オープンソースソフトウェアを活用した IPv4/v6 共存システムの実装と性能評価

岡田　正*　　安藤健人**　　山本恭平***

Implementation and Performance Evaluation of an IPv4/v6 Interconnect System Using Open Source Software

Tadashi OKADA, Kento ANDO and Kyouhei YAMAMOTO

We implement an IPv4/v6 interconnect system in low specification hardware with open source software (OSS) as IPv6 network environment spread. The interconnect system is composed by adopting an IPv6-to-IPv4 transport relay translator (TRT), and using OSS such as FreeBSD, Jail, faithd and BIND. The performance of the system is evaluated by a viewpoint of a mutual packet transformation of IPv4 and IPv6 and a durability when flash memory is used. About the packet transformation, it is confirmed to operate normally by measuring packets in the system at a high load. In addition, the system durability that replaced a hard disk with a flash memory is evaluated by a continuous writing examination.

Key Words: IPv6, IPv4, Open Source Software, Packet Transformation, Flash Memory

１．はじめに

我々の研究室は，オープンソースソフトウェア（OSS）[1]を活用することで，安全・安定な情報システムを安価に構築するための研究を行っている．地域支援の一環で構築した「津山瓦版」[2]，学生向け情報を提示する「学内掲示板」[3,4]など，実用的な情報システムを実現してきた．これらのシステムでは，必要最小限の機能のみを効率よく実装し，低機能のハードウェア上でも安全に運用できるよう工夫している．本論文では，同様の考え方で構築した IPv4/v6 共存システムを述べる．

本論文で取り上げるネットワーク専用機器は，IPv4 と IPv6 とのネットワークを相互接続して共存させるシステムである．これは，近年のインターネットの急激な普及に伴い，IPv4 アドレスの枯渇が現実問題となり[5,6]，IPv6 ネットワークが徐々に拡大していくなかで，既存の IPv4 資産を有効活用することができる IPv4/v6 共存システムが必要と考えたからである．

ネットワーク専用機器の開発にあたっては，OSS や適切な性能のコンピュータを用いることで，低コスト化と安定化を図っている．さらに，ネットワーク専用機器に安定性・省エネルギー性を実現する技術を実装し，高信頼性を実現することも行っている．今回は，低消費電力化・低発熱化と障害発生時の迅速な対応を目的に，フラッシュメモリを導入したときの耐久性を検討した．

本論文では，IPv6 の概要と IPv4/v6 共存システムについて２章で説明する．次にシステム評価として，３章で IPv4/v6 共存システムのパケット交換時の性能について，４章でフラッシュメモリ導入の耐久性について，それぞれの目的・方法と結果について述べる．

２．IPv4/v6 共存技術とシステム実装

２．１　アドレス枯渇と IPv6

IP（Internet Protocol）は，経路選択や交換制御を行うための基礎を担い，TCP/IP ベースのネットワークを実現する上で，最も重要なプロトコルである．永らく 32 ビット長の IP アドレス（IPv6 が策定され

原稿受付　平成２６年９月１日

*情報工学科
**専攻科電子・情報システム工学専攻平成２５年度修了生
***専攻科電子・情報システム工学専攻

たので，区別するため IPv4 と呼ばれる）が使われていた．しかし，インターネットの急激な拡大に伴い，アドレス数の不足が予想されたため，新しい IP として 128 ビット長の IPv6 が策定された[7]．

IP アドレスの枯渇に関しては，2010 年 2 月 3 日，世界各地域に IP アドレスを分配する IANA（Internet Assigned Numbers Authority）は，IPv4 アドレスの在庫を全て払い出したと発表した[5]．また，翌年の 2011 年 4 月 15 日に，アジア太平洋地域のアドレスを管理する APNIC（Asia-Pacific Network Information Center）が，IPv4 アドレスの在庫が底を尽きたとしている[6]．在庫がなくなったからといって，直ちに IPv4 が使えなくなったり，IPv6 ネットワークに移行するわけではない．しかし，IPv4 空間の中に IPv6 空間が作られ，徐々に IPv4 空間が消えていくと予想される．

IPv6 ではアドレス長を 128 ビットにすることで，2 の 128 乗という事実上無限といえるアドレスが存在する．さらに，IPv6 はただ単にアドレス総数を増やすだけでなく，IP というプロトコル自体に下記のようなさまざまな機能が追加されている[8]．

