広島大学

IPv6-Mostly 実証環境の構築とクライアント OS 検証について

Internet Week 2025

2025/11/25 (火)

広島大学 先進理工系科学研究科

横尾 和真

<kzm-yokoo@hiroshima-u.ac.jp>

自己紹介

- 横尾 和真 / よこお かずま
  - 広島大学 先進理工系科学研究科 情報科学プログラム
  - 博士後期課程 在学中
  - 普段の研究内容（キーワード）:
    - IPv6、サイトマルチホーム、NPTv6、ルータ広告 (RA)
      SRv6、可用帯域推定、etc..
  - IPv6 との関わり:
    - IPv6 プロトコルスタック自作（SecHack365）
    - IPv6 基礎検定（広島地域IPv6推進委員会 支援）
    - 『ラズパイで試す！ IPv6⇔IPv4相互通信』執筆
      - Interface 2024年2月号

@yy_kzm [↗]

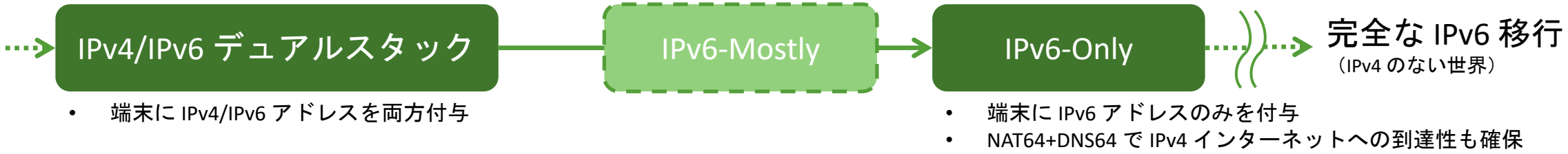
@y-kzm [↗]

目次

1. 本講演の概要
2. IPv6-Mostly 登場の背景と要素技術
3. Raspberry Pi を使った実証環境の構築
4. クライアント OS における動作検証

本講演の概要

- IPv6-Mostly ネットワークとは
  - IPv6-Only で問題なく動作するホストは IPv4 を無効にし、そうでないホストには、今まで通り IPv4 アドレスを必要に応じて配布するようなネットワークのこと

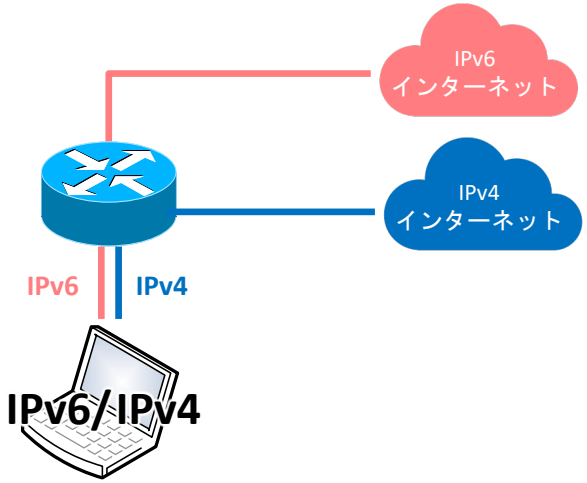


- 本講演では、実証環境の構築を具体的に紹介するとともに、主要クライアント OS における動作検証の結果を共有する
  - ✓ Raspberry Pi を用いた、自宅で簡単に試せる実証環境を紹介
  - ✓ DHCP IPv6-Only Preferred や CLAT の実装状況を調査

2. IPv6-Mostly 登場の背景と要素技術

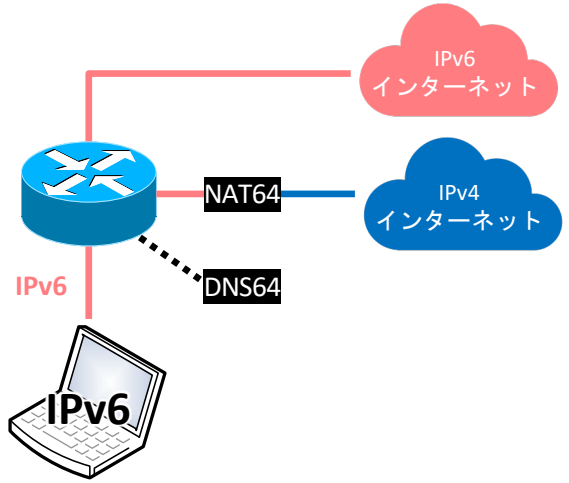
IPv6-Mostly 登場の背景

- IPv6 の導入 = IPv4 + IPv6（デュアルスタック）
  - IPv4 アドレス枯渇問題を解決していない
  - デュアルスタックが長期的な解決策になると、完全な IPv6 移行の足枷になってしまう



