WIDE Technical-Report in 2014

合宿ネットワークにおけるIPv6 only 化実験

wide-tr-camp1409-v6-experiment-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on WIDE documents, please contact to board@wide.ad.jp

Title: 合宿ネットワークにおける IPv6 only 化実験
Author(s): 櫨山寛章 (hiroa-ha@is.naist.jp), 田川真樹 (tagawa.masaki.td4@is.naist.jp), 石原知洋 (sho@c.u-tokyo.ac.jp), 垣内正年 (masato@itc.naist.jp)
Date: 2014-09-23

合宿ネットワークにおけるIPv6 only 化実験

櫨山 寛章 (hiroa-ha@is.naist.jp)　田川 真樹 (tagawa.masaki.td4@is.naist.jp)
石原 知洋 (sho@c.u-tokyo.ac.jp)　垣内 正年 (masato@itc.naist.jp)

2014 年 9 月 19 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が加速しているが IPv6 の普及はいまだ 6 %程度である．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が，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]．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．また，Android など，OS によっては IPv4 のアドレス割り当てがないと無線 LAN 接続を開始しなかったり，DNS クエリを IPv4 パケットでしか送信できないなどの OS 側での IPv6 対応化の遅れも確認できた．

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 [2] 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．WIDE 合宿での murakami-4RD の実験の後，IETF softwire WG にて MAP-E と名称を変え，MAP-E として JANOG や Interop Tokyo などでの相互接続試験が行われた．また，2014 年 9 月現在では MAP-E に関するインターネットドラフトは RFC として発行を待つ段階（Publication Requested）のステータスになっており [3]，一部の商用ルータではすでに MAP-E が利用できる．

2011 年 9 月の WIDE 研究会以降，2013 年 3 月の WIDE プロジェクトの研究会の実験ネットワークまで，オープンソースによる 464 技術の検証や DNS64/NAT64 の検証を毎回実施している．それらの実験結果の一部は奈良先端科学技術大学院大学の Georgescu らによる 464 技術検証フレームワークの提案と実践的評価 [4] や，live-with-ipv6 WG の担当ボードである慶応義塾大学の中村 修 教授らによる DNS64/NAT64 環境下で IPv4 Address Literal を扱うための特殊な DNS に関するインターネットドラフト [5] として発表が行われている．

これらの過去 3 年間の実験成果と成果展開を踏まえ，DNS64/NAT64 を用いたネットワーク構築方法の参照手法をまとめることを目的とし，2014 年 9 月の WIDE 研究会の合宿ネットワークチームでは，2011 年 9 月の WIDE 研究会と同様の運用方針で IPv6 only 環境を提供し，IPv6 対応や HappyEyeball の挙動の把握と 2011 年 3 月からの 3 年間での差分の洗い出しの解析に重点を置いてネットワークを設計運用することとした．具体的には，live-with-ipv6 WG と NTT Advanced Technology（NTT-AT）の協力により 商用仮想ルータ，商用仮想アプライアンスを用いた DNS64/NAT64 環境

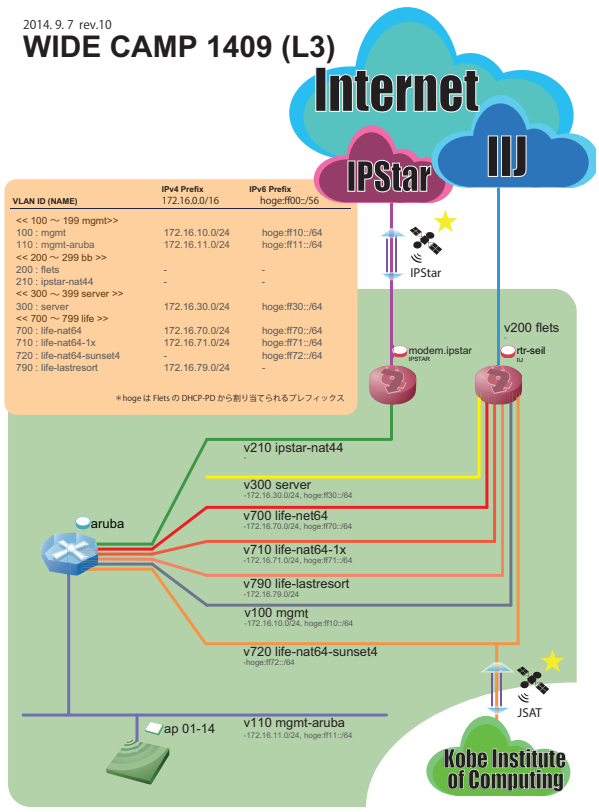

図 1: 合宿ネットワーク（レイヤ 3 トポロジー）

を提供し HappyEyeball 対策である A Record Filter [6] の商用アプライアンスでの実装方法やその有効性の検証，課題の洗い出しを行った．また，NTT-AT の協力により，IPv6 バックボーンでのコンテンツキャッシュ装置の動作検証も実施した．

さらに，2011 年 9 月の WIDE 研究会ネットワークと同様に，wlanops WG と協力し 802.1x 認証による無線 LAN 認証のデバイス対応の状況の把握や設定，運用知見の収集実験も平行して実施した．以降，本稿では，live-with-ipv6 WG らによる DNS64/NAT64 環境の構築手法と検証実験の詳細について報告する．

2　合宿ネットワークの基本構成情報

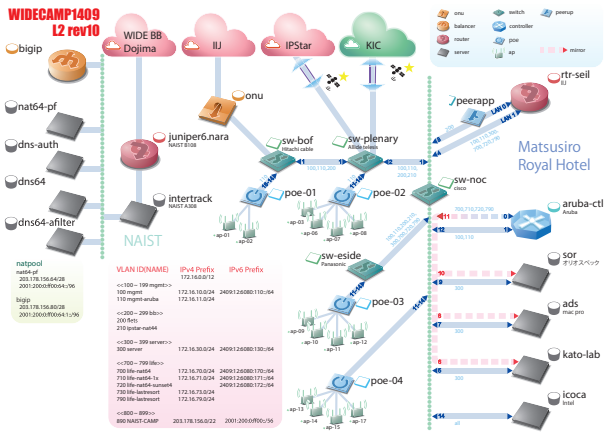

図 2: 合宿ネットワーク（レイヤ 1, 2 トポロジー）

2.1　レイヤ 3 構成

図 1 は合宿ネットワークのレイヤ 3 トポロジーである．ルータは 1 つで，ネットワークごとに VLAN を分割した小規模オフィスの構成である．ルータには IIJ SEIL を用い，フレッツ回線の IPoE 接続と IIJ からの DHCP-PD client として/56 の IPv6 アドレス広報を受け，フレッツ回線の IPv4 PPPoE 接続と NAT44 を構築した．また，IPStar による IPv4 ネットワークも別途構築した．