・IPSec の標準搭載によるセキュリティ向上
・IP アドレス自動構成機能による利用者・管理者ともに負担軽減可能
・個々の端末にグローバルアドレスを割り当てることで P2P（Peer to Peer）通信が可能
・マルチキャスト方式による情報配信の実現
・アドレス構造の厳密な階層化により，インターネットバックボーンでの効率的なアドレス集約可能
・IPv4 ヘッダよりも簡素化された固定長の IPv6 ヘッダにより，ルータやスイッチが効率よく IPv6 パケットを処理可能

2．2　IPv6 と IPv4 との共存技術

IPv4 と IPv6 は IP アドレス体系，IP ヘッダの違いにより互換性がなく，直接通信することができない．そこで，IPv6 共存システムを IPv4/IPv6 ネットワーク境界に配置することで，そのシステムを通過するパケットに対して何らかの処理を加え，IPv4/IPv6 間の通信を実現することになる．その処理は IPv4/v6 共存システムの種類によって異なり，次に 3 種類の共存技術が存在する[8]．

(1)デュアルスタック

デュアルスタックとは，単一の端末に対して IPv4/IPv6 の両方に対応させ，どちらのネットワークにも接続を可能にする．IPv4 と IPv6 の両方を有効にしておき，IPv4 で通信要求があれば IPv4 で応答し，IPv6 で通信要求があれば IPv6 で応答する仕組みである．IPv6 ネットワークと IPv4 ネットワークの境界に位置して，トンネリングやプロトコル変換を行うデバイスは，必ずデュアルスタックデバイスである．

(2)トンネリング

IPv4 ネットワークを経由して IPv6 の通信を行う仕組みである．IPv6 ネットワークや IPv6 ホスト同士が直接つながっていない場合に使用される．パケットを送信する際，トンネルの入口において IPv4 ヘッダが付加されることで，本来 IPv4 ネットワークで扱うことのできない IPv6 パケットを使用することができる．トンネルの出口に到達したパケットは IPv4 ヘッダを外され，IPv6 ヘッダ情報をもとにルーティングされる．そのため，ユーザ拠点と ISP（Internet Services Provider）をつなぐ回線上には，IPv4 パケットだけが流れることになる．また，この逆の IPv6 ネットワークを経由して IPv4 の通信を行うことも可能である．

(3)トランスレータ

トランスレータは，IPv4/IPv6 パケットを相互変換することで IPv4/IPv6 間での通信を確立する．トランスレータには代表的な方式としてプロキシ方式と TRT（Transport Relay Translator）方式があり，それぞれ変換する部分が異なる．

プロキシ方式では，アプリケーションレベルでの変換を行うので，従来のプロキシサーバと基本的に動作は同じで，IPv4/IPv6 ネットワーク境界に配置する．例えば IPv6 アドレスを設定したインタフェースから宛先ネットワークが IPv4 のパケットを受信すると，プロキシ型変換サーバが自身のインタフェースに設定された IPv4 アドレスを使用して代わりに宛先へ問い合わせを行い，宛先から応答が返ってくると送信元へそれを転送する．この方法では，プロキシ方式のトランスレータ上で，あらかじめ IPv4 アドレスと IPv6 アドレスを対応づけておく必要がある．

一方，TRT 方式は，トランスポート層レベルで変換する方式である．宛先 IPv4 アドレスを IPv6 アドレスに写像させることで，IPv4 アドレスと IPv6 アドレスを対応づける．そのため，プロキシ方式と異なり，IPv4 アドレスと IPv6 アドレスをあらかじめ対応づけておく必要がない．しかし，この方法はコネクション型の TCP 通信に限られてしまうという欠点がある．

2．3　IPv4/v6 共存システムの実装

今回構築した IPv4/v6 共存システムには，トランスレータの TRT 方式を採用した．この理由としては，IPv4 端末と IPv6 対応端末がお互いにプロトコルの違いを意識せず通信できることに利点があると考えたからである．デュアルスタックやトンネリングでは，IPv4 端末は IPv4 端末と，IPv6 端末は IPv6

