Vol.2017-IOT-36 No.13
2017/3/3

クライアントOSのIPv6実装検証とネットワーク運用における課題

北口 善明[1,a)]　近堂 徹[2]　鈴田 伊知郎[3]　小林 貴之[4]　前野 譲二[5]

概要：各オペレーティングシステムのIPv6実装が進んだことで，IPv6ネットワークが提供されればクライアント端末はIPv6通信が優先となるケースが増えてきている．一方で，現在IPアドレス自動設定の手法としてはRAやDHCPv6を用いたステートレス・ステートフル設定がRFCで策定され，その組み合わせによってはクライアントの意図しない挙動を誘発するだけでなく，ネットワーク環境へ与える影響も考慮しなければならない．本発表では，各種クライアントOSにおけるIPv6実装状況の検証結果を報告するとともに，これらがネットワーク運用管理に与える影響について考察する．

キーワード：IPv6, Operating System, デュアルスタック, ネットワーク管理, ネットワーク運用

Verification of an IPv6 Implementation in Operating Systems for Network Management

Yoshiaki Kitaguchi[1,a)]　Tohru Kondo[2]　Ichiroh Suzuta[3]　Takayuki Kobayashi[4]　Joji Maeno[5]

Abstract: Recently, cases where IPv6 communication of clients take priority are increasing if IPv6 networks are available because of the IPv6 implementations of each operating system have proceeded. On the other hand, the stateless or stateful IP address configuration using RA and DHCPv6 had been defined by RFCs. This may result in not only an unintended behavior of clients but also an impact on network environment from these combinations. In this paper, we report some verification results for IPv6 implementation of each client operating system and consider about influence for network operation and management.

Keywords: IPv6, Operating System, Dual-stack, Network Management, Network Operation

1. はじめに

IPv6は，現行のインターネットプロトコルであるIPv4におけるアドレス枯渇問題を解消することを目的として，1995年に最初の仕様がRFCにて策定されてたが，20年以上経た現在においても，継続的な仕様更新・追加がなされている．ほとんどのネットワーク機器やクライアントおよびサーバ用OSでは，IPv6のプロトコルスタックが既に実装されており，多くのネットワークサービスやサービスもIPv6に対応している．特にGoogleやFacebook等のハイパージャイアントを中心にIPv6化が完了しており，30%のトラフィックがIPv6通信となる状況となっている[1]．

その一方でインターネットにおける多くの通信は未だIPv4であり，IPv6対応していないネットワークやサービスのほうが優勢な状況のままである．このIPv6対応が進まない原因としては，IPv6がIPv4機器との直接通信するための互換性を持っていないことによるIPv6対応のコスト増が考えられる．IPv6に対応することは，すなわち，IPv4と別のIPv6ネットワークを二重に運用することを意味しており，ネットワーク運用コストが単純に増加する．また，利用者がIPv6を積極的に利用するメリットがない

[1] 金沢大学 総合メディア基盤センター
Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan
[2] 広島大学 情報メディア教育研究センター
[3] アラクサラネットワークス株式会社
[4] 日本大学 文理学部
[5] 早稲田大学 情報教育研究所
[a)] kitaguchi@imc.kanazawa-u.ac.jp

ことから，追加コストを支払ってまで IPv6 を利用する動機が発生せず，ネットワーク運用コストの増加分を回収する術がない点が大きいと考えられる．しかし，インフラストラクチャとして位置付けられるようになっている現在のインターネットについて，IPv6 導入に向けた課題を正しく把握し，効率的に運用するための知識・経験を養っていくことが必要となる．

そこで，本稿では，クライアント OS の IPv6 実装に焦点を当て，IPv6 アドレス自動設定や IPv6 オンリーネットワークへの対応状況などの実装状況を検証した結果を報告する．また，この検証結果を元に，現時点におけるネットワーク運用管理における課題を考察し，今後の IPv6 対応ネットワークの普及促進に向けて提言を行う．

2. RFC における IPv6 アドレス自動設定とアドレス選択関連の仕様

IPv6 では，ルータ広告（RA: Router Advertisement）に含まれるプレフィックス情報を利用する SLAAC（StateLess Address AutoConfiguration）[2] と，DHCP の IPv6 版である DHCPv6（Dynamic Host Configuration Protocol for IPv6）[3] を用いる二種類のアドレス自動設定手法が存在する．さらに，DHCPv6 には，IPv4 における DHCP と同様に IP アドレスの割り当てまでを実施するステートフルモードと，IP アドレス設定以外の情報（DNS サーバ情報など）を配布するだけのステートレスモードが定義されている．これらのアドレス自動設定は，RA に含まれる 3 つのフラグにより制御される仕様 [4] となっており，それぞれ下記の動作を制御している．

