックが連続している箇所だけに注目すると、6 箇所中 4 箇所が正しく認識できている。トラフィックが連続している箇所だけが High になっている場合でも、3 箇所中 2 箇所が正しく認識できている。このことから、連続して音が鳴る場合は、多量と判断する傾向にあると言える。

また、トラフィックの変化を漠然と把握したり、変化が起こったことを認識する目的には向いているが、変化が無くなる場合、例えば定期的に鳴っていた音が鳴らなくなるようなイベントの認識は遅れる傾向がある。

実験 B

実験 B の目的は tcpdump との比較実験である。

まず回数を重ねても学習効果が現れないことについては、文献 [59] での実験結果と一致する。

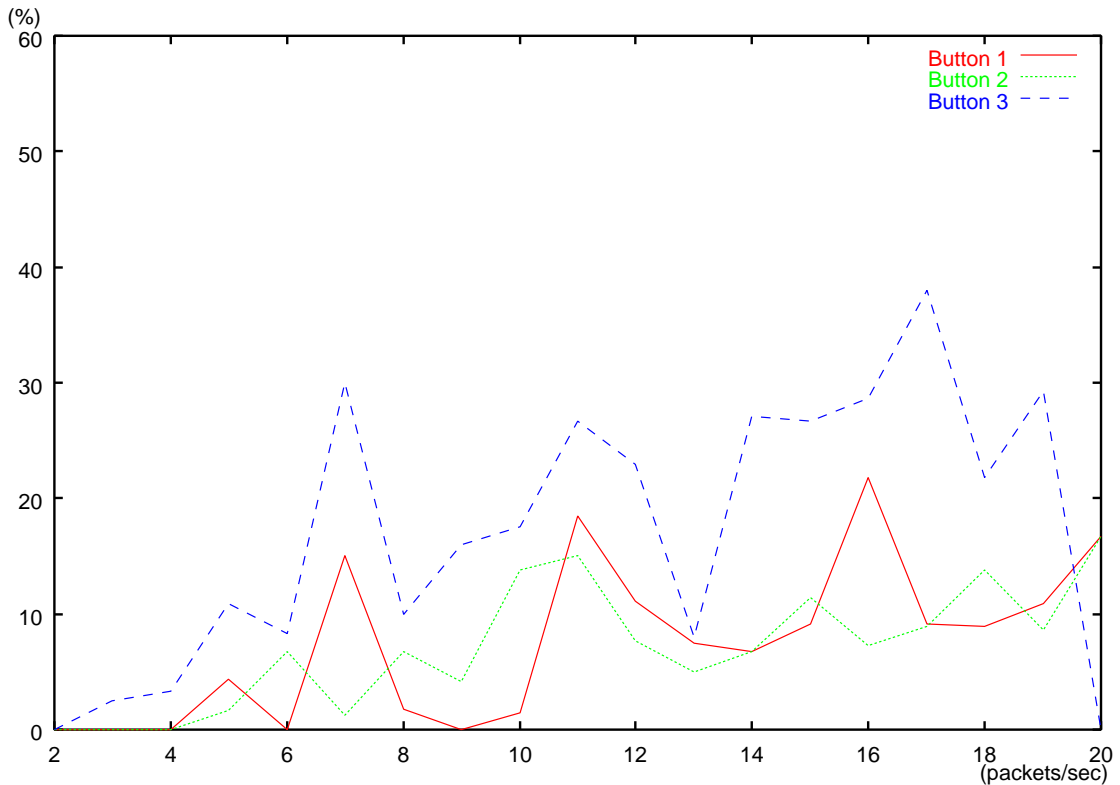


図 5.12: 実験 B 発生速度に対するボタンごとのロスト率 (exp3)

図 5.9 から、パケットの発生速度が上がると反応速度が若干向上していることが見て取れるが、これは次々に発生するパケットに対応するために反応が速くなっているものと考えられる。全体を平均すると、exp1 が 812msec、exp2 が 648msec、exp3 が 591msec であり、stetho の音声を聞く場合が最も反応時間が速い。

次にロスト率であるが、exp2 ではほとんどロストが発生していない。これは、対象の行に色が表示されているため、画面上をスクロールしている間は容易にバックトラックが可能であるためである。色が付いていない場合は、文字を読み取らなければならないため、同じように画面に表示されていてもバックトラックはできない。言い換えれば、exp2 についてはパケットの文字ではなく色のみで判別している。exp1 と exp3 のロスト率を比較した場合、全体では exp1 が 13.0 %、exp3 が 11.9 % となり、stetho の音声を聞いた方が 1 割ほど認識率が高い。ただし、この場合 stetho の音声は確実に耳に入っているにも関わらず、それを分解してボタンを押すという操作が追い付かないのに対して、tcpdump の出力は表示が速過ぎて知覚できていないという差異がある。

次に、exp1 と exp3 のボタンごとのロスト率に注目する。exp1 では SSH のロスト率が高いが、これは字面が http に類似しているため読み取りにくいと考えられる。逆に表示形式が異なる ICMP のロスト率は低い。exp3 については ICMP のロスト率が高い。これは、ICMP に割り当てられている音が鐘の単音であり、他の音と比較して目立ちにくく聞き取りにくいことが原因であると考えられる。

まとめると、stetho の音声はパケットの発生にすばやく反応できるが、多数のパ

ケットが同時に発生した場合に目立ちにくい音を聞き逃しやすい。通常の tcpdump と比較すると反応速度、ロスト率ともに優れている。しかし目的のケットに色をつけた tcpdump の出力では、秒間 20 パケット程度の量ではほぼケットを見落とさないのに対し、stetho では 1 割程度のケットを聞き落していることになる。

しかし、色付き tcpdump のロスト率の低さはバックトラックという、いわば巻き戻しに相当することを行なっているためである。言い替えれば stetho に巻き戻しや早送りといったインターフェースを用意することが、ロスト率の改善などに繋がる可能性があるとも言える。

5.6 今後の課題

stetho の今後の課題について、四点を挙げる。

最初は音声の割り当て手法の検討である。音声割り当ては大別して連続的に発生するケットと、断続的に発生するケットとに分けられる。前者のケットに対しては、弦楽器などの減衰の長い音や、アルペジオの連続などが適している。後者のケットに対しては、ベルなどの音が適している。また、全体として不協和音にならないように音を割り当てなければならない。stetho での音の割り当てを行なう際には、観測地点のトラフィック傾向や、観測する内容などの条件を加味する必要がある。tcpdump の出力結果と観測条件から適度に適切な設定ファイルを自動生成する機構を作成し、この作業を支援することを考えている。

次に、表現力の向上である。現在の stetho では、時系列での変化もしくは累積などを表現できない。トラフィック量の変化については擬似的に表現可能にしているが、例えば「あるケットが 1 時間内に流れた合計が一定量を超過したら音を発生させる」などの表現ができない。設定ファイル中で「演算」の記述もできないため、トラフィック情報になんらかの演算処理を施した結果に応じた発音もできない。この改良については、設定ファイルの表現力の向上を図る方針と、外部モジュールとして実装する方法の二通りが考えられる。

つぎが自動作曲システムという方向性の検討である。いわゆる自動作曲システムの中には、作曲理論に基づいてコード進行や旋律の遷移の情報を知識ベースとして持ち、乱数などのイベント列に応じてコードや旋律を自動生成するものがある。このようなシステムの入力情報として、ネットワークトラフィックを用いる試みを検討している。これは、先に挙げた stetho の目標のうちの、「音楽として心地よいものである」ことを更に進めたものである。

最後に stetho の有効性の更なる検証が挙げられる。仮に stetho の機能を tcpdump と同等のものであると限定したとしても、tcpdump から得られる情報をもとに何を抽出してどのような判断を下しているかという管理のシナリオを分析し、そのシナリオに基づいた判断を stetho を用いてできるか否かが評価の要となるであろう。

第6章 本研究がもたらした効果と ISNA 管理モデルの妥当性

6.1 要素技術が現在の技術に与えた効果

6.1.1 統一環境が与えた効果

今後増加するであろう小規模なネットワークの管理においては機器構成の管理が1つの重要な課題であり、適切な構成管理を行なうためにはOSやアプリケーションを含めた統一環境の構築が解決方法となり得る。PICKLESは当初の公衆端末という方向性から、管理が容易なサーバ用途に方向が遷移しているが、そこで提案した管理手法は、ISNAの重要性が増している現在でもなお有効である。

PICKLES SYSTEMをFreeBSDに移植したFreePICKLESは、通信総合研究所非常時通信グループが開発したDDoSシミュレータ[70]に用いられている。これは分散DoS攻撃をシミュレートするために100台の攻撃用計算機を用いるもので、この100台の計算機にFreePICKLESが用いられている。すべての計算機は前面から交換可能な2台のSCAドライブを搭載しており、それぞれがシステムディスクとユーザディスクになっている。このため、OSのアップグレードなどは前面からディスクモジュールを交換することで行なえる。また、システムディスクの内容だけをネットワーク経由で更新する機構も用意されている。FreeBSDではソフトウェアRAIDによるミラーリングが利用できるため、システムのアップグレードの際にミラーリングを停止し、片側のディスクの内容だけ更新して同期をとる方法や、単純に二つのパーティションを用意して交互に更新をかけていくことで可用性を維持する方法などが提案されているが[8]、どちらの場合でも、PICKLESとPOPSで提案したシステムディスクに相当する部分を切り分ける手法を導入することで、更に安全な更新が実現できる。

論文中で述べた統一環境を構築するための2つの技術的解決方法は、他のOS、例えば他のBSDやLinuxなどへも応用可能であると同時に、他のクラスタリング技術などと共存させることで更に管理作業を円滑なものにする。

この設計の妥当性については、PICKLES SYSTEMの運用実績からその有用性を示した。FreeBSDの世界でこのようなディストリビューショナルなアプローチがどの程度認められるか否かについて、ダウンロード件数を参考にすることにする。2002年3月にFreeBSD-Users-JPメーリングリスト(2003年7月時点の参加者数は約5400名)にPOPSをアナウンスしたところ、その後の4週間でダウンロード件数は約500件であった。2003年5月に公開した最新のFreeBSD 4.8-RELEASE対応のものは、公開後10週間で110件のダウンロードがあった。後者については、いわば

固定ユーザとして利用してもらえているものと考えられる。また、2003年6月に開催された「オープンソースのつどい2003 in 名大」にて、30枚のPOPSのCD-ROMが配布されたという報告も受けている。

今後は、手作業で行なっている部分をいかに省力化するなども含めた体制作りも重要になるであろう。

本文中ではPICKLESとPOPSの技術に対して、「Linuxでいうところのディストリビューションのようなもの」という表現を用いたため、Linuxの各種ディストリビューションと比較しての優劣を議論すべきだという指摘があるかもしれない。これの対しての回答は「違いを考察する価値はあるが、優劣を判断する必要はない」である。

BSD/OSのPICKLESについては、開発時期がLinuxが普及していく時期と重なっていることをまず念頭におく必要がある。1995年の開発当初のLinuxの安定度、性能、PCカードのサポート状況などから、当時はベースのOSとしてLinuxを選ぶという選択肢は考えられなかった。その結果、Linuxのディストリビューションの開発が進み、世の中の認知度が上がっていくのと並行して、BSD/OSベースでのPICKLES SYSTEMの開発を進めていくことになった。PICKLESは当初公衆端末としての用途を実現するためのIDカードを用いた認証技術などが組み込まれており、また独自のインストーラやデスクトップ環境などを提供していることから、仮にLinuxをベースにしたとしても結果的には独自のディストリビューションを開発することと同義になったであろう。そして、その有効性については、既に述べた通りである。

POPSを実運用に用いるためには、システムインテグレーション作業を加えて環境を構築する必要がある。POPSは、FreeBSDの世界で既に存在している枠組の中で、最小の労力で共通環境を作り、それを皆で共有しようという方針の元に開発した。PICKLESで包含するシステムインテグレーションのポリシーはPOPSからは分離されている。

さて、それでは上記の内容をすべて含めた上で、Linuxの世界に優れたディストリビューションが存在するのにBSD用に似たものを作る意義があるのかという問いがあるかもしれない。これの答えは「意義がある」である。BSDの世界で問題が存在し、それを解決する手段を作り上げたというのであれば、仮にそれと類似でしかも優れたものがLinuxの世界に存在したとしても、BSDの世界の問題を解決し一歩前進させたという点で、技術的には、特に運用技術という点では価値のあるものである。その価値が認められたからこそ、このテーマが論文誌に採録されたものと言える。したがって、PICKLESとPOPSについてLinuxのディストリビューションとの比較をすることは、PICKLESとPOPSをより良いものにするための考察の過程では必要なことではあるが、優劣を付けて意義の有無を議論することは、それこそ意義のないことである。

6.1.2 IPv6 対応 ISDN ルータが与えた効果

SOHO 向け ISDN ルータの IPv6 スタックの開発については、ISDN 自体は減少傾向にあり現在では ADSL や FTTH などが主な需要になっているが、著者らが開発したプロトコルスタックは ADSL 用のルータにも利用できる。ISDN 自体も全体数は減少しているが、国内でもいまだに ISDN しか利用できない地域は存在しており、逆に ISDN 対応のルータを他社が製造しなくなっている現在著者の成果が製品化されたヤマハ社の ISDN ルータの存在意義は相対的にむしろ高くなっていると言っても良い。

開発当時は ISDN の 128Kbps 程度の速度を念頭に置いていたが、通信速度は CPU の処理速度やメモリの帯域の向上で対応できる問題であり、実際共同研究を行なったヤマハからは 12Mbps ~ 数十 Mbps の速度に対応した ADSL 用ブロードバンドルータが製品化されている。同時に、現在 TACA などで議論されている組込み機器の性能が開発当初の WS-One の動作プラットフォームとほぼ同等であることから、現在であっても WS-One の開発により得られた技術は活用できる。

今後利用者が要求する様々なサービスの利用に対して、適切なネットワークの構築を行なうためには IPv6 を視野に入れざるをえなく、この研究は ISNA に対する IPv6 の導入を支援するものであるといえる。一般利用者向けの IPv6 接続サービスが既に開始されているが、価格帯にして 5 万円程度 of 家庭用ルータで IPv6 に対応している製品は 2003 年末の時点で国内ではヤマハ社の製品のみであり、共同研究という形で早期から IPv6 対応を進めてきたこそ実現できた成果である。

6.1.3 ネットワーク可聴化システムが与えた効果

ネットワークトラフィックの何を監視するのは、作業内容や監視者によって違って来る。stetho ではどの情報を音声に変換するかなどは、設定ファイルで任意に指定できるようにし、本論文中では特定のシナリオ、特定の条件に基づいた例についてのみ評価実験を行なった。stetho の使いかたとしては、実験で行なったもの以外にも色々考えられるし、究極的な目的としての管理者向け BGM の作成にはまだまだ遠く及ばない。現時点は、「管理作業への音声の導入」の最初の一步を行なったにすぎない。

また、stetho の開発を始めた当初は管理ツールとしては、パケットダンパや SNMP エージェント程度のイベントを取り扱うものしか存在しておらず、現在の IDS のように抽象的なイベントを検出するツールは少なかった。このため stetho はその当時の設計方針をひきずっているが、抽象的なイベントを音声化の対象とすることにより、よりの確なイベントだけを抽出して認識可能になると思われる。

監視対象についても、従来の組織内ネットワークの監視程度の規模から、広域ネットワークの複数拠点のイベントの監視が必要なケースが出てきている。例えば Telecom ISAC[34] は「通信業界に属する企業のサービス基盤において発生したインシデントに関する情報を収集・分析し、その結果を業界内で共有すること」を目的としており、その際には広域ネットワーク上の複数拠点での監視作業が必要になる。ここで

問題になるのは、その伝達方法である。監視情報はセキュリティ上重要な情報でもあり、安全に伝達しなければならない。もちろん暗号化して送信するというのも一つの方法だが、stethoの音声(またはMIDIイベント)のように抽象度の高い情報で送信する方法も隠匿するという点では有効かもしれない。

