Live with IPv6 Working Group 2012年度活動報告

櫨山寛章，加藤朗，廣海緑里，中村修，宇多仁，尾上淳，石原知洋，神明達哉

平成 25 年 2 月 23 日

1 はじめに

World IPv6 day，World IPv6 Launch などを通して，世界的に商用ベースでの IPv6 デプロイメントの波が加速している．日本でも 2011 年 5 月前後からホームユーザ向けの商用 IPv6 ネットワークサービスが開始されている．しかしながら，ホームユーザ向け IPv6 サービスは IPv4 サービスのオプション，または IPv4/IPv6 デュアルスタックで提供される場合が多い．また，多くの商用 OS や商用デバイスの基本設定が IPv4/IPv6 デュアルスタックを前提としている．

2011 年 9 月の WIDE プロジェクト研究会で実施した実験では，DNS64, NAT64, DHCPv6 で基本的に構築された IPv6 アドレスのみ提供されるアクセスネットワーク（IPv6 only ネットワーク）を構築し，どの程度の研究会参加者が IPv6 only ネットワークのみで “生活” できるのかを調査した [1, 2]．実験結果として，IPv6 only ネットワークのみを利用し続けた研究会参加者は 20 名程度であった．古い OS の利用者ほど IPv6 only ネットワーク向けの設定に手動対応が必要であり，技術者ではない参加者は自動設定が行われる IPv4 ネットワークの提供を強く希望していた．また，IPv6 の自動設定機能を保持している新しい OS の利用者であっても，VPN や音声チャットなど，IPv6 に対応していないアプリケーションやメールサーバなどを利用するために，IPv4 の提供を希望するというのが現実であった．2011 年 9 月の WIDE プロジェクト研究会の実験では，IPv4 over IPv6 技術の一つである murakami-4RD [3] による IPv4 の提供も行った．実験を通して，IPv4/IPv6 移行期におけるカプセル・アドレス変換技術の実装上，運用上の課題も浮かび上がった．

上記の 2011 年 9 月の WIDE プロジェクト研究会の実験の結果は，IETF 82 Taipei, Global IPv6 Summit Taipei，AWFIT2011，Internet Week 2011，JANOG29 などで発表し，多くの反響を受けた．また，JANOG29 にて 2012 年 3 月 WIDE 研究会における IPv6 実験の協力者を募ったところ，WIDE プロジェクト内外から協力者を得るに至った．そこで，末端利用者レベルでの快適な IPv6 利用環境の構築を目指し，IPv6 only アクセス技術やアドレス変換技術とその問題点，実装上，運用上の TIPS などを WIDE プロジェクト内外で情報共有する分科会として，2012 年 2 月 2 日 Live with IPv6 Working Group を発足した．

2　本年度の活動

2.1　2012 年 3 月 WIDE 合宿での実験

Live with IPv6 Working Group では，2012 年 3 月 WIDE 合宿にて IPv4/IPv6 移行・共存技術を用いたネットワーク構築・運用実験，およびワークショップを開催した．検証のため，対外回線として NTT 東日本のフレッツ光ネクストの PPPoE 方式，IPoE 方式を光電話オプション付きで合宿期間中契約した．

2012 年 3 月 WIDE 合宿では，JANOG29 ライトニングトークセッション [4] などにて実験参加者を広く募った．その結果として，下記の企業の皆様に協力していただいた：

- JPIX，NEC アクセステクニカ：464XLAT の試作機の提供
- IIJ，NTT 東日本，Internet MultiFeed ：フレッツ網を用いた murakami-4RD の試作機の提供
- NTT アドバンステクノロジー：F5 BigIP による HTTP ロードバランサーと NAT64 機能の提供，pivot3 を用いたトラフィック計測用のサーバとストレージの提供
- 富士通，富士通コンピュータテクノロジーズ，富士通九州ネットワークテクノロジーズ：SA46T 試作機 3 種類の提供
- コナミデジタルエンターテイメント：独自拡張 STUN による NAT 検証ツールによる試験

このほか，合宿ネットワーク構築チーム (camp-net) にて bind および unbound を用いた DNS64，linuxnat64 を用いた NAT64，LISP+VXLAN, L3 mesh による v6only ネットワーク環境を用意した．合宿ネットワーク全体のレイヤ 2 トポロジ図を図 1 に，PPPoE 方式下のレイヤ 3 トポロジを図 2 に，IPoE 方式下のレイヤ 3 トポロジ図を 3 に示す．また，コナミデジタルエンターテイメントの提供による STUN を用いた検証トポロジを図 4 に示す．

実験では，昼食時，夕食時に提供するネットワークの種別を変更して参加者に提供した．また，参加者に先入観を持たせないために，ESSID は利用している技術が特定できない名前で提供を行った．表 1 は提供した ESSID とネットワークの対応表である．また，図 5 は参加者の接続推移である．実験では，多くの参加者が IPv4 接続性のあるネットワークを探し，輻輳の発生を感じると別の IPv4 ネットワークを探して移るという傾向が観測された．また，20 名から 60 名程度の参加者が IPv6 only ネットワークに接続しており，その多くは DHCPv6 client を持つ Windows 7 および Mac OS Lion ユーザであった．IPv6 only network への常時接続者数は 2011 年 9 月合宿では最大 20 名であったため，IPv6 only 環境でも不都合なく過ごせるユーザが増加したと考えられる．

実験を通して得られた知見としては，

- コナミデジタルエンターテイメント佐藤氏の指摘により，NAT，NAPT 実装は RFC4787 [5] に準拠すべきである．
- MTU ブラックホールはネットワーク機器の MTU の扱い方だけでなく OS 側の MTU の扱い方まで考慮して検証を行わなければならない．