IPv6-Mostly 登場の背景

- IPv6 の導入 = IPv4 + IPv6（デュアルスタック）
  - IPv4 アドレス枯渇問題を解決していない
  - デュアルスタックが長期的な解決策になると、完全な IPv6 移行の足枷になってしまう



→ **IPv6-Only** というアプローチ

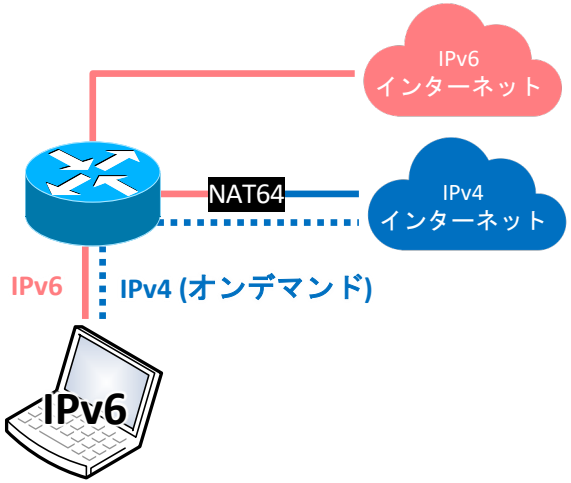
- LAN に IPv6 のみを利用するが、NAT64+DNS64 で IPv4 インターネットへの接続性も提供

いきなりデュアルスタックから IPv6-Only に完全移行するのはハードルが高い

- IPv4 に依存するアプリケーションや OS は動作しない
- IPv4 リテラルを用いた通信ができない
- IPv6-Only を別 SSID で提供するなどの複雑性を生む etc..

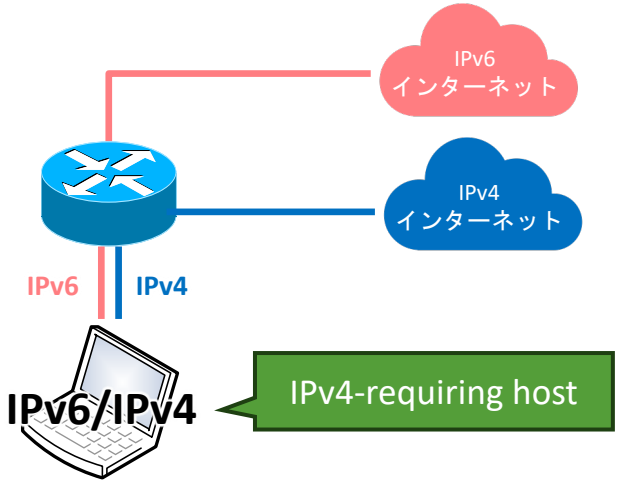
IPv6-Mostly 登場の背景

- **IPv6-Mostly** というアプローチ
  - 基本的には、LAN に IPv6 のみを利用するが、必要に応じてこれまで通り IPv4 も利用する
- IPv6-Mostly におけるクライアント種別