また論文中では評価の対象外としたが、芸術作品としての発展の可能性は未だにある。トラフィックに限らず、可聴化の研究の多くは聞きやすいかという点については懐疑的な結果を出しているようだが、同時に音作りに音楽家が参加していないことも多い。音楽的心地よさについては、音楽家の力抜きには実現できない。無論本文中で述べたように、前衛音楽が必ずしも「楽しい音」であるとは言えないのも事実であるが、それであっても音のまとまり方などは音楽家の手にかかる立派な作品と呼べるものが出来上がる。stethoはそのような音楽家が作品を作るための、下地のシステムとしても位置付けられるであろう。

6.2 ISNA の管理モデルと管理技術の考察

6.2.1 実現できた要件

本節では、先に挙げたISNAの管理モデルの要件のうち、実現できた点について論じる。

ソフトウェア構成の一元管理は、PICKLESによって実現した。またPICKLESでは保存する必要がある管理情報を明確に分離しており、これによりバックアップなどの作業が明快かつ確実に実行できる。例として、2章で挙げた管理コストの試算を行なう。2章の例では20台のFreeBSDのホストのセキュリティ対策を行なうのに必要なコストは1,100万円であった。PICKLESの場合は、更新作業を行なうのはマスターの一台だけになる。残りは19台には同じ内容を複製すれば良い。複製作業にHDDデュプリケータを用いれば転送の待ち時間を除くと操作に要するのは合計しても1時間程度である。ディスクの交換作業を含めても2時間程度であろう。しかも後半のディスクの複製と交換には高度な技術力を必要としないので、技術者0.5人分程度のコストと見積もって良い。したがって、合計で2人時間の作業になる。コストを見積もると年間では110万円となり、10%のコストに押えられた。

また、文中では述べていないが研究室の活動として、管理作業や構成情報に関するドキュメント作成の徹底とその内容の周知をする機構を提供したことにより、より確実な構成管理が実現したことを付記しておく。

障害発生時の迅速で低コストな復旧については、PICKLESによって代替機の立ち上げ作業を省力化できた。また、機材の基本構成を統一することで、使いまわしを可能にし、使われていない代替機を常に用意しておく必要がなくなった。特にPICKLES SYSTEM上に構築されたRebornシステムを用いることで、代替機能の切替えが容易かつ迅速に行なえるようになった。

例えば著者の研究室での事例を挙げると、1998年1月にメールサーバの機体にディスクのトラブルが発生した。この対処にあたったのが当時管理者権限を与えられたばかりの新米管理者であったが、Rebornシステムを用いることにより動作確認を

めて 30 分程度の作業で、他の機体で機能を代替させることができた。このうち 15 分は Reborn システムのマニュアルを読むのに要したとの報告を受けているので、実作業は 15 分程度であった。

安全かつ安心であるという点については、PICKLES は公式のセキュリティ監査を受けており、その PICKLES を使う範囲では全員が同じように安全であるという安心感が得られる。また、研究室ネットワークの対外接続点では、長期に渡ってファイアウォールを運用してきたことも付記しておく。

利用者の要求に応えるためには、End to End の通信が必要であり、IPv6 対応の SOHO 向けルータを開発したことにより、これを現実的に可能にした。

6.2.2 ISNA の管理技術の今後の課題

逆に ISNA の管理モデルの中で、本論文で述べた範囲で実現できていない項目を、今後の課題として挙げる。

構成管理については、移動する機器の位置情報なども含めた構成管理を行なう必要がある。今後 RF-ID タグ [55] などとの連系をとることが考えられる。また、通信基盤の構成管理が不足している。ここには VLAN の構成管理やスイッチ、ルータなどの構成管理が含まれるはずである。VLAN を導入することにより、物理的障害と論理的障害の症状との切り分けが難しくなることから、ISNA のような環境ではこういった機能は排除してシンプルなネットワーク構成を目指すことが本来は望ましいと著者は考える。しかし今後はこれらも視野にいれる可能性はある。

障害管理については、動作記録などに基づいた障害予測を行なう必要がある。既に述べたように、運用コストとのバランスを考えて行なう必要がある。

ISNA のセキュリティに関しては次節で述べるが、本論文で述べた範囲ではセキュリティポリシーの策定と運用に関して言及しなかった。ISNA ではポリシーとその実装が極めて近い距離にあるため、実際に動いているシステムからポリシーを作成する機構か、ポリシーから実際に動作するシステムを作成する機構の構築が必要であろう。

性能管理については、二点挙げられる。一つは、動作記録やトラフィック記録などに基づいた適切な性能管理を、本論文の範囲では行なっていない点である。無論研究室での教育の一環としては、構成員にこらの適正な評価が出来るような教育を行なうなどの対応はしてきたが、定量的な評価方法の確立は行なっていない。もう一つは、利用者からの要求に応えるためにはインターネットから ISNA への資源のアクセス方法を提供する必要があるという点である。資源にアクセスする方法については、P2P アプリケーションなどでは IP アドレスとポート番号でない上位の識別子やサービス記述言語を用いる方法などが提案されている。上位の機構が必要なのか出来るだけ簡単な機構で十分なのかについては更に議論する必要がある。

図 2.5 をもとに不足している機能を論ずると、まず通信基盤の障害管理が挙げられる。これは、スイッチやケーブルなどの通信機能の障害の検出とその対応である。

情報共有とアプリケーションの機密管理については、前述のセキュリティポリシーの策定や管理システムの充実などが課題として挙げられる。また性能管理について

は、前述の内容が挙げられる。

残るは障害管理における情報共有、すなわち BML の項目である。これには例えばトラブルチケットシステムの構築などが挙げられるであろう。

6.2.3 ISNA のセキュリティ

「IPv6 の時代には、ファイアウォールではなく End To End でのセキュリティの確保を行なうべきである」という意見がある。確かに IPv6 の導入と、末端のネットワークにも高速な回線が普及することによって、アドレス空間も通信速度もフラットなものになり、現在では制限されることの多いホスト間の直接の通信が当り前になる。加えて、IPSec を利用することで、その通信路を安全なものにすることも可能であろう。

しかし現時点では IPSec は信頼関係がある二つのノード間の通信路を安全にするための目的に使われているに留まっているし、仮に通信路が安全だったとしてもその通信内容が安全であるとは限らない。IPSec だけではなく、もっと上位の層でホストを護る概念として (ファイアウォールに対する言葉として) ファイアスーツ (防火服) がある。ネットワーク単位で護られていることに甘んじるのではなく、ホスト単位で防御するべきであるという考えである。これは最早当り前のことであり、個々のホストで必要のないサービスは無効にするなどの対応は当然行なっている。

しかし、組織としてのセキュリティポリシーが確立しつつある昨今、それを組織として実装し運用する必要もあり、それを束ねる 1 つの要素がファイアウォールである。IPv6 の時代になっても、組織のセキュリティポリシーを実装する手段の一つとして、ファイアウォールは必要である。

ファイアウォールとファイアスーツは併用し共存しなければならない。ホスト同士の対等な通信モデルでは、すべてのホストがサーバにもクライアントにもなりうる。実際のセキュリティホールとは不必要なサービスに対するアクセスではなく、必要なサービスに対して不正なアクセスが可能な場合に発生するため、すべてのアプリケーションを安全なものに保つ必要がある。すなわち、安全なシステムやソフトウェアの構築技術と、運用技術の両方が必要になる。

安全なソフトウェアの開発については、バッファオーバーフローなどを発生させないようなライブラリ、コンパイラ、ランタイム環境に関する研究 [51] などが行なわれているが、まだ一般的に用いられているとは言えない。運用技術については、著者は本論文で「それなりの強度のシステムを容易に導入できるようにする」目的で PICKLES と POPS を開発した。これは底上げという目的では適切な解法だが、副作用も起こしている。一つは PICKLES SYSTEM の通常の運用手順に従った場合に、セキュリティホールに迅速に対応できないというものである。通常の手順であれば、セキュリティホール対策をした新しいリリースを作成し、それにシステム全体を更新するという手順をとる。このリリース作業のコストはそれなりにかかるため、セキュリティ対策が後手に回りがちになる。変更部分だけ差分という形でリリースすれば良いのだが、バージョン管理を厳密にしなければならないという点と、差分作成のコストも経験上それなりに大きなものになる点が問題であり、BSD/OS

版の PICKLES SYSTEM ではそこまでの体制は達成できなかった。もう一つの問題は多様性が確保できないという点である。仮に世の中のすべての OS が単一であったとしたら、その OS のセキュリティホールを狙った攻撃がまたたく間に世界中を席捲することになる。複数の OS が混在しているからこそ、攻撃が難しくなる。すべての OS を PICKLES SYSTEM にすることで「それなりの強度のセキュリティ」は確保できるが、PICKLES SYSTEM にセキュリティホールが発見された場合はすべての OS に脆弱性が存在することになるというジレンマを抱えている。これについては、著者はあくまでシステム構築の手法として PICKLES SYSTEM 的コンセプトを提示し、その実装として BSD/OS 版 PICKLES SYSTEM を開発したという立場をとる。同じコンセプトを様々な OS の上で展開することは可能であり、この点は管理者なりベンダーなりの実装に期待する。

IPv6 の世界では、ネットワークに接続される機器についても様々なものが想定されるし、そもそも今後も現在のパーソナルコンピュータの延長にある機器を一般利用者に利用させる必要があるのか否かも含めて議論する必要があるだろう。

6.2.4 ISNA の管理モデルの妥当性

最後に、本論文で提案した ISNA の管理モデルの妥当性について述べる。管理項目については、力を入れる項目とそうでない項目という区別をしたが、それぞれの力点の配分が妥当であったか否かを議論する。以下の議論での具体的な事例としては、著者が所属していた研究室ネットワークの構築と運用経験 [89] を主に挙げる。

力を入れた項目は以下である。それぞれについて、達成できたか否か、達成できていない内容はなにか、達成したものはどの程度の利益があったのかについて述べていく。

構成管理 重点項目である構成管理については、PICKLES SYSTEM を用いた構成管理を実現した。ソフトウェア構成を統一することにより、管理すべき対象を集約させた。PICKLES SYSTEM のマスターの管理は従来通りの労力が必要であるが、その他のホストの構成情報については個別に管理する必要がない。PICKLES SYSTEM では管理対象を集約し、それを容易に分配する機構を提供した。この分配作業がすなわちシステムの導入更新作業であり、PICKLES ではネットワークに依存せずにこれらの作業が行なえる。このため、ノート PC や、仕事を離れた自宅でも同じ環境を容易に導入できる。PICKLES SYSTEM の保守とリリースエンジニアリングのコストは必要だが、このコストは分散できる。全体としてみたときの「対労力効果」は高いと言え、構成管理に注力した効果があったと言える。

障害管理 PICKLES と Reborn を用いた障害管理の基本コンセプトは「障害の発生を防ぐのではなく、障害が起きたら迅速に低コストで回復する」である。すなわち、MTBF(平均故障間隔時間)を延ばそうとしない代わりに、MTTR(平均故障復帰時間)を短くし、結果的に可用性を上げようというものである。故障

からの復旧作業が、新米管理者であっても短時間で行なえるようになったことは既に述べた。

先にコストを押えた障害管理が必要であることを述べた。障害予測と予防については、研究室ネットワークの運用などではコスト対利益の点から、導入しなかった。規模の大きなネットワークでは、二重系にするなどの方法を用いて、可用性を限界まで高めると同時に、障害予防も行なう必要がある。研究室ネットワーク程度の規模では、上記のポリシーの元で実装した PICKLES SYSTEM による方式が有効に機能したと言える。

機密管理 著者が所属していた研究室ネットワークでは、1997年の時点で学科のネットワークとの間にファイアウォールを設置し、独立したネットワークの運用を開始した。当時、学内の他研究室ではファイアウォールを導入する例は少なく、学内でも最も早い時期にファイアウォールを導入した研究室であった。その後、学内では数度に渡って外部からのネットワーク攻撃事件が起こり、外部の攻撃者に不正侵入されるという事件が発生した。これらの多くは、ファイアウォールを設置していなかったり、利用しているサーバアプリケーションを古いままにしておいたりといった、セキュリティ意識の低い組織が被害にあった事例である。著者らは、早くから安全性に配慮してネットワークを構築しており、これらの攻撃を回避することができた。

機密管理については、重きを置いた成果が十分に発揮されたと同時に、PCベースの低価格なファイアウォールの実装であっても、研究室ネットワークの規模であれば、機能的に十分であったと実証できた。

逆に力の入れ方を押えた項目は以下である。それぞれについて、生じたメリットとデメリットについて述べる。

性能管理 性能管理については、以下の三点について議論する。

1. 「適度な性能」の成否。
2. 性能解析の必要性。
3. 利用者が求めるサービスの提供。

1. は、ネットワークインフラと、機器の性能という二つの点を考える。研究室ネットワークの運用については、対外接続は大学などの上位組織の LAN に接続して運用したため、性能上制約を感じることもなく、同時にそれに対して評価を行なうこともしなかった。機器の性能については、その時その時で「適度な性能」の PC ベースのサーバおよびクライアントの運用を行なった。WWWサーバとメールサーバについては、約3年程度同じ計算機を用いたが、実用上性能に不具合は感じられなかった。最高性能を求めず、サービスが機能していることが重要であり、この目標で十分であったと言える。

このため研究室ネットワークの運用においては、2. の性能解析は皆無であったと言って良い。PC ルータが 100BASE Ethernet で用いるに十分であるかとい

う評価や、クライアント PC などのベンチマーク検査などを行なうことにより、おおまかな性能検査は行なったが、統合的な評価は行なっていない。むしろ構成員各自の技術者としてのセンスに任せた部分がある。それでも不具合が生じなかったのは、構成員全員が技術者であり適切な判断ができるという前提があったため（現実にはそのための教育に要したコストは大きなものであったが）あり、限定的な状況であったとは言える。

ISNA 一般論に移した場合、性能評価は難しい。なぜなら、特に民生用の機器ではカタログスペックに出ないところでの性能や利便性に大きく差が出る場合があるからである。極めて概念的な例であるが、著者らが IPv6 スタックの開発に参与した YAMAHA 社のルータは、転送速度と価格の点では他社の製品よりも劣る。しかし、操作性や細かな設定が可能である点などの理由で「玄人好み」の製品という評価を得ているが、このような点はカタログスペックや一般向け書籍のレビューには現れないし、定量的に評価することも難しい。

しかし一方で、研究室ネットワークのサーバの対負荷性能が十分であるか否かといった定量的評価は可能であり、機材投資などの際にはこのような指標が重要になる。これらのことからすると、現時点では ISNA では性能評価は不十分でも問題ないが、今後は数値的な点と数値に現れない点の両面からの評価を行なう「性能アセスメント」のようなサービスが、ISNA の運用支援としては必要になるのではないかと予測できる。

3. については、論文中では、IPv6 を前提とした End to End の通信を実現できる環境を利用者が求めていると述べた。研究室ネットワークを運用していた当時を背景に議論するとすると、IPv6 を定常的な対外接続には利用していなかったことと、当時は P2P アプリケーションが存在しなかったことから、IPv4 での議論になる。その際、テレビ会議などの「普段は利用していない」「End to End の通信を必要とする」アプリケーションを用いる場合は、適宜ファイアウォールのフィルタ設定を変更する対応を取った。これは構成員のほぼ全員が必要な管理作業をできるという、研究室ネットワークならではの対応といえる。ファイアウォールの運用などのセキュリティモデルについては、今後の検討課題であろう。

総じて、性能評価に力を入れなくても研究室ネットワークの運用には障害は発生しなかった。しかしこれは構成員のほとんどが技術者であり管理者であるという ISNA の中でも一部の特例であると考えている。今後は、性能評価と改善の提案を行なう性能アセスメントなどを外部サービスとして導入する必要があるであろう。

課金管理 研究室ネットワークの運用において、課金管理は一切行なわなかった。同時に、課金管理が必要になる局面が発生することもなかった。

予測される事態としては、プリンタの不正利用などの消耗品の管理の問題などが挙げられるが、個人の責任の範囲で管理するのは、ISNA を対象とした場合は妥当な方針ではないかと考える。