- A (Autonomous address configuration) Flag:
  プレフィックス情報オプションに含まれるフラグで，このフラグが 1 の場合に当該プレフィックスを利用した SLAAC による IPv6 アドレス設定が可能となる．
- O (Other configuration) Flag:
  このフラグが 1 の場合，IPv6 アドレス設定以外のネットワーク情報を DHCPv6 にて用意されていることを示しており，ステートレス DHCPv6 のプロセスが起動する．
- M (Managed address configuration) Flag:
  このフラグが 1 の場合，DHCPv6 にて IPv6 アドレスを払い出す用意があることを示しており，ステートフル DHCPv6 のプロセスが起動する．なお，ステートフル DHCPv6 では，IPv6 アドレス以外の情報も配布されるため，M フラグが利用される場合には O フラグの値は意味をなさない．

本節では，IPv6 アドレス自動設定とアドレス選択に関連する仕様について RFC を元に整理する．

2.1 SLAAC

IPv6 アドレスは 128 ビットからなり，IPv6 アドレス自動設定を使用する場合には，上位 64 ビットをサブネットプレフィックス，下位 64 ビットをインタフェース ID として扱う [5]．SLAAC では，まず 64 ビットのインタフェース ID を生成し，fe80::/64 のサブネットプレフィックスを付加したリンクローカルアドレスを設定する．このリンクローカルアドレスを用いて，RA を促すルータ要請（RS: Router Solicitation）を送信し，RA を受信する．RA に含まれるプレフィックス情報からプレフィックス長を取得し（一般的には 64 ビット），A フラグが 0 でない限りそのプレフィックス情報とインタフェース ID を組み合わせることで IPv6 アドレスを得ることができる．また，RA の送信元アドレスがデフォルトルートとして設定され，これにより IPv6 通信が可能となる．

インタフェース ID の生成方法として，多くの場合は MAC アドレスを元にした Modified EUI-64 フォーマット [5] が利用される．Modified EUI-64 フォーマットは，重複しないという利点があるが，常に同じ値であることからセキュリティとプライバシに関する懸念が指摘されていた．そこで，インタフェース ID をランダムに生成し定期的に更新するプライバシ拡張アドレス [6] が策定され，Modified EUI-64 フォーマットによるアドレスに追加する形で利用されることとなった．このプライバシ拡張アドレスは，ほとんどのクライアント OS にて実装されており，デフォルトで利用されている．また，Modified EUI-64 フォーマットによるアドレスに対するセキュリティ上の問題が指摘され [7]，MAC アドレスの推測を不可能にし，プレフィックス情報が変化しない限り固定となるインタフェース ID の生成方法 [8] が規定された．

2.2 DHCPv6

DHCPv6 は，前述したように RA のフラグにより二種類の動作が規定されている．

ステートレス DHCPv6 は，RA の O フラグが設定されることで起動され，IPv6 アドレス設定以外のネットワーク情報を DHCPv6 により配布する．DNS サーバの IPv6 アドレスを端末に自動設定する手法として，当初は DHCPv6 しか規定されていなかったが，RA にて DNS サーバアドレスを通知するオプション機能（RDNSS : Recursive DNS Server オプション）[9] が追加規定されたため，ステートレス DHCPv6 が必須ではなくなった．

ステートフル DHCPv6 は，IPv4 の DHCP のように IPv6 アドレス設定とネットワーク情報の配布を DHCPv6 で実施するモードで，RA の M フラグを設定することで利用される．ステートフル DHCPv6 で設定される IPv6 アドレスにはプレフィックス情報がなく，加えて，デフォルトルートを設定する機能も DHCPv6 には規定されていないため，

DHCPv6 単独ではなく RA との併用が必須となり，この点が IPv4 の DHCP と異なる点であり注意が必要である．

2.3 デフォルトアドレス選択

IPv6 では，同一セグメント内の通信に用いられるリンクローカルアドレスとグローバル通信のためのグローバル IPv6 アドレスが利用されるため，一つのインタフェースに複数のアドレスが設定される．また，プライバシ拡張アドレスなどの追加により，通信に利用するアドレスを選択するルール規定が必要となり，IPv4 アドレスも含めた形で通信に用いるアドレス選択の仕組みが規定された [10]．

送信元アドレス選択では，以下の 8 つのルールを元に利用する送信元アドレスを決定する．

- Rule 1: Prefer same address.
  自身への通信には同じアドレスを使用．
- Rule 2: Prefer appropriate scope.
  アドレススコープ（グローバル，サイトローカル，リンクローカルなど）が同じアドレスを優先．
- Rule 3: Avoid deprecated addresses.
  非推奨アドレス（自動アドレス設定におけるアドレスの有効期限が近づいているアドレス）を回避．
- Rule 4: Prefer home addresses.
  モバイル IP で利用されるホームアドレスを優先．
- Rule 5: Prefer outgoing interface.
  宛先アドレスへ向かうインタフェース（Next-hop が設定されているもの）に設定されているアドレスを優先．
- Rule 6: Prefer matching label.
  ポリシーテーブル（表 1）のラベル（Label）が一致するアドレスを優先．
- Rule 7: Prefer temporary addresses.
  プライバシ拡張アドレスなどの一時アドレスを優先．
- Rule 8: Use longest matching prefix.
  プレフィックスが最も長く一致するアドレスを優先．