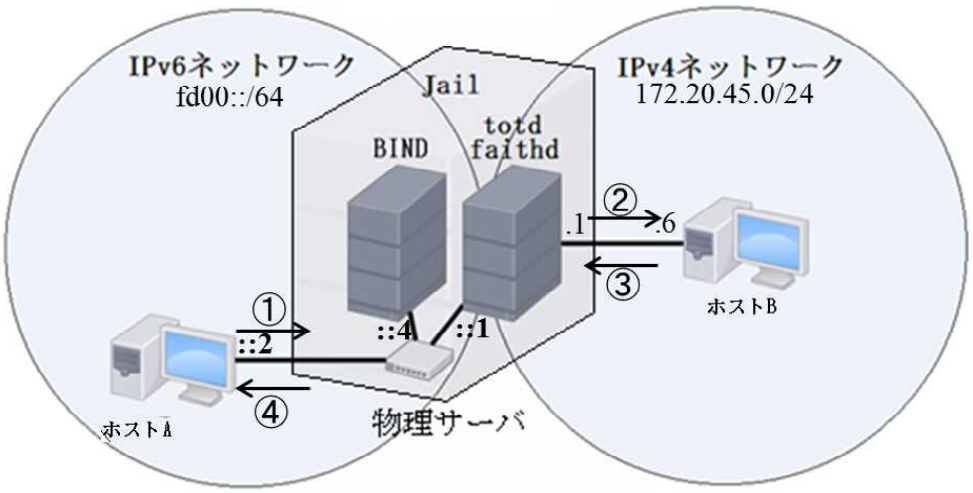

図 1　IPv4/v6 共存システムの構成（性能評価用情報を含む）

端末としか通信することができない．そのため，例えば IPv6 に移行すると既存の IPv4 サービスを利用することができない．TRT 方式トランスレータであれば，IPv6 へ移行したとしても，IPv6 と IPv4 のサービスの両方を利用することができるため採用した．

本システムは，図 1（この図には 3 章で使う性能評価用の情報が含まれている）に示すように OSS である FreeBSD・faithd・BIND・Jail を用いて構成されている．OSS は，再頒布の自由，ソースコードの公開，個人やグループに対する差別の禁止などの特徴があり [1]，また使用許諾料を必要としない．そのため，OSS を用いて本システムを構築することで，システム構築費用を削減でき，また今回構築したシステムを別の環境で，著作権を気にせず再現することも可能になる．これにより，地域の企業等に対するシステムの提供も容易になると考える．

TRT 方式トランスレータは，faithd と totd で構成され，BIND と連携して動作する [9]．faithd は，FreeBSD にも標準搭載されているトランスレータ型のデーモンである．あらかじめルーティングテーブル上に，宛先 IP アドレスとして後述する faith プレフィックスと，出力インタフェースとして faithd デーモンを対応付けたエントリを登録しておく．これで faithd が IPv6 ホストから送信されたパケットを代理で受け取る．

その後 IP プロトコル変換を行い，宛先となっている IPv4 ホストへ転送する [9]．トランスポート層レベルでの変換を行うので，運用では利用するポートごとに設定が必要となる．faithd では，トランスポート層プロトコルとして TCP を使用するアプリケーションでなければ IP プロトコルの変換を行うことができない．また，IPv6 ネットワークに属する端末から通信を開始した場合に限り，パケットの送受信を行うことができる．

また，totd はプロキシ DNS とも呼ばれ，IPv6 ホストが DNS サーバに問い合わせしたとき，その問い合わせをプロキシ DNS サーバが受け取り，DNS サーバへ問い合わせを転送する．そして応答が IPv4 アドレスであれば，IPv6 アドレスへと変換して問い合わせ元に転送する [10]．

IPv4 アドレスに 96 ビットの faith プレフィックスを組み合わせることで，128 ビットの IPv6 アドレスを形成する．そして，宛先 IP アドレスのプレフィックスが，faithd プレフィックスのパケットを IP プロトコルの変換対象とする．

また，今回のシステムで利用する DNS サーバには BIND[11] を用いた．BIND は最も普及している DNS サーバのソフトウェアで，ドメイン名（コンピュータを識別する名称）と IP アドレスを対応させるアプリケーション層のプロトコルである．今回構築した IPv4/v6 共存システムにおいて，DNS サーバは Proxy DNS から受け取った DNS 問い合わせ要求（URL）を IP アドレスへ対応させ，応答する．

しかし，totd と BIND は，どちらのサービスも同じポート番号で待機するので，同一サーバ上に実装することができない [12]．そのため，仮想化システムを導入することで，この問題を解決した．仮想化システムとは，物理的なサーバ上に論理的なサーバを動作させるための仕組みである．今回は仮想化システムの方式として FreeBSD 上の Jail[13] を導入し，faithd と totd および BIND を同一サーバ上で実装した．