ディスク容量などの計算機資源については、経験上、要求量と民生部品の適正価格はほぼ同じような変化を見せるため、適切な性能評価がなされて機器投資がなされれば、制限を設ける必要はそれほどないように考える。しかし一方で、末端の利用者のデータ量に関する感覚は、ソフトウェアのインタフェースが抽象的なもののみになるに従って麻痺している(例えば自分が編集している画像ファイルのファイルサイズをまったく試算および理解できない利用者は、著者が芸術大学で勤務していた際の経験上、多数存在する)ため、教育的配慮という点での制約や資源の管理は必要なのかもしれない。

以上をまとめると、技術力が乏しい管理者であっても、全体として見れば低い管理コストで、安全で安心なネットワークを構築できたことから、ISNA の管理モデルにおける力点の配分は、適切なものであったと言える。

6.3 今後の展開

今後の展開について、短期的課題と中期の展開のそれぞれについて述べる。短期の課題としては、以下が挙げられる。

- 名前付の問題
- どういうドメインを取るべきか
- DNS サーバの運用

中期の展開については様々な方向性が考えられるが、著者は以下を重要な課題として挙げる。

- 移動する ISNA のモデル
- 更に小規模な ISNA

以下ではそれぞれについて述べる。

まず名前空間の問題である。言い替えれば、家庭などの末端のネットワークに接続されている資源を、どのような名前で参照するべきか、その名前サービスをどのように運用するべきかという問題である。

これに対しては、CORBA の IDL や WEB サービスにおける WSDL[19] を用いて資源やサービスを記述し、上位のアプリケーションでサービス探索などを行なう方法などが提案されている。もっと単純に DNS を用いたサービス探索を行なう方法も提案されている [30]。後者のほうが機構としては単純であるが、以下の二つの問題がある。

一つは ISNA ではどのようなドメインを取得するべきかという点である。2000 年から汎用 jp ドメインが利用できるようになったため、国内でしか移動しないのであれば jp ドメインで十分であるし、海外へ移動するのであれば .org や .net などの下のドメインが適当であろう。ドメインは所属のプレゼンスでもあるので、主に組織

としてどういう立場をとるかというポリシーの部分が多い。DNS サーバへの負荷を考えると、国内だけに留まる組織が不必要に.orgなどのドメインを使うべきでないし、国内でも地域ドメインなどを用いた方が負荷が分散されるという利点があるが、回線速度や計算機能力などの時間が解決しそうな技術的制約を利用者に課すべきでない意見もまた正論である。私見としては、この類の問題については、利用していきながらバランスを取るしか解がないと考えるし、実際 DNS はそうやってなんとか世界規模の分散データベースを実現している。

次に DNS サーバをどのように運用するかである。自組織内でサーバを運用する方法もあるし、ホスティングサービスに依頼する方法もある。後者については、組織内 LAN を持っている企業であってもメールサーバと Web サーバは外部のホスティングサービスを利用するケースもあるので、DNS も同様な運用をする選択もありうるであろう。実際既に、商用または無料のサービスで、静的または動的な DNS のプライマリサーバを代行するサービスが登場している。しかし、これらは今のところ、本論文で述べたような柔軟な運用を行なうには機能が不足しているため、可能であれば自前のネームサーバを立ち上げて運用するべきである。

次に中期的展開について述べる。

今後の中期 (5 年程度) の展開としては、本論文で想定したよりも更に小さなネットワークが移動するモデルについて検討していきたい。

例えばウェアラブルコンピューティングなどが実用的になれば、身にまとった機材がアドホックネットワークを形成し、外部のネットワークを移動するという利用形態が求められるであろう。そのような状況下でのネットワークの安全性の確保や管理技術などが、将来課題になるだろうと考えている。

6.4 今後の社会に与える効果

ISNA では技術力を持たない利用者が自らのネットワークやホストを管理しなければならず、更にこれを低コストで行なえなければならない。この課題を考えていくと、そもそもコンピュータやネットワーク機器をどのように一般利用者に提示するかという問題がある。セキュリティ上問題がない単機能の機器のみを利用者に使わせればよく、現在のパーソナルコンピュータのような汎用的な装置は十分な技術力を持つ者のみが使うべきだという意見もあるだろう。少なくともパーソナルコンピュータという名称で当初思い描かれたダイナブック [54] などのデバイスや、TRON[20] の電脳住宅で提示される理想の世界は、ソフトウェアのバグや OS の更新に煩わされる現在の計算機の姿とは異なるものである。一般利用者に提示すべき計算機環境を、根本から見直す必要も将来は出てくると思われる。

以下ではそのような将来の像を予想しながら、その中において本研究がどのように位置付けられるかを考察する。本節の内容はあくまで主観的な未来予想図に基づくものであり数値的な根拠などはないが、一つのビジョンとしては荒唐無稽なものではないと考えている。

現在先進国内においても、コンピュータを使える者と使えない者の間のデジタル

デバイド¹が問題になっている。この傾向は残念ながら今後も改善されないどころか、更に広がる可能性がある。そもそも「パーソナルコンピュータ」という存在は、一般利用者の利便性の向上に寄与していないのではないかと著者は考える。一種の技術者の玩具として、試作段階のものが市場に流れてたものがパソコンの発端であり、製品としては完成されたものではない。仮に一般利用者向けとなりうる完成品であれば、利用者が自力でソフトウェアを導入したりカスタマイズする必要はないはずである。そういう作業は本来、技術力のある技術者が楽しみの範囲だけでやれば良いのではないだろうか。

デジタルデバイドが進むと、人々は大きく二種類に分かれる可能性がある。1) 技術力を持ち計算機を理解して使える人間と、2) 技術力を持たない人間である。しかし二分化したとしても、依然としてその中間の層である、3) コンピュータを理解しようとするモチベーションはあるものの技術者と呼べるほどの技術力をもたない人々は存在するだろう。

1) の層は自らの技術力で自らの欲求を満たせるので、何らかの技術的支援の類を与える必要はない。2) の層に対しては、汎用的なパソコンを無理に使わせる必要はないものと著者は考える。家庭用であればウェブブラウザと電子メール程度が出来るアプライアンス端末、事務用ならビジネスアプライアンス、その他専門機能に特化した単機能端末が利用できれば、ほとんどの要求は満足できる。

3) の層については、モチベーションを礎に1) の層に昇華することが望まれる。この層の人々をいかに救い上げるかが重要な課題である。この層については、パーソナルコンピュータの汎用性とあわせて潜在的な問題がある。パソコンは汎用的であるが故に、最初はインターネット端末程度の目的で購入した利用者であっても、アプリケーションを追加して新しい取り組みに発展的に挑戦できるという利点がある。しかしその反面、「あるアプリケーションを入れたら OS の動作が不安定になった」といった苦情が発生する事態も起こる。

ここで議論を飛躍させるが、日本の戦後民主主義の一つの骨子に「すべての人々は平等に可能性を持っている」というものがあつた。しかし著者はこれは誤りであると考え。才能を伸ばす機会を平等に与えられるべきであるが、与えられている才能は平等ではない。パソコンの利用者の裾野が広がると同時に、「誰でも簡単に

できる」といったことを特長にするソフトウェアが増加してきた。同時にコンピュータの本質を理解しないまま、その「簡単さ」の恩恵だけを受けようとする利用者も増加している。理想的には、アプリケーションを利用する際に、そのアプリケーション利用方法以外の知識は必要とされないはずである。ところが現状のパソコンは、OS の特性やハードウェアの特性(多くの場合、それらはブラックボックスになっている)を理解しないと快適に利用できないことが多い。

ここで必要になることは、3) の層を1) と2) の二つに分離することである。2) の層には、簡単さだけを安定して享受できるようなシステムを提供する必要がある。拡張性や発展性よりも、このような安定性のほうが重要視されるべきであると著者は考える。

¹ デジタルデバイドという言葉は利用機会の不均等を指す場合が多いが、ここでは技術力の不均等についてのみ言及する。

さてここで本節の題目に立ち帰り、上記の3つの層の人々に対して、本論文の技術がどのように貢献できるかを考えることにする。

最初に1)の層の人である。第3章で述べたように、熟練者であっても、導入作業のような定型的作業の負担を減らす要求はある。こういった層にPICKLES またはその後継の技術は訴求するであろう。

次に2)の層に対してである。第3章で述べたように、PICKLESの管理モデルは利用者と管理者を明確に分離するものである。すなわち、末端の利用者である2)の人々は、アプライアンス端末の管理に留意する必要はないし、その体制が実現できてこそ始めてアプライアンス端末の意義があるといえる。

3)の層に対しては、第3章で述べているように、参照となるような環境を提示することで、その育成を支援する必要がある。1980年代から1990年代前半にかけてのUNIXコミュニティのような大学内コミュニティでの技術継承だけでなく、ネットワーク経由での情報交換が増えている現状にあって、先人たる技術者は適切なリファレンスを提示しなければならない。

ここで上げた貢献の可能性と、著者の実際の興味の対象とは必ずしも一致しないため、これらを今後の展望に挙げることはあえてしない。前述のようにパーソナルコンピュータは、必ずしも利用者の利益に即した形で発展しているとは言いがたい。このような中で著者の開発した技術が有効に働く可能性はあると考える。

第7章 結論

より高速に、より広帯域で、より新しい技術でという方針の、基幹系を始めとした大規模ネットワークの研究には多くの研究者が取り組んでいる。しかし冒頭に述べたように、この数年で数として著しく増加しているのは、SOHO や家庭などの小規模なネットワークであり、今後はこういった末端のネットワークでサーバ機能を稼働させるような運用があたりまえに行なわれるようになる。にも関わらず、このような小規模なネットワークには専任の管理者がいない場合が多く、非専門家が本業の片手間や趣味の延長として管理している場合が多いのが現状である。

本論文ではこのようなネットワークを自立運用ネットワーク (ISNA) と定義し、ISNA の管理技術の確立と、それによる安全で安心な ISNA の普及を目的とした。

第1章では導入として研究の概要を述べ、本研究がもたらした効果を示した。

第2章では背景となるインターネットの現状を量の変化と質の変化の二つの側面から述べ、そこから ISNA の重要性を導いた。ISNA の要件から、TMN 管理モデルとその適用事例を参考にして ISNA に適した力点の配分を行なった ISNA の管理モデルを提案し、これを実現する技術的解決方法を示した。また、この実装である要素技術として、著者らが開発したものを列挙した。

第3章では、要素技術のうちの統一環境の構築について述べた。容易な構成管理を実現するための二つの技術的解決方法を示し、これを実現する PICKLES SYSTEM とその技術の FreeBSD 上での実装方法を述べた。これらの技術によって、システムの導入、更新、復旧作業の時間を短縮し作業手順を簡略化できたことを、運用経験から示した。

第4章では IPv6 対応 ISDN ルータの実装について述べた。組み込み機器という制限のある条件下で、IPv6 プロトコルスタックを実装した。乏しい計算機資源、開発期間などの制限のもとで、開発効率を重視した設計方針を採用した。この方針は性能的には不利であったが、開発当初の要求性能は十分に満たせることを性能評価実験から示した。また、今後の小型組み込み機器の IPv6 スタックに求められる性能も満たせることを外挿し示した。

第5章ではネットワークトラフィック可聴化システムの実装と評価について述べた。評価実験の結果から、ある限定された条件の下では、音声を用いたトラフィックの把握が、通常のパケットダンパーと同程度に有効であることを示した。また実験に基づいて、今後の展開の方針を導き出した。

最後に第6章にて、ISNA の管理技術に関する考察を行なった。まず個々の要素技術が現在に与えた効果を示した。次に ISNA 管理モデルのうち実現できた点の考察と、今後の課題を示した。また ISNA の事例の一つである著者らの研究室ネットワークなどの運用経験から、ISNA 管理モデルの妥当性を示した。

今後の課題としては、短期的には名前サービスの適切な運用技術の確立、中期的には移動する ISNA の運用技術の確立が挙げられる。後者については、携帯するすべての電子機器が小さな ISNA をアドホックに形成し、ネットワーク間を渡り歩くような、将来のユビキタスコンピューティングの世界において重要になるであろう。

ISNA では技術力を持たない利用者が自らのネットワークやホストを管理しなければならない、更にこれを低コストで行なえなければならない。第 6 章で述べたように、将来は一般利用者に提示すべき計算機環境を根本から見直す必要も出てくるかもしれない。その場合でも、一般利用者が行なうべき管理作業を最少にする技術は必要である。著者が提案した ISNA 管理モデルとそれを実現する管理技術によって、技術力のない管理者であっても容易かつ安全・安心なネットワークの構築と運用が可能になり、本論文の内容は現在と将来の ISNA の管理支援に有効であると言える。

謝辞

謝辞には常識的な形式というものがあり、本来ならそれに則って謝意を述べるべきであるのだろう。しかしいざ謝辞を書く段になると、本論文の執筆の過程にまつわる様々な感情が著者の胸を駆け巡り、冷静な心持ちで定型に沿った謝辞を書くことは難しい。

ここでは全ての感情を押し殺すと同時に、記述形式もあえて無機的なものにしたと思う。すなわち、「順不同の箇条書」である。どうかこれを失礼な行為であると受け取らないで頂きたい。感情と様式美とニヒリズムとに押し潰されないように心身のバランスを取った結果なのである。

以下の方々と事物に、心からの謝意を表する。

- 博士論文主査であり、論文執筆の過程で指導して下さった、数理・計算科学専攻の佐々政孝教授。
- 東工大時代の指導教官であり、その後も実質的な指導をしていただいた、独立行政法人通信総合研究所情報通信部門セキュアネットワークグループ 大野浩之グループリーダー。
- 博士論文審査委員の先生方。柴山悦哉教授 (数理・計算科学専攻)、松岡聡教授 (数理・計算科学専攻)、櫻井成一朗助教授 (計算工学専攻)、脇田建助教授 (数理・計算科学専攻)。
- 旧 数理・計算科学専攻 大野研究室の歴代のメンバー。
- 独立行政法人通信総合研究所情報通信部門 旧非常時通信グループの皆様。
- 独立行政法人 通信総合研究所 情報通信部門 主任研究員の中川晋一様。
- 研究室で同期だった、小野木渡くん、中嶋良彰くん、成田哲也くん。
- 無線研究部 OB メーリングリストの面々。
- 友人の草刈千晶さん。
- 東工大時代以降も継続して様々な助言を下さった、飯島昭博さん、清水亮博さん、新美誠さん。
- WIDE プロジェクトの皆様。

- FreePICKLES 開発者メーリングリストの皆様。特に、菊池豊助教授を始めとする高知工科大学情報システム工学科 菊池研究室の皆様。
- WS-One 開発プロジェクトの共同研究者である、ヤマハ株式会社の関係者の皆様。
- 大野研究室の多くの共同研究プロジェクトに関わった皆様。
- 大野グループリーダーの夫人である、大野宏美様。
- 入社直後でありながら、休職して論文を執筆することを許して下さった、田口社長を始めとする株式会社創夢の皆様。
- 数々の有形無形の支援をしてくれた、両親と妹夫婦。
- 滝本竜彦メーリングリストの皆様。
- 幾多の小説や書籍たち。特に滝本竜彦氏の小説と斎藤環氏の著作は、私の精神構造に大きな影響を与えた。
- 詳しくは書けないが、さまざまな薬たち。

第6章で触れたように、著者は現在のパーソナルコンピュータのありかたに疑問を感じている。著者が見てきた世界は、技術者の良心などあざ笑うかのように好ましくない方向に向かうことが多々あった。その一方で、そんな世の中を良い方向に向かわせるように貢献したいと考えることもまた、技術者の良心である。

しかし著者は半ば迷いを捨て、自らの閉鎖的な幸福と技術的興味の追求のみに生きようとしている。なぜなら「好ましい世の中」が何かと考えた時に、自分が求めるのは最終的に技術者が幸福になれる世界であると認識したからである。大局的な技術者の理想郷が手に入らないのであれば、せめて自らの殻の中の小さな幸福を自らの技術で手に入れようとするくらいは許されるのではなかろうか。

だがこれは自閉的エゴイズムであり、決して褒められたものではない。

然れば、八方塞がりである。

ならば全ての価値観を無にしよう。

世界の全てに全身全霊の愛情をこめ、同時にそのことの無意味さをも胸に抱く。

——曰く、世の中はすべからく、生も死も等しく無価値である。

——ならば世界なんか、消えてなくなってしまう方がいい。

参考文献

- [1] Analog: Www logfile analysis. <http://www.analog.cx/>.
- [2] Dolly+ home page. <http://corvus.kek.jp/manabe/pcf/dolly/index.htm>.
- [3] EtherApe, a graphical network monitor. <http://etherape.sourceforge.net/>.
- [4] The ethereal network analyzer. <http://www.ethereal.com/>.
- [5] Fractal tune smithy home page. <http://www.tunesmithy.connectfree.co.uk/>.
- [6] Hearing the mandelbrot set. <http://members.aol.com/dshp3/mandelmaps.html>.
- [7] ITU Telecommunication Standardization Sector. <http://www.itu.int/ITU-T/>.
- [8] Mails in FreeBSD-users-jp[72132, 72133, 72136, 72140, 72142,72156 など]. <http://home.jp.FreeBSD.ORG/cgi-bin/showmail/FreeBSD-users-jp/72132>, etc.
- [9] The multi router traffic grapher (mrtg). <http://www.mrtg.org/>.
- [10] NetPC Specification. <http://web.jf.intel.com/design/netpc/netovr.htm>.
- [11] Network computer inc. <http://www.nc.com/>.
- [12] Oss: 4front technologies. <http://www.opensound.com/>.
- [13] The packet capture library. <http://www-nrg.ee.lbl.gov/nrg.html>.
- [14] RedHat linux kickstart information. <http://wwwcache.ja.net/dev/kickstart/>.
- [15] Sensorium. <http://www.sensorium.org>.
- [16] Timidity++: Midi to wave converter / player. <http://www.goice.co.jp/member/mo/timidity/>.
- [17] Ttcp. <ftp://ftp.sgi.com/sgi/src/ttcp/>.
- [18] Ttt: Tele traffic tapper (version 1.6). <http://www.csl.sony.co.jp/person/kjc/software.html#ttt>.