これらのルールに則ると，SLAAC と DHCPv6 にて IPv6 アドレスを設定する場合，Rule 7 による一時アドレス優先によりプライバシ拡張アドレスが選ばれ，DHCPv6 による IPv6 アドレスなどは基本的に通信に利用されないことになる．

宛先アドレス選択においても同様のルールが規定されており，DNS での名前解決を行うリゾルバにて複数の IPv6 アドレスが解決された場合の順位付けに利用される．宛先アドレスの優先順位は表 1 に示すポリシーテーブルの優先度（Precedence）にて制御することができる（Rule 6: Prefer higher precedence.）．例えばデュアルスタック環境において IPv4 通信を優先させたい場合には，“::ffff:0:0/96” の優先度を “::/0” より大きな値（例えば 45）とすることで制御できる．このポリシーテーブルの値は手動で変更することができるだけでなく，DHCPv6 の拡張オプション [11] を用いることでネットワーク側からの制御も可能となるが，現時点においてサーバアプリケーションおよびクライアント OS における実装を確認できていない．

表 1　RFC 6724 で定義されたデフォルトポリシーテーブル
Table 1　Default policy tables defined in RFC 6724.

Prefix	Precedence	Label	Description
::1/128	50	0	ループバックアドレス
::/0	40	1	IPv6 アドレス
::ffff:0:0/96	35	4	IPv4 アドレス
2002::/16	30	2	6to4 アドレス
2001::/32	5	5	Teredo アドレス
fc00::/7	3	13	ULA
::/96	1	3	IPv4 互換アドレス
fec0::/10	1	11	サイトローカルアドレス
3ffe::/16	1	12	6bone アドレス

2.4 Happy Eyeballs

TCP 通信では，何らかの原因で通信ができない場合に，別のアドレスを用いた通信に切り替えるフォールバック機構を有しており，特に IPv6 と IPv4 を切り替えて代替通信することを IPv6/IPv4 フォールバックと呼ぶ．この機能により，どちらかのプロトコルで障害があった場合でも最終的に通信を確立することが可能となるが，切り替えのための遅延が発生するため，ユーザ体験が低下することになる．

そこで，フォールバックによる影響を緩和するために，IPv4 と IPv6 の通信を同時に開始して先に接続できたプロトコルを利用する Happy Eyeballs[12] が策定された．この Happy Eyeballs では，DNS 名前解決手順も含め IPv4 と IPv6 による接続方法が定義されているが，複数の IPv6 アドレスを有する場合などの動作は定義されておらず，実装により差異が生じる仕組みとなっている．そのため，Apple 社では iOS9 以降の実装において IPv6 通信をより優先するための仕組みが追加されており，Happy Eyeballs の動作において IPv4 側に 25ms の遅延挿入を実施している [13]．

3. IPv6 導入におけるネットワーク形態の整理

2 章で述べたように，IPv6 の自動アドレス設定に関しては，SLAAC+RDNSS オプション，SLAAC+ステートレス DHCPv6 などの様々なパターンを取ることができる．また，IPv6 を導入する際に用いられる一般的なデュアルスタックでは，IPv4 と IPv6 双方のネットワーク運用が必要となり，ネットワーク障害が発生した際の問題切り分けが複雑になること [14] や，ネットワーク運用に際しても双方のプロトコル監視が必要となるため，運用コストやセキュリティリスクの増加も課題となる．そのため，ネットワークを IPv6 で構築し，トランスレータ機能を用いて IPv4 を 1 つのサービスとして提供する “IPv6 オンリーネットワーク” の利用も現実的なものとなっている．トランスレータ

表 2　デュアルスタック環境を提供するサービス形態
Table 2　List of dual-stack network models.

Type	IPv4	IPv6		
		RA flags	Address	DNS
0	DHCP	A	RA	IPv4 only
1		A, RDNSS	RA	RA
2		A, O	RA	DHCPv6
3		M	DHCPv6	DHCPv6
4	DNS64/ NAT64	A, RDNSS	RA	RA
5		A, O	RA	DHCPv6
6		M	DHCPv6	DHCPv6

機能としては DNS64[15] と NAT64[16] を組み合わせた仕組みがデファクトスタンダートとなっており，2016 年 6 月から，Apple 社は iOS 用のアプリケーションに対して，この DNS64/NAT64 環境を持つ IPv6 オンリーネットワークでも問題なく動作することをアプリケーション審査における必須条件としている [17].