- IPv4/IPv6 移行・共存技術が本格的に普及すると旧来の IPv4 プロトコルに依存した VPN（GRE, PPTP, IPSec など）は利用できなくなる可能性がある．特に GRE トンネルを用いた VPN に関しては 464XLAT および 4RD (MAP-E) では規格上利用できない．IPSec に関しては VPN ゲートウェイ側が IPv6 に対応すれば問題ない．
- IPv4 に依存した OS のネットワーク設定が存在していることが確認された．特に Smart Phone 端末で IPv4 に依存した設定が多く確認された．
- DNS64 の fallback に対応しない不適切な AAAA 応答を返す authoritative DNS サーバの実装が存在することが確認された．実験では unbound の方が bind より FormError に寛容であることも確認された．

これら，2012 年 3 月合宿の検証結果の一部は internet draft [6] および IETF83 v6ops WG での口頭発表（英語） [7]，JANOG30 でのパネルセッション（日本語） [8] で発表した．また，4RD (MAP-E) や 464XLAT に関してはその後 JANOG softwire WG や Interop Tokyo での相互接続性検証が行われており，IETF での標準化も進められている．

表 1: 提供ネットワークと ESSID の対応

時間	sakura	ichigo	momo	nashi	ringo
3/5 昼食後	464XLAT (IPoE)	4RD (IPoE)	Native IPv6 only (IPoE)	N/A	LISP + VXLAN
3/5 夕食後	Native IPv6 only (IPoE)	4RD (IPoE)	Native IPv6 only (IPoE)	N/A	LISP + VXLAN
3/6 昼食後	L3 mesh IPv6 only	SA46T-FA	464XLAT (PPPoE)	Native IPv6 only (IPoE)	LISP + VXLAN
3/6 夕食後	Native IPv6 only (PPPoE)	464XLAT (IPoE)	SA46T-FK	N/A	LISP + VXLAN
3/7 昼食後	Native IPv6 only (PPPoE)	4RD (IPoE)	Native IPv6 only (IPoE)	N/A	LISP + VXLAN
3/7 夕食後	4RD + ULA v6 addr (PPPoE)	4RD (IPoE)	Native IPv6 only (IPoE)	rouge	LISP + VXLAN
3/8 午前	Dual Stack	N/A	N/A	N/A	LISP + VXLAN