- [19] Web services description language (wsdl) version 1.2.
<http://www.w3c.org/TR/wsdl12/>.
- [20] www.tron.org. <http://www.tron.org>.
- [21] は音楽. <http://web.kyoto-inet.or.jp/people/haselic/pi/pai.htm>.
- [22] 日本の adsl の歴史. <http://www9.plala.or.jp/demu-1/page025.html>.
- [23] 情報通信ネットワークマネジメント — スタンドアートの活用法 —. 電気通信協会, 1999.
- [24] Peep(The Network Auralizer): Monitoring Your Network With Sound. In *2000 LISA XIV*, Dec. 2000. <http://peep.sourceforge.net/docs/lisa2000.pdf>.
- [25] 敷田幹文 and 井口寧, 丹康雄, 松澤照男. 大規模分散システムの効率的管理法. 第一回インターネットテクノロジーワークショップ (WIT'98) 論文集. ソフトウェア科学会, Aug 1998.
- [26] Donnie Barnes. RPM How/To. <http://www.redhat.com/support/wpapers/rpm-howto.pdf>.
- [27] M. Barra, T. Cillo, and A. Santis. “webmelody: Sonification of web servers”. In *WWW9*, May 2000.
- [28] Stephen A. Brewster, Peter C. Wright, and Alistair D.N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. In *INTERCHI'93*, pp. 222–227, April 1993.
- [29] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, December 1998. RFC 2460.
- [30] Discovering Named Instances of Abstract Services using DNS. Internet Draft. <http://files.dns-sd.org/draft-cheshire-dnsextnias.txt>.
- [31] SNMPv2 Working Group, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*, January 1996. RFC 1902.
- [32] Hideki Honma, Akio Noda, and Hiroyuki Ohno. An alternative user interface for the IAA system: Using OCR/OMR as on-ramp gateway for the Internet. In *Proceedings of IEICE Internet Workshop '98*, March 1998.
- [33] ITU-T. Overview of TMN Recommendations. ITU-T M.3000. <http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-M.3000>.
- [34] Telecom-ISAC JAPAN. Telecom-isac japan.

- [35] Masahiko KIMOTO and Hiroyuki OHNO. Design and Implementation of Stetho — Network Sonification System. In *Proceeding of ICMC2002*, pp. 273–279, Sep. 2002.
- [36] Masahiko KIMOTO, Satoshi SHIRASUNA, Hiroki SUENAGA, Toshihiro KIMURA, and Hiroyuki OHNO. Construction of small IPv6 network using INS router. In *Proc. of INET 2000*. ISOC, Jul. 2000.
- [37] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Willson. On the Self-Similar nature of Ethernet Traffic (Extended Version). In *IEEE/ACM Transactions on Networking*, 2(1):1-15, 1994.
- [38] Luke Mewburn. The Design and Implementation of the NetBSD rc.d system. In *Usenix Annual Technical Conference*. USENIX, June 2001.
- [39] Sun Microsystems. Sun Ray Software 1.3 Advanced Administrator’s Guide. <http://docs.sun.com/db/doc/806-7713-10>.
- [40] Philip M.Papadopoulos, Mason J.Kats, and Greg Bruno. NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters. In *IEEE Cluster 2001*, pp. 258–267, 2001.
- [41] Monique Noirhomme-Fraiture, Olivier Scholler, Christophe Demoulin, and Simeon Simoff. Sonification of time dependent data. In *PKDDD2002*, August 2002.
- [42] Open Cluster Group. OSCAR: A packaged cluster software stack for high performance computing. <http://oscar.sorcegorge.net/>.
- [43] J. Postel. *Internet Protocol*, September 1981. RFC 791.
- [44] R. Rivest. *The MD5 Message-Digest Algorithm*, April 1992. RFC 1321.
- [45] Carla Scaletti and Alan B.Craig. Using sound to extract meaning from complex data. *Extracting Meaning from Complex Data:Processing, Display, InteractionII*, 1991.
- [46] Gustavo Noronha Silva. APT HOWTO. <http://www.debian.org/doc/manuals/apt-howto/index.en.html>.
- [47] Rob Snevely. JumpStartTM: NIS と sysidcfg, Oct. 1999. <http://www.sun.co.jp/blueprints/1099/jumpstart.pdf>.
- [48] Nobuhiko TADA, Yukimitsu IZAWA, Masahiko KIMOTO, Taro MARUYAMA, Hiroyuki OHNO, and Masaya NAKAYAMA. IAA System (“I Am Alive”): The Experiences of the Internet Disaster Drills. In *Proceedings of INET 2000*. ISOC, July 2000.

- [49] Mark Weiser. The Computer for the Twenty-First Century. In *Scientific American*, September 1991.
- [50] Mark Weiser. Ubiquitous computing. *IEEE Computer*, October 1993.
- [51] こがよういちろう, 江藤博明, 佐藤広生, 小飼弾. セキュアプログラミングのすすめ. BSD MAGAZINE, No. 13, 株式会社アスキー, 2002.
- [52] ばるばら. 教科書には載らないニッポンのインターネットの歴史. <http://blogdex.tripod.co.jp/encyclopedia/>.
- [53] まつもとゆきひろ. オブジェクト指向スクリプト言語 ruby. <http://www.ruby-lang.org/>.
- [54] アラン・C・ケイ. アラン・ケイ. アスキー, 1992.
- [55] オート ID センター. Auto-id center - japan. http://www.autoidcenter.org/japanese_main.asp.
- [56] 伊藤純一郎, 横手靖彦, 所真理雄. Apertos オペレーティングシステムによる IPv6 ルータの構築. コンピュータソフトウェア. 日本ソフトウェア科学会, Jan. 1997.
- [57] 井澤志充, 木本雅彦, 多田信彦, 三輪信介, 大野浩之, 篠田陽一. Iaa システムの現状とその課題. コンピュータソフトウェア, 第 18 巻, pp. 27–32. ソフトウェア科学会, Nov. 2001.
- [58] 井澤志充, 木本雅彦, 多田信彦, 大野浩之, 篠田陽一. Iaa システムの現状とその課題. インターネットコンファレンス 2000, Nov. 2000.
- [59] 河本敏志, 竹村治雄, 片山喜章, 萩原兼一, 横矢直和. 音声メニュー同時提示方式の提案と評価. 第 59 巻, pp. 43–50. 情報処理学会ヒューマンインタフェース研究会, March 1995.
- [60] 村井純監修. インターネットの歴史: インターネットの歴史年表. <http://terakoya.yomiuri.co.jp/shiro/rekishi/nenpyo/90s.html>.
- [61] 高宮安仁, 真鍋篤, 松岡聡. Lucie: 大規模クラスタに適した高速セットアップ・管理ツール. 先進的計算基盤システムシンポジウム SACSIS2003 論文集, pp. 365–372, May 2003.
- [62] 児玉憲造, 廣安知之, 三木光範, 谷村勇輔 and 上川純一. Dcast: 大規模クラスタにおける簡易セットアップ・管理ツールの提案. 同志社大学理工学研究報告, 第 43 巻, pp. 187–196, 2002.
- [63] 上田仁, 木本雅彦, 大野浩之. 小規模組織に適した標準ネットワークとその管理支援系の構築. 情報処理学会分散システム運用技術研究会, Sep. 1998.

- [64] 上田仁. 多様な利用形態に対応した小規模組織用ネットワークに関する研究. Master's thesis, 東京工業大学 情報理工学科 数理・計算科学専攻, 1998.
- [65] 成田哲也, 大野浩之. stetho:ネットワークトラフィック可聴化システム. In *DSM-951150*. 社団法人 情報処理学会マルチメディア通信と分散処理研究会, Nov. 1995.
- [66] 成田哲也, 大野浩之. ネットワーク可聴化システムの利用例. 第 52 回 (平成 8 年後期) 全国大会 公演論文集 (3), p. 473. 社団法人情報処理学会, 1996.
- [67] 篠原正紀, 藤崎智宏, 浜田雅樹. サービス管理に着目したネットワーク管理モデル. 研究報告「分散システム運用技術」, No. 3. (社) 情報処理学会, 1996.
- [68] 岡部宣夫, 石山政浩, 井上 淳他. 非 PC 系デジタル機器への適用に向けた IPv6 最小要求仕様の検討. *情報処理*, Vol. 42, No. 9, pp. 920 – 925, 2001.
- [69] 総務省. 平成 15 年度版 情報通信白書. <http://www.johotsusintokei.soumu.go.jp/whitepaper/ja/h15/index.html>.
- [70] 大野浩之, 武智洋, 永島秀己. インターネットの脅威に対抗しうる脆弱性データベースと検証システムの構築. *DSM シンポジウム 2001 予稿集*, pp. 121–126. (社) 情報処理学会, Feb. 2001.
- [71] 関口智嗣, 水田 秀行他. 特集: グリッドコンピューティング. *情報処理*, Vol. 44, No. 6, pp. 574 – 621, 2003.
- [72] 中嶋一雄, 木村誠吾, 木本雅彦, 大野浩之. NMWシステムを用いた公衆情報端末の管理. 情報処理学会第 57 回 (平成 10 年後期) 全国大会 講演番号 1G-07.
- [73] 小坂直敏. なぜ嫌われる音楽を創り続けるのか - 芸術音楽の創作姿勢とその普及 -. 研究報告「音楽情報科学」, No. 41. (社) 情報処理学会, 2001.
- [74] 小野木渡. NMW System によるネットワーク管理とその評価. Master's thesis, 東京工業大学 大学院 情報理工学研究科数理・計算科学専攻, 1996.
- [75] 小野木渡, 清水亮博, 大野浩之. ネットワークワームを利用したネットワーク管理手法. *コンピュータソフトウェア*, 第 16 巻. 日本ソフトウェア科学会, May 1999.
- [76] 二ノ宮寿之, 木本雅彦, 大野浩之. 公衆情報端末と超小型形態端末の連携によるアンケート調査システムの構築. 情報処理学会第 57 回 (平成 10 年後期) 全国大会, 講演番号 4H-03.
- [77] 日本 ヒューレット・パカード 株式会社. HP OpenView. <http://www.jpn.hp.com/openview/>.
- [78] 風間一洋, 佐藤孝治. 並列プログラムの聴覚化. In *WISS '94*, pp. 125 – 134. 日本ソフトウェア科学会, 1994.

- [79] 木本雅彦. PICKLES TERMINAL SPECIFICATION.
<http://www.ohnolab.org/researches/pickles/spec1996.html>.
- [80] 木本雅彦. POPS: Package Of the PackageS ~ 誰でも簡単 マイパッケージ集 ~ . FreeBSD PRESS, 毎日コミュニケーションズ, No. 9, pp. 96 – 99, 2002.
- [81] 木本雅彦, 井澤志充, 多田信彦, 丸山太郎, 大野浩之. 非常時情報流通システム (IAA システム) の現状と今後の展開. 情報処理学会 DSM シンポジウム, Feb. 2000.
- [82] 木本雅彦, 山内崇圭, 持田啓, 大野浩之. RFC2305 に準拠したシンプルモードインターネット FAX の設計と実装. 情報処理学会報告書 98-DPS-2. 情報処理学会マルチメディアと分散処理研究会, May 1999.
- [83] 木本雅彦, 新美誠, 大野浩之. 可搬型被災者安否情報登録検索サーバ群の設計と実装. 情報処理学会 DSM 研究会, Oct. 2001.
- [84] 木本雅彦, 川部勝也, 中嶋一雄, 持田啓, 大野浩之. 非常時情報流通におけるインターネットと公衆電話網の連携 — インターネット災害訓練の経験から —. 情報処理学会 全国大会 (講演番号 3X-02), Sep. 1999.
- [85] 木本雅彦, 大野浩之. 街角公衆情報端末計画 ~ PICKLES の概要 ~ . 第 52 回全国大会 講演番号 3Y-2. (社) 情報処理学会, Mar. 1996.
- [86] 木本雅彦, 大野浩之. 学内情報システム ~ citrus の概要 ~ . 情報処理学会第 52 回 (平成 8 年前期) 全国大会 大会論文集 (1), pp. 277–278, Mar 1996. 大会優秀賞受賞.
- [87] 木本雅彦, 大野浩之. 公衆情報端末の図書館における利用例. 第 53 回 (平成 8 年後期) 全国大会 大会論文集 (3), pp. 475–476. (社) 情報処理学会, Sep. 1996.
- [88] 木本雅彦, 大野浩之. 公衆情報端末計画 (PICKLES) におけるシステム設計と管理技法. 情報処理学会研究報告 (96-DSM-2), pp. 13–18, July 1996.
- [89] 木本雅彦, 大野浩之. 機動性に配慮した小規模ネットワークの構築経験 — (4) 運用および管理 —. 第 55 回全国大会 講演番号 4S-8. (社) 情報処理学会, Sep. 1997.
- [90] 木本雅彦, 大野浩之. 小規模な組織の運営を支える情報共有機構-(4) 情報登録・抽出インタフェース-. 情報処理学会第 55 回 (平成 9 年後期) 全国大会講演論文集 (4), pp. 343 – 344,, Sep. 1997.
- [91] 木本雅彦, 大野浩之. 自律型ネットワーク端末 (PICKLES) を用いたシステム運用技法. DSM シンポジウム'98 予稿集, pp. 93–99. (社) 情報処理学会, Feb. 1998.
- [92] 木本雅彦, 大野浩之. 管理情報可聴化システム (stetho) の拡張とその評価. 分散システム運用技術シンポジウム, pp. 115–120. 社団法人情報処理学会, Feb. 2001.

- [93] 木本雅彦, 大野浩之. POPS:FreeBSD における統一された利用者環境の構築. 情報処理学会研究報告 2002-DSM-25, No. 25, pp. 43–48. (社) 情報処理学会, June 2002.
- [94] 木本雅彦, 大野浩之. 自立運用ネットワークの管理を支援する統一環境の構築. 情報処理学会論文誌, Vol. 45, No. 1, pp. 24–32, Jan. 2004.
- [95] 木本雅彦, 末永浩樹, 野田明生, 白砂哲, 木村俊洋, 大野浩之. 小規模ネットワークの構築に適した IPv6 対応 ISDN ルータの実装と評価. 電子情報通信学会論文誌 D-I, Vol. J85-D-I, No. 4, pp. 371–379, April 2002.
- [96] 中嶋良彰. 管理情報を自動的に収集・分析するネットワーク管理支援系の設計と実装. Master's thesis, 東京工業大学 情報科学科 卒業論文, 1995.
- [97] 中嶋良彰. システム管理に必要な情報を自動的に収集・分析するための機構 (magP) に関する研究. Master's thesis, 東京工業大学 大学院 情報理工学研究科 数理・計算科学専攻, 1996.