以上のことから，デュアルスタック環境を提供するサービス形態を整理し，表 2 にまとめた．IPv4 と IPv6 のアドレスをそれぞれ設定する場合と，IPv4 をトランスレーションで提供する構成に大きく分けることができる．IPv6 のアドレス設定に関しては，RA における冗長なフラグ構成を排除し，IPv6 アドレスやネットワーク情報を RA もしくは DHCPv6 のどちらかで提供する形態のみとしている．また，クライアント OS がデュアルスタック環境に接続した際（表 2 における Type 2 のケース），ウェブ通信を開始するまでの典型的なパケットフローを図 1 に示す．矢印が途中で止まっているものはマルチキャスト通信を意味している．このように，自動アドレス設定などの必要最低限の通信フローであっても，IPv4 だけの場合と比較して複雑な動作となっていることが分かる．

これに加え，IPv6 アドレスとして一般的なグローバルアドレス提供に加え，ULA（Unique Local IPv6 Unicast Address）[18] と NAT（Network Address Translarion）による提供方法も存在する．IPv6 は基本的に NAT がなくとも潤沢なアドレス空間を有することからグローバルアドレスを利用することが一般的であるが，テザリングを提供する環境などで IPv4 と同様の形態を取る可能性が考えられる．IPv6 における NAT の手段としては，標準化されていないが IPv4 の NAPT（Network Adress Port Translation）と同様の仕組みである NAPT6 と，Experimental State ではあるが RFC にて標準化されている NPTv6（IPv6-to-IPv6 Network Prefix Translation）[19] が Linux の実装として存在している．NPTv6 は，チェックサムを維持しつつ IPv6 アドレスを 1 対 1 でステートレスに変換する仕組みで，マルチホーミング環境などでの利用が想定されている．

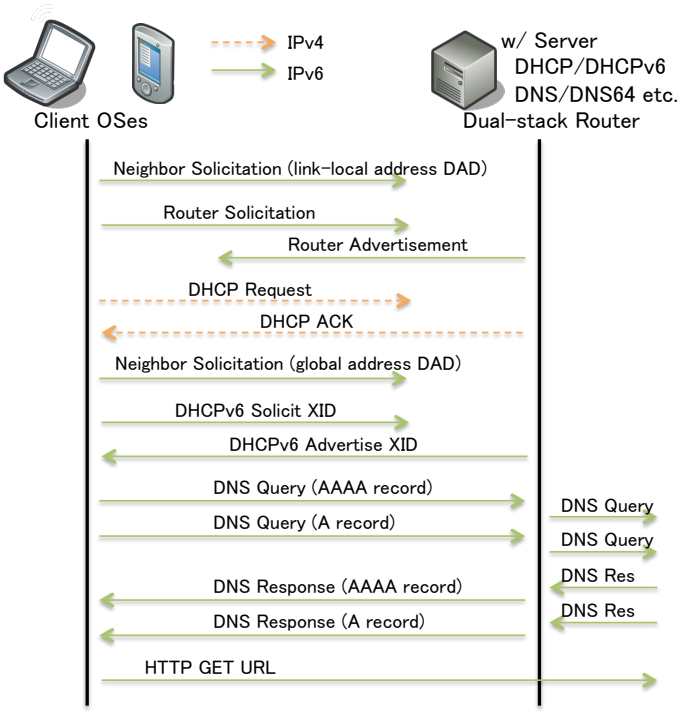

図 1　典型的なデュアルスタック環境でのクライアント OS 起動時のパケットフロー
Fig. 1　Typical packet flow at booting client OS on dual-stack.

4. クライアント OS の実装検証実験

IPv6 における自動アドレス設定は複雑な仕様であることに加え，少しずつ追加・改良されて来たこともあり，すべてのクライアント OS において均質に仕様が実装されていない．そのため，RA に設定されるフラグによる挙動がクライアント OS 毎に異なることが指摘されており，仕様上の整理が IETF にて進められている状況である [20].

以上のことから，主要なクライアント OS における IPv6 自動アドレス設定の挙動と，IPv6 オンリーネットワークにおける動作検証を行った．なお，Happy Eyeballs の実装に関しては，クライアント OS だけではなくブラウザによる実装も存在するため，検証の組み合わせが多岐に渡ることとなる．今回の検証では，時間的な制約のため，Happy Eyeballs の動作検証は見送っており，今後の課題としている．

4.1 検証環境

今回の検証を進めるために，様々な IPv6 自動アドレス設定環境を提供するためのルータを Linux OS を利用して構築した．スマートフォン端末の検証も可能とするため，無線 LAN アクセスポイントとしての機能を提供する構成としている．今回は，デバイスとして Raspberry Pi 3 Model B を利用して構築しており，表 3 に利用したアプリケーションとバージョン情報をまとめる．

表 3　利用したルータ用アプリケーションのバージョン
Table 3　Version of applications for router.

Application	Version	use
Raspbian	November 2016	OS
hostapd	1:2.3-1+deb8u4	Wireless LAN
radvd	1:1.9.1-1.3	RA
isc-dhcp-server	4.3.1-6+deb8u2	DHCP/DHCPv6
unbound	1.5.9-1~bpo8+1	DNS/DNS64
jool	3.5.2	NAT64
iptables	1.4.21-2	NAPT NPTv6/NAPT6

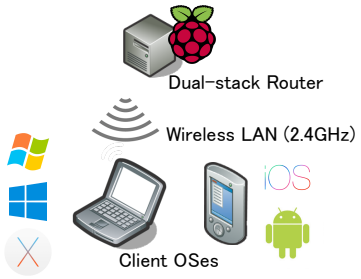

図 2　検証環境のネットワーク構成
Fig. 2　Network configuration of the verification.