Jail は，FreeBSD 上に実装された仮想化システムである．Jail は，ファイルシステムやカーネルへのアクセスが制限されたプロセスの集合であり，Jail を使用することで，FreeBSD 内に本体とは独立して動作する別の FreeBSD システムを構築運用することができる [14]．プロセス自体はホスト上のカーネルで動作するため，ホストと同じ速度で動作する．

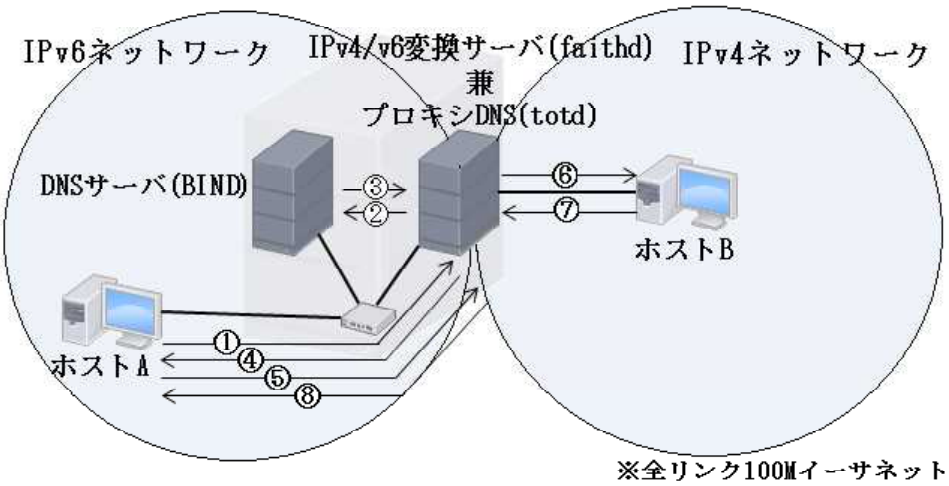

図２　TRT 方式 IPv4/v6 システムの動作

さらに，Jail 内で完結しているため，セキュリティが高く，管理も簡単である．

これにより，totd と BIND を同一ハードウェア上に実装すると同時に，稼働するサーバ数の削減によるランニングコストの削減，また IPv4/v6 共存システムのためのハードウェア調達コストの削減を図ることができると考える．

３．IPv4/v6 共存システムの性能評価

３．１　性能評価の目的と動作

本システムは Jail を用いることで，IPv4/v6 共存機能と DNS 機能を統合し，一つのサーバ上に実装した．構築に用いたサーバのハードウェア仕様を表１に示す．今回使用したハードウェアは性能の低いものであり，このハードウェア仕様で正常にサービスが提供できれば，ハードウェアを調達するときの指標が得られる．そこで，動作解析とパケット数測定を行い，問題なく動作しているかどうかの評価を行った．

表１　IPv4/v6 共存システムのハードウェア仕様

CPU	Intel Celeron(R) CPU E1200 @1.60GHz
メモリ	Transcend DDR2-667 512MB
HDD	日立 Deskstar80GB Serial ATA II300
NIC	ASUS P5GC-MX/1333 （10/100M） INTEL 945GC+ICH7 BUFFALO LGY-PCI-TXD（ 10/100M） PCI2.0

まず，図２により TRT 方式 IPv4/v6 共存システムの動作を述べる．

① ホスト A は，入力された URL から IPv6 アドレスを取得するため，プロキシ DNS として動作しているサーバへ問い合わせを行う．

② 問い合わせを受けたサーバは，DNS サーバへ問い合わせを転送する．

③ DNS サーバは AAAA レコード（IPv6 情報）があればそれを，なければ A レコード（IPv4 情報）を返す（今回 DNS サーバには A レコードのみ設定）．

④ プロキシ DNS は受け取った応答が IPv4 情報なので，あらかじめ設定した 96 ビットの faith プレフィックスと 32 ビットの IPv4 アドレスを組み合わせ 128 ビットの IPv6 アドレスを形成し，ホスト A へ転送する．