付録A PICKLES SYSTEM管理の手引き

この文書はBSD/OS版 PICKLES SYSTEMの「管理の手引き」である。この文書は1999年の時点のものであり、一次情報は<http://www.ohnolab.org/research/pickles/admin-guide/>である。なお、原文のうち不十分な記述は削除して整理したものを収録している。ただし加筆修正は行っていない。

A.1 PICKLES SYSTEMの導入

A.1.1 インストールの概要

PICKLESのインストール方法は大きくわけて以下のものがあります。

- PICKLES配布者からシステムディスクとユーザディスクの配布を受ける方法
- PICKLES配布者からシステムディスクと未初期化のユーザディスクの配布を受ける方法
- CD-ROMからインストールする方法
- システムディスクの複製を作成する方法

基本的にPICKLES SYSTEMは利用者がインストール作業などを行う必要がない形態での配布を目標としています。したがって、設定済のディスクを配布する方法が最も推奨される導入方法です。

A.2 基本的な設定

A.2.1 configuserの使いかた

初期設定を行ったあとのユーザディスクはconfiguserプログラムで行います。`/usr/pickles/bin/configuser`を実行してください。以下では、ユーザディスクの設定手順について述べます。configuserを実行するとメニューが表示されます。

- 1. Install BSD/OS License

- 2. User Administration
- 3. Host Configuratio
- 4. Setting up X-WindowSystem
- 0. Exit 'configuser'

それぞれの項目について説明します。

0: configuser プログラムを終了します。

1: BSD/OS License の入力 (Install BSD/OS License Code) BSD/OS のライセンスコードを入力してください。分からない場合はここでは”none”を入力して先に進み、調べた後に/etc3/license というファイルにライセンスコードを書き込んでください。

2: 利用者管理 (User Administration) 利用者の追加/情報変更を行います。最初に追加する方法を聞かれます。単純に利用者を追加するのなら adduser を選択してください。細かく利用者情報を変更する場合は vipw を利用してください。cancel を入力すると前の画面に戻ります。

3: ホストの設定 (Host Configuration)

1. ホスト名を入力します。ドメイン名を持っている場合はドメイン名込みで入力してください。
2. ネットワークインタフェースを選択します。デスクトップ等で内蔵型の EthernetCard を一枚だけ利用している場合は”auto”を選択してください。PCMCIA EthernetCard を利用する場合などはインタフェース名を入力してください。
3. IP アドレス, サブネットマスクを入力します。
4. デフォルトルータ, ネームサーバの IP アドレスを入力します。
5. 1. から 4. までの内容を確認します。適切な設定が分からない場合は管理者に問い合わせてください。
6. 5. で yes と返答すると、その内容を/etc3/hostconfig, /var/qmail/control に書き出します。

4: X-WindowSystem の設定 (Setting up X-WindowSystem) PICKLES SYSTEM 2.00DR8 では Accelerated-X と XFree86 の 2 種類の X サーバを利用できます。Accelerated-X は version 4.2.1, XFree86 は version 3.3.4 がインストールされています。

最初にぷらっとホームから提供されている PICKLES 仕様計算機か否かを聞かれます。ぷらっとホームから提供されている PC の場合は、ここで yes と返答すると XFree86 の設定ファイルを適当なものを適用します。

上記で no と返答した場合、手動で X の設定を行います。XFree86 の設定手順については別途発行するドキュメントを参照してください。

A.2.2 電子メールの設定

configuser を使ってホストの設定を行った場合は、電子メールの設定も同時に完了しています。ただし、その時点ではメールサーバを経由せずにメールを送信する設定になっています。もし、すべてのメールがメールサーバを中継するようにしたい場合は以下の設定を行う必要があります。

(smtp サーバとして mail.domain.org を利用する場合) /var/qmail/control/smtproutes の内容を:mail.domain.org とする。

次に各ユーザのメール環境の設定を行います。imsetup を実行してください。いくつか質問されますが、すべて提示する内容で構わないので Eter キーを押して行ってください。

以上でメールの送信の環境が整いました。もし正常にメールが送られない場合は、ホームディレクトリの直下の.im というディレクトリにある Config というファイルを調べてください。Smtpservers=の値が localhost になっている必要があります。

また、POP 経由で電子メールを受け取る場合は POP サーバの設定を行う必要があります。PICKLES SYSTEM 2.00DR7 以降は IM を用いたメールの送受信が標準環境になっています。各ユーザのホームディレクトリの直下の.im というディレクトリにある Config というファイルを編集し、POP サーバを指定してください。PopAccount=の行で POP サーバを指定します。

以下は記述例です。ユーザ名を username, POP サーバの pop.domain.org とします。

通常の POP を用いる場合 PopAccount=/POP:username@pop.domain.org

APOP を用いる場合 PopAccount=/APOP:username@pop.domain.org

A.2.3 ネットワークの設定

/etc3/hostconfig

/etc3/hostconfig にはホスト名, IP アドレスなどが記述されます。各種サービスを立ち上げるか否かの指定も、このファイル中で記述します。

/etc3/hostconfig 中のネットワークの設定は大別して 2 種類あります。

```
hostname="ホスト名"  
defroute="デフォルトルートの IP アドレス"  
ipaddr="IP アドレス"  
netmask="ネットマスク"  
iface="使用するインターフェース"
```

/etc3/resolv.conf

参照するネームサーバの指定は/etc3/resolv.confで行う。書式は以下である。

domain ドメイン名

nameserver ネームサーバの IP アドレス

A.2.4 X-WindowSystem の設定

PICKLES SYSTEM 2.00DR7以降では、Accelerated-X と XFree86 の両方をサポートしています。ただし、現在は XFree86 のサポートに重点が置かれています。

XFree86 の設定方法

XFree86 の設定方法は別のドキュメント (XFree86 の設定方法) にまとめられています。

Accelerated-X の設定

この節では、Accelerated X の設定の仕方を説明します。Accelerated X の設定方法は二通りあります。

- Xsetup プログラムによる方法
- Xaccel.ini を編集する方法

通常は Xsetup プログラムを用いて設定します。Xsetup は root 権限で実行してください。

Xsetup コマンドを実行するとメニュー画面が出て来るので、各項目の設定を行います。設定する項目は下記の通りであります。印をつけた項目は、優先して設定すべき項目で、印をつけた項目は、通常の利用では変更する必要のない項目を表しています。

Graphics Board
Monitor
Colors
Resolutions
Desktop
Visual
Gamma Correction
EnergyStar/DPMS
Keyboard Layout
Mouse Type

Mouse Device
Mouse Buttons
RGB File
Font Path

各行は、「項目名: 現在の設定」となっています。矢印の上下キーで各項目は選択でき、Enter キー (もしくは Return キー) を押すと各項目を変更できます。

設定が全て終わったら、メニュー画面で Esc キーを押してください。いままで行った設定を保存するかどうか聞かれるので、「Exit with save」を選ぶ。次に設定を保存するファイル名を聞かれます。通常は Return キーを押して推奨値をそのまま用いてください。

ディスプレイマネージャの設定

`/etc/pxdm/*` 実体は、`/etc3/pxdm/*` である。Xaccess、Xresources、Xservers 等のファイルがあり、ホスト全体で共通の設定をする時に用いる。

A.2.5 WWW の設定

`/etc/proxies` に書かれた内容がすべてのユーザの環境変数に反映されます。このファイルの中では Web ブラウザを立ち上げた時に最初に表示するページの URL や、プロキシサーバの URL を設定します。

A.3 利用者管理

A.3.1 利用者の新規登録/削除

PICKLES システムでのユーザアカウントの登録は、基本的に BSD/OS と同様の手順で行なう。アカウント作成の主な方法として次の 2 つがある。configuser から利用者管理が行えるが、基本的に以下の 2 種類のプログラムを呼び出している。

- adduser コマンドによる方法
- vipw コマンドによる方法

adduser による方法

adduser コマンドを用いると、必要項目を順に入力していくことで新規にユーザのアカウントを作成できる。rmuser コマンドを用いてアカウントの削除を行なうことも可能である。adduser コマンドと rmuser コマンドは `/usr/sbin/` に置かれている。adduser を用いたアカウント作成は以下の手順で行なう。

1. ユーザのログイン名

2. パスワード (2 回入力)
3. 所属グループ
4. フルネーム
5. オフィス
6. オフィス電話番号
7. 自宅電話番号
8. ホームディレクトリ
9. シェル

このうち、1,2,3,8,9 は必ず登録する必要がある。これらの情報のうち、4,5,6,7,9 は `chfn` コマンドで各ユーザが変更が可能である。ユーザが自分自身のパスワードを変更する際には `passwd` コマンドを用いる。

vipw による方法

`vipw` コマンドを用いた方法では、パスワードファイルを直接編集する。編集に用いるエディタは環境変数 `EDITOR` で指定することができる。特に指定されていない場合は `vi` エディタによって作業を行なう。編集対象のファイルは `/etc/master.passwd` と同じ形式である。このファイルは一行につき一人のアカウント情報が記録されている。ユーザ情報には `:` で区切られた次のエントリがある。

Login : ログイン名
Password : 暗号化されたパスワード
Uid : ユーザ ID
Gid : グループ ID
Change :
Exipre :
Class :
Full Name : ユーザの本名
Location : オフィス
Office Phone : オフィス電話番号
Home Phone : 自宅電話番号
Home Directory : ホームディレクトリ
Shell : シェル

(ただし、Location, Office Phone, Home Directory は `'` で区切る。) Change, Exipre, Class は通常 `'0'` を記述する。

例:Login:Password:Uid:Gid:Change:Exipre:Class:Full Name,Office,Office Phone:Home Directory:Shell

A.3.2 標準環境 (設定ファイル等の雛型)

- PICKLES システムでは、ユーザの標準環境の設定を行なうための設定ファイルに dot ファイル (ファイル名が dot('.') で始まるファイル) を用いる。
- 設定ファイルは基本的に各ユーザのホームディレクトリに置く。
- 各設定ファイルは、基本的に UNIX における設定ファイルに準拠している。PICKLES システムの設定ファイルで、UNIX システムと異なる点は、.xsession が .pxsession に置き換わっていることである。
- 代表的な設定ファイルとして次のものが挙げられる。
 - .pxsession – X ウィンドウ稼働時に、ログインウィンドウからログインしたユーザの .pxsession がシェルスクリプトとして実行される。
 - chmod により実行許可の属性にある必要がある。
 - .cshrc – csh や tcsh がログイン時に読み込んで実行する。
 - シェル変数・環境変数のセット、エイリアスの定義、端末設定を行なう。
 - csh が起動されるたびに実行される。
 - .login – csh や tcsh がログイン時に .cshrc を実行した後に読み込み実行する。
 - シェル変数・環境変数のセット、エイリアスの定義、端末設定を行なう。
 - ログイン時に 1 度だけ実行される。
 - .logout – csh や tcsh がログアウト時に読み込んで実行する。
 - .emacs – Emacs が起動時に Emacs-Lisp のプログラムとして実行する。
 - 主に変数の値の指定、キーバインディングの定義、ライブラリの読み込みの指定等を行なう。
 - .Xresources – フォント、色等のクライアントアプリケーションの外観設定をデータベースに登録するための設定ファイル。
 - .twmrc – ウィンドウマネージャ twm の設定を行なう。
 - .fvwm2rc – ウィンドウマネージャ fvwm2 の設定を行なう。
 - .rhosts – そこに書かれたホストの同じユーザに対してアクセス許可を与える。

A.3.3 利用者情報が格納されているファイル群

adduser を実行した際に用いられるドットファイルの雛型は /usr/pickles/skel/home に置かれています。

パスワード情報は/etc3/master.passwdに格納されています。パスワードに関する情報は他に/etc3/passwd, /etc3/pwd.db, /etc3/spwd.dbに格納されていますが、どれもmaster.passwdから生成されたファイルです。

A.4 ディスクの管理運用

A.4.1 ディスク管理の方針

PICKLES 端末は管理作業の省力化することを目標に設計されている。

PICKLES 端末におけるディスクの管理方針は、端末が故障してもハードディスクを差し換えるだけで復旧できるようにすることである。そのため、端末間で共有できる情報とできない情報を分離し、別々のディスクにそれらの情報をいれることにした。

PICKLES 端末では管理情報を次の4つに分類した。

端末ハードウェア固有の情報 PICKLES 端末の仕様では、ハードウェア構成に適度な自由度を持たせている。そのため端末ごとに異なる情報が存在する。イーサネットインターフェースやビデオボードの種類などの情報がこれに含まれる。

ホスト固有の情報 IP アドレス、ネットマスク、ホスト名などのホスト固有の情報がこれに含まれる。

利用者が管理する情報 ホームディレクトリやスプールディレクトリなどの利用者が持つ情報がこれに含まれる。

端末間に共通の情報 すべての端末に共通の情報である。OS やアプリケーションプログラムなどがこれに含まれる。

A.4.2 ハードディスク構成

PICKLES 端末は「systemdisk」と「userdisk」の2台で構成される。システムディスクに端末間に共通の情報を格納し、ユーザディスクに端末ハードウェア固有の情報、ホスト固有の情報、利用者が管理する情報を格納する。

A.4.3 ハードディスク構成の種類

ハードディスク構成は

- 標準 PICKLES
- MIX PICKLES
- SMALL PICKLES

- MINI PICKLES
- SCSI PICKLES

に分類できる。

標準 PICKLES

標準の PICKLES 端末のディスク構成は 540MB のハードディスクを 2 台用いる。パーティション構成は以下のようになる。

```
systemdisk
    wd0a    /        48MB
    wd0b    swap     96MB
    wd0h    /usr     372MB
```

```
userdisk
    wd1e    /etc3    6MB
    wd1f    /var     96MB
    wd1g    /local   414MB
```

userdisk の /etc3 のディレクトリには /etc の下に置かれるファイルのうち、ホスト固有の情報が格納される。/etc3 の下の各ファイルは /etc ディレクトリからシンボリックリンクが張られている。

MIX PICKLES

MIX PICKLES は 1.2BG のハードディスクに標準 PICKLES の systemdisk と userdisk を共存させたものである。

パーティション構成は以下のようになる。

```
systemdisk
    wd0a    /        48MB
    wd0b    swap     96MB
    wd0h    /usr     372MB
```

```
userdisk
    wd0f    /etc3    6MB
    wd0g    /var     96MB
    wd0h    /local   414MB
```

SMALL PICKLES

SMALL PICKLES は 850MB のハードディスクに標準 PICKLES の systemdisk と小サイズの userdisk を共存させたものである。SMALL PICKLES はハードディスクを 1 台しか搭載できないような PC を用いるときに有効である。

MINI PICKLES

MINI PICKLES は 540MB のハードディスクに、標準 PICKLES の systemdisk からいくつかのファイルを削除した小サイズの systemdisk と小サイズの userdisk を共存させたものである。MINI PICKLES はハードディスクを 1 台しか搭載できないような PC を用いるときに有効である。

SCSI PICKLES

SCSI ハードディスクに対応させた PICKLES である。

A.5 各種サーバの設定

A.5.1 WWW サーバ

使用する www サーバには、特に指定はない。bsdi で動くものであればよい。

www サーバに関する実行形式、設定ファイル、ログ、www コンテンツ等は物理的に system/user ディスクとは別に用意するべきである。最低でも www コンテンツは別にしたほうがよい。

例. wd0: system ディスク
sd0: user ディスク
sd1: www サーバ用ディスク

これにより、www サーバ用ディスクさえ保持していれば、障害等の際にも別の pickles マシンで www サーバを代用することが容易である。

www サーバを起動するためには、以下の設定を行なう。

- www サーバを用意し、設定等を行なう。
- /etc3/fstab.userdisk に www サーバ用ディスクのマウント情報を記述する。マウントポイントは、/local の下に用意するのが望ましい。

例.
/dev/sd1h /local/www ufs rw 0 5

- /etc3/rc.user に www サーバの起動について記述する。

例.

```
# for WWW
if [ -f /local/www/conf/httpd.conf ];then
    echo -n 'apache_httpd '
    /local/www/bin/httpd -f /local/www/conf/httpd.conf
fi
```