Aruba によって SSID は次の 5 つを無線 LAN 上で広報した：

- **camp1409-nat64**：DNS64/NAT64 の基本環境．2 日目以降は hidden に設定．
- **camp1409-nat64-1x**：DNS64/NAT64 の基本環境に 802.1x 証明書認証を加えた環境
- **camp1409-nat64-sunset4**：DNS64/NAT64 の基本環境に ProxyARP による sunset4 検証を実施した環境
- **camp1409-ipstar**：IPStar による高遅延 IPv4 only ネットワーク
- **camp1409-lastresort**：SEIL の IPv4 PPPoE 接続と NAT44 設定による低遅延 IPv4 only ネットワーク．hidden に設定し，IPv4 接続の要望のあったユーザへの口頭での通知と，アンケートページにのみ公開．

2.2 レイヤ 1，レイヤ 2 構成

図 2 は，機器の接続図と VLAN 構成図である．ここでは，IPv6 実験に関連するものだけ紹介する．

DNS64，NAT64 は今回，パブリック DNS64/NAT64 サービスを想定し，WIDE Nara NOC に設置した．オープンソース実装として BIND を用いた権威サーバ，DNS64 サーバ，DNS A Record Filter サーバの 3 種類と KVM の仮想マシンとして用意し，NAT64 として OpenBSD Packet Filter の仮想マシンを用意した．また，NTT-AT の協力により，BIG-IP LTM/GTM Virtual Edition on KVM Hypervisor を DNS64/NAT64 として用意した．基本的には合宿期間中は BIG-IP を DNS64/NAT64 を利用した．

PeerApp の UltraBand は SEIL の上流に透過型コンテンツキャッシュとして設置した．VLAN としては ONU と SEIL の間の VLAN 200 に挟み込み，物理的には sw-noc（cisco catalyst 3750）と SEIL の間に挟みこんだ．また，SEIL では DHCP4, DHCP6, DNS Forwarder の設定を行った．他，icoca を NTP サーバや SYSLOG サーバ，BIND による DNS forwarder のバックアップ，SNMP による計測サーバなどとして利用した．合宿地のサーバには基本的には IIJ から delegate された IPv6 によってグローバルアクセスを設定し，ユーザからは DNS の名前解決によってアクセスすることを推奨した．

camp1409-lastresort (vlan 730,790) 以外の IPv4 プライベートアドレスは，ローカルサブネットに閉じ，基本的にグローバルアクセスのない環境である．アクセス側の VLAN では，SEIL が DNS forwarder として DHCP4 で広報され，名前解決のみ IPv4 でも実施できるように設定した．

また，SEIL の仕様により default gateway を広報しない形式で DHCP4 を設定できなかったため，到達性のない IPv4 アドレス（1.1.1.1 や 192.0.2.1，DHCP4 で配布するサブネットでのホストに割り当てないアドレス）を検証環境の修正やユーザからの変更依頼（VPN アプリケーションが利用するアドレスと衝突するなど）により，適宜変更しながら設定した．

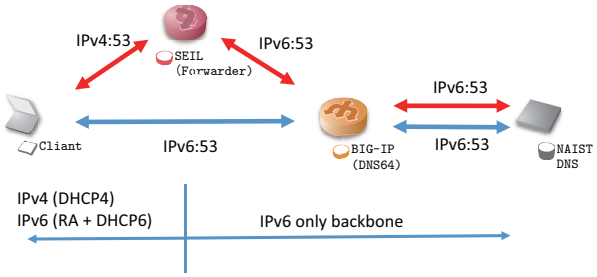

図 3: DNS クエリのダイアグラム（パターン 1）

```
1: when DNS_RESPONSE {
2:   set prefix "2001:200:0:ff00:64:1:"
3:   set rrs [DNS::answer]
4:   if { [DNS::question type] equals "ANY" } {
5:   foreach rr $rrs {
6:     if { [DNS::type $rr] equals "A" } {
7:        log local0. "A Record is \[ $rr \]"
8:        set a_addr [getfield $rr "A" 2]
9:        DNS::answer remove $rr
10:       DNS::answer insert "[DNS::question name]. 111
          [DNS::question class] AAAA $prefix[format %02x
          [lindex [split $a_addr .] 0]][format %02x
          [lindex [split $a_addr .] 1]]:[format %02x
          [lindex [split $a_addr .] 2]][format %02x
          [lindex [split $a_addr .] 3]]"
11:    }
12:  }
13: }
```

図 4: BIG-IP iRule を用いた A Record Filter (rev.0)

3 DNS64/NAT64 環境の構築手法とその検証

ここでは，BIG-IP を用いた DNS64/NAT64 と BIG-IP での A Record Filter の実装方法の検証を実施した．

3.1 構築手法その 1

まずは，bind での A Record Filter パッチの挙動をもとにしながら，BIG-IP の A Record Filter での実装方法と，その効果を検証することを主眼として，実験を行った．

3.1.1 構成

図 3 がパターン 1 の構成図である．パターン 1 では IPv4 の DNS クエリは SEIL による DNS fowarder

```
;; QUESTION SECTION:
;www.ntt-at.co.jp. IN ANY
;; ANSWER SECTION:
www.ntt-at.co.jp. 47488 IN A 114.179.21.42
```

図 5: A Record Filter 適用前の DNS Response

```
;; QUESTION SECTION:
;www.ntt-at.co.jp. IN ANY
;; ANSWER SECTION:
www.ntt-at.co.jp. 111 IN AAAA 2001:200:0:ff00:64:1:72b3:152a
```

図 6: A Record Filter 適用後の DNS Response

機能で仲介し，IPv6 による DNS クエリは直接 BIG-IP による DNS64 を参照するように SEIL の Stateless DHCP6 機能でクライアントに通知する形式をとった．

BIG-IP の iRule を用いて A Record Filter を実装するにあたり，図 4 のような，ANY クエリに含まれる A レコードの結果を mapped address に変換するスクリプトを実装した．この A Record Filter 適用前の DNS Response は図 5 のようになり，A Record Filter を適用すると図 6 のように mapped IPv6 に変換した AAAA として返すように iRule を実装した．

次に，図 4 のフィルタでは，question type A で DNS クエリが発行されていた場合，そのままクライアントに A レコードを返してしまう．そのため，question type A のフィルタリングを行うフィルタとして図 7 に示す iRule も実装した．この改良版 A Record Filter の適用結果は図 8 のようになる．

3.1.2 検証結果

検証として図 7 の A Record Filter の状態で正常にインターネットアプリケーションが利用できるかの調査を行った．検証の結果，Facebook の画像が取得出来ない問題が発生した．BIG-IP のログなどを解析した結果，[Browser] IPv4:AAAA Query → [BIG-IP] AAAA Response (Native IPv6 address) → [SEIL] SERVFAIL, と SEIL にて SERVFAIL に変換されてクライアントに返されていることがわかった．SEIL 側を確認すると FormError が出力されていた．SERVFAIL に置き換わるクエリとしては，Akamai のキャッシュサーバであることが確認された．

```
 1: when DNS_RESPONSE {
 2:   set prefix "2001:200:0:ff00:64:1:"
 3:   set rrs [DNS::answer]
 4:   if { [DNS::question type] equals "ANY" } {
 5:   foreach rr $rrs {
 6:     if { [DNS::type $rr] equals "A" } {
 7:        log local0. "A Record is \[ $rr \]"
 8:        set a_addr [getfield $rr "A" 2]
 9:        DNS::answer remove $rr
10:        DNS::answer insert "[DNS::question name]. 111
           [DNS::question class] AAAA $prefix[format %02x
           [lindex [split $a_addr .] 0]][format %02x
           [lindex [split $a_addr .] 1]]:[format %02x
           [lindex [split $a_addr .] 2]][format %02x
           [lindex [split $a_addr .] 3]]"
11:     } elseif { [DNS::question type] equals "A" } {
12:        log local0. "A Record \[ $rrs \] is DELETED. "
           DNS::answer clear
13:     }
14:   }
15: }
```