⑤ IPv6 アドレスを受け取ったホスト A は，その受け取ったアドレスと自身のインタフェースに割り当てられた IPv6 アドレスを比較し，別ネットワークに所属していることがわかるので，デフォルトゲートウェイである IPv4/v6 変換サーバへデータ（HTTP 要求）を送信する．その際受けとったアドレスを宛先 IP アドレスとして送信する．

⑥ IPv4/v6 変換サーバは，データを受け取るとルーティング処理を行う．ルーティングテーブルを参照し，宛先 IP アドレスが faith プレフィックス宛のデータを受け取ると仮想インタフェースである faithd0 に転送する．

⑦ faithd がそのデータを受け取るとプロトコルを変換し，宛先 IP アドレスに faith プレフィックスを取り除いた IPv4 アドレスをセットして転送する．

⑧ HTTP 要求を受け取った Web サーバは応答を返す．

⑨ IPv4/v6 変換サーバはホスト A へ中継する．

３．２　検証の結果と考察

今回の検証は，IPv4/v6 共存システムに最も負荷がかかる状況を作り，その際パケット数をカウント

し，通信が正常に行えているか確認する（検証に用いた構成は，前掲の図１参照）．

IPv6 の独自割り当て領域セグメント(fd00::/64)に属するホスト A(::2)から，IPv4 の学内アドレス空間(172.20.45.0/24)に配置されたホスト B(172.20.45.6)へと，パケットを 100Mbps で転送し，その際 IPFW[15]を用いて，共存システム(::1 & 172.20.45.1)への入出力パケット数を計測する．測定するのは図１の①から④の部分の４か所である．これら４か所の数値を比較することで，IPv4/v6 共存システムにて，パケットロスは起こるのか，また起こるとすればどれぐらいなのか測定できる．

また，IPv4/v6 共存システムの動作より，システムに対して最も負荷がかかるのは，単位時間当たりの変換対象となるパケットが最も多いこと，並びに帯域をすべて使い切るときであると考えた．変換対象となるパケットが最も多いのは，MTU(Maximum Transmission Unit)が最小の時である．IPv6 では IPv4 と異なり MTU の最小サイズが 1280 バイトと定められている．そのため MTU=1280 バイトで検証を行った．全リンクとも 100M イーサネットで接続されているため，100Mbps の帯域を使い切るようにデータ通信を行った．

全リンクとも MTU を 1280 バイトに設定し，計測を行った結果を図３に示す．

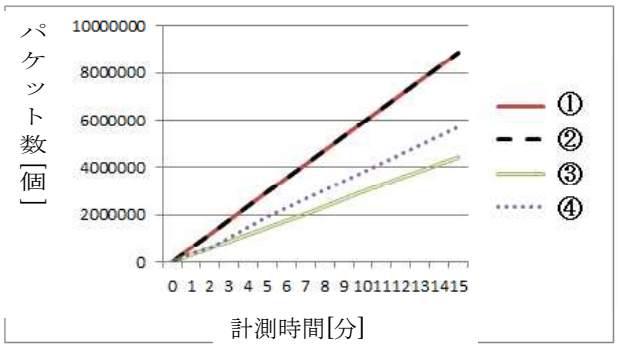

図３　パケット数－計測時間の関係(MTU=1280)

図３より，パケット数と時間は比例関係にある．ここで IPv4/v6 共存システムに対する入力の帯域実効値を考える．15 分測定した際の IPv4/v6 共存システムが受信するパケット数は，8865911 個であった．そのため，１秒当たりの平均パケット数は 8865911/15/60 ＝ 9851 個となる．さらに，一つのフレームのサイズは，MTU が 1280 バイトであり，それにイーサネットヘッダとトレーラが付加されるので，1280+14+4 ＝ 1298 バイトとなる．このフレームが 9851 個あるので，総データ量は 9851 × 1298 ＝ 12786598 バイト＝ 102292784 ビット＝ 97.0M ビットとなる．これより IPv4/v6 共存システムに対する入力の帯域実効値は平均 97.0Mbps となり，100Mbps の帯域をほぼ使い切るようにデータ通信を行えていることがわかる．

次に，各部の通信フローをパケットキャプチャにて調べたところ，①で 11，②で 11，③で 6，④で 7 となった．ここで，①の地点ではパケットの損失は起こっていない（もし起こっていても IPv4/v6 共存システムは関与していない）と仮定し，図３の測定結果を，①のパケット数が 11 になるように正規化すると，表２のような結果が得られる．