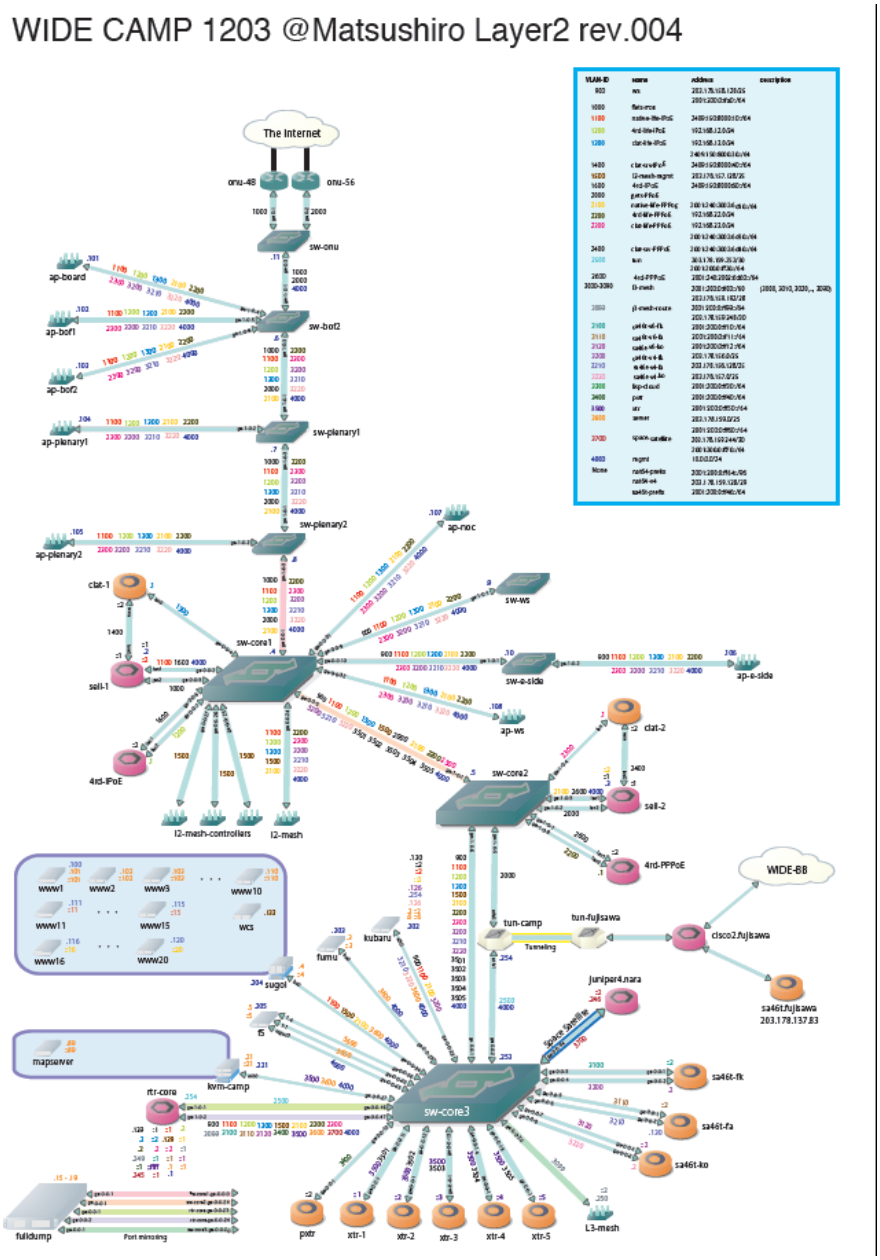

図 1: 合宿ネットワークトポロジ（レイヤ 2）

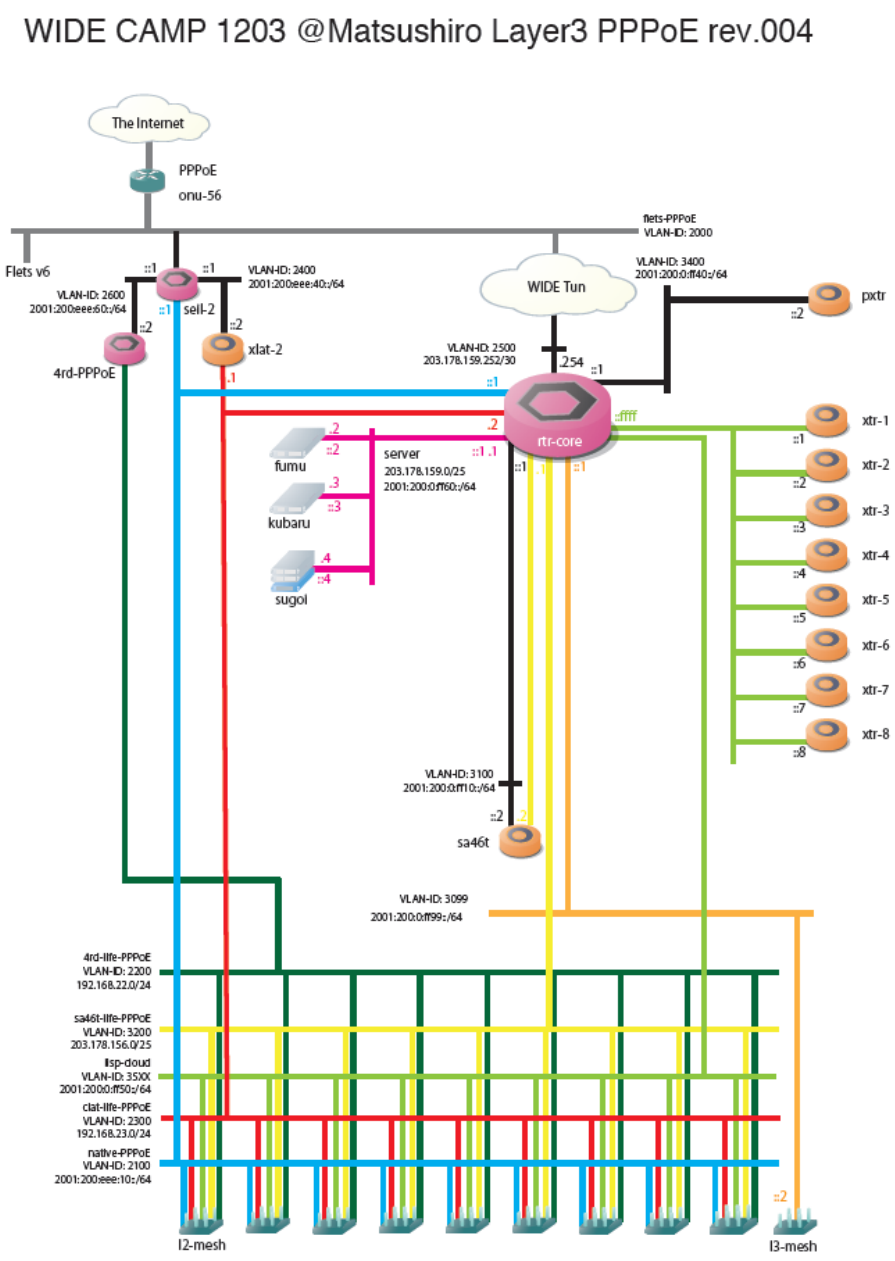

図 2: 合宿ネットワークトポロジ（レイヤ 3 PPPoE 方式）

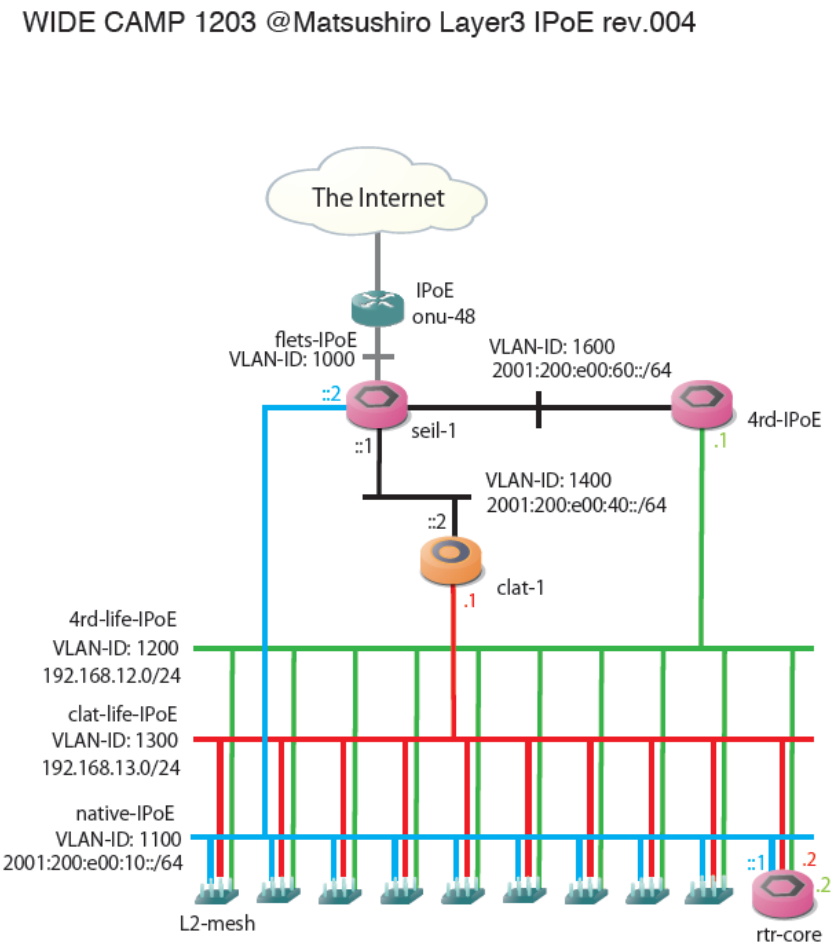

図 3: 合宿ネットワークトポロジ（レイヤ 3 IPoE 方式）

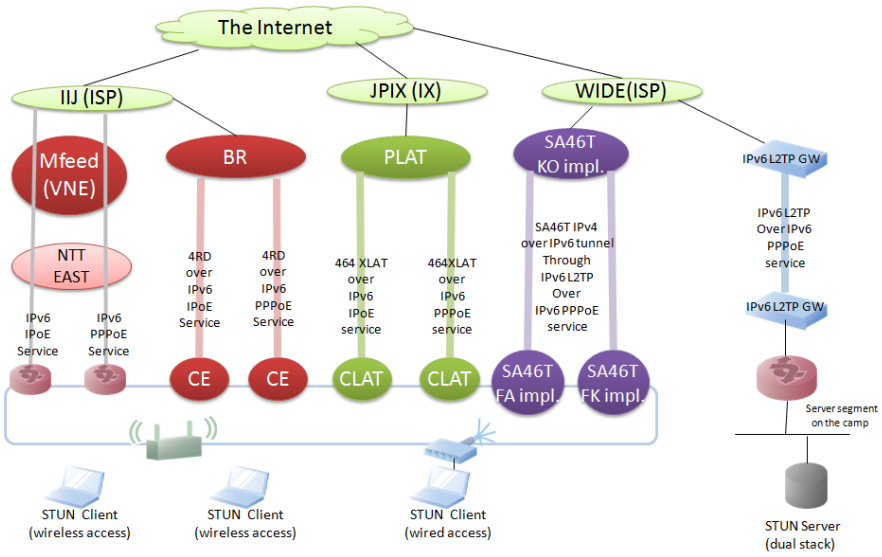

図 4: STUN によるテスト環境

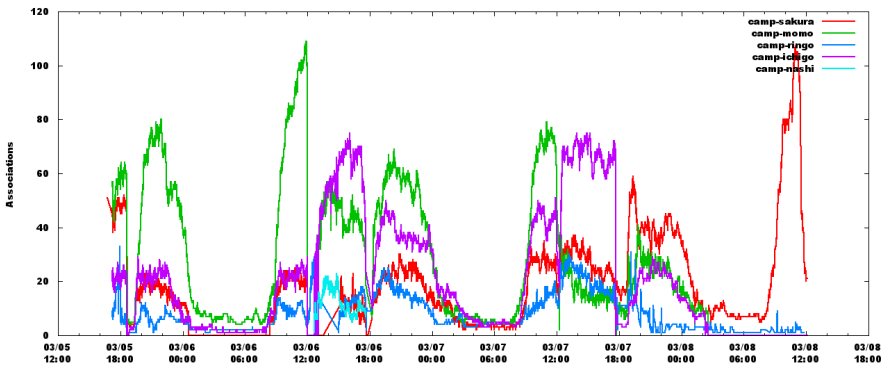

図 5: 参加者の接続推移

2.2　2012 年 9 月 WIDE 合宿での実験

2012 年 3 月 WIDE 合宿に引き続き，2012 年 9 月 WIDE 合宿でも IPv4/IPv6 移行・共存技術を用いたネットワーク構築・運用実験，およびワークショップを開催した．2012 年 9 月 WIDE 合宿では主に DNS64/NAT64 による IPv6 only 環境での OS ごとの振る舞いの差の調査と work around の開発に注力して行った．図 6 は 9 月合宿の実験ネットワークトポロジである．対外線としては，9 月 3 日および 9 月 4 日はフレッツ光ネクスト IPoE 方式で IPv6 オプションのみの/64 プレフィックスが RA によって割当たるサービスを使い，9 月 5 日および 9 月 6 日は光電話オプションを拡充し DHCP-PD を使って /56 プレフィックスが割当たるサービスを利用した．また，サーバのほとんどは StarBED 上に配置し，合宿地のホテルには DHCP-DP クライアントや stateless DHCPv6 サーバを行うための PC ルータと alaxala3630 を配置した程度のシンプルなネットワークトポロジで実験を行った．