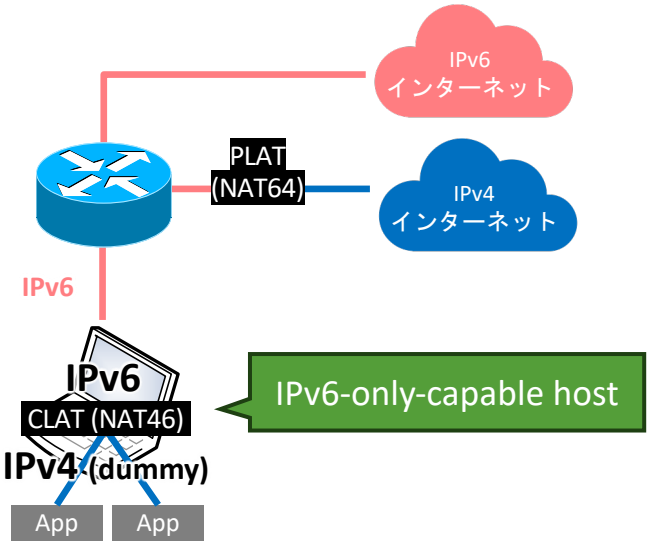
IPv6-Mostly 登場の背景

- **IPv6-Mostly** というアプローチ
  - 基本的には、LAN に IPv6 のみを利用するが、<u>必要に応じて</u>これまで通り IPv4 も利用する
- IPv6-Mostly におけるクライアント種別
  - **IPv4-requiring host**（IPv4を必要とするホスト）
    - IPv6-Only 環境では正常に動作しない
    - 一部アプリケーションや OS が IPv4 前提の場合、この種別に該当