クライアント OS 検証におけるネットワーク構成を図 2 に示す．検証を実施する際には，ルータにおける設定を変更した後，下記の手順にてクライアント OS の挙動データを取得した．

(1) 無線 LAN インタフェースの停止
(2) DNS キャッシュデータの削除
　Windows: ipconfig /flushdns or 再起動
　OS X, macOS: sudo dscacheutil -flushcache
　iOS: 再起動
　Android: 再起動
(3) ルータにおけるパケットキャプチャの開始
(4) 無線 LAN インタフェースの有効化
(5) IPv4 および IPv6 ウェブサーバへの接続

表 4 に，今回の検証に用いたクライアント OS のバージョン情報を記す．

4.2　検証結果と考察

表 5 に，各クライアント OS 毎における SLAAC で用いるインタフェース ID の生成手法，RDNSS オプションの実装，DHCPv6 の実装および IPv6 オンリーネットワーク環境での検証結果をまとめる．インターフェース ID（IID）の生成手法の記載は，

- RFC 2464：Modified EUI-64 IID
- RFC 7217：Semantically Opaque IID
- Microsoft：Random IID（Microsoft 独自実装）
- Apple：Random IID（Apple 独自実装）

をそれぞれ意味している．その他は，動作を確認できたものを “○”，確認できなかったものを “×” としている（“△” は問題があったことを示しており，後述する．）．また，表 6 には，RDNSS オプションと RA の M フラグを同時に設定した場合の各クライアント OS 毎の DNS 関連の検証結果をまとめている．表には，デュアルスタック環境と IPv6 オンリーネットワーク環境での DNS サーバの利用順と DNS クエリの順番を記している．いずれも左側に記載したものが先に利用されていることを示していて，DNS サーバの利用における “Random” は，設定されている DNS サーバからどれか一つをランダムに選んで利用していることを示す．

以下に各クライアント OS 毎に検証結果をまとめ，考察する．

4.2.1　Windows

Windows の実装では，今回検証に用いたどのバージョンにおいても，RA における RDNSS オプションが実装されておらず，DHCPv6 はステートフル，ステートレス双方への対応が確認できた．DHCPv6 に関しては，Windows

表 4　検証に用いたクライアント OS のバージョン
Table 4　Version of client OSes on the verification.

OS	Version
Windows 7 Home Premium	6.1 (7601, SP1)
Windows 8.1 Pro	6.3 (9600)
Windows 10 Pro Insider Preview	1607 (14931.1002)
OS X 10.10 (Yosemite)	10.10.5
OS X 10.11 (El Capitan)	10.11.6
macOS 10.12 (Sierra)	10.12.2
iOS 7	7.1.2
iOS 8	8.4
iOS 9	9.3.5
iOS 10	10.2
Android 4.4 (Xperia Z Ultra)	4.4.2
Android 5 (NEXUS7 2012)	5.1.1
Android 6 (NEXUS7 2013 LTE)	6.0

表 5　クライアント OS 毎の挙動
Table 5　Behavior each client OSes.

OS	IID	RDNSS	DHCPv6	v6only
Windows 7	Microsoft	×	○	○
Windows 8.1	Microsoft	×	○	○
Windows 10	Microsoft	×	△	△
OS X 10.10	RFC 2464	○	○	○
OS X 10.11	RFC 2464	○	○	○
macOS 10.12	RFC 7217	○	○	○
iOS 7	Apple	○	○	○
iOS 8	Apple	○	○	○
iOS 9	Apple	○	○	○
iOS 10	RFC 7217	○	○	○
Android 4.4	RFC 2464	N/A	×	×
Android 5	RFC 2464	○	×	○
Android 6	RFC 2464	○	×	○

表 6　クライアント OS 毎の DNS 利用順序
Table 6　DNS order each client OSes.

OS	DNS Order		Query Order
	Dual-stack	IPv6 only	
Windows 7	DHCPv6	DHCPv6	(A), AAAA
Windows 8.1	DHCPv6, IPv4	DHCPv6	(A), AAAA
Windows 10	DHCPv6, IPv4	DHCPv6	(A), AAAA
OS X 10.10	IPv4	Random	A, AAAA
OS X 10.11	Random	Random	A, AAAA
macOS 10.12	DHCPv6	Random	AAAA, A
iOS 7	IPv4	DHCPv6	AAAA, (A)
iOS 8	IPv4	DHCPv6	AAAA, (A)
iOS 9	DHCPv6	Random	AAAA, (A)
iOS 10	RA	Random	AAAA, A
Android 4.4	IPv4	N/A	AAAA, A
Android 5	RA	RA	AAAA, A
Android 6	RA	RA	AAAA, A

※(A) は IPv6 オンリーネットワーク環境で実施されないことを表す.