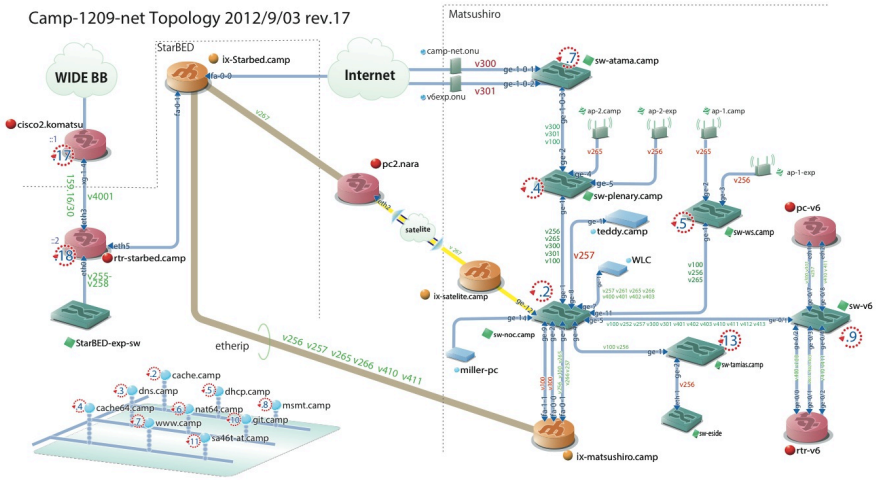

図 6: 2012 年 9 月合宿のトポロジ

2.2.1　Stateless DHCPv6 による DNS 情報の上書き実験

DNS64/NAT64 によるアドレス変換を有効にするためには，DHCP v6 で DNS64 のアドレスをクライアントに広報しなければならない．9 月 3 日および 9 月 4 日の RA による環境では，フレッツ網にある stateless DHCP v6 サーバから，フレッツ網の IPv6 only（AAAA しか返さない）DNS のアドレスがクライアントに広報される形式になっていた．そこで，ローカルに DNS64 のアドレスを広報する stateless DHCP v6 サーバを設置し，DHCP v6 サーバのオプションである priority を最大にすることで，DNS 情報を上書きができるかどうかを試みた．結果としては，上書きはできるものの，OS によって，無線 LAN 初期接続時や DHCP 情報のリフレッシュのタイミングでローカルの DHCP v6 サーバが優先される場合とフレッツ網の DHCPv6 サーバが優先される場合があり，非常に不安定になることが確認された．9 月 5 日以降は DHCP-PD によるアドレス取得に変更した

ため，ローカルの stateless DHCP v6 サーバによる DNS64 アドレスの広報のみとなり，上記の問題は解決された．

2.2.2 IPv6 only ネットワーク下での IPv4 依存性を緩和するための Work around 開発

9 月 5 日からは，ネットワーク設定が IPv4 に依存している OS を IPv6 only 環境でも快適な通信が行えるようにするための work around の開発を行った．Work around の開発は，尾上淳氏，石原知洋氏，神明達哉氏によって行われた．まず，OS によって次のような IPv4 への依存性が 9 月 5 日までの実験を通して確認された．

- **初期のネットワーク接続で DHCPv4 によるアドレス取得と IPv4 アンカーサーバへの到達性確認のための Long Fallback**

  Windows 7, Mac OS X Lion 以降，Android OS, iOS など，多くの OS にて確認されている．

- **Mac OX X Lion および Mountain Lion における TCP Fallback**

  Mac OX X Lion および Mountain Lion では On link Assumption による TCP fallback と，Happy Eyeball のような getaddrinfo の挙動に起因する TCP fallback が確認されている．On link Assumption による TCP fallback は，Mac OS X では IPv4 Link Local Address が付いたインターフェースを経路表の default に入れてしまう．同時に，Mac OS X の Happy Eyeball のような getaddrinfo の挙動ではまず IPv4 を優先し，IPv4 が到達不能であることが確認されたのちに 3 分から 5 分（おそらく ARP テーブルおよび ND テーブルの更新間隔）に基づいて IPv6 を優先する．この挙動は中村修氏がワークショップ中に作成した Mac OS X 上で getaddrinfo を直接操作するサンプルプログラムによって確認されている．また，Object C 経由でアドレス選択するアプリケーション（Safari) 等では，TCP Fallback は軽減されるものの，正常に通信ができるようになるまで，3，4 分かかるという状況になることが確認された．

- **iOS 5 でのネットワーク設定**

  iOS 5 のバージョンや SIM 配布キャリアによって，IPv6 の設定が不十分になる場合や DHCPv4 による IP アドレスの設定のみを優先する挙動，IPv4 アンカーサーバへの通信が確認されないと定期的に設定をリセットする挙動が確認された．

- **DHCP v6 未対応 OS**

  Windows XP や snow leopard 以前の Mac OS X, バージョンの古い iOS や Android OS では DHCP v6 未対応 のため，RA により IPv6 アドレスがインターフェースに割当たったとしても，名前解決が IPv4 に依存している．そのため，実質的に通信が行えないという状況が確認された．また，古い iOS や Android OS では，IPv6 による DNS の手動設定インターフェースが提供されていない場合，そもそも設定できない場合があることが確認された．

上記の問題を緩和するために，次のように Work around を追加していった．

DHCPv4 サーバの設置　まず，単純に IPv4 プライベートアドレスを配布する DHCP v4 サーバをサブネット上に配置して，どの問題が緩和されるのかを検証した．DHCP v4 サーバはゲートウェイ上に設置したが，IPv4 は転送せず，全て破棄する設定にした．結果としては，DHCPv4 のアドレス取得待ちによる Fallback のみ解決され，Windows 7 のみ Fallback が生じない状況となった．その他の Fallback 問題は一向に解決されない状況であった．

Bind 9 による DNS forwarder の設置　次に，Bind 9 を用いた DNS forwarder をサブネット上に設置し，DHCPv4 および DHCPv6 で DNS forwarder のアドレスを配布するように設定した．DNS forwarder の設定には，

```
deny-answer-addresses  0.0.0.0/0; ;
```