注) 標準の PICKLES は、上記とは別に/etc/rc で www サーバの起動についての記述がある。混乱を招かないために、これをコメントアウトすることを勧める。

A.5.2 proxy サーバ

proxy サーバに関する実行形式、設定ファイル、ログ、cache は物理的に system/user ディスクとは別に用意するべきである。最低でも cache は別にしたほうがよい。

例. wd0: system ディスク
sd0: user ディスク
sd1: proxy サーバ用ディスク

これにより、proxy サーバ用ディスクさえ保持していれば、障害等の際にも別の pickles マシンで proxy サーバを代用することが容易である。

proxy サーバを起動するためには、以下の設定を行なう。

1. proxy サーバを用意し、設定等を行なう。
2. /etc3/fstab.userdisk に proxy サーバ用ディスクのマウント情報を記述する。マウントポイントは、/local の下に用意するのが望ましい。

例.

```
/dev/sd1f          /local/proxy ufs rw    0 4
```

3. /etc3/rc.user に proxy サーバの起動について記述する。

例.

```
# for PROXY
if [ -f /local/proxy/etc/proxy.conf ];then
    echo -n 'proxy '
    /local/proxy/bin/runcache
fi
```

注) 標準の PICKLES は、上記とは別に/etc/rc で proxy サーバの起動についての記述がある。混乱を招かないために、これをコメントアウトすることを勧める。

A.5.3 IP ルータ

PICKLES を IP ルータにするためには以下の設定をする。

1. /usr/src/sys/i386/conf/PICKELS の

```
#option GATEWAY
```

の行を

```
option GATEWAY
```

にする。その後カーネルを再構築する。

または、/etc/rc.local

```
#echo -n "IP forwarding: "; sysctl -w net.inet.ip.forwarding=1
```

の行を

```
echo -n "IP forwarding: "; sysctl -w net.inet.ip.forwarding=1
```

にする。

2. routed を起動する場合は /etc3/hostconfig に記述されている routedflags の値を修正する
3. 必要のないプロセスを消す。具体的には、プロセスの起動が記述されているファイル /etc/rc, /etc/rc.local, /etc3/inetd.confなどを修正する。

A.5.4 DHCP サーバ

1. /etc/dhpc.pool, /etc/dhcp.relay を用意する。
2. /etc3/rc.user に以下の行を加える

```
if [ -x /usr/contrib/bin/dhcps ]; usr
    /usr/contrib/bin/dhcps de0 ef0 > /dev/console
    echo 'dhcp server started.' > /dev/console
fi
```

A.6 アプリケーションの保守

A.6.1 新しいアプリケーションのインストール

PICKLES SYSTEM の設計方針では、システムディスクは原則として変更しません。従って、`/usr/local` の下にアプリケーションのインストールをするべきではありません。新しいアプリケーションをインストールしたい場合はユーザディスクにある `/local` を使用してください。

例えば、最近のアプリケーションの多くは GNU-autoconf を利用しています。この場合 `configure` を実行する際に `-prefix=/local` を付けて下さい。また、設定ファイルは原則として `/etc3`、スプール、ロックファイルなどは原則として `/var` に格納します。これらのディレクトリの指定も、`configure` の引数で指定するようにしてください。

A.7 バージョンアップ

A.7.1 バージョン情報の参照

システムディスクのバージョン情報は `/etc/.systemdisk` に記述されています。ユーザディスクのバージョン情報は `/etc3/.userdisk` に記述されています。

A.7.2 システムディスクのバージョンアップ

システムディスクの更新作業は、インストール作業と同様にいくつかの方法があります。

ディスクモジュールごと更新する場合 この場合は古いディスクモジュールを取り出して、新しいディスクモジュールに入れ替えるだけです。

CD-ROM を使って更新する場合 インストール作業と同様の作業を行いますが、”install”ではなく”update”を選択してください。この場合、`/usr` パーティションの空き容量に注意してください。

A.7.3 ユーザディスクのバージョンアップ

ユーザディスクは、システムディスクを Update すれば、起動時に自動的に更新されます。

A.7.4 syncuser スクリプト

ユーザディスクの自動更新の仕組みは `syncuser` スクリプトによって実現されています。

A.8 カーネルの再構築

PICKLES SYSTEM 標準のカーネルは、利用が保証されている機器のほとんどをそのまま利用できます。しかし、機器の設定を変更した場合や、一部の特殊な機器の場合を利用する場合はカーネルを再構築する必要があります。また、大規模なサーバとして用いる場合など、カーネルの再構築が推奨される場合もあります。この章では PICKLES SYSTEM のカーネルを再構築する手順について述べます。

A.8.1 再構築の手順

カーネルのソースをコピーする

はじめてカーネルを再構築する場合は、カーネルのソースをユーザディスクにコピーする必要があります。各ユーザのホームか `/local/src/` にコピーしてください。コピー方法は

```
cp -Rpp /usr/src/sys.local /local/src/
```

で良いでしょう。以下ではカーネルのソースの先頭のディレクトリを `/local/src/sys` として説明します。

`config` の修正

カーネルの `config` ファイルを変更します。カーネルの `config` ファイルは `/local/src/sys/i386/conf/` におかれています。PICKLES 標準カーネルの `config` ファイル名は PICKLES です。これを適当な名前 (一般的にはそのホスト名。ここでは SAMPLE とする) にコピーして編集してください。

`config` の実行とコンパイル

`config` ファイルが置かれているディレクトリに移動し、以下の手順を順番に実行してください。

```
# cd /local/src/sys/i386/conf
# vi SAMPLE    # Edit config file
# config SAMPLE
# cd ../../compile/SAMPLE
# /usr/bin/make clean depend all
# cp /bsd /bsd.old
# cp bsd /
```

A.8.2 カーネルの再構築に関してよくある質問

本来はFAQ集として独立させるべきであるが、カーネルの設定には子細な知識や経験が要求される局面があるため、ここにまとめておく。

timezone の設定はどうすればいいのですか 一般解としては、CMOS の時計が GMT の場合は kernel の timezone は 0 に、CMOS の時計が localtime の場合は kernel の timezone は GMT からの時差にしてください。PICKLES の場合は、後者が義務づけられています。

maxusers ってなんですか kernel が内部で用いるいろいろなテーブルの大きさを決定するためのパラメータです。決して同時に login できる最大ユーザ数ではありません。この値は大規模なサーバで用いるのでなければ変更する必要はありません。

PCIC(PCMCIA コントローラ) が割込みを利用しないような設定はできますか？
できます。

3COM の 3C509 を利用しています。システム起動時には認識されますが実際には使えません。なぜなのでしょう？ 3C509 付属の設定ユーティリティを使って、kernel の config で指定している I/O 領域や IRQ と同じ値に設定されているか確認してください。3C509 はボードの検出用に実際の動作とは異なる I/O ポートを参照しているため、検出されているからといって正しく動くとは限らないという癖があります。

SMC の Ethernet カードを使っています。使っているとコンソールにエラーメッセージが沢山でできます。なぜでしょうか？ 送受信バッファに用いているメモリ領域が、ボードの設定と kernel の config とでずれている可能性があります。確認してください。

私の持っている Ethernet カードだと IPv6 が使えません。なぜですか？ それは、そのカードがマルチキャストを正しく拾えない悪い子だからです。多分。

純正 SoundBlaster 以外の SoundCard は使えませんか？ 使えます。詳しくは HCL を参照ください。

カーネルの再構築にはどのくらいの空きディスク容量が必要ですか？ ソースの他に 16MB 程度の領域が必要です。デバッグ用カーネルを作成する場合はさらに 10MB 程度必要になります。

付録B PICKLES FAQ

この文書はBSD/OS版 PICKLES SYSTEMに関する「よくある質問と回答」をまとめたものである。この文書は1999年の時点のものであり、一次情報は<http://www.ohnolab.org/researches/pickles/pickles-faq/>である。なお、原文のうち不十分な記述は削除して整理したものを収録している。ただし加筆修正は行っていない。

B.1 PICKLES 概要

このFAQの目的は？ PICKLESに関して頻繁に尋ねられる質問とそれに対する答えを網羅し、より多くの人達にPICKLESというプロジェクトを理解してもらうことがこのFAQの目的です。それと同時に、PICKLESの利用者、管理者、所有者にとって有益な情報源となることも目指しています。

もしこのFAQに加えることが適当と思われる内容があったらPICKLESの管理者までご連絡ください。

PICKLESって何？ PICKLESとは東京工業大学大野研究室がすすめるプロジェクトの名称です。

どうしてPICKLESと呼ばれているのですか？ もともとはPublic/Personal Information Catering Kiosk and Literacy Education Systemの略でしたが、現在ではこの事実は忘れ去られようとしています。またPICKLESという単語が英語的にnegativeな意味があるのと、普通名詞なため登録商標がとれないという理由で、名称については再検討の余地があると考えています。

ロゴの意味を教えてください PICKLESのロゴは開発責任者の一人である木本(kimoto@ohnolab.org)が友人に依頼してデザインしてもらいました。このロゴの周囲の丸は「どこにいても行っても利用できる環境」を意味しており、中央の鎖は「インターネットの接続性」を意味しています。さらに中央にはPICKLESの「P」の文字を配しています。

PICKLESのリリースはいつ作られるのですか？ PICKLES SYSTEMのリリースは大きくわけて次の3種類があります

DR(Developer's Release): 開発者向けリリース。その内容は後のリリースで大幅に変更になる可能性があります。完全に内輪での利用を念頭においているため、ライセンス的に問題のある内容が含まれる可能性もありえます。つまりあんまり責任もてません、ということ。

ALPHA: 内部テストバージョン。この後は内容的には大幅な変更はしない。

BETA: 外部の人にも評価してもらえるバージョン。

RELEASE: 正式リリース。

また、各リリース内で複数の版があるときは、名前の後ろに数字をつけて DR3, ALPHA2 などというバージョン名をつけます。

PICKLES のリリースに有効期限はありますか？ 2.00DR5 からは、正式に有効期限という概念が導入されます。この情報は/etc/.systemdisk というファイルに記述されています。あるリリースの有効期限は、おおよそ以下を目安に決められています。

DR,ALPH,BETA: 2ヵ月 (最大3ヵ月)

正式リリース: 無期限

誰が PICKLES の責任者ですか？ PICKLES Project は東京工業大学 情報理工学研究科 大野研究室の活動の一つです。最高責任者は同研究室の責任者である大野浩之講師になります。システム開発は、同研究室博士過程所属の木本雅彦が主に携わっています。

1999年7月の大野講師の郵政省への異動により、東工大からは大野研究室はなくなりました。しかし、1999年度いっぱいには学生はこれまで通りの活動をつづけますし、PICKLES のアプローチ自体は今後も継続して研究されます。

PICKLES の著作権は？ PICKLES SYSTEM で独自に開発した部分については、大野研究室 PICKLES Project が、その著作権を所持します。BSD/OS については BSDI 社が著作権を所持します。PICKLES SYSTEM に導入されている各アプリケーションについては、各々の著作権表示に従います。

どこで PICKLES の情報を手に入れることができますか？ PICKLES に関する最新情報は Web ページを参照してください。

URL は <http://www.ohnolab.org/resaerches/pickles/> です。

ソースコードの配布条件は？ PICKLES SYSTEM に関して、なにをもってソースコードと呼ぶかはむずかしいので、「パッケージ」の配布条件のことを指しているものとしてお答えします。現在のところ PICKLES SYSTEM のパッケージは PICKLES Project による一時配布のみになります。2次配布等は認めておりません。

PICKLES のメーリングリストはありますか？ 内部連絡用のものはいくつかありますが、外部との窓口は以下のアドレスです。

`pickles-talk@ohnolab.org` : PICKLES 開発グループへの連絡用

PICKLES のマークはどなたがデザインされたのですか？ デザインした人については前述の項目を参照してください。PICKLES ロゴマークのシールははすでに小数ですが作成されています。PICKLES ロゴ入りの PC 用エンブレムは大野研のすべてのルータ機器や端末につけられています。大野研メンバーが持っているノート PC を良く見ると、このエンブレムがついたものを見付けられるでしょう。また、なんらか形で、公的に認められたマークにすることは検討しています。

PICKLES は BSD/OS をベースにしていると聞きましたが、どうして高価な商用 OS をベースにしているのでしょうか？ この質問はまさに「良く聞かれる質問」です。PICKLES 開発当初は、以下の理由から BSD/OS を選択しました。これが歴史的な理由です。

- 当時 PCMCIA サポートが一番安定していた PC-UNIX は BSD/OS (WILDBOARD というドライバセット) であった。
- 同様にノート PC のサポートも BSD/OS が一番優れていた。
- 当時はフリー PC-UNIX に対する認知がそれほど高くなく、企業での利用を考えると、むしろ商用の方が信頼されやすかった。
- 教育機関での利用の範囲では、フリー UNIX と BSD/OS との間にコストの差はなかった。(サイトライセンスがあるため)

しかしその後状況は変化しました。BSD/OS の認知度は著しく低下し(これは開発元である BSDI の販売戦略の失敗によるものが大きいでしょう) フリーの PC-UNIX は Linux を筆頭に社会的に認知されています。広く多数の人に使ってもらうためにはフリー PC-UNIX ベースの PICKLES が必要でしょう。日程などの具体的なプランは今のところありませんが、その必要性については認識しています。むしろ、今後の PICKLES はベースとなる OS にはこだわらずに、その上にどういう環境を構築するかという点に注力すべきです。

B.2 インストール

PICKLES のインストールに必要なディスク容量は？ 現在 (2.00DR8) では、システムディスクだけで最低 1GB 程度必要です。この他にユーザディスクの領域が最低 200MB 程度必要です。ユーザディスクの大きさは、必要とする利用者の作業領域の大きさが大きくなれば、大きな容量を必要とします。

PICKLES のインストールについての説明書はどこにありますか？ 作成中です。PICKLES のホームページ、PICKLES CDROM、インストール FD に置く予定です。

インストーラを日本語表示して下さい。これはもっともな要求だとは思いますが、現時点ではその予定はありません。インストーラについては、抜本的な修正を検討しており、いくつかの具体的なアイデアも既に持っています。新しいインストーラは、目標としてはすべての BSD-UNIX で共通して利用できるものであることを目指しています。

どのようなメディアからインストールできますか？ 導入手順は以下の4つがあります。

- 導入済のハードディスクモジュールをプロジェクトマネージャに送付してもらう
- CD-ROM から (一般的な方法で) インストールする
- ネットワーク経由で他のマシンの CD-ROM からインストールする
- すでに稼働しているシステムの複製をつくる

ネットワーク経由でインストールできますか？ 可能です。インストール時に remote インストールか local インストールかを聞いてきます。その際に remote を選択してください。ただし利用可能な Ethernet インタフェースが用意されていることが前提です。

同じマシンで Windows95,98 と共存できますか？ 可能です。こつとしては、最初に Windows{95,98} をインストールしてから PICKLES をインストールすることを推奨します。

自分の PC に複数のオペレーティングシステムを入れるには？ インストール時に、「他の OS と共存するか？」と聞かれます。その際に yes と答えてください。

Windows{95,98} を入れたら PICKLES が起動しなくなりました。Windows のインストーラが MBR(MasterBootRecord) を破壊してしまったためです。PICKLES のインストールフロッピーを使って起動した後、local 又は remote の CD-ROM を mount し、disksetup コマンドを用いて MBR を書き直して下さい。

パーティションはどのように切ればいいですか？ 各パーティションの容量は推奨値が決められています。この条件を満たしていれば、ユーザが希望のパーティション構成を設定できます。

ハードディスクドライブには、どのジオメトリを使うべきでしょうか？ BSD/OS を扱う上で、必ず聞かれるのがこの質問です。disksetup を実行すると、最初にジオメトリを質問されます。通常は CMOS と同じ値にすれば良いようです。BSD/OS のジオメトリ情報が書かれていない状態では、自動検出 (Probe) を選択すれば良いです。

B.3 ハードウェアコンパチビリティ

PICKLES を使うのにどのようなハードウェアが必要ですか？

<http://www.ohnolab.org/researches/pickles/spec1998.html> を参照してください。また (HCL) ハードウェアコンパチビリティリストを作成しています。