図 7: BIG-IP iRule を用いた A Record Filter (rev.1)

```
C:\bind>dig @BIG-IP.naist.camp.wide.ad.jp www.ntt-at.co.jp a
; <<>> DiG 9.8.7-W1 <<>> @BIG-IP.naist.camp.wide.ad.jp www.ntt-at.co.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59729
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.ntt-at.co.jp. IN A
;; Query time: 23 msec
;; SERVER: 2001:200:0:ff00::16#53(2001:200:0:ff00::16)
;; WHEN: Wed Sep 10 23:10:13 東京 (標準時) 2014
;; MSG SIZE rcvd: 34
```

図 8: A Record Filter rev.1 適用後の DNS Response

3.2 構築手法その 2

パターン 1 で確認された，SEIL の ParseError や SERVFAIL に書き換える問題の切り分けを行うために，パターン 2 では bind での DNS forwarder に入れ替えて挙動を確認してみることにした．

3.2.1 構成

図 9 がパターン 2 の構成図である．パターン 1 の構成で SEIL で発生する FormError や SERVFAIL が発生する理由が不明瞭だったため，切り分けとしてパターン 1 の fowarder を SEIL から BIND に置き換え，IPv4, IPv6 両方の DNS クエリを BIND に通す形に変更した．

3.2.2 検証結果

パターン 2 での検証の結果 Windows 7 端末でのブラウザ (IE11, Chrome) が「インターネットに接続出来ない」と表示する問題が発生した．tcpdump や BIG-IP 側のログを解析してみると

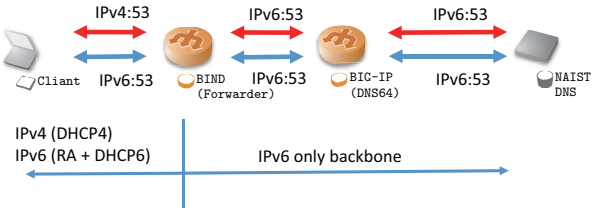

図 9: DNS クエリのダイアグラム（パターン 2）

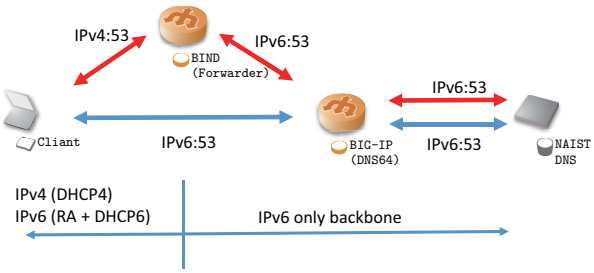

図 10: DNS クエリのダイアグラム（パターン 3）

1: [Browser] IPv4:A Query → [BIG-IP] A Filtered Response → [BIND] SERVFAIL

2: [Browser] IPv6:A Query → [BIG-IP] A Filtered Response → [BIND] SERVFAIL

3: ブラウザアクセス不可

という挙動になっていた．また，問題が発生していないパターン 1 の場合も解析した結果，A クエリ に対し A Record Filter を適用した場合，NOERROR が返ってくればブラウザは AAAA を投げ直すということが判明した．パターン 2 の環境では，BIND の forwarder が NOERROR の Respose を SERVFAIL に書き換えていることが判明した．

3.3 構築手法その 3

パターン 2 の検証結果を踏まえ，IPv6 でのクエリに関しては BIND を通さず直接 BIG-IP にクエリを行うと何が発生するのかを検証した．

3.3.1 構成

```
1: when DNS_RESPONSE {
2:   set prefix "2001:200:0:ff00:64:1:"
3:   set rrs [DNS::answer]
4:   if { [DNS::question type] equals "ANY" } {
5:   foreach rr $rrs {
6:     if { [DNS::type $rr] equals "A" } {
7:        log local0. "A Record is \[ $rr \]"
8:        set a_addr [getfield $rr "A" 2]
9:        DNS::answer remove $rr
10:       DNS::answer insert "[DNS::question name]. 111
          [DNS::question class] AAAA $prefix[format %02x
          [lindex [split $a_addr .] 0]][format %02x
          [lindex [split $a_addr .] 1]]:[format %02x
          [lindex [split $a_addr .] 2]][format %02x
          [lindex [split $a_addr .] 3]]"
11:    } elseif { [DNS::question type] equals "A" } {
12:       log local0. "A Record \[ $rrs \] is DELETED. "
13:       DNS::answer clear
14:       DNS::authority clear
14:    }
16:  }
17: }
```

図 11: BIG-IP iRule を用いた A Record Filter (rev.2)

```
C:\bind>dig @BIG-IP.naist.camp.wide.ad.jp www.ntt-at.co.jp a
; <<>> DiG 9.8.7-W1 <<>> @BIG-IP.naist.camp.wide.ad.jp www.ntt-at.co.jp a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59729
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.ntt-at.co.jp. IN A
;; Query time: 23 msec
;; SERVER: 2001:200:0:ff00::16#53(2001:200:0:ff00::16)
;; WHEN: Wed Sep 10 23:10:13 東京 (標準時) 2014
;; MSG SIZE rcvd: 34
```

図 12: A Record Filter rev.2 適用後の DNS Response

図 10 がパターン 3 の構成図である．ちょうど，パターン 1 の fowarder を SEIL から BIND に置き換えた形である．また，FormError や SERVFAIL をいたずらに発生させないように，図 11 のように A Record Filter では Authority Secition をクリアするルールに変更した．Authoritiy Section をゼロクリアする理由は，さまざまな authoritative DNS に対するクエリを解析した結果平均的に Authoritative section が 0 の場合 FormError が発生しにくいという経験則から実施することとしたため，明確な論拠は今のところない．改変した A Record Filter 適用後の DNS Response は図 12 のようになった．

3.3.2 検証結果

検証の結果，Windows 7 端末でブラウザ接続可能となった．シーケンスとしては，

1. [Browser] IPv6:A Query → [BIG-IP] A Filtered Response → [Browser] Fallback

2. [Browser] IPv6:AAAA Query → [BIG-IP] AAAA Response

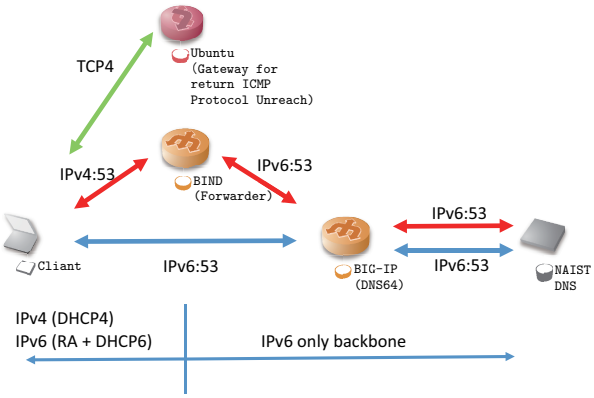

図 13: DNS クエリのダイアグラム（パターン 4）

3. ブラウザアクセス可

パターン 2 で見受けられた IPv4 での DNS クエリは無しという結果になった．この結果から，ブラウザ (OS) と割り当てられた DNS のアドレスによって挙動が異なることが明らかとなった．