と IPv4 による応答に対しては ServFail を応答する設定を投入した．

結果としては，Android での IPv4 設定に依存した問題は解決された．iOS では問題は解決されたものの，定期的にネットワーク設定がリセットされる問題は引き続き生じた．Mac OS X での HappyEyeBall のような getaddrinfo の挙動による TCP fallback 問題は引き続き生じた．Windows XP 等では DNS クエリが全て ServFail となるため通信が実質的に行えない状況となった．

A フィルタの投入　次に，神明氏により Bind 9 に備わっている，AAAA レコードを全てフィルタする “AAAA フィルタ” を A レコードを全てフィルタするように変更した単純な “A フィルタ” が実装され，実装した A フィルタを DNS forwarder 上に設定した．AAAA フィルタを A フィルタに変更する Bind 9.9.2-P1 用の Patch ファイルと A フィルタ付き DNS forwarder の設定ファイルは Appendix に添付する．

結果として，Windows XP, MacOS X variants, iOS および Android で生じていた問題はほぼ解決された．残課題として，1) 外部の DNS の設定・実装の不備による DNS64 の ServFail 問題，2) 一部の OS での，無線 LAN への初期接続における RS (Router Solicitation) 発行タイミングの不備による待ち状態の発生，の 2 つの問題が確認された．問題 1) に関しては，DNS の設定を RFC4074 [9] に準拠するように啓蒙するか，ServFail を受けても A レコードの有無を調査して 64 変換を行う機能を DNS64 上に実装することが work around として考えられる．問題 2）に関しては，RA の発行間隔を 5 秒から 10 秒程度に短くするか，AP で観測されたアソシエーションをトリガーとして RS または RA を発行するように OS やルータの挙動を変更することが work around として考えられる．

上記の実験結果は，実験全体を説明する internet draft [10] と A フィルタに絞った internet draft [11]，廣海らによる IC2012 での口頭発表 [12] にて発表した．

3　KREONET Workshop 2012

前述の WIDE 合宿での実験や，日本での IPv6 普及状況，JAIST，東京大学，慶応大学での IPv6 デプロイメントにおける知見の紹介などを KREONET Workshop 2012 での特別セッションとして KISTI の協力を得て行った．発表者としては，慶應義塾大学 加藤朗，慶

應義塾大学 中村修，JAIST 宇多仁，NAIST 櫨山寛章が参加した．発表内容を KREONET Workshop 2012 の報告書として英文文書にまとめ，再度 WIDE Project technical report として `wide-tr-live-with-ipv6-wg-ipv6-depolyment-in-japan.pdf`(DVD 収録) にまとめた．

4 おわりに

本年度 Live with IPv6 Working Group を発足し，IPv6 移行・共存技術を用いたネットワークの構築・運用に関する知見を WIDE 合宿での実験を通して収集した．今後は，現在 NAIST 内に構築を進めている常設の IPv6 only ネットワークテスト環境や MAP-E テスト環境を通した運用知見の蓄積や，WIDE 合宿での実験ネットワーク構築を通して，IPv6 移行・共存技術の運用知見をより深め，広く公開していく予定である．

参考文献

[1] 櫨山寛章, 上野幸杜, 佐藤弘崇, 石橋尚武, 横石雄大, 山岸祐大, 石原知洋. IPv6 only Network 構築と検証実験. WIDE Technical-Report `http://member.wide.ad.jp/tr/wide-tr-hazeyama-ipv6-only-network-00.pdf`, 2012 年 1 月.

[2] Hiroaki Hazeyama, Yukito Ueno, Hirotaka Sato, Yudai Yamagishi, Takehiro Yokoishi, and Hisatake Ishibashi. How much can we survive on an IPv6 network? - Experience on the IPv6 only connectivity with NAT64/DNS64 at WIDE camp 2011 autumn. In *Proceedings of Asia Workshop on Future Internet Technologies (AWFIT2011)*, November 2011.

[3] T. Murakami, O. Troan, and S. Matsushima. IPv4 Residual Deployment on IPv6 infrastructure - protocol specification. Internet Draft (expired) `http://tools.ietf.org/html/draft-murakami-softwire-4rd-01`, September 2011.

[4] 櫨山寛章. IPv6 only access network 検証実験に参加しませんか？JANOG 29 Meeting ライトニングトーク `http://www.janog.gr.jp/meeting/janog29/program/wide.html`, 2012 年 1 月.

[5] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), January 2007.

[6] Hiroaki Hazeyama, Ruri Hiromi, Tomohiro Ishihara, and Osamu Nakamura. Experiences from IPv6-Only Networks with Transition Technologies in the WIDE Camp Spring 2012. Internet Draft `http://tools.ietf.org/html/draft-hazeyama-widecamp-ipv6-only-experience-01`, March 2012.

[7] Hiroaki Hazeyama, Ruri Hiromi, Tomohiro Ishihara, and Osamu Nakamura. Experiences from IPv6-Only Networks with Transition Technologies in the WIDE

Camp Spring 2012. Presentation at v6ops in IETF 83 Paris http://www.ietf.org/proceedings/83/slides/slides-83-v6ops-0.pdf, March 2012.

[8] 櫨山寛章, 末永洋樹, 川島正伸, 松平直樹, 佐藤良. 「IPv6 時代の IPv4 を考える」〜第二章〜. JANOG 30 Meeting パネルディスカッション http://www.janog.gr.jp/meeting/janog30/program/v64.html, 2012 年 7 月.

[9] Y. Morishita and T. Jinmei. Common Misbehavior Against DNS Queries for IPv6 Addresses. RFC 4074 (Informational), May 2005.

