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

木本雅彦[†,☆] 大野浩之[††]

昨今，末端のネットワークでもひととおりのサーバを独自に運用する事例が増えており，このような自立運用されたネットワークの管理モデルの確立が求められている．特にその中の機器構成管理に着目すると，サーバから利用者端末までに利用できる，OS やアプリケーションを含めた統一環境の構築が望ましい．著者らは統一環境を実現する際の要件を考察し，その技術的解決法として「共通の利用者環境の提供」と「ホストに依存した情報と共有できる情報との明確な分離」の 2 つをあげる．これを FreeBSD 上で実現するものとして，前者についてはパッケージ集である POPS を開発し，後者についてはファイルシステム上での構成技術を実装した．本論文では，両者の設計と実装，および開発と運用経験から得られた知見について述べる．

Constructing Common Environment for Independent Networks

Masahiko Kimoto[†,☆] and Hiroyuki Ohno[††]

There are many small scale networks which are using their own servers recently. We need to establish the administration model for those networks. Especially, to improve configuration management, providing common environment including OS and applications is recommended. We introduce two techniques, development of the common environment and clarification of information which depends on each hosts. In this paper, we describe the design and implementation of those techniques and consideration from experiences of the development and the operation.

1. はじめに

インターネットの利用形態は，この数年間で量と質という 2 つの側面から大きく変化している．

量の側面については，普及率の増加から見てとれる．総務省が発行している情報通信白書[1] によると，平成 13 年末における日本国内のインターネット利用者数は 5,593 万人（対前年比 18.8%増）と推計され，人口普及率は 44.0%となっている．この資料で注目するべきは，世帯・企業・事業所でのインターネット普及率である．企業普及率は平成 13 年末で 97.6%であり，ほぼすべ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 2 つは，ほぼ同じ曲線を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 13 年ではそれぞれ 60.5%と 68.0%になっている．つまりこの 4 年間では，家庭と事業所という小規模な組織での普及率が顕著な増加を見せている．

質の側面では，利用形態の多様化があげられる．1 つはアクセス網の多様化である．1998 年に CATV によるインターネット接続が始まって後，ADSL や FTTH などの複数の高速回線が選択できるようになった[2]．同時にアプリケーションも多様化しており，いわゆる P2P（Peer To Peer）アプリケーションや，ネットワークゲーム，VoIP（Voice over IP）など多様なアプリケーションを利用者が求めている．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どうしが直接通信するモデルを用いているという特徴がある．また，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．たとえば家庭の 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すでに登場している．

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さら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 の通信が行われる．そのうえで，セキュリティ

† 東京工業大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† 独立行政法人通信総合研究所情報通信部門非常時通信グループ
Emergency Communications Group, Information and Network Systems Division, Communications Research Laboratory
☆ 現在，株式会社創夢
Presently with SOUM Corporation

モデルや管理モデルを確立する必要がある．

しかし TMN[3] などの従来の管理モデルはその仕様が大きすぎることから，こういった末端の小規模なネットワークの管理にはそのままでは導入できない．管理者の技術力は期待できないし，運用コストも制限されるからである．小規模なネットワークの管理を容易で安全かつ低コストに行うためには，たとえば安全性が保証された統一環境を簡単かつ大量に作成する技術などが必要になる．

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かかえるものを探すと，研究室や SOHO のように，ひととお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あげられる．このようなネットワークを自立運用ネットワークと呼称するとすると，自立運用された小規模なネットワークの運用技術が今後のネットワーク運用に欠かせないといえる．

このような位置付けの中で，著者らは小規模なネットワークの運用技術を確立し自立運用ネットワークの普及を目指している[4]．

本論文では，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．

まず最初に自立運用ネットワークでの OS の構成管理に必要な要件をあげ，その要件を満たす技術的解決方法として,「共通の利用者環境の提供」と「ホストに依存した情報と共有できる情報の明確な分離」とあげる．次にこれらを FreeBSD 上で実現する方法について述べ，実装し運用した経験に基づいて評価と今後の展開について議論する．