また，`mirror.centos.org` が A クエリ に対しては応答可能であるが，AAAA クエリ に対して SERVFAIL を返してくるという現象を確認した．これは，過去の合宿でも確認され，WIDE プロジェクトの森下氏，神明氏らから RFC4074 [7] にて指摘されている不適切な DNS 応答に由来する問題ではないかと推定される．このように、クライアントが最初に AAAA を聞いてきた場合に 権威サーバが AAAA に対して SERVFAIL を返してくると，BIG-IP は DNS64 の仕様 [8] により A (DNS64 AAAA) を投げ直さないため，ブラウザ表示不可となる．

3.4 構築手法その 4

パターン 3 でおおむね解決できたかに見えたが，iOS 7 上で Facebook App Store アプリにおいて，明らかに何かのタイムアウトを待っているフォールバック問題が発生してると，ユーザから指摘を受けた．そのため，この解析とワークアラウンドを live-with-ipv6 WG 有志により実施した．

3.4.1 構成

図 10 が試行錯誤の結果たどり着いたワークアラウンドを組み込んだ，パターン 4 の構成図である．Facebook App Store アプリが送信するパケットダンプを解析したところ，ARP が定期的に流れていることから，ARP のタイムアウトを待ち，IPv6 に fallback する HappyEyeball が Facebook App Store アプリ実装されているのではないかと推定したため，DHCP4 で配布する IPv4 ゲートウェイ（Ubunutu 14.4LTS) で iptables を用いてフィルタを適用し ICMP4 type 3 Destination Unreachable メッセージを通じて到達性がないことをクライアントに通知すれば，IPv6 に即座に fallback するのではないかという推論のもと，検証することとした．

3.4.2 検証結果

ICMP4 type 3 Destination Unreachable メッセージの code を色々試したところ，以下の結果となった．

- **code 0 - Network Unreachable :** 依然としてタイムアウトが発生した．TCPDUMP の結果，Facebook App Store アプリは名前解決せずに Facebook の IPv4 アドレス，ポート番号 443 宛（31.13.17.49:443, 31.13.19.101:443）に TCP4 にて再送を繰り返していることが明らかとなった．
- **code 3 - Port unreachable :** ポート番号 443 に対してフィルタを適用したところ，タイムアウトは発生せず，体感で 1 秒程度で IPv6 にフォールバックし，コンテンツを閲覧できるようになった．
- **code 2 - Protocol unreachable :** Facebook App Store アプリケーションを調査し，ポートごとに設定するのは大変なため,「TCP, UDP, ICMP など各種 Protocol に到達性がない」という code 番号を試してみたところ，Port unreachable と同様の結果を得ることが出来た．
- **code 9 - Communication with Destination Network is Administratively Prohibited :** 「ネットワークポリシーとして許可されていない」という code 番号を試してみたところ，Protocol Unreachable と同様の結果を得ることが出来た．

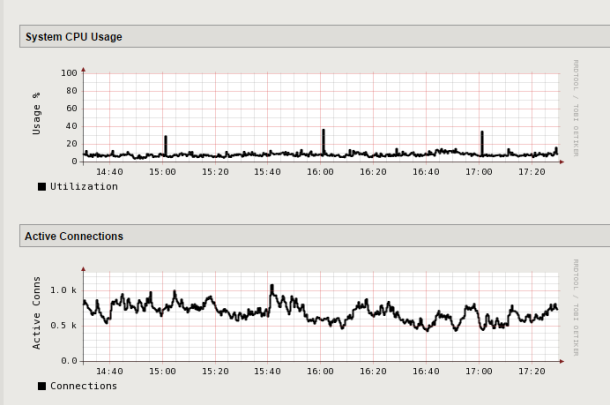

図 14: BIG-IP の CPU 利用率とアクティブコネクション数の推移

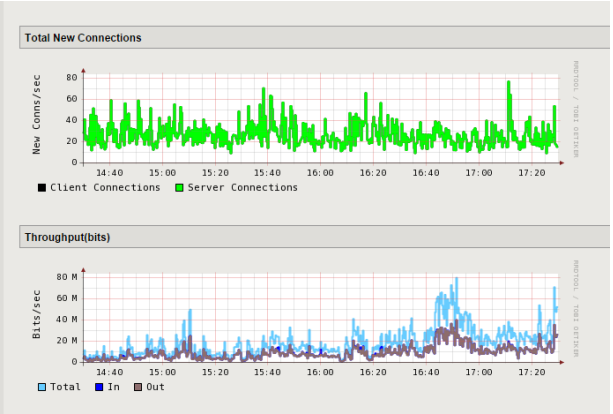

図 15: BIG-IP で処理した新規コネクション数とスループットの推移

3.5 BIG-IP の性能

ここでは，BIG-IP の性能面に関して報告する．今回は，BIG-IP LTM/GTM Virtual Edition on KVM Hypervisor を用いた．ライセンスとしては Throughput 1Gbps License を利用した．

ハードウェアとしては Dell PowerEdge R620, Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz 2 プロセッサ，物理メモリ 64 ギガバイトのサーバを用いた．BIG-IP に対しては仮想 CPU としては 2 コア，メモリは 4 ギガバイト割り当てた．

図 14 は研究会期間中の BIG-IP の CPU 利用率とアクティブコネクション数の計測結果，図 15 は BIG-IP で処理した新規コネクション数の推移とスループットの計測結果である．CPU 利用率は平均 10 %であり，アクティブコネクションの最大数は 1,400 コネクションであった．新規コネクションは最大で 80 コネクション/秒であり，スループットは最大 80Mbps であった．後述の PeerApp UltraBand でパススルーしたトラフィックが最大 17Mbps 程度であったことと，研究会ネットワーク以外からの DNS スキャンなども受け付けていたため，おおむね妥当なスループットであり，性能面で特に問題はなかったといえる．

3.6 考察

今回の実験で商用実装である BIG-IP の機能を用いて A Record Filter を構築できることが明らかとなった．また A Record Filter のルールに関しても，DNS 実装の Parse 方法や Form の Validation の違いにより，クライアント側の DNS Forwarder にて FormError や SERVFAIL に変更されることも明らかとなった．

また，A Record Filter の組み合わせやルール変更だけでは対応できなかったフォールバック問題に対して，ICMP4 type 3 Destination Unreachable によりフォールバックにかかる時間を大幅に短縮できることが明らかとなった．その際，code 3 - Port unreachable よりも code 2 Protocol Unreachable もしくは code 9 - Communication with Destination Network is Administratively Prohibited を返すことが，「各種プロトコルに到達性がない」もしくは「運用上許可されていない」というメッセージであるため，クライアント側でも納得感が出るのではないかという結論になった．live-with-ipv6 WG としての推奨は code 9 - Communication with Destination Network is Administratively Prohibited により IPv4 に到達性がない IPv6 only network であるということを通知するのが DNS64/NAT64 環境としては適切ではないかという結論とした．