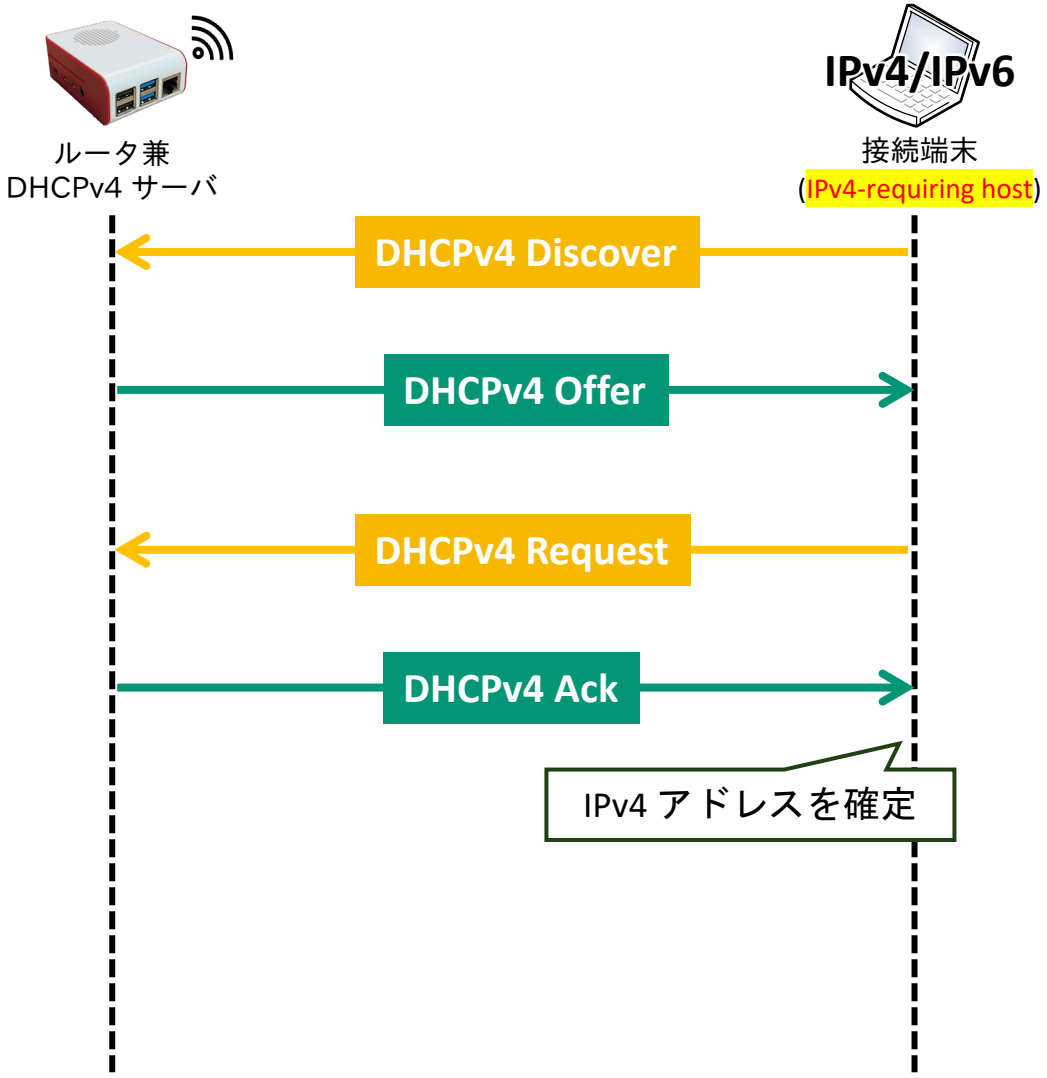
IPv6-Mostly 登場の背景

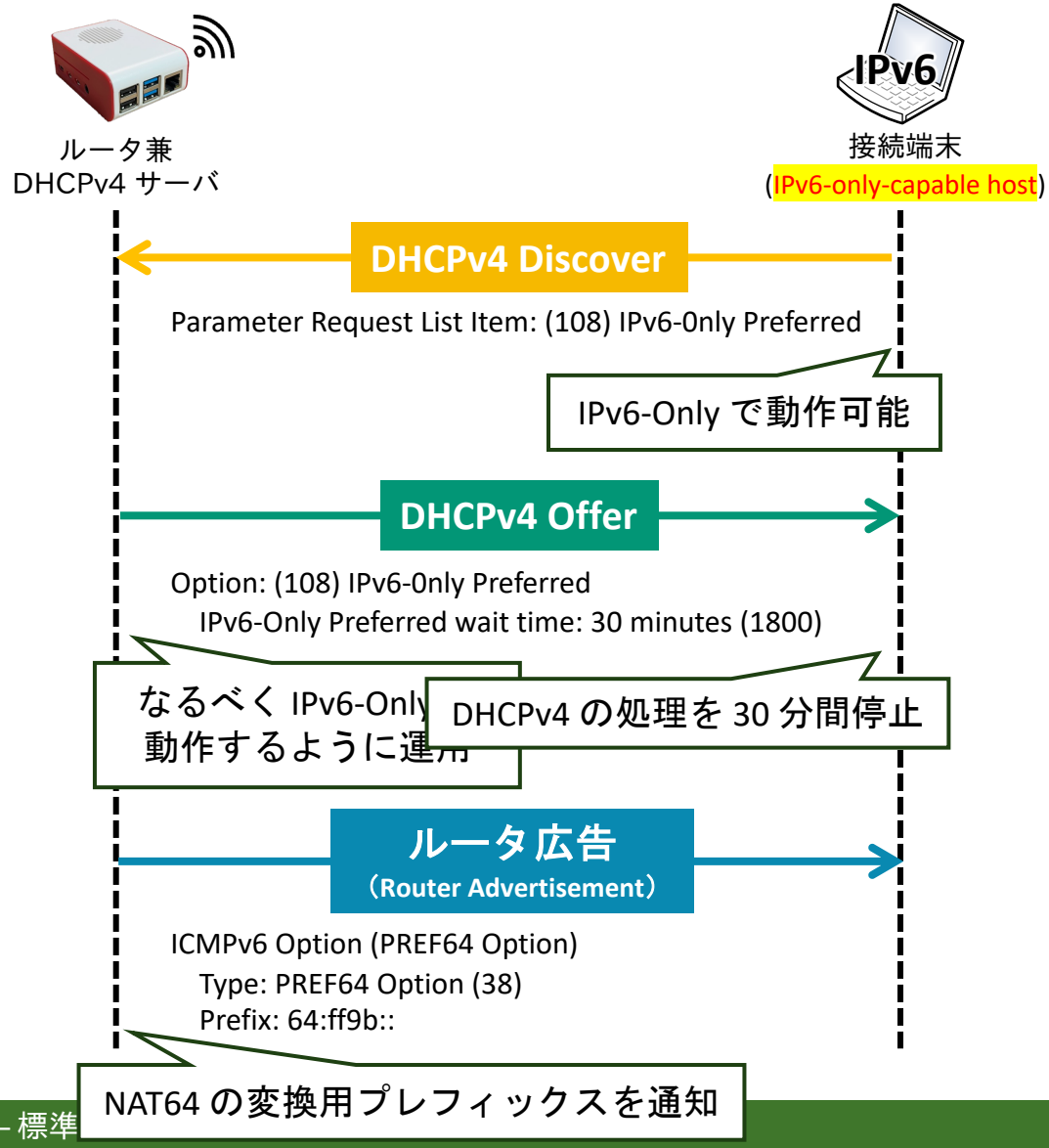
- **IPv6-Mostly** というアプローチ
  - 基本的には、LAN に IPv6 のみを利用するが、<u>必要に応じて</u>これまで通り IPv4 も利用する
- IPv6-Mostly におけるクライアント種別
  - **IPv4-requiring host**（IPv4を必要とするホスト）
    - IPv6-Only 環境では正常に動作しない
    - 一部アプリケーションや OS が IPv4 前提の場合、この種別に該当
  - **IPv6-only-capable host**（IPv6-Only 対応ホスト）
    - IPv4 しか対応していないサーバへは、ネットワークが提供する NAT64 を使って到達
    - 一部アプリケーションが IPv4 を前提としていても、CLAT が有効なら IPv6-Only として動作可能