2. 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，たと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，理想的にはすべてのアプリケーションがつねに最新のものに追従されているべきであるが，現実的にはまった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．むろん，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あ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すべ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いっせい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．

さらに付け加えると，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，上記にあ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ることがないため，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[5]．これは現在では Linux☆1でいうところの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/OS☆2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わた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[6] をはじ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 と一言でい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，KAME☆3による先進的な IPv6 の実装を持つことや，NetBSD☆4では多数のアーキテクチャに対応していて組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 FreeBSD☆5を採用することとした．以下では FreeBSD 上で上記の解決手段を実現する方法について述べる．

☆1 http://www.linux.org
☆2 http://www.bsdi.com
☆3 http://www.kame.net
☆4 http://www.netbsd.org
☆5 http://www.freebsd.org

3. 共通の利用者環境の提供

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する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[7] や APT[8] 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ていたり，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てパッチが正常に適用できなかったりといった原因がある．対して package はコンパイル済み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検査が厳しく，本来問題なく動作する程度の差異しかないにもかかわらず，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．

3.2 POPS 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よ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[9),10)]．POPS をインストールするだけでひととお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ある．しかしまずは，著者ら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) アプリケーションどうし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のが適当である．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 人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 1** 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．いったん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．いい換えれば，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．

表 1 アプリケーションの利用傾向
Table 1 Usage of applications.

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

(2) については，すでに述べたように 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．さらにいうと，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．とはいえ，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．したがって，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とることにした．

(4) については，バイナリパッケージを配布する際

には，つね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 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600 MB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8,900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9 GB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-ROM1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3 GB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 500 MHzと24倍速CD-ROMの組合せのシステムで約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) 通常どおりpkg_addを実行すればよいもの．
(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，たとえば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．

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」「すでに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．

さらに，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[11]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．

4. 情報の明確な分類と分離

4.1 情報の分離の方針

すでに述べたように，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替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，たと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．

たとえば/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．そのほかについては，/varはその名称が示すとお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

表 2 情報の分類
Table 2 Classification of the information.

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

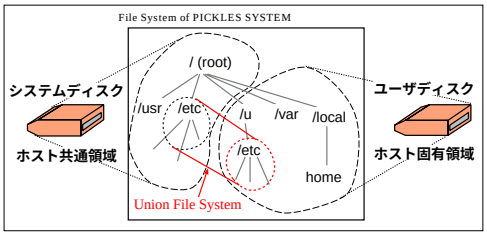

図 1 ファイルシステムの構成
Fig. 1 Structure of the filesystem.

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**表 2** になる．

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と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りないため，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．たとえば/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（**図 1**）．

次の課題はユーザディスクの発見方法である．すでに述べたように，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する．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 を読み込めるようになった．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，たとえば `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 を再度読み込む．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う必要がある．続いて/etc/rc の残りの起動手順を続行する．

5. 評価と考察

5.1 他の機構との比較

多数の計算機に同一の環境を構築する機構としては，Linux 用の LUCIE [12] や SUN の JumpStart [13] などがある．どちらも LAN に接続された初期状態の計算機に，ネットワーク経由で自動的にシステムをインストールする機構であり，クラスタを実装する場合などに有用である．しかし POPS とその元になった PICKLES SYSTEM では，導入作業場所自体がネットワーク的に疎結合な状況を主に想定しており，この場合クラスタリング技術だけでは解決できない．

また，これらのクラスタ環境を構築する技術においても，更新するべき箇所と上書きしてはいけないホスト固有の情報を切り分けなければならないという問題は存在しており，本論文で述べたような情報の切り分けという設計方針と共存することでより効果的な運用が実現できる．

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ときにはおよそ 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実行さ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[14]．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[15] 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われた．

POPS は実用的な環境の提供を目指してはいるが，だれ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う必要がある．たと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．