[10] Hiroaki Hazeyama, Ruri Hiromi, Tomohiro Ishihara, and Osamu Nakamura. Experiences from IPv6-Only Networks with Transition Technologies in the WIDE Camp Autumn 2012. Internet Draft http://tools.ietf.org/html/draft-hazeyama-widecamp-ipv6-only-experience-02, October 2012.

[11] Ruri Hiromi, Hiroaki Hazeyama, Atsushi Onoe, and Osamu Nakamura. A workaround for termination of IPv4 network services. Internet Draft (individual submittion) http://tools.ietf.org/html/draft-hiromi-sunset4-termination-ipv4-00, October 2012.

[12] 廣海緑里, 櫨山寛章, 尾上淳, 中村修. 仕様や実装の異なる多数の端末を IPv6 only ネットワークに接続するための課題と考察. インターネットコンファレンス 2012(IC2012), 2012 年 11 月.

A Bind 9 A filter patch

神明氏のご厚意により，2012 年 WIDE 合宿で開発した，Bind 9.9.2-P1 向け A filter patch をここに添付する．利用方法としては，AAAA filter のコンパイルオプションでコンパイルし，添付の `named.conf` のような設定で AAAA フィルタオプションを付けた DNS forwarder として起動させることである．`quey.c` および `message.c` の AAAA フィルタ部分を単純に A フィルタに変更しているため，オプションの付け方は AAAA フィルタと全く同じである．なお，NAIST での実験で DNS64 とこの A filter を同じ bind で動かすと 64 変換での Segmentation Fault が発生することが確認されている．

A.1 Bind 9.9.2-P1 A filter patch

```
diff -ru bind-9.9.2-P1.org/bin/named/query.c bind-9.9.2-P1.a-filter/bin/named/query.c
--- bind-9.9.2-P1.org/bin/named/query.c 2012-10-26 13:50:34.000000000 +0900
+++ bind-9.9.2-P1.a-filter/bin/named/query.c 2013-02-23 19:14:08.000000000 +0900
@@ -1348,7 +1348,7 @@

  if (qtype == dns_rdatatype_a) {
 #ifdef ALLOW_FILTER_AAAA_ON_V4
-isc_boolean_t have_a = ISC_FALSE;
+ isc_boolean_t have_aaaa = ISC_FALSE;
 #endif

  /*
@@ -1390,7 +1390,7 @@
  if (result == ISC_R_SUCCESS) {
  mname = NULL;
 #ifdef ALLOW_FILTER_AAAA_ON_V4
-have_a = ISC_TRUE;
+ have_aaaa = ISC_TRUE;
 #endif
  if (!query_isduplicate(client, fname,
         dns_rdatatype_a, &mname)) {
@@ -1444,7 +1444,7 @@
   * There's an A; check whether we're filtering AAAA
   */
 #ifdef ALLOW_FILTER_AAAA_ON_V4
-if (have_a &&
+ if (have_aaaa &&
       (client->filter_aaaa == dns_v4_aaaa_break_dnssec ||
       (client->filter_aaaa == dns_v4_aaaa_filter &&
        (!WANTDNSSEC(client) || sigrdataset == NULL ||
@@ -5224,6 +5224,7 @@
 #ifdef ALLOW_FILTER_AAAA_ON_V4
 static isc_boolean_t
 is_v4_client(ns_client_t *client) {
+ return (ISC_TRUE);
  if (isc_sockaddr_pf(&client->peeraddr) == AF_INET)
  return (ISC_TRUE);
  if (isc_sockaddr_pf(&client->peeraddr) == AF_INET6 &&
@@ -6756,7 +6757,7 @@

  if (type == dns_rdatatype_any) {
 #ifdef ALLOW_FILTER_AAAA_ON_V4
-isc_boolean_t have_aaaa, have_a, have_sig;
+ isc_boolean_t have_a4, have_aaaa, have_sig;

  /*
   * The filter-aaaa-on-v4 option should
@@ -6765,8 +6766,8 @@
   * even in if it is not in our cache.  This assumption could
```

```
   * be wrong but it is a good bet.
   */
-have_aaaa = ISC_FALSE;
-have_a = !authoritative;
+ have_a4 = ISC_FALSE;
+ have_aaaa = !authoritative;
  have_sig = ISC_FALSE;
 #endif
  /*
@@ -6803,10 +6804,10 @@
   * that AAAAs can be hidden from IPv4 clients.
   */
  if (client->filter_aaaa != dns_v4_aaaa_ok) {
-if (rdataset->type == dns_rdatatype_aaaa)
+ if (rdataset->type == dns_rdatatype_a)
+ have_a4 = ISC_TRUE;
+ else if (rdataset->type == dns_rdatatype_aaaa)
  have_aaaa = ISC_TRUE;
-else if (rdataset->type == dns_rdatatype_a)
-have_a = ISC_TRUE;
  }
 #endif
  if (is_zone && qtype == dns_rdatatype_any &&
@@ -6866,7 +6867,7 @@
  if (client->filter_aaaa == dns_v4_aaaa_break_dnssec)
  client->attributes |= NS_CLIENTATTR_FILTER_AAAA;
  else if (client->filter_aaaa != dns_v4_aaaa_ok &&
- have_aaaa && have_a &&
+  have_a4 && have_aaaa &&
   (!have_sig || !WANTDNSSEC(client)))
    client->attributes |= NS_CLIENTATTR_FILTER_AAAA;
 #endif
@@ -6934,10 +6935,10 @@
       (!WANTDNSSEC(client) || sigrdataset == NULL ||
       !dns_rdataset_isassociated(sigrdataset))))
  {
-if (qtype == dns_rdatatype_aaaa) {
+ if (qtype == dns_rdatatype_a) {
  trdataset = query_newrdataset(client);
  result = dns_db_findrdataset(db, node, version,
-     dns_rdatatype_a, 0,
+      dns_rdatatype_aaaa, 0,
       client->now,
       trdataset, NULL);
  if (dns_rdataset_isassociated(trdataset))
@@ -6977,7 +6978,7 @@
   * if the recursion for the A succeeds.
   */
  result = query_recurse(client,
-dns_rdatatype_a,
+ dns_rdatatype_aaaa,
  client->query.qname,
  NULL, NULL, resuming);
  if (result == ISC_R_SUCCESS) {
@@ -6988,7 +6989,7 @@
  }
  }

-} else if (qtype == dns_rdatatype_a &&
+ } else if (qtype == dns_rdatatype_aaaa &&
     (client->attributes &
      NS_CLIENTATTR_FILTER_AAAA_RC) != 0) {
  client->attributes &=
Only in bind-9.9.2-P1.a-filter/bin/pkcs11: Makefile
Only in bind-9.9.2-P1.a-filter/bin/python: dnssec-checkds.py
Only in bind-9.9.2-P1.a-filter/bin/python: Makefile
Only in bind-9.9.2-P1.a-filter/bin/tests/system/dlz: prereq.sh
```

```
Only in bind-9.9.2-P1.a-filter/bin/tests/system/ecdsa: prereq.sh
Only in bind-9.9.2-P1.a-filter/bin/tests/system/gost: prereq.sh
Only in bind-9.9.2-P1.a-filter/bin/tests/virtual-time: conf.sh
Only in bind-9.9.2-P1.a-filter/bin/tests/virtual-time: Makefile
Only in bind-9.9.2-P1.a-filter/contrib: check-secure-delegation.pl
Only in bind-9.9.2-P1.a-filter/contrib: zone-edit.sh
diff -ru bind-9.9.2-P1.org/lib/dns/message.c bind-9.9.2-P1.a-filter/lib/dns/message.c
--- bind-9.9.2-P1.org/lib/dns/message.c 2012-10-26 13:50:34.000000000 +0900
+++ bind-9.9.2-P1.a-filter/lib/dns/message.c 2013-02-23 18:36:17.000000000 +0900
@@ -1811,14 +1811,14 @@
 norender_rdataset(const dns_rdataset_t *rdataset, unsigned int options)
 {
   switch (rdataset->type) {
-case dns_rdatatype_aaaa:
+ case dns_rdatatype_a:
   if ((options & DNS_MESSAGERENDER_FILTER_AAAA) == 0)
   return (ISC_FALSE);
   break;

   case dns_rdatatype_rrsig:
   if ((options & DNS_MESSAGERENDER_FILTER_AAAA) == 0 ||
-    rdataset->covers != dns_rdatatype_aaaa)
+     rdataset->covers != dns_rdatatype_a)
   return (ISC_FALSE);
   break;

Only in bind-9.9.2-P1.a-filter/lib/dns/tests: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/dns/include/dns: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/dns/include/dst: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/dns/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/dns: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/irs/include/irs: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/irs/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/irs: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/include/isc: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/nls: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/nothreads/include/isc: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/nothreads/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/nothreads: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/unix/include/isc: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/unix/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isc/unix: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isccfg/include/isccfg: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isccfg/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/isccfg: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/samples: Makefile
Only in bind-9.9.2-P1.a-filter/lib/export/samples: Makefile-postinstall
Only in bind-9.9.2-P1.a-filter/lib/irs/include/irs: Makefile
Only in bind-9.9.2-P1.a-filter/lib/irs/include/irs: netdb.h
Only in bind-9.9.2-P1.a-filter/lib/irs/include/irs: platform.h
Only in bind-9.9.2-P1.a-filter/lib/irs/include: Makefile
Only in bind-9.9.2-P1.a-filter/lib/irs: Makefile
Only in bind-9.9.2-P1.a-filter/lib/isc/tests: Makefile
```