表２　MTU ＝ 1280 の時の通信フロー(11 パケットで正規化)

		パケットの測定箇所			
		①	②	③	④
計測時間[分]	0.25	11	10.99651	5.50449	7.07334
	0.5	11	10.99877	5.50232	7.07313
	1	11	10.99792	5.50546	7.07792
	2	11	10.99970	5.50177	5.50288
	5	11	10.99860	5.50257	7.07303
	7	11	10.99888	5.50216	7.07258
	10	11	10.99873	5.50244	7.07305
	15	11	10.99744	5.50431	7.07296

表２より 15 分の場合，②の地点では，0.02 %の減少，③の地点では，8.3 %の減少，④の地点では 1.04 %の増加という結果が得られた．この結果より，表１のハードウェア仕様でもシステムが正常に動作していると考える．しかし，戻りパケット，つまり TCP の確認応答の数に誤差がでた．これは，TCP では，全てのパケットに対する確認応答を送信するのではなく，ある程度まとめた範囲を指定し，確認応答を返している．その確認応答を送り返す条件は以下の通りであり，このためにパケット数のばらつきが発生したと考えられる．

1) 最後に確認応答を送ってから一定の個数パケットを受信した
2) 最後に確認応答を送ってから一定の時間 T が経過した

４．フラッシュメモリによる運用の耐久性

４．１　フラッシュメモリ導入の目的と対象

IPv4/v6 共存システムに対して，フラッシュメモリの導入を検討する．これにより，低消費電力化・低発熱化が図れる．さらにフラッシュメモリとして USB メモリを採用し，あらかじめ別の USB メモリにバックアップを作成しておけば，知識のない人でもディスクデバイスの取り替え作業が容易にできる．

今回開発した IPv4/v6 共存システムのように，常

時接続して運用するネットワーク機器は，耐久性が問題である．ハードウェアの再利用や安価なハードウェアで構築することができても，平均故障間隔が短ければ実際に運用することはできない．コンピュータ内部のパーツの故障率について，CPU：44%，メモリ：29%，ハードディスクドライブ（HDD）：16%，マザーボード：9%，電源：2%というデータがある[16]．

このうち，簡単に交換可能で，ハードウェアの再利用の場面で重要な，HDD の故障率について検討する．2 万 5 千台の一般消費者向け HDD を，24 時間稼働させた時の HDD の生存率と経過年数の関係が調査されている[16]．経過年数とともに HDD の生存率は低下し，４年後には生存率が約 80 %，６年後には生存率が 50 %になると予想されている．

そのため，HDD の代わりのディスクデバイスとしてフラッシュメモリを導入したとき，実用的な耐久性が得られるかどうか評価した．フラッシュメモリを IPv4/v6 共存システムに導入したとき，まず書き換え回数の制約を検討しなければならない．フラッシュメモリはその原理上，書き込みや消去の際，絶縁体に高電圧をかけ電子つまりデータの格納，取り出しを行う[17]．また，書き込みや消去を繰り返すと，酸化膜が次第に劣化し電子の格納ができなくなる．そのため，書き換え回数に制限がある．

また，フラッシュメモリは，書き込み／読み出しを「ページ」と呼ばれる単位で行なう一方，書き換えは，複数のページをまとめた「ブロック」という単位で行なう[18]．これらのサイズは一般的にページが 8KB，ブロックがページを 64 個集約した 512KB とされている．あるブロック内の 1 ページの内容を書き換えるときは，ブロックの全内容を制御チップ内の作業用メモリに読み出して当該ページを更新し，作業用メモリの内容を別のブロックに書き出し，元のブロック全体を消去する．例えば，１ビットを上書きしても，ページ単位でなく書き換えをブロック全体で行わざるを得ない．

このように耐久性に不安を抱えるフラッシュメモリを，IPv4/v6 共存システムに導入することが可能かどうか評価を行う．今回の検証では，フラッシュメモリとして安価で入手できる USB メモリと対象とする．また，近年 SSD（Solid State Drive）の価格も低下してきており，容量の少ないものであれば HDD と同程度の価格で入手できるため，候補の一つと考え同様の検証を行う．

4．2　IPv4/v6 共存システムのディクスアクセス

IPv4/v6 共存システムにフラッシュメモリを導入するため，このシステムの単位時間当たりの書き込み回数と書き込み容量を測定する．また，フラッシュメモリの書き込み回数と書き込み容量を測定し比較することで，ディスクデバイスとしてフラッシュメモリを導入できるかどうか判断できると考えた．