5.3 技術的考察

5.3.1 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えなかった理由から，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，依存関係の矛盾をなくし，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．

5.3.2 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う．通常の FreeBSD の使いかたでは，ports を最新にしたうえで 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，たとえば ghostscript が日本語版 5.50 からバージョン 6 に更新されたときのように，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，定期的に全体の入換えを行う必要があり，POPS を用いることでこの作業を容易に行うことができる．そのうえでアプリケーションをつね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いえる．

5.3.3 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元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われ，繁雑なものであった．その後の版では少しずつの修正にとどまっているが，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．

5.3.4 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もの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．さらに，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 ができあがる．いうまでもなく，使うコンパイラによっても異なるバイナリができあがるであろう．

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．

5.3.5 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[16] 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．

6. おわりに

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うる．

本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，たとえば他の 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さら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ン的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 5,400 人）に POPS をアナウンスしたところ，その後の 4 週間での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」で，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．最後に POPS に関する情報は，以下の URL から入手できる．
http://www.ohnolab.org/~kimoto/freebsd/pops.html

謝辞 FreePICKLES 開発メーリングリストのメンバには，POPS の開発とその過程の議論において，多くの助言と助力をいただいた．特に高知工科大学の菊池豊助教授には，POPS の改良方針と論文執筆の過程で多くの助言をいただいた．また，富士通研究所の富田憲範氏には POPS のカスタマイズに関する助言をいただいた．ここに感謝する．

参 考 文 献

1) 総務省：平成 14 年度版情報通信白書．http://www.johotsusintokei.soumu.go.jp/whitepaper/ja/h14/index.html
2) 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
3) ITU-T: Overview of TMN Recommendations, M.3000 (2000).
4) 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 (1997).
5) 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 (1996).
6) 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，Vol.18, No.6, pp.27–42, ソフトウェア科学会 (2001).
7) Barnes, D.: RPM How/To. http://www.redhat.com/support/wpapers/rpm-howto.pdf
8) Silva, G.N.: APT HOWTO. http://www.debian.org/doc/manuals/apt-howto/index.en.html
9) 木本雅彦：POPS: Package Of the PackageS—誰でも簡単マイパッケージ集，*FreeBSD PRESS*, No.10, pp.96–99, 毎日コミュニケーションズ (2002).
10) 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 (2002).
11) Rivest, R.: *The MD5 Message-Digest Algorithm*, RFC 1321 (1992).
12) 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 (2003).
13) Snevely, R.: JumpStartTM: NIS と sysidcfg (1999). http://www.sun.co.jp/blueprints/1099/jumpstart.pdf
14) 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 (1998).
15) 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 (2001).
16) Mewburn, L.: The Design and Implementation of the NetBSD rc.d system, *Usenix Annual Technical Conference*, USENIX (2001).

(平成 15 年 5 月 6 日受付)
(平成 15 年 10 月 16 日採録)

木本　雅彦（正会員）

1995 年東京工業大学情報科学科卒業．2002 年同大学院数理・計算科学専攻博士課程満期退学．東京芸術大学芸術情報センター非常勤講師を経て，現在，株式会社創夢に勤務．管理運用面に着目した OS 構築手法の研究開発を専門とする．日本ソフトウェア科学会，電子情報通信学会，ACM 等の会員．

大野　浩之（正会員）

1994 年東京工業大学大学院情報理工学研究科数理・計算科学専攻講師．1999 年より通信総合研究所通信システム部非常時通信研究室長（現，情報通信部門非常時通信グループリーダー）．1980 年代後半より新しい情報提供機構の研究開発に従事．近年は，情報通信分野と危機管理分野の境界領域を活動場所に定め，大規模災害時におけるコミュニケーション支援や，電子政府の情報セキュリティ確保等に強く興味を持つ．WIDE プロジェクトボードメンバー．ITU-T SG17 Q.10 ラポータ，新情報セキュリティ技術研究会技術部会長ほか．博士（理学）．