A.2 named.conf for DNS forwarder by Bind 9.9.2-P1 A filter patch

```
key "rndc-key" {
        algorithm hmac-md5;
        secret "HXE6UugyLhLiRcHttJ2R7e7rEwnDt6x2pjE38aB3EH8=";
};

controls {
```

```
        inet 127.0.0.1 port 953
        allow { 127.0.0.1; } keys { "rndc-key"; };
};

logging {
        channel remote_log {
                severity info;
                print-category yes;
                print-severity yes;
                syslog local1;
        };
        category resolver { remote_log; };
        category security { remote_log; };
        category queries { remote_log; };
        category default { remote_log; };
};

options {
        directory "/etc/namedb";

allow-recursion { 203.178.136.0/26; 127.0.0.1; ::1; 10.0.0.0/8; 2409:12:6080:102::/64; };
        recursion yes;
        pid-file        "/var/run/named/pid";
        dump-file       "/var/dump/named_dump.db";
listen-on-v6 { any; };
// deny-answer-addresses { 0.0.0.0/0; };
forward only;
forwarders { 2001:200:0:ff10::4 port 53; };
filter-aaaa-on-v4 yes;
filter-aaaa { any; };
/*
dns64 2001:200:0:ff64::/96 {
clients {
::1;
};
mapped { !rfc1918; any; };
suffix ::;
recursive-only yes;
break-dnssec yes;
};
*/
};

acl rfc1918 { 10/8; 192.168/16; 172.16/12; };


zone "." {
type hint;
file "named.root";
};


zone "rpz.camp.wide.ad.jp" {
type master;
file "rpz.camp.wide";
allow-query { any; };
};
```