この実験結果から，A Record Filter を適用しなくても ICMP4 type 3 code 9 を返すだけでほとんどのフォールバック問題は発生しないもしくはフォールバックにかかる時間を短縮できるのではないか？という議論が新たに浮上した．しかしながら，今回の検証で調査したのは一部の OS やアプリケーションであり，ア

プリケーションによっては普通は使われない code 9 に対応していない可能性もある．

また，A Record Filter は DNS クエリの結果を AAAA レコードだけにするため，A レコードの結果を優先し IPv4 を選択するタイプの HappyEyeball によるフォールバック問題を発生させないメリットがあるが，設定や挟み込み方法が難しいことや DNS Forwarder での FormError, SERVFAIL などを発生させやすいことも，今回の検証で明らかになり，適切な検証無しには A Record Filter は不要と断定することはできない．

そのため，live-with-ipv6 WG では，引き続き奈良先端科学技術大学院大学の DNS64/NAT64 WiFi 実験環境を用いて A Record Filter と ICMP4 type 3 destination unreach の組み合わせに関する追検証を実施し，もっとも設定，運用コストの少ない方法を検討することにした．その報告は WIDE 2014 年 12 月研究会にて発表する予定である．

アンケート結果やヘルプデスクへの苦情・要望から，VPN アプリケーションや Skype，Dropbox など，通常の IPv4 ネットワークで多くのユーザがビジネスなどで利用しているアプリケーションが利用できないため，合宿参加者から不満の声が上がり，Skype や Polycom RealPressence などを用いてビデオカンファレンスを行う必要のあったユーザから低遅延の IPv4 インターネットを使いたいという要望がヘルプデスクにあがってきた．そのため，合宿ネットワークチームでバックアップ用に用意しておいた NAT44 をもちいた IPv4 ネットワーク環境を要望のあったユーザに提供した．

また，中には IPv6 でトンネルを張って IPv4 を手元に持ってくるユーザ，そもそも合宿ネットワークに接続せずに，会場に存在した au-WiFi などの公衆無線 LAN サービスやテザリングで IPv4 からのインターネット接続を使うユーザが多く存在した．また，3DS, PS Vita ではプラットフォームが IPv4 対応していないことや，Final Fantasy XIV などのゲームではクライアント側が IPv4 対応していない，DNS64/NAT64 の変換による遅延に耐えられないため，ゲームユーザから不満の声が上がった．

このため，キラーアプリケーションと呼ばれる skype, polycom, dropbox などのアプリケーションや 3DS や PSVita などのゲームプラットフォームやオンラインゲームサービスが IPv6 対応することが望まれる．

4 Mac OS X における IPv4 Link Local Assamption に由来するフォールバック問題への ProxyARP による対策手法の検証

前節 3 節で説明した検証環境では IPv6 only 環境を目指しながらも，DHCP4 を用いている．この理由は，

- IPv4 link local assumption (RFC3927 [9]) に MAY で定義されている仕様により，Mac OS X では IPv4 リンクローカルアドレスの着いたインターフェースに default route を向けてしまい，フォールバック問題が発生する．

  具体例を Mac OS X 10.8 の場合で示す．IPv4 DHCP 設定の Mac OS X 10.8 をネットワークに接続すると，DHCP 要求を limited broadcast する．DHCPv4 サーバの応答がないと，IPv4 link-local address をインターフェースに設定して、default route を向け，その際のホスト上の経路表は次のようになる．

```
Internet:
Destination   Gateway    Flags   Refs   Use   Netif Expire
default       link#6     UCS       31     0     en3
1.2.3.4       link#6     UHLWIi     0     0     en3
```

  この状態で，IPv4 グローバルアドレス宛にパケットを送信しようとすると，on link assumption で解決しようとして，Mac OS X 10.8 のホストは ARP 要求をブロードキャストする．しかしながら，ARP には誰も応答しないため，タイムアウト待ちが発生することになる．

  この挙動に対して，DHCP4 で IPv4 プライベートアドレスのみを配布し，default gateway のアドレスを通知しないことで，IPv4 default route がクライアントの経路表に載らないようにすることができ，フォールバック問題の発生を抑えることが出来る．

- Android などの一部の OS では，IPv4 アドレスが設定されないと無線 LAN の設定が終了しない．それに対し，DHCP4 で IPv4 プライベートアドレスのみを配布し，default gateway のアドレスを通知しないことで無線 LAN 接続を完了させる

ことが出来，且つフォールバック問題も発生しにくくなる．

- また，過去の実験では Android などでは，RA は受け取り IPv6 での通信を行うものの DNS クエリは IPv4 パケットでしか送信できない実装が存在することが確認されている．また，HappyEyeball の実装によっては IPv4, IPv6 両方で DNS クエリを発生させるものも存在する．

  そのため，無線 LAN 上に DNS Fowarder（DNS Proxy）を設置して，IPv4 での DNS クエリを仲介する必要がある．その際，A レコードを返すと IPv4 で通信しようとしてフォールバックが起こるため，前節で説明した A Record Filter にて AAAA の結果のみがクライアントに通知すると，フォールバック問題が発生しにくくなる．

という過去の WIDE 研究会での実験から得た知見により設定している．

しかし，「IPv6 only 環境」をめざしながら，IPv4 アドレスを配布することは IPv6 完全移行の障害となる．実際に IETF sunset4 WG では，前節で紹介したような DNS64/NAT64 環境を IPv6 only network と呼称することには反論の声が上がっている．

そこで，主に Mac OS X の IPv4 Link Local Assamption によるフォールバック問題の解消を主眼として，「DHCP4 の代わりに ProxyARP を用いるとフォールバック問題も発生せず，且つ IPv4 アドレスも配布しないので sunset4 WG でも IPv6 only network と呼べる形式になるのではないか？」という提案が奈良先端科学技術大学院大学 垣内 助教から提案された．そこで，実際に ProxyARP を用いてもフォールバック問題が改善されるのか否かの検証を行った．

4.1 構成

図 16 が sunset4 の構成図である．前節 3 での構成のように DHCP4 を SEIL で配布せず，代わりに SEIL にて Proxy ARP の設定を行った．また，DNS は DHCP6 にて BIG-IP の IPv6 アドレスを通知する形をとり，BIG-IP では前節での実験と同様の A Record Filter を設定し，AAAA Record のみ通知する形式としている．