IPv4/v6 共存システムのディスクアクセスを測定する方法として iostat を用いた．iostat は，１秒当たりの読み込み回数や書き込み回数，平均 I/O サイズなどの I/O に関する統計情報を取得することができるコマンドである．IPv4/v6 共存システムにおいて，crontab を用いて iostat を実行することで，１秒に１回統計情報を取得し，それを 72 時間行った．その結果，下記の値が得られた．

平均書き込み回数：4.61 回/s
平均書き込み容量：58.72KB/s

この結果を１日当たりに変換すると，

平均書き込み回数：398304 回/日
平均書き込み容量：4924.8MB/日＝ 4.81GB/日

となる．

4．3　フラッシュメモリの性能測定

フラッシュメモリの耐久性評価のために，実際に書き込むことで確認を行う．フラッシュメモリとして，8GB の USB メモリ，16GB の SSD を使った．IPv4/v6 共存システムのディスクアクセスは，一度システムを構築してしまうと，そのほとんどがログの書き込みであることが予測される．このことを考慮し，プログラムを作成した．

作成したプログラムは，評価対象となるデバイス上に１つのファイルを生成し，そのファイルにランダムアクセスで，ランダムな値を書き込んでいく．空き容量がなくなったら，そのファイルへ上書きしていき，これをエラーが出るまで繰り返す．そして，フラッシュメモリに書き込んだ容量，経過時間，SSD の容量分書き込んだ回数を，別のディスクデバイスへ保存する．この動作を行うプログラムを，およそ 100 行の C 言語により作成した．

エラーが生じたときまでの，書き込み全容量と，フラッシュメモリ全容量分書き込んだ回数を，表３に示す†．

表３　フラッシュメモリのエラーまでのディスクアクセス

	USB メモリ	SSD
書き込み全容量［GB］	12319.45	258688.73
メモリ容量分書き込んだ回数［回］	1727	14024

フラッシュメモリは同じブロックに書き込みを続

† SSD に関しては，原稿提出時までにエラーが発生せず，紀要発行までに表３と本文の数値を修正した．

けると,そのブロックのみ壊れてしまうことがある．しかし，SSD また近年の USB メモリでは，ウェアレベリングが実装されており，プログラムで同じ場所に書き込んだとしても，自動的に書き込み頻度が低い場所に割り当てを行う．例えば，空き容量 1GB の SSD に，512MB のファイルを書き込んで消すのを 4 回繰り返したとしても，SSD の前半 512MB に 4 回書き込まれるのではなくて，全体の 1GB に 2 回書き込まれることになる．そのため，今回の検証で得られた SSD の容量分書き込んだ回数は，１ブロックへの書き込み回数と同程度であると考えられる．

４．４　フラッシュメモリ耐久性に対する考察

各ベンダはフラッシュメモリへの書き込みアルゴリズムに独自の工夫を凝らし長寿命化を図っているので，実際の書き込み量を知ることは困難である．このため，最悪の場合を想定し，一つの目安として評価する．また，書き込みデータ量と書き込み回数の２つから寿命を予測する．

(1)書き込みデータ量からの推定

４．２より，4.61 回／秒の書き込みがあり，すべてが異なるブロックに書き込まれるとする．また，書き換えによるブロック単位の操作であり，ブロックサイズは 512KB とする．

・1 秒間の書き込み量は，512 × 4.61 ≒ 2361KB
・1 日 24 時間電源を入れているとしたときの書き込み量は，2361KB × 3600 × 24=194.5GB

なので,

・USB メモリの書き込み量が表３の値になる日数は，12319.45/194.5=63.3 日
・SSD の書き込み量が表３の値になる日数は，258688.73/194.5=1330.0 日

(2)書き込み回数からの推定

すべてのデータが異なるブロックへの書き込みであると仮定して，単位時間の書き込み数から寿命を推定する．また，書き換えによるブロック単位の操作であり，ブロックサイズは 512KB とする．

・１秒間の書き込み回数は，4.61 回
・１日 24 時間電源を入れているとしたときの書き込み回数は，4.61 × 3600 × 24=398304 回
・8GB の USB メモリの総ブロック数は 8192/0.512 =16000 個で，１年当たり１ブロックへの平均書き込み回数は，398304/16000 × 365=9086.3 回/年
・16GB の SSD の総ブロック数は 16384/0.512 =32000 個で，１年当たり１ブロックへの平均書き込み回数は，398304/32000 × 365=4513.2 回/年