7 と Windows 8.1 および Windows 10 にて実装の差異が見られた．Windows 7 は RA における O フラグおよび M フラグの指定がない限り DHCPv6 クライアントが起動しないが，Windows 8.1 と Windows 10 ではインタフェース起動時に DHCPv6 クライアントが起動される実装となっている．さらに，Windows 10 では，起動時に DHCPv6 でのアドレス設定や DNS サーバ取得ができない現象を確認した．コマンドプロンプトにて “ipconfig /renew6” を実行することで取得することができるが，[21] に記載されているように，Windows 8 以降の実装に含まれる BUG である可能性が高い．

DNS サーバの利用は，Windows 7 では DHCPv6 で取得した DNS サーバしか利用しないが，Windows 8.1 および Windows 10 では取得した DNS サーバすべてに同じ DNS クエリを出していた．また，名前解決の順番はデュアルスタックでは A, AAAA クエリの順で同時に実施しており，IPv6 オンリーの場合には A クエリを省略していた．

Windows の実装で設定される IPv6 アドレスには，Modified EUI-64 によるインタフェース ID が用いられず，ランダムに生成されることを確認した．この値は RFC 7217 の実装とは異なり，ネットワークアドレスが変更されても同じものが利用されている．

以上のように Windows は，Windows 7 では RFC による仕様に忠実な実装になっていたと言えるが，Windows 8.1 および Windows 10 では DHCPv6 が “勝手に” 起動する実装となり，さらには BUG が存在する状況となっている．

4.2.2　OS X, macOS

OS X および macOS の実装では，10.10，10.11 および 10.12 いずれのバージョンにおいても，RDNSS オプションと DHCPv6 の実装を確認できた．また，今回の検証では，IPv6 オンリーネットワーク環境ではすべてのバージョンにおいて，設定された DNS サーバをランダムに利用することが確認できた．

macOS と名称が変更されたバージョン 10.12 では，DNS のクエリ順が後述の iOS と同様に AAAA,A クエリと変更されており，Apple 社の IPv6 優先方針が伺える．また，インターフェース ID の生成も Modified EUI-64 から RFC 7217 に変更されていることが確認できた．

4.2.3　iOS

iOS の実装では，今回検証に用いたすべてのバージョンにおいて，RDNSS オプションと DHCPv6 の実装を確認できた．iOS 9 以降では，デュアルスタック環境における IPv6 DNS サーバの優先利用などの若干の差異が見受けられた．

iOS 10 では，IPv6 オンリーネットワーク環境における A クエリ省略を行わない実装となっており，IPv6 オンリーネットワーク環境での IPv4 通信を許容する実装になったと想定できる．加えて，インターフェース ID の生成が RFC 7217 に変更されていることを確認した．なお，iOS 9 以前の実装では，Modified EUI-64 ではなく，Windows の実装と同様にランダムな独自実装であると考えられる．

4.2.4　Android

Android の実装では，いずれのバージョンにおいても DHCPv6 の実装が確認できず，RA で設定される IPv6 のインタフェース ID は Modified EUI-64 であった．RA の RDNSS に関しては，バージョン 5 と 6 での動作を確認できたが，バージョン 4.4 では確認できなかった．また，Android 4.4 では，無線アソシエーション確立後に DHCP で IPv4 アドレスが取得できないとインタフェースをダウンさせる挙動となっており，IPv6 オンリーネットワーク環境で動作できなかった．

なお，[22] に記載されているように，NAT64（PLAT: Provider-side translator）と連携して IPv4 通信を可能にする CLAT（Customer-side translator）が CLATd として Android 4.3 以降で実装されている．NAT64（PLAT）と CLAT により IPv6 ネットワークを介して IPv4 通信を可能にする仕組みは 464XLAT として RFC 6877[23] にて標準化されているものである．この CLATd は，IPv4 アドレスしか登録されていない “ipv4only.arpa” に対して AAAA レコード解決を実施し，IPv6 アドレスの回答を得ることで NAT64 配下であることを確認したのちに起動され，clat インタフェースが作成されることを確認した．したがって，今回検証に用いた Android 4.4 の端末：Xperia Z Ultra 固有の設定があり，IPv6 オンリーネットワーク環境で動作しなかったのではないかと考えられる．

以上のように，Android の実装では DHCPv6 利用ができない状況であり，IPv6 オンリーネットワーク環境で動作させるためには RA の RDNSS オプションが必須となる．

また，CLAT の実装がされている唯一の OS で，Skype などの IPv4 にしか対応していないアプリケーションを IPv6 オンリーネットワーク環境で利用可能なクライアント OS となっている．

4.2.5 まとめ

今回検証に用いたクライアント OS すべて（Android 4.4 を除く）を IPv6 オンリーネットワーク環境で動作させるためには，RDNSS オプションと O フラグを設定した RA を利用し，ステートレス DHCPv6 を用いる運用が最低限必要となる．複数の DNS サーバが設定された場合に，それらを利用する仕組みがクライアント OS 毎で異なる状況であることが分かった．