PICKLES はどのようなハードディスクドライブをサポートしているのですか？ 標準的な PICKLES では EIDE ドライブ 2 台の構成ですが、SCSI ハードディスクドライブもサポートしています。詳しくは PICKLES-HCL を参照してください。

どの SCSI コントローラをサポートしているのですか？ NCR 製の SCSI コントローラや、Adaptec 製の SCSI コントローラなどをサポートしています。詳しくは PICKLES-HCL を参照してください。

どんな CD-ROM ドライブをサポートしているのですか？ ATAPI 接続のドライブや、SCSI 接続のドライブのほとんどをサポートしています。詳しくは PICKLES-HCL を参照してください。

ZIP ドライブをサポートしていますか？ SCSI 接続のドライブをサポートしています。パラレルポート接続のドライブはサポートしていません。詳しくは、PICKLES-HCL を参照してください。

JAZ や EZ、それからその他のリムーバブルドライブはサポートしていますか？ SCSI 接続のドライブであればほとんどの製品が利用できます。PD については ATAPI 接続のドライブもサポートします。詳しくは PICKLES-HCL を参照してください。

どのマルチポートシリアルカードをサポートしていますか？ SDL RISCCom/8, Digiboard PC/Xem などのマルチポートシリアルカードをサポートしています。詳しくは PICKLES-HCL を参照してください。

マウスのホイールやボタンを、PICKLES で使うことはできますか？ 現在のところ、利用できません。ただし、3 ボタンのマウスとして使うことは可能です。

USB キーボードや USB マウスを利用利用できますか？ BIOS が Legacy モードをサポートしていれば利用できます。通常の USB デバイスとしては、現時点では正式サポートしていません。しかし、2.00DR5 以降では試験版のデバイスドライバは組み込まれています。

ラップトップ PC のマウス/トラックボール/タッチパッドは使えますか？ PS/2 タイプのものはサポートしています。東芝製，ソニー製，Panasonic 製，日立製のラップトップ PC の多くの製品で実績があります。詳しくは PICKLES-HCL を参照してください。

どんなテープドライブをサポートしていますか？ SCSI 接続の DDS ドライブなどをサポートしています。詳しくは PICKLES-HCL を参照してください。

どんなテープチェンジャーをサポートしていますか？ SCSI 接続の DDS チェンジャーをサポートしています。詳しくは PICKLES-HCL を参照してください。

どんなサウンドカードをサポートしていますか？ Sound Blaster 16 などのカードをサポートしています。ただし，PnP 専用のカードは基本的にはサポートしていませんが、使える場合が多いようです。詳しくは PICKLES-HCL を参照してください。

PCMCIA カードは何をサポートしていますか？ NE2000 互換の Ethernet card や Adaptec 製の SCSI card などをサポートしています。詳しくは PICKLES-HCL を参照してください。

ラップトップのパワーマネージメント機能を使うことができますか？ 可能です。

B.4 システム管理

システムが無反応になりました。どう対処すればよいでしょう キーボードで Enter キーを押して反応がある場合は、Ctrl キーと Alt キーと Del キーを同時に押してください。reboot するかどうか確認されるので、'y' を押してください。そのしばらくまっても再起動しない場合は、再度 Ctrl+Alt+Del を押した後、'y' を押してください。

キーボードも反応しない場合はどう対処したらいいのでしょうか。 ネットワーク経由でのログインもできず、キーボードも無反応の場合は電源を停止させてください。この際ハードディスクの内容の損傷を最低限に押えるため、HDD のアクセスランプに注意し、書き込みが行われた後に電源を停止させてください。30 秒待っても HDD のアクセスランプが点灯しない場合は、あきらめて電源を停止してください。

この結果システムディスクが破損した場合は、PICKLES SYSTEM の配布者に連絡をとってください。代替ディスクを送付します。

ルートのパスワードを忘れてしまいました。

システムが起動時に参照するファイル群はどこに置かれていますか？ ほとんどの設定ファイルは/etc3 に置かれています。詳しくは管理ドキュメントを参照してください。

簡単にユーザを追加するにはどうすればいいのですか？ 2.00DR8からは、`/usr/pickles/bin/configuser` を実行し、ユーザ管理を選択すればよいです。実際には configuser のユーザ管理は vipw, adduser のいずれかのプログラムを実行しているので、これらを直接実行しても構いません。

新しいリムーバブルドライブを持っていますが、どうやって使うの？ そのリムーバブルドライブが SCSI であれば問題なく利用できる可能性が高いです。少なくとも 128/230MB の MO ドライブと、PD ドライブは動作実績があります。ATAPI の場合、一部の PD/CD ドライブであれば利用することができます。もしあなたが持っている ATAPI の PD/CD ドライブが認識されなかったら、連絡を下さい。積極的に対応する用意があります。

他のシステムのファイルシステムを PICKLES でマウントすることはできますか？ BSD/OS がマウントできるファイルシステムならば、マウントできます。CD-ROM や MS-DOS のファイルシステムについては確認してあります。

その他の各種設定はどうすれば良いですか？ 詳しくは管理ドキュメントを参照してください。

B.5 ユーザアプリケーション

PICKLES で利用可能なアプリケーション一覧はありますか？ 最新 PICKLES にインストールされているアプリケーションリストを作成しています。

FreeBSD や Linux 用のアプリケーションは実行できますか？ PICKLES SYSTEM 2.00DR3 からは、FreeBSD 2.2.x 用の実行形式はほぼ問題なく実行できます。ただし一部のアプリケーションでは日本語の表示や、動的ローディング処理で問題が発生するかもしれません。Linux 用の実行形式については現在 BSDI 社が LAP というシステムを開発しています。LAP を使えば Linux 用のさまざまなアプリケーションが実行できるようになります。現時点の PICKLES では LAP はサポートしていません。

基本的なアプリケーションとしてエディタ・Web ブラウザ・E-MAIL ソフトが一種類ずつ挙げられていますが、他のアプリケーションは使用できますか？ 利用できます。

エディタ・Web ブラウザ・E-MAIL ソフトが基本的なアプリケーションである理由情報 KIOSK である PICKLES 端末では、その主な役割として情報の閲覧と交換のプラットフォームとなることが挙げられます。そのために必要なソフトウェアが、PICKLES における基本的アプリケーションです。現在の所、情報閲覧のためのソフトウェアとしては Web ブラウザが、情報交換のためのソフトウェアとしては E-MAIL ソフトとエディタが一般的ですので、基本的アプリケーションとして以上の3つが採用されています。

Java アプレットの表示は可能ですか PICKLES SYSTEM 2.00DR8 では、netscape-3.01 では表示可能です。netscape-4.7 では現在のところ成功していません。

Web ブラウザのプラグインは何をサポートしていますか？ MIDI, PDF, Flash などは外部ブラウザでの閲覧が可能です。プラグインという形ではサポートしていません。

B.6 ウィンドウシステム

起動した時にウィンドウシステムが立ち上がっていません。 login プロンプトが出ている状態で、'pxdm' と入力してください。パスワードは要求されません。ウィンドウシステムが立ち上がり、ログイン画面が表示されます。

起動時に自動的にウィンドウシステムを立ち上げるには、/etc3/hostconfig を編集して'PXDM=YES'を追加してください。

パスワードでログインするには？ ウィンドウシステムが起動した時のログイン画面ではIDカードの挿入を要求してきます。ログイン名とパスワードによるログイン画面に切り替えるには ctrl+'q' を押して下さい。

ウィンドウシステムを停止するには ログイン画面でパスワードによるログインに切り替えた後、Ctrl+'r' を押してください。コマンドプロンプトになります。

Window Manager には何がありますか？ PICKLES に標準添付されているのは、twm,fvwm, fvwm2, ctwm,vtwm, WindowMaker の5種類です。adduser した時の標準環境では、fvwm2 が立ち上がるようになっています。

Window Manager の設定はどのようにするのですか？ Window Manager に異なります。WindowMaker では設定を変更する GUI が用意されています。

PICKLES のデスクトップ環境は？ 現在のところ GUI シェルの試作品と、いくつかのファイルマネージャは用意されていますが、統合的なデスクトップは整備されていません。gnome ベースのデスクトップ環境を予定していますが、今のところ次の段階への課題となっております。

B.7 サーバ構築

ルータとして運用したい 2枚のネットワークカードを PICKLES マシンに差せば、`routed` を使うことによりルータとして使用可能です。現行の PICKLES 2.00DR8 では、以下の作業を行う必要があります。

- `/etc/hostconfig` で、2枚のネットワークインタフェースに明示的にアドレスを割り振る。
- `/etc/rc.user` に以下の記述を追加する。

```
echo -n "IP forwarding: "; sysctl -w net.inet.ip.forwarding=1
/usr/old/routed -s
```

NAT 機能を利用したい PICKLES SYSTEM 2.00DR8 からは NAT 機能を標準装備しています。この機能は IP Filter 3.1 を用いて実現されています。ipfilter の設定は `/etc3/ipfilter` に置かれています。`/etc3/ipfilter/natrules` を記述した上で、`/etc/rc.user` に以下の記述を追加してください。

```
echo -n "IP forwarding: "; sysctl -w net.inet.ip.forwarding=1
/usr/local/sbin/ipnat -f /etc3/ipfilter/natrules
```

詳しくは管理者ガイドを参照ください。

Firewall を構築したい パケットを選択的に遮断/通過させる方法として、`screend` と IP Filter の 2 種類の機構を提供しています。詳しくは管理者ガイドを参照ください。

PICKLES をメールサーバにしたい PICKLES には `qmail` というメールサーバがインストールされています。PICKLES に導入されている `qmail` は IPv6 対応になっていますので、そのまま IPv6 対応メールサーバとしても利用できます。

設定情報は `/var/qmail/control/` 以下にまとめられています。通常は `/usr/pickles/bin/configuser` を実行し、ホストの設定を行えば `qmail` の設定も書き込まれます。ただし、メールサーバとして利用するためには以下の点を修正する必要があります。

- `locals`, `rcpthosts` にメールアドレスのドメインを追加する。例えばホスト名が `mail.domain.com` で `user@domain.com` 宛てのメールを処理する場合、`domain.com` を追加する。
- `defaultdomain`, `defaulthost` が `domain.com` になっていることを確認する。
- メールを受け取る全ユーザのアカウントを作成する。ユーザのホームディレクトリは十分な容量がなければならない。

- メールの中継を行う場合、中継を許可するホストに対してのアクセス制限を解除しなければならない。これは `tcpserver` の設定を修正する必要がある。

注) `rcpthosts` を空にすると、全てのメールを中継可能になるため設定は簡単です。しかし、この設定では SPAM メールの踏台になれてしまうため、絶対にこの設定は行わないでください。設定について詳しくは管理者ガイドを参照ください。

qmail のドキュメントは `/usr/contrib/qmail/doc/` にまとめられています。

PICKLES を WWW サーバにしたい PICKLES には `apache` という WWW サーバがインストールされています。PICKLES に導入されている `apache` は IPv6 対応になっているので、そのまま IPv6 対応 WWW サーバとして利用できます。

`apache` の設定ファイルは `/var/www/conf/` に置かれています。`apache` を立ち上げるには、`/var/www/conf/httpd.conf`, `/var/www/conf/srm.conf` を編集した後、`/usr/contrib/bin/httpd -f /var/www/conf/httpd.conf` を実行してください。

システム起動時に `apache` を立ち上げるためには、以下の修正を行う必要があります。

- `/var/www/docs/index.html` を用意する。
- `/var/www/bin/start-apache` を用意する。通常は `start-apache.sample` をそのまま利用すれば良い。

`apache` に関するドキュメントは `/var/www/docs/apache/` 以下に HTML 形式で置かれています。

B.8 PICKLES の入手について

どこから PICKLES SYSTEM を入手できますか？ 現在 PICKLES は無制限には配布していません。PICKLES 開発者に直接連絡してください。

PICKLES の入手に必要なライセンスについて BSD/OS 3.1 のソースライセンスが必要です。この他に標準搭載されている `DynaFont` のライセンスが必要になります。

2.00 正式リリースまでには、バイナリライセンスでのパッケージと、商用フォントの分離を行う予定です。

PICKLES 対応ハードウェアを入手するには 仕様書と HCL を参考にして、部品を集めて購入することも可能ですが、現在ぷらっとホーム社から PICKLES の仕様を満たしたハードウェアを一式購入できます。ぷらっとホーム社の `EntrySystem` をベースにしており、価格は 20 万円程度になります。これについては、個別にご連絡ください。

付録C ぶらっとホーム製 PICKLES対応PC仕様書

EntrySystem/PICKLES version 1.2 設定仕様書

第1版 1999年
3月4日

東京工業大学大学院情
報理工学研究科
数理・計算科学専攻 大
野研究室
PICKLES Project

ハードウェア構成

CPU: Celeron 300Mhz

RAM: 64MB (SDRAM DIMM × 1)

MB: ASUSTeX P2L97

Video: ATI XPERT98

HDD ケース: OWL-MR30AE × 2

SoundCard: SB16DV

PCMCIA I/F: RATOC REX-5051FV

HDD: IBM DTTA-350430 (4.3GB EIDE) × 2

Ethernet: Intel EtherExpressPro/100

本体ケース:

補足事項)

- 上記の構成はぶらっとホーム株式会社製の EntrySystem/PICKLES version 1.2 と同一である。

ハードウェアの接続

ハードウェアは以下のように接続した状態で出荷される。下記での上下は、通常利用する状態の上下とする。

ドライブベイ：

- 3つ5インチベイには上から順に removableHDD ケース×2, CD-ROM の順に収納する
- PrimaryIDE インタフェースに2台の removableHDD ケースを接続する。
- SecondaryIDE インタフェースには CD-ROM を接続する。
- 2台の3.5インチベイには PCMCIA I/F と FDD を収納する。順番は上から PCMCIA I/F, FDD の順番とする。

拡張スロット：

- PCI 拡張スロットには VGA カードと Ethernet カードを接続する。順番は上から VGA カード, Ethernet カードとする。
- ISA 拡張スロットには PCMCIA I/F カードと SoundCard を接続する。順番は上から SoundCard, PCMCIA I/F カードとする。

BIOS 設定項目一覧

STANDARD CMOS SETUP

時計：日本 (JST) 時間に合わせる (GMT ではない)

HDD：

Primary Master	: TYPE=Auto MODE=LBA
Primary Slave	: TYPE=Auto MODE=LBA
Secondary Master	: TYPE=Auto MODE=AUTO
Secondary Slave	: TYPE=Auto MODE=AUTO

Drive A : 1.44M, 3.5 in.

Drive B : None.