なので,

・USB メモリの１ブロックへの平均書き込み回数が表３の値になるのは，1727/9086.3 ＝ 0.190 年＝ 69.4 日
・SSD の１ブロックへの平均書き込み回数が表３の値になるのは，14024/4513.2 ＝ 3.107 年＝ 1134 日

書き込みデータ量と書き込み回数ともに，ほぼ同じ寿命を推定できている．この結果から，USB メモリの２ヶ月程度の寿命では，長期利用の共存システム導入するのは難しい．しかし，システムの停止期間を短くするため，システム情報を USB メモリに事前に書き込んでおき，短時間で置き換えためには利用可能である．

SSD に関しては，３年以上は使えると予想できるので，HDD との正確な比較は困難なものの，実用レベルの耐久性を持っている考えられる．消費電力の低減効果と価格の問題を考慮して，共存システムに対して適応することは可能である．

５．あとがき

本研究では OSS と低機能なコンピュータを活用し，Jail 環境を導入した TRT 方式の IPv4/v6 共存システムを構築した．実装したシステムに対して最も負荷のかかる状態で，各インタフェースに対する入出力パケットを測定した．その結果を比較することで低機能なコンピュータ上でも，問題ない動作をすることを示すことができた．

また，コンピュータ利用の耐久性に関する問題を解決するために，フラッシュメモリ導入の検討と書き込み実験を通した検証を行った．IPv4/v6 共存システムのディスクアクセス量と，フラッシュメモリに対して書き込みを行った結果をもとに，書き込みデータ量と書き込み回数から寿命を推定した．USB メモリを短時間でのシステム復旧に，SSD を HDD との置き換えに，それぞれ利用できることが予想できる．

IPv6 ネットワークの拡大に伴い，IPv4 と IPv6 とを共存させる必要性が増大するなかで，共存専用システムを簡単に安価に実現できた．本システムを実際に運用するためには，周辺部分を含めたセキュリティの検討が残っており，今後の課題としたい．

参　考　文　献

1）The Open Source Initiative：http://opensource.org/

2）田渕哲也，佐野真澄，岡田正：Web サイト公開支援システムの構築とオープンソースの効果，津山工業高等専門学校紀要 第 48 号，pp.55-60（2006）.

3）宮岡樹，衣笠昭弘，岡田正：Web ベース業務効率化システムのオープンソースソフトウェアによる構築，津山工業高等

専門学校紀要 第 46 号，pp.59-66（2004）.
4）福田大賀，井上靖久，岡田正：学内掲示板システムの更新，津山工業高等専門学校紀要 第 52 号，pp.73-77（2010）.
5）Available Pool of Unallocated IPv4 Internet Addresses Now Completely Emptied：http://www.icann.org/en/news/press/releases/release-03feb11-en.pdf
6）APNIC IPv4 Address Pool Reaches Final /8：http://www.apnic.net/ publications/news/2011/final-8
7）Internet Protocol, Version 6（IPv6）Specification：https://www.ietf.org/rfc/rfc2460.txt
8）Gene, 松田千賀：Cisco CCNP ROUTE テキスト：日経 BP 社，（2011）.
9）KAME Project：http://www.kame.net/~suz/freebsd-ipv6-config-guide.txt
10）The FreeBSD Project：http://www.freebsd.org/cgi/ports.cgi?query=totd& stype =all
11）Internet Systems Consortium：http://www.isc.org/downloads/bind/
12）増田康人，長橋賢吾，有賀征爾：使って学ぶ IPv6，ASCII，pp.129-143（2002）.
13）FreeBSD Handbook：http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/jails.html
14）佐々木宣文，後藤大地，佐藤広生：実践 FreeBSD サーバ構築・運用ガイド，技術評論社，pp.118-158（2012）.
15）Chapter 29. Firewalls:http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html
16）5th USENIX Conference on File and Storage Technologies：https://www.usenix.org/legacy/events/fast07/tech/schroeder/schroeder_html/index.html
17） Backblaze： http://blog.backblaze.com/2013/11/12/how-long-do-disk-drives-last/
18）日経 PC：http://pc.nikkeibp.co.jp/article/basic/20091109/1020283/?rt=nocnt