名前解決の順序は，IPv6 が先に行われる実装が主流となりつつあり，いずれの場合も同時にクエリを送信していた．IPv6 オンリーネットワーク環境では IPv4 の名前解決を行わない実装もあるが，Android のように IPv6 非対応アプリケーションのために CLAT を動作させ，ローカルでの IPv4 通信を提供している実装も存在している．

SLAAC によるアドレス設定では，プライバシ拡張アドレスがどのクライアント OS においてもデフォルトで利用されていた．そのため，インタフェースに複数の IPv6 アドレスが設定されていたとしても，IPv6 通信に利用されるアドレスは，2.3 節のルールに示されるようにプライバシ拡張アドレスのみが利用されていた．

5. ネットワーク運用に与える影響

5.1 アドレス自動設定と運用管理コスト

4 章ではアドレス自動設定の OS 毎の実装の差異について示した．IPv4 と IPv6 をそれぞれ提供する場合は，各端末ともにインターネットに接続できない状況は発生しない．しかしながら 3 章でも述べた通り，IPv4 と IPv6 双方のネットワーク運用が必要となり，ネットワーク障害が発生した際の問題切り分けが複雑になることなどから，恒久的なネットワーク運用やセキュリティリスクへの対応によるコスト増は避けられない．

一方で，端末に IPv6 アドレスのみを割り当て，IPv4 をトランスレーションで提供する場合，OS 実装の差異がネットワーク運用者側へ影響を与える可能性がある．本実験の結果からもわかるように，Android も含めた全ての OS を接続可能にするには，RDNSS オプション付きの RA とステートレス DHCPv6 を併用する必要がある．このとき，IP アドレスは RA により割当てられるものになるため，管理者側で割当制御を行うことができない．昨今のセキュリティインシデントの対応の迅速化が求められるなか，ステートレス IP アドレス割当により端末の特定が困難になる状況は避けなければならず，DHCPv6 のみで運用できることが望ましいが，現在の OS 実装状況においてはまだ不十分な状況にあると言える．

5.2 ネットワーク機器

本節では，ネットワーク提供側の機器仕様の面から考える．各端末においてはなんらかの手法でアドレスを取得し DNS サーバの情報が得られれば通信が可能となるが，ルータ及び L3 スイッチにおいては各端末の MAC アドレスと IP アドレスの対応テーブルを持つ必要がある．IPv4 の場合は ARP テーブルとして MAC アドレスに対して基本的には一つの IPv4 アドレスが対応しているのみであったが，IPv6 アドレスについては MAC アドレスに対してリンクローカルアドレスと通信用のアドレス，さらにはプライバシ拡張アドレスが割り当てられる．その上，OS の実装によっては RA と DHCPv6 の双方からアドレスを取得し保持する OS があることが今回の検証でも明らかになった．IPv6 通信に利用されるアドレスは 2.3 節のルールに従うものの，プライバシ拡張によるアドレス変更やリンクローカルアドレスによる通信などにより，状況によっては ARP より多くのエントリを占めることとなる．

YAMAHA 社の RTX1200 にてデュアルスタック運用を行い，一週間後そのテーブルを調べたところ，1 MAC アドレスあたり macOS 10.12 で 4 つほど，iOS 10 に至っては 10 個の IPv6 アドレスを用いていることがわかった．当然ながら IPv4 は 1 MAC アドレスあたり 1 つの IPv4 アドレスであった．

一方，受ける側の L3 スイッチの収容数はこの実情を反映しているとはいえない状況にある．例えば ALAXALA 社の 1U ボックス型タイプの L3SW である AX3650S を例にとると [24]，IPv4 のみの運用であれば ARP は 11,000 強収容できるにもかかわらず，デュアルスタックにすると NDP は 2,000 しか入らず，ARP も 2,000 と減ってしまう．また IPv6 のみの端末収容を優先する仕様も用意されていない．現在の IPv6 の普及状況からは妥当な状況とも言えるが，今後の改善が期待されるとともに，機器導入の際には留意すべき観点である．

6. おわりに

本稿では，クライアント OS の IPv6 実装における動作検証を，IPv6 アドレス自動設定および IPv6 オンリーネットワーク環境下での挙動を中心に評価した結果を報告した．現在のクライアント OS 実装では，いずれもデュアルスタック環境での動作が可能であり，IPv6 通信も問題なく実現できていることが確認できた．また，IPv6 オンリーネットワーク環境では，一部のクライアント OS において課題があることが分かり，クライアント OS 毎の実装差異による問題も明らかとなった．今回検証対象としたすべてのクライアント OS を IPv6 オンリーネットワーク環境で動作させるためには，RDNSS オプションと O フラグを設定した RA を利用し，ステートレス DHCPv6 を用いる運用が最低限必要となることを確認した．

検証実験により，IPv6 対応時にクライアント OS に設定される IPv6 アドレスは，IPv4 のみの時と大きく異なり，ネットワーク管理における課題として運用面と実装面から整理して考察した．運用面では，一様にステートフルでのアドレス管理ができない状況であることから，セキュリティインシデント対応時のトレーサビリティに関して課題があると考えられる．実装面では，デュアルスタック運用で機器のメモリ資源が切迫することが予想でき，機器導入時において考慮が必要であることを指摘した．