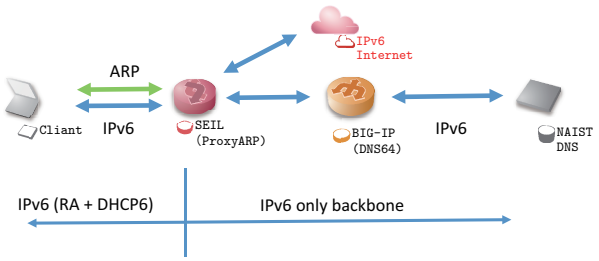

図 16: ProxyARP を用いた IPv6 only 環境

4.2 検証結果

検証の結果は以下のとおりとなった．

- Mac OS X や iOS 7 では，即座に無線 LAN の接続が完了する．インターフェース設定を確認すると見かけ上 IPv4 アドレスが設定されず IPv6 アドレスのみが確認できる．tcpdump で確認すると MAC アドレスが衝突しない IPv4 リンクローカルアドレスを延々と探索していることが判明した．

  IPv6 に DNS クエリも通信も対応しているアプリケーション（ウェブブラウザなど）は問題なく利用できるが，Facebook App Store アプリなどの IPv4 の通信からはじめる HappyEyeball 実装を行っているアプリケーションや，Skype や dropbox など IPv4 のみにしか対応していないアプリケーションは軒並み利用できない結果となった．

- Windows 7 や Windows 8 では，無線 LAN への接続を開始してから 10 秒程度あとにアドレス衝突が発生した旨を通知するポップアップが描画され，無線 LAN の接続設定がいったん中断する．ポップアップの確認ボタンを押した後，1 分程度 DHCP4 のタイムアウトを待って，無線 LAN の接続が完了する．体感では 1 分 30 秒程度無線 LAN 接続完了までユーザは待たされることになる．

  Mac OS X の場合と同様に，IPv6 に DNS クエリも通信も対応しているアプリケーション（ウェブブラウザなど）は問題なく利用できるが，IPv4 の通信からはじめる HappyEyeball 実装を行っているアプリケーションや，Skype や dropbox など IPv4 のみにしか対応していないアプリケーションは軒並み利用できない結果となった．

- Android 4.4 では，過去の実験で確認された現象

と同様に，無線 LAN 接続シーケンスが 30 秒ごとにリセットされ，通信が行えない結果となった．通信が開始できないため，アプリケーションの利用は確認できなかった．

4.3 考察

検証した結果，Mac OS X ユーザの体感は DNS64/NAT64 環境とほぼ同じだが，Window 7/8 ユーザの体感は無線 LAN 接続の完了まで 1 分 30 秒待たされるため QoE が低下すると感じ，Android ユーザは無線 LAN に接続できないため，QoE が著しく低下することが明らかとなった．

また，IPv6 に完全に対応したアプリケーションは，ssh などのコマンドラインツールやウェブブラウザなどで，VPN アプリケーションや Skype など，多くのユーザが利用しているアプリケーションが利用できないため，合宿参加者からやはり不満の声が上がった．

当初 ProxyARP による sunset4 については，すべての ARP に応答し，送られてきた IPv4 パケットに対して Network is Administratively Prohibited を返せば，ARP タイムアウト待ちがなくなってすぐに IPv6 にフォールバックするだろうという期待であった．しかしながら，実際は自動生成した IPv4 link-local address のアドレス重複検査にも Proxy ARP が応答してしまい，IPv4 アドレスが付かない (バックグラウンドで IPv4 link-local address 自動生成が走り続ける）という結果になった．これについては今後，`169.254.0.0/16` には Proxy ARP 応答しないようにして再検証を奈良先端大の検証環境にて実施する．

5 IPv6 バックボーンにおけるコンテンツキャッシュの有効性の検証

合宿ネットワークでは，IPv6 バックボーンにコンテンツキャッシュ装置を挟み込んだ場合の有効性に関して検証を行った．検証には NTT-AT 長嶺氏に協力していただいた．

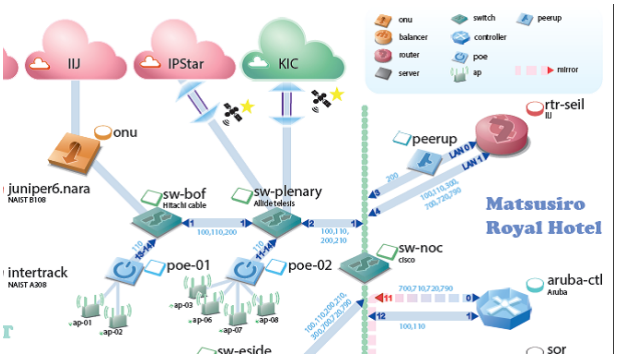

図 17: UltraBand の設置場所

5.1 構成

コンテンツキャッシュ装置としては PeerApp 社製 UltraBand を利用した．UltraBand は透過型コンテンツキャッシュ装置でインライン設置を想定している．UltraBand は IPv4，IPv6 の区別なくキャッシュ対象コンテンツを自動判別しコンテンツのハッシュ値を取って保存し，3 回目以上のキャッシュヒットが発生した場合，一旦リクエストはサーバ側に配送し，その後 RST を発行してサーバからの実際のコンテンツ配信をキャンセルした後，ローカルのキャッシュをクライアントに配信する．これにより，上位回線の帯域消費は少なく，下位の回線の帯域利用率を向上することにより，帯域削減と QoE (Quality of Experience) 向上の両立を達成することが出来る．

今回の実験は Proof fo Concept の確認として，IPv6 環境でのキャッシュ機能の有効性を検証した．検証では，1) 帯域削減効果，2)QoE 向上効果，3) 各プロトコル，サービスへの対応状況に関して注力して実施した．また，そのほかの機能確認としてグラフ表示やログ出力など運用に必要な機能が十分備わっているかの確認も行った．図 17 に示すように，UltraBand はルータ（SEIL）の上流側に挟み込む形で設置した．

5.2 検証結果

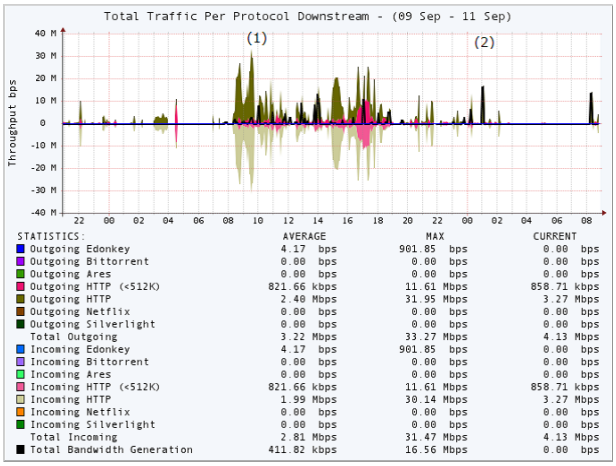

図 18: UltraBand でのコンテンツキャッシュのグラフ

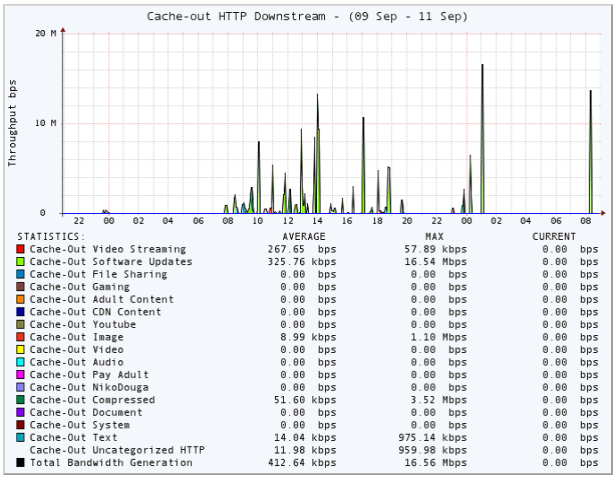

図 19: UltraBand での帯域削減効果

図 18 は UltraBand での合宿期間中のキャッシュ効果のグラフである．キャッシュ対象トラフィックについて, プラス側がコンテンツキャッシュ装置から下流へ出ていく Outgoing トラフィックの流量，マイナス側が上流からコンテンツキャッシュ装置へはいってくる Incoming トラフィックの流量である．プラス側の値とマイナス側の値の差分が大きいほどキャッシュの効果が出ているコンテンツが多く含まれていることになる．