Floppy 3 mode Support : Disable

Video : EGA/VGA

Halt On : All Errors

BIOS FEATURE SETUP

Boot Virus Detection	: Disabled
CPU L1 cache	: Enabled
CPU L2 cache	: Enabled
CPU Level2 Cache ECC	: Disabled
BIOS Update	: Enabled
Quick Power On Self Test	: Enabled
HDD Sequence SCSI/IDE First	: IDE
Boot Sequence	: A,C
Boot Up Floppy Seek	: Disabled
Floppy Disk Access Control	: R/W
IDE HDD block Mode Sectors	: HDD MAX
HDD S.M.A.R.T. capability	: Disabled
PS/2 Mouse Function Control	: Auto
OS/2 Onboard Memory > 64M	: Disabled
PCI/VGA Palette Snoop	: Disabled
Video ROM BIOS Shadow	: Enabled
C8000 - CBfff Shadow	: Disabled
CC000 - CFfff Shadow	: Disabled
D0000 - D3fff Shadow	: Disabled
D4000 - D7fff Shadow	: Disabled
D8000 - DBfff Shadow	: Disabled
DC000 - DFfff Shadow	: Disabled
Boot Up Numlock Status	: Off
Typematic Rate Setting	: Disabled
Typematic Rate	: 6
Typematic Delay	: 250
Security Option	: System

CHIPSET FEATURE SETUP

EDO Auto Configuration	: 60ns DRAM
SDRAM Configuration	: 12ns SDRAM
SDRAM RAS to CAS Delay	: Auto
SDRAM RAS Precharge Time	: Auto
MA Wait State	: Auto
SDRAM Banks Close Policy	: Arbitration

16-bit I/O Recovery Time	:	1 BUSCLK
8-bit I/O Recovery Time	:	1 BUSCLK
Graphics Aperture Size	:	64MB
Video Memory Cache Mode	:	UC
PCI 2.1 Support	:	Enabled
Memory Hole At 15M-16M	:	Disabled
Onboard FDC Controller	:	Enabled
Onboard FDC Swap A & B	:	No Swap
Onboard Serial Port 1	:	3F8H/IRQ4
Onboard Serial Port 2	:	Disabled
Onboard Parallel Port	:	378H/IRQ7
Parallel Port Mode	:	EPP
Onboard PCI IDE Enable	:	Both
IDE Ultra DMA Mode	:	Auto
IDE0 Master PIO/DMA Mode	:	Auto
IDE0 Slave PIO/DMA Mode	:	Auto
IDE1 Master PIO/DMA Mode	:	Auto
IDE2 Slave PIO/DMA Mode	:	Auto

POWER MANAGEMENT SETUP

Power Management	:	User Define
Video Off Option	:	Suspend Off
Video Off Method	:	DPMS OFF
HDD Power Down	:	Disable
Suspend Mode	:	Disable
PWR Button < 4 secs	:	Soft Off
PWR Up On Modem Act	:	Disabled
AC PWR Loss Restart	:	Disabled
Wake On LAN	:	Disabled
Automatic Power UP	:	Disabled

PNP/PCI CONFIGURATION

PNP OS Installed	:	No
------------------	---	----

Slot 1 IRQ	: 10
Slot 2 IRQ	: 10
Slot 3 IRQ	: 10
Slot 4 IRQ	: 10
PCI Latency Timer	: 32 PCI Clock
IRQ-3 Used by ISA	: NO/ICU
IRQ-4 Used by ISA	: NO/ICU
IRQ-5 Used by ISA	: Yes
IRQ-7 Used by ISA	: NO/ICU
IRQ-9 Used by ISA	: NO/ICU
IRQ-10 Used by ISA	: NO/ICU
IRQ-11 Used by ISA	: NO/ICU
IRQ-12 Used by ISA	: NO/ICU
IRQ-14 Used by ISA	: NO/ICU
IRQ-15 Used by ISA	: NO/ICU
DMA-1	: Yes
DMA-3	: NO/ICU
DMA-5	: Yes
ISA MEM Block BASE	: No/ICU
USB IRQ	: Disabled
VGA BIOS Sequence	: PCI/AGP

付録D PICKLES対応PC仕様書 (1996年版)

ここで述べる仕様は1996年に制定したものであり、当時の文書をそのまま掲載している。

PICKLES TERMINAL SPECIFICATION (1996)

1. INTRODUCTION

この文書は、PICKLES 端末の仕様を示したものである。PICKLES 端末は東京工業大学大野研究室で行なわれている PICKLES プロジェクトで開発している端末で、インターネットを利用するための公衆端末として利用を念頭において開発されたものである。管理の容易さから、公衆端末としてのみならず、広く利用者向け端末としても有益である。なお、この仕様は1996年に制定したものであり、将来変更される可能性がある。

2. AT A GLANCE

部品	class	条件
CPU	B	Pentium class 90MHz 相当以上
memory	B	32MB 以上
BIOS	B	IDE HDD 容量の自動検出が可能なもの
MotherBoard	B	
2ndCache	D	256KB 以上推奨
拡張バス	B	PCI/ISA
ChipSet	D	
VideoCard	B	AcceleratedX または XFree86 で 利用可能で、1152x864 以上の解像度 で 65536 色以上の表示が可能なもの。
HDD 搭載する。	A	WesternDigital 製の 540MB 以上の容量のものを 2 台
FDD	D	3.5" 2HD 2mode
PCMCIA	A	Wildboar で利用可能なもの
SerialPort	B	16550A 互換のもの。2 ポート

PararelPort	B	EPP 利用可能なもの。1 ポート
IDE Interface	D	PCI 接続のもの
HDD Case	A	指定機種を用いる
PC Case	D	5 インチベイが二つ。3.5 インチベイが一つ以上。
電源	D	
CD-ROM	D	特に必要なし
FaxModem	C	データ通信 28.8Kbps 以上の FaxModem HylaFax で利用可能なもの。
PointingDevice	D	3 ボタンのマウスまたはトラックボール
KeyBoard	B	101/104 キーボード
Soundcard	B	SoundBlaster16。Plug and Play 非対応のもの。
EthernetCard	B	ISA または PCI。BSD/OS で利用可能なもの。
Display	D	21 インチサイズ。1152x864 ドットの 解像度がリフレッシュレート xx 以上で利用可能なもの。
Speaker	D	
Microphone	D	

class とは以下の 4 種類である。

- class A 絶対にこのメーカーのこの型番でなければならないもの
- class B メーカー型番不問だが、条件を要するもの
- class C メーカー型番不問だが、設計変更を行う必要があるもの
- class D メーカー型番はまったく不問

3. SPECIFICATION

3-1. BASIC REQUIREMENT

ARCHITECTURE

IBM-PC/AT アーキテクチャを先祖とした、いわゆる AT 互換機を用いる。

CPU

Intel 社 Pentium 90MHz よりも高速なものである必要がある。

これは利用者に対して端末の処理能力が低いことによって、ネットワーク自体に対して誤った認識を与えないためである。

また他社の Pentium 互換 CPU についても、BSD/OS で動作が確認されているものは利用可能である。

MEMORY

32MB 以上必要。可能であれば EDO-RAM を推奨する。

2 次キャッシュは SRAM 256KB 以上 (可能であれば PB-SRAM) を推奨する。

MOTHERBOARD

特に指定はない。

ATX 準拠のもので電源の制御ができるものを推奨する。

BIOS

IDE HDD のパラメータを自動的に検知してくれるものが必要。

APM BIOS については version1.1 以上を推奨。

次にあげる項目が設定可能である必要がある。

- ・ PCI のボードが利用する IRQ
- ・ ISA バスのボードが利用する IRQ

CHIPSET

CASE

ATX 準拠のマザーボードを用いる場合は、ATX 準拠の電源を利用する。

5 インチベイが二つ以上、3.5 インチベイが一つ以上ある必要がある。

さらに CD-ROM や PCMCIA スロットを本体に内蔵する場合は、必要なだけのスロットを追加する。

背面部拡張スロットはシリアル、パラレル、キーボード、PS/2 などを除くと 5 つ以上必要である。

3-2. BUSES

ISA

サウンドカード、PCMCIA インタフェース、モデムカード、Ethernet カードのが ISA バスに接続される。PCI 接続の Ethernet カードを用いる場合は、これを除く 3 種類になる。

PCI

ビデオカードが PCI バスに接続される。PCI 接続の Ethernet カードを用いる場合は、

これら 2 種類が PCI バスに接続される。

ATA

マスターとスレイブにハードディスクを一台ずつ接続する。ATAPI CD-ROM を接続する場合は、セカンダリの IDE コントローラのマスターに接続する。

PCMCIA

PCMCIA TypeII のカードが一台以上利用可能であることを推奨する。

PCMCIA コントローラは Wildboar で利用可能なものである必要がある。

3-3. DEVICES

SERIAL

16550 相当のシリアルコントローラを利用し、2ポート以上利用可能である必要がある。

PARALLEL

EPP 利用可能なものを推奨する。

3-4. INPUT DEVICES

KEYBOARD

AT または PS/2 のキーボードが一台必要である。

101 キーボードを推奨する。

POINTING DEVICE

シリアルまたは PS/2 のインタフェースで接続された、3 ボタンのマウスまたはトラックボールを一台必要とする。

Accelerated-X か XFree86 で 3 ボタンが利用可能である必要がある。

キーボードが PS/2 である場合は、ポインティングデバイスも PS/2 に統一することを推奨する。

PS/2 インタフェースは BSD/OS で安定して利用可能である必要がある。

注) 富士通 FM-V の一部の製品で、PS/2 が安定して利用できないものがあるという報告をうけている。FMV-5xxx の 1996 年前半のロットにこの現象が現れるらしい。

3-5. GRAPHIC ADAPTERS AND DISPLAYS

GRAPHIC ADAPTER

Accelerated-X (version1.2) で 1152 ドット × 864 ドット以上の解像度で 65536 色以上表示可能なものである必要がある。またこのときのリフレッシュレートが 以上であることを推奨する。

この条件を満たすコントローラとしては以下のものが挙げられる。

DISPLAY

多くの人が同時にのぞきこみながら利用することを想定して、21 インチのマルチシンクのモニターを推奨する。

1152 ドット × 864 ドットの解像度で

3-6. AUDIO

AUDIO DEVICE

全二重 16bitPCM の入出力が可能である必要がある。また Plug and Play に対応していないものである必要がある。

SPEAKER

ステレオスピーカ

MICROPHONE

モノラルマイクロフォン

3-7. STRAGES

FLOPPY DISK

3.5 インチ 2 モード (1.44MB/720KB) のフロッピーディスクドライブの一台搭載する。

HARD DISK

540MB 以上の容量のドライブを 2 台搭載する。WesternDigital 社製のドライブを推奨する。

3-8. COMMUNICATION DEVICES

MODEM

データ通信 28.8Kbps 以上の FAX モデムを搭載する。

内蔵のもので I/O ポートとして COM3, COM4 に対応するもの、IRQ 9 を選択可能なものを推奨する。

データ通信 33.8Kbps 以上の Voice/FAX modem を推奨する。この場合コントロールチップとして Rockwell の*****を採用 (またはこれと完全互換のチップを採用している必要がある)

ETHERNET

10BASE-T のポートをもつ Ethernet カードを搭載する。

Plug and Play に対応していないもの、または無効にできるものである必要がある。ISA バスでも PCI バスでも構わないが、BSD/OS で利用できるもの。現在のところ以下のカードが挙げられる。

4. SAMPLE CONFIGURATION

4-1. STANDARD KIOSK

もっとも一般的なキオスク型の端末の参照機に用いられている機器構成は以下のようになっている。

CPU	Pentium90MHz
memory	32MB
BIOS	Phoenix

MotherBoard	Micronics M54Pi
2ndCache	SRAM 256KB
バス	PCI*3,ISA*4
ChipSet	
VideoCard(Chip)	S3 86C968
	(RAMDAC) IBM
HDD	WD2540 *2
FDD	3.5" 2HD/2DD 2mode
PCMCIA	Bullion II
Serial	マザーボード付属
Pararel	マザーボード付属
IDE	マザーボード付属
HDD Case	SI-115A * 2
PC Case	フルタワー。3.5"ベイ*1,5"ベイ*3
電源	300W AT タイプ
CD-ROM	なし
FaxModem	PRAGMATIC H1428
PointingDevice	Genius EasyTrak (3 ボタン トラックボール)
KeyBoard	101 キーボード (スピーカ、マイク内蔵型)
Soundcard	SoundBlaster16 /V
EthernetCard	SMC ElitePlus16T
Display	NANA0 77F
Speaker	キーボード内蔵
Microphone	キーボード内蔵

4-2. DESKTOP

4-3. NOTEBOOK

PICKLES 端末の設計コンセプトを反映させた NOTE BOOK の構成は示す。標準のキオスク型の端末との主な相違点は以下のものである。

- ・ ディスプレイの大きさ
- ・ ディスプレイの解像度
- ・ ハードディスクが一台である点
- ・ Ethernet や FAX MODEM は PCMCIA カードとして供給される点
- ・ 場合によっては FDD は内蔵されない点

PICKLES 仕様の NOTE BOOK のサンプルは次のような構成になっている。

TOSHIBA PORTAGE 620CT

CPU Pentium100MHz

memory	24MB
2ndCache	SRAM 256KB
バス	PCI*3,ISA*4
ChipSet	
VideoChip	Chips&Technology 65548
HDD	E-IDE 1.5GB
FDD	3.5" 2HD/2DD 2mode 外付け
PCMCIA	本体内蔵
Serial	本体内蔵
Pararel	本体内蔵
IDE	本体内蔵
CD-ROM	なし
FaxModem	PCMCIA
PointingDevice	IBMのトラックポイントみたいなやつ
KeyBoard	TOSHIBA 変則キーボード
Soundcard	内蔵 SB 互換
EthernetCard	3C589D PCMCIA 経由
Display	本体内蔵 12.1" TFT 800x600
Speaker	本体内蔵
Microphone	外付け

5. RESOURCE ASSIGNMENT

5-1. IRQs

標準的な IRQ の割り当ては以下の通りである。

0	内蔵タイマ
1	キーボード
2	カスケード
3	COM2(tty01)
4	COM1(tty00)
5	SoundBlaster16
6	FDD
7	Parallel
8	RTC
9	内蔵 MODEM
10	Ethernet
11	PCMCIA
12	PS/2
13	数値演算プロセッサ

- 14 primary IDE
- 15 secondary IDE

ただしこれでは PCMCIA のカードに割り当てる IRQ がないので以下の対策をとる。

1. PS/2 マウスを用いる場合

内蔵モデムを COM2 にするか、外付けモデムを COM2 に接続する
PCMCIA のカードを IRQ12 に割り当てる

2. シリアルマウスを用いる場合

PCMCIA を IRQ12 に割り当てる
シリアルポートのあきが必要なければ、モデムを COM2 に割り当てても良い。

3. 内蔵 MODEM を用いない場合

IRQ 9 を PCMCIA で利用可能

5-2. I/O ADDRESSES

標準的な I/O ポートの割り当ては次の通りである。

0x0000-0x000f	DMA コントローラ
0x0020-0x0021	割り込みコントローラ
0x002e-0x002f	82306
0x0040-0x0043	タイマ
0x0060	キーボード (データ)
0x0061	スピーカ
0x0064	キーボード (コマンド/ステータス)
0x0070(bit7)	NMI 有効
(bit6:0)	RTC
0x0071	RTC (データ)
0x0080-0x008f	DMA ページレジスタ
0x00a0-0x00a1	割り込みコントローラ
0x00b2-0x00b3	APM
0x00c0-0x00de	DMA コントローラ
0x0170-0x0177	Secondary IDE
0x01f0-0x01f7	Primary IDE
0x0200-0x0207	(Joystick)
0x0220-0x022f	(WSS)
0x0270-0x0273	(Plug and Play I/O)
0x0278-0x027b	(Parallel 2)
0x02e8-0x02ef	(COM4)
0x02f8-0x02ff	COM2
0x0330-0x331	MIDI (MPU401)
0x0376	Secondary IDE Command

0x0377	Secondary IDE Status
0x0378-0x37f	パラレルポート
0x0388-0x038b	OPL(WSS)
0x03e0	PCIC (Vadem468)
0x03e8-0x03ef	(COM3)
0x03f0-0x03f5	FDD
0x03f6	Primary IDE Command
0x03f7	FDD Command
0x03f7(bit7)	FDD DISK CHANGE
0x03f7(bit0:6)	Primary IDE Status
0x03f7-0x03ff	COM1

追加デバイスは次の設定を推奨する

3C509	0x250
SMC Elite16	0x300
NE2000	0x340
	0x300
RE2000	0x340
	0x300 (FMV183)

5-3. DMA

標準的な DMA チャンネルの割り当ては次の通りである。

0	
1	SB PRO
2	FDD
3	
4	Cascade
5	SB16
6	
7	

5-4. BAYS

最低一つの 3.5 インチベイと二つの 5 インチベイを必要としている。
 3.5 インチベイにはフロッピーディスクドライブを格納する。
 二つの 5 インチベイにはそれぞれハードディスクケースが格納される。

5-5. SLOTS

6. CONCLUSION

APPENDIX A. HOW TO BUY A NEW TERMINAL

APPENDIX B. OPTIONAL DEVICES

B-1. VIDEO INPUT

画像の入力装置として、Connectix 社の QCAM を利用する。これは parallel ポートに接続するタイプのカメラである。

B-2. IrDA

IrDA の入出力を行なうために、シリアル接続型の IrDA インタフェースを用いる。

B-3. MIDI

SoundBlaster 搭載の MIDI インタフェースを利用する。

B-4. BARCODE READER

キーボード接続型、シリアル接続型のインタフェースユニットを利用する。

B-5. MAGNATOCARD READER

キーボード接続型、シリアル接続型のインタフェースユニットを利用する。