IPv6 の利用は，世界的に見ると堅実に伸びており，欧州での利用率増加が近年顕著になっている [25][26] こともあり，一般的に利用する機会が増大すると予想されている．日本においても，モバイルキャリアにおける IPv6 デフォルト提供が 2017 年度より開始されると発表されており [27]，これに伴い，多くのサービスでの IPv6 導入が進むと期待される．したがって，今回のような機器実装の検証実験は，広く共有することが今後重要と考えており，Happy Eyeballs のようなよりアプリケーションに近いレベルの実装に対する評価・検証を今後継続して進めていきたいと考えている．

謝辞 本研究は，科学研究費補助金（15K00118）の助成を一部受けたものである．

参考文献

[1] 小林貴之, 鈴田伊知郎, 前野譲二, 近堂徹. 2016 年度における IPv6 の大学等における利用状況. 大学 ICT 推進協議会 2016 年度年次大会予稿集, December 2016.

[2] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), September 2007. Updated by RFC 7527.

[3] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. Updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283, 7550.

[4] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), September 2007. Updated by RFCs 5942, 6980, 7048, 7527, 7559, 8028.

[5] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), February 2006. Updated by RFCs 5952, 6052, 7136, 7346, 7371.

[6] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), September 2007.

[7] A. Cooper, F. Gont, and D. Thaler. Security and Privacy Considerations for IPv6 Address Generation Mechanisms. RFC 7721 (Informational), March 2016.

[8] F. Gont. A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC). RFC 7217 (Proposed Standard), April 2014.

[9] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli. IPv6 Router Advertisement Options for DNS Configuration. RFC 6106 (Proposed Standard), November 2010.

[10] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default Address Selection for Internet Protocol Version 6 (IPv6). RFC 6724 (Proposed Standard), September 2012.

[11] A. Matsumoto, T. Fujisaki, and T. Chown. Distributing Address Selection Policy Using DHCPv6. RFC 7078 (Proposed Standard), January 2014.

[12] D. Wing and A. Yourtchenko. Happy Eyeballs: Success with Dual-Stack Hosts. RFC 6555 (Proposed Standard), April 2012.

[13] M. Geniar. Apple favours ipv6, gives ipv4 a 25ms penalty. `https://ma.ttias.be/apple-favours-ipv6-gives-ipv4-a-25ms-penalty/` (参照 2017-01-27), July 2015.

[14] IPv6 普及・高度化推進協議会 IPv4/IPv6 共存 WGIPv6 導入に起因する問題検討 SWG. IPv6 導入時に注意すべき課題. `http://www.v6pc.jp/jp/pdf/20111124_v6fix.pdf` (参照 2017-01-27), November 2001.

[15] 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.

[16] M. Bagnulo, P. Matthews, and I. van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146 (Proposed Standard), April 2011.

[17] Apple Inc. Supporting ipv6-only networks. `https://developer.apple.com/news/?id=05042016a` (参照 2017-01-27), May 2016.

[18] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. RFC 4193 (Proposed Standard), October 2005.

[19] M. Wasserman and F. Baker. IPv6-to-IPv6 Network Prefix Translation. RFC 6296 (Experimental), June 2011.

[20] B. Liu, S. Jiang, X. Gong, W. Wang, and E. Ray. *DHCPv6/SLAAC Interaction Problems on Address and DNS Configuration*, August 2016. draft-ietf-v6ops-dhcpv6-slaac-problem-07.txt.

[21] M. Jackson. Update microsoft windows dhcp bug causing internet problems for users. `http://www.ispreview.co.uk/index.php/2016/12/microsoft-windows-dhcp-bug-causing-internet-problems-users.html` (参照 2017-01-27), December 2016.

[22] J. Zorz. Skype on android works over ipv6 on mobile networks using 464xlat. `http://www.internetsociety.org/deploy360/blog/2013/11/skype-on-android-works-over-ipv6-on-mobile-networks-using-464xlat/` (参照 2017-01-27), November 2013.

[23] M. Mawatari, M. Kawashima, and C. Byrne. 464XLAT: Combination of Stateful and Stateless Translation. RFC 6877 (Informational), April 2013.

[24] ALAXALA Networks Corp. AX3650S のテーブルエントリ数. `http://www.alaxala.com/jp/techinfo/archive/manual/AX3650S/HTML/11_14/CFGUIDE/0021.HTM#ID00076` (参照 2017-01-27), 2015.

[25] RIPE NCC. IPv6 Enabled Networks. `http://v6asns.ripe.net/v/6/` (参照 2017-01-27).

[26] Akamai Technologies. IPv6 の普及状況の可視化. `https://www.akamai.com/jp/ja/our-thinking/state-of-the-internet-report/state-of-the-internet-ipv6-adoption-visualization.jsp` (参照 2017-01-27).

[27] 安力川幸司, 伊藤孝史, 茂庭智. ついに決定！スマートフォンでの IPv6 提供開始. JANOG39 ミーティング, January 2017.