Outgoing トラフィックのピークは 9 月 10 日 9 時半から 10 時の間で 33.27Mbps を UltraBand で透過している．キャッシュ効果のピークは 9 月 11 日 1 時から 1 時半の間であった．図 19 は UltraBand のキャッシュから配信したコンテンツの流量を表示したものである．主なコンテンツとしては Windows Update のパッチデータであり，合計 17Mbps 中 16.54Mbps を UltraBand から配信していることが解析結果からわかった．

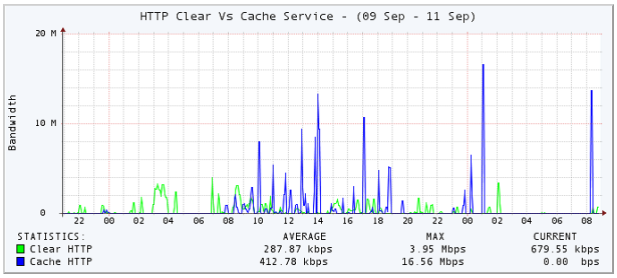

図 20: UltraBand での QoE 向上効果

図 20 は UltraBand による QoE 向上効果を示すグラフである．UltraBand によるキャッシュがない場合は 1Mbps，キャッシュがある場合は 16.56Mbps にスループットが向上しており，約 16 倍の QoE 向上効果が確認された．

また，対応プロトコル，コンテンツの確認を行ったところ，HTTP，Window Update, iOS アプリのダウンロードなど，コンテンツを配送するプロトコルや仕組みに対してはほぼ動作し，HTTPS など暗号化されたコンテンツは想定どおりにコンテンツキャッシュが行えないことも確認できた．機能面の確認に関しては，一部グラフが表示されないことが確認され，NTT-AT 長嶺氏により PeerApp 社に報告と修正依頼が行われた．

6 アンケート結果

IPv6 実験に関する感想についてもアンケート項目として設定した．2014 年 9 月 17 日の時点での回答数は合宿参加者 122 名のうち 46 名で，回答率は 37.7 %であった．回答した 46 名が利用した無線 LAN の内訳は

- camp1409-nat64-1x：39 名
- camp1409-nat64-sunset4：18 名
- camp1409-ipstar：9 名
- camp1409-lastresort：5 名

であった．

IPv6 実験に関する好意的な感想と否定的な感想は下記のようになった．

6.1 好意的な感想

- facebook の happyeyball 問題が解決されて快適になった.
- とりあえず v6 only でなんとか生活できた
- OpenVPN で udp6 を指定して NAT64 で会社の VPN に繋ぎに行ったら普通に IPv4 も使えるようになって快適になった

6.2 否定的な感想

- 接続できない SSID

  iPhone や Android 端末から接続できない SSID が存在したと複数名の参加者が回答していた.
- IPv6 対応していないキラーアプリケーションが利用できないことへの不満

  やはり skype とか polcom mobile presence が v6 で使えないことはつらい. または，skype や polycom，dropbox などが使えないので，公衆無線 LAN や lastresort，IPv4 over IPv6 のトンネルを設定して IPv4 を利用していた，という回答が見受けられた.
- 普段の高品質な IPv4 インターネット環境と比較した QoE の低下

  Thunderbird の IMAP での fetch がかなりもっさりしていた，などフォールバック問題や HappyEyeball の挙動に起因していると思われる QoE の低下が感想として挙げられた.

7 まとめと今後の展開

今回の実験では，DNS64/NAT64 環境の構築手法において，A Record Filter は FormError を発生させないようなフィルタリングを行うか DNS Forwader 側で厳密な FormError の判定を行わないようにすることが必要であることが確認できた.

また，従来の A Record Filter のほかに，ICMP protocol unreach や ICMP administratively prohibited をゲートウェイで返すことで，HappyEyeball による IPv4 でのタイムアウト待ちの時間を短縮し，IPv4 から IPv6 へのフォールバックに要する時間を短縮する効果が確認できた. また IPv6 バックボーンでのコンテンツキャッシュの有効性の確認も行えた.

今後の展開としては，A Record Filter および ICMP による Happy Eyeball IPv4 fallback の追検証を奈良先端科学技術大学院大学内の DNS64/NAT64 検証無線 LAN 環境で実施し，追検証結果を合わせてインターネットドラフトを執筆し，IETF v6ops WG や sunset4 WG にて報告する予定である.

```
iptables -A FORWARD -i eth0 -d 0.0.0.0/0 -j REJECT \
--reject-with icmp-proto-unreachable
```

図 21: iptables の場合 (code 2 : Protocol Unreachable)

```
iptables -A FORWARD -i eth0 -d 0.0.0.0/0 -j REJECT \
--reject-with icmp-admin-prohibited
```

図 22: iptables の場合 (code 9 : Administrativly Prohibited)

8 Appendix

8.1 Destination Unreach ゲートウェイの作成，設定方法

8.1.1 Ubuntu 14.4 LTS を用いた場合

3.4 節で説明したワークアラウンドである ICMP Destination Unreach を応答するゲートウェイは，Ubuntu 14.14 LTS の iptables を用いて実装した.

1： DHCP サーバで配っているデフォルトルートに Ubuntu サーバを置く

2： sysctl で v4 の forwarding=1 に設定する

3： Linux サーバの iptable の FORWARD CHAIN 上に問題となるパケットに対して destination **port** unreach を返すような IP フィルタを書く

```
term accept-limited-broadcast {
    from {
        source-address {
            0.0.0.0/32;
        }
        destination-address {
            255.255.255.255/32;
        }
    }
    then accept;
}
term accept-to-local {
    from {
        destination-address {
            172.31.0.0/16;
        }
    }
    then accept;
}
term reject-last {
    then {
        reject administratively-prohibited;
    }
}
```

図 23: JUNOS の場合 (code 9 : Administrativly Prohibited)

4： 何らかの経路がないと先に destination network unreachable を返して FORWARD CHAIN まで上がらないので、到達しない IPv4 アドレス（たとえば 192.0.2.1 や null インターフェースなど）にトンネルを設定し、 デフォルトルートをトンネルの先の IPv4 アドレス向ける.

このように設定することで v4 パケットに destination port unreach を返すルータが完成する．図 21 が Protoco Unreachable を設定する場合の iptables の設定であり，図 22 が Administratively Prohibited を設定する場合の iptables の設定例である．なお，unix OS であれば，他の linux distribution でも BSD 系 OS でも同様の設定が行える.

8.1.2 Juniper EX9208 を用いた場合

研究会終了後，奈良先端大 垣内助教により，奈良先端大内に構築されている DNS64/NAT64 試験環境にて administratively prohibited による IPv4 フィルタが実装された．なお，DNS64/NAT64 試験環境では現在 IPv4 クライアント向け DNS fowrder は実装しているが A Record Filter は実装していない.

1. IPv4 DNS forwarder つき DNS64/NAT64 環境を構築する.

   NAIST では 172.31.0.1/16 を DHCP4 / IPv4 DNS forwarder（OpenMicroServer OMS-AL400/128）に，172.31.0.2/16 を IPv6 gateway (Juniper EX9208) に設定．DNS64 は BIND 9.9，NAT64 には ObenBSD Packet Filter を用い，NAT64 prefix には `64:ff9b::` を利用している．DHCP6 は ISC DHCP 4.2 の機能を用いて設定している.

2. 172.31.0.2（Juniper EX9208）にて以下のファイアウォールフィルタを設定する.

   1. accept from 0.0.0.0/32 to 255.255.255.255/32
   2. accept from any to 172.31.0.0/16
   3. administratively-prohibited from any to any

3. DHCP4 にて 172.31.0.2 を IPv4 default gateway として，DNS を 172.31.0.1 としてクライアントに 172.31.0.0/16 の IPv4 アドレスとともに配布する設定を行う.

体感では，A Record Filter ありの場合より若干 IPv6 へのフォールバックに数秒（1 秒から 8 秒くらい）必要としているように見受けられるが，administartively-prohibited フィルタなしの場合に比べて著しく availability が高まっていることが確認できた.

A Record Filter がある場合・ない場合の差分については引き続き調査が必要である.

8.2 Firefox での SOCKS プロキシ設定方法

1. In firefox type this in your address bar: `about:config`
2. Click that you promise to be careful.
3. In the filter textbox, `type:proxy`
4. Find the preference name called "`network.proxy.socks_remote_dns`". Double click it to set it to `true`.
5. `ssh -D your-sock-server.your.net`

8.3 DNS64/NAT64 環境で利用できないことが確認されたアプリケーション

ここでは，研究会参加者からヘルプデスクの担当者に対して利用できないことが報告されたアプリケーションを列挙する．利用できない原因の切り分けは行っていないため，VPN ゲートウェイなどは単なる設定ミスの可能性もあることをここに明記しておく．

8.3.1 VPN

- Cisco AnyConnect (iOS App Store version)

  慶應義塾大学加藤 朗 先生から報告を受ける．VPN ゲートウェイは KMD 設置のゲートウェイで，機種は ASA5505．あて先のアドレスは Mapped Address である．別の ASA5505 の VPN ゲートウェイ（Native IPv6）でも接続が出来なかったとのこと．

- OpenVPN / Tunnelblick (Windows 7, Mac OS X)

  奈良先端科学技術大学院大学 櫨山 寛章 から報告を受ける．VPN ゲートウェイは NAIST 設置のゲートウェイで，機種は FreeBSD．あて先のアドレスは Mapped Address である．

- Fortinet FortiClient SSL VPN (Windows, Mac OS X)

  奈良先端科学技術大学院大学 櫨山 寛章 から報告を受ける．VPN ゲートウェイは北陸 StarBED 技術センター設置のゲートウェイで，機種は Fortigate．あて先のアドレスは Mapped Address である．

8.3.2 Voice / Video conference

- Skype

  多数の参加者から「Skype が快適に使える IPv4 インターネットが欲しい」との要望がでる．camp1409-lastresort を使うようにヘルプデスクでは対応した．IPv6 トンネルごしに大学や社内ネットワークに接続して，IPv4 インターネットを手元に持ってきて対応している参加者も存在した．

- Polycom Real Presence

  数名の参加者から「Polycom Real Presence が快適に使える IPv4 インターネットが欲しい」との要望がでる．ヘルプデスクでは，Skype の場合と同様の対応を行った．

8.3.3 アンチウィルス製品

- ノートンアンチウィルスのシグニチャアップデート

  東京大学 江崎 浩 先生から報告を受ける．

8.3.4 クラウドストレージ

- DropBox

  多数の利用者から報告を受ける．

8.4 その他

- VMWare Fusion の NAT44 設定

  JAIST の岩橋氏から報告を受ける．「VMwareFusion の NAT の下にかかえていた開発環境はインターネット接続が不能でした。(VMware Fusion は NAT44 しか行わない模様). VMware Fusion のゲスト OS にて Mac OS X の Wi-Fi インタフェースをブリッジした場合もゲスト OS には有線インタフェースに見えていてほぼ接続が確立しない.」という解析結果も報告された．

- POPFile

  千葉商科大 渡辺 恭人教授から報告を受ける．

8.5 sunset4 環境で利用できないことが確認されたアプリケーション

注意事項として，DNS64/NAT64 環境で動作しなかったアプリケーションは同様に動作しないことがおおむね確認された．その中で，著しく利用できなくなったアプリケーションに関して live-with-ipv6 WG の有志から報告を受けた．おそらく，IPv4 を優先するもしくは IPv4 の存在を前提としている HappyEyeball の挙動により IPv6 にフォールバックしていないことが原因ではないかと推測される．

8.5.1 iOS 上のアプリケーション

- iTune Store

  NTT-AT 西田さんから報告を受ける．メッセージとして「iTunes Store に接続できません．iTunes Store にアクセスするには Wi-Fi ネットワークに接続する必要があります」となる．

- Facebook アプリ

  NTT-AT 西田さんから報告を受ける．

- (Facebook) Messenger

  NTT-AT 西田さんから報告を受ける．

- App Store のダウンロード

  NTT-AT 西田さんから報告を受ける．ブラウジングは出来る．

参考文献

[1] 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.

[2] T. Murakami, O. Troan, and S. Matsushima. IPv4 Residual Deployment on IPv6 infrastructure - protocol specification. Internet Draft (expired), September 2011.

[3] O. Troan (Ed.), W. Dec, X. Li, C. Bao, S. Matsushima, T. Murakami, and T. Taylor (Ed.). Mapping of Address and Port with Encapsulation (MAP). Internet Draft (Publication Requrested), June 2014.

[4] Marius Georgescu, Hiroaki Hazeyama, Youki Kadobayashi, and Suguru Yamaguchi. Empirical Analysis of IPv6 Transition Technologies Using the IPv6 Network Evaluation Testbed. In *The 9th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM 2014)*, May 2014.

[5] Osamu Nakamura, Hiroaki Hazeyama, Yukito Ueno, and Akira Kato. IPv4 Address Literal in URL. Internet Draft (will be revised to 02), January 2014.

[6] Hiroaki Hazeyama, Tomohiro Ishihara, and Osamu Nakamura. DNS A Record Filtering for the migration from dual stack networks to IPv6 only networks. Internet Draft (expired), July 2013.

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

[8] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147 (Proposed Standard), April 2011.

[9] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.