既存のデュアルスタックとの互換性を最大限保ちつつ移行が可能

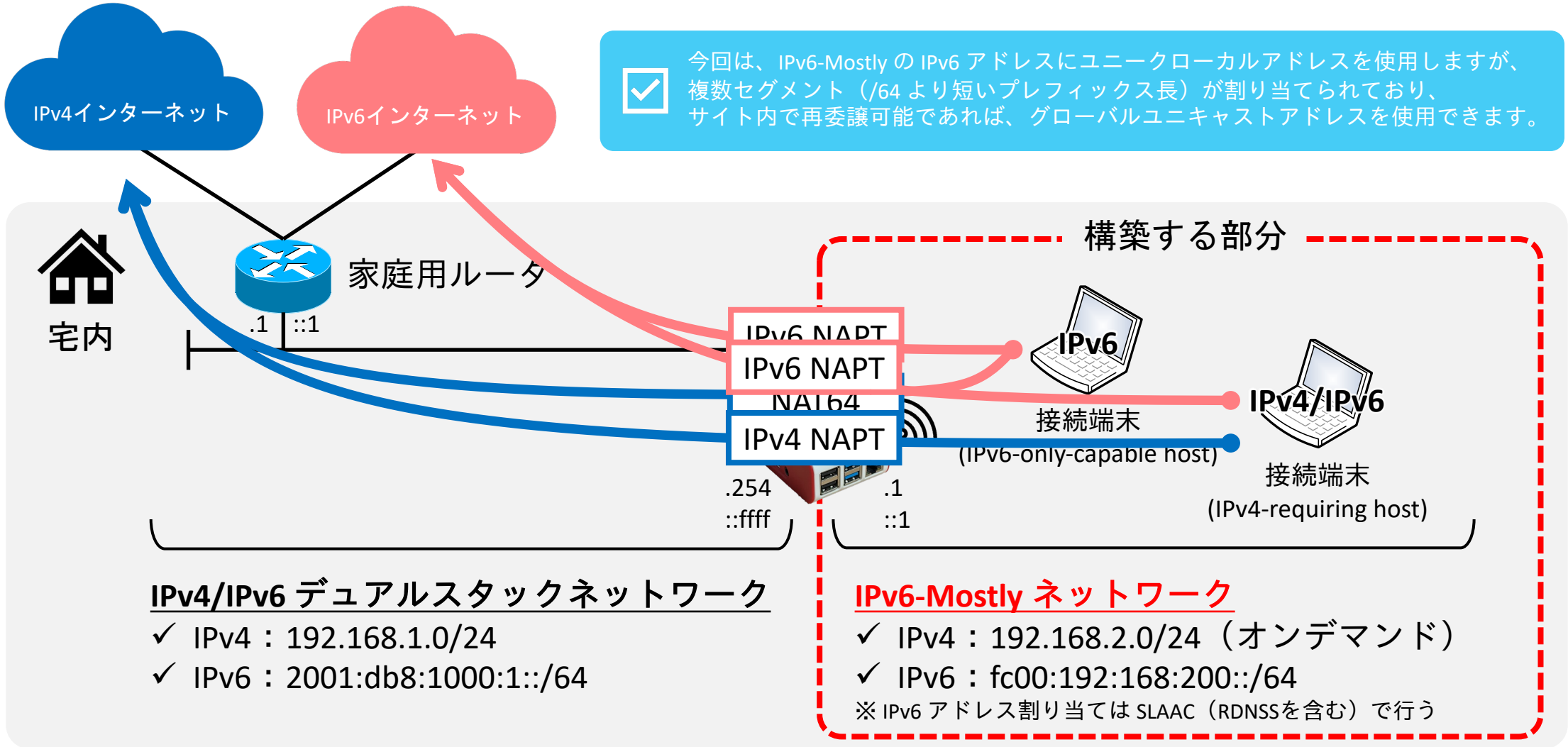
IPv6-Mostly におけるアドレス割り当て





3. Raspberry Pi を使った実証環境の構築

構築するネットワーク構成



使用する機器とソフトウェア

- 使用する機器
  - Raspberry Pi 4 Model B Rev 1.5

- 使用するソフトウェア

用途	ソフトウェア	バージョン
OS	Ubuntu	24.04.3 LTS
Raspberry Pi の無線 AP 化	hostapd	v2.10
IPv4/IPv6 NAPT	nftables	v1.0.9
DHCPv4	Kea	3.0.1
NAT64	Jool	4.1.11.0
ルータ広告デーモン	radvd	2.20

※ 今回使用するソフトウェアの各種設定ファイルは以下から取得可能です。
https://github.com/y-kzm/IPv6-Mostly_ConfigurationFiles

構築の手順

手順1. hostapd を用いた Raspberry Pi の無線 AP 化

手順2. nftables を用いた IPv6 NAPT (Masquerade) の設定

手順3. nftables を用いた IPv4 NAPT (Masquerade) の設定

手順4. Kea を用いた DHCPv4 の設定

手順5. Jool を用いた Stateful NAT64 の設定

手順6. radvd を用いたルータ広告デーモンの設定

構築の手順

手順1. hostapd を用いた Raspberry Pi の無線 AP 化

手順2. nftables を用いた IPv6 NAPT (Masquerade) の設定

手順3. nftables を用いた IPv4 NAPT (Masquerade) の設定

手順4. Kea を用いた DHCPv4 の設定

手順5. Jool を用いた Stateful NAT64 の設定

手順6. radvd を用いたルータ広告デーモンの設定

手順1. hostapd を用いた Raspberry Pi の無線 AP 化

- hostapd のインストール
  - Raspberry Pi を
    無線アクセスポイントにする

```
ubuntu@raspi:~$ sudo apt install hostapd ↩
...
ubuntu@raspi:~$ $ hostapd –v ↩
hostapd v2.10
...
```

```
ubuntu@raspi:~$ cat /etc/hostapd/hostapd.conf ↩
interface=wlan0
ssid=IPv6-Mostly
wpa_passphrase=passphrase
wpa=2
hw_mode=g
channel=11
auth_algs=1
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

```
ubuntu@raspi:~$ sudo systemctl unmask hostapd.service ↩
ubuntu@raspi:~$ sudo systemctl start hostapd.service ↩
ubuntu@raspi:~$ sudo systemctl enable hostapd.service ↩
ubuntu@raspi:~$ sudo systemctl status hostapd.service ↩
● hostapd.service - Access point and authentication server ...
```

※ 説明は省略

手順2. nftables を用いた IPv6 NAPT の設定

- IPv6 NAPT の設定
  - 今回は IPv6-Mostly の IPv6 アドレスにユニークローカルアドレスを使用しているため、IPv6 のアドレス変換が必要
  - nftables を使用

```
ubuntu@raspi:~$ sudo systemctl start nftables.service ↩
ubuntu@raspi:~$ sudo systemctl enable nftables.service ↩
```

```
ubuntu@raspi:~$ sudo nft add table ip6 mostly-nat ↩
ubuntu@raspi:~$ sudo nft add chain ip6 mostly-nat postrouting \
            '{ type nat hook postrouting priority srcnat; }' ↩
ubuntu@raspi:~$ sudo nft add rule ip6 mostly-nat \
            postrouting ip6 saddr fc00:192:168:200::/64 \
            oif eth0 masquerade ↩
```

```
ubuntu@raspi:~$ sudo nft list ruleset ↩
...
table ip6 mostly-nat {
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;
    ip6 saddr fc00:192:168:200::/64 oif "eth0" masquerade
  }
}
...
```

```
ubuntu@raspi:~$ cat /etc/sysctl.d/99-sysctl.conf ↩
...
# Uncomment the next line to enable packet forwarding for IPv6
net.ipv6.conf.all.forwarding=1
...
ubuntu@raspi:~$ sudo sysctl –p ↩
```

※ 説明は省略

手順3. nftables を用いた IPv4 NAPT の設定

- IPv4 NAPT の設定
  - IPv6-Mostly では、IPv6-Only 非対応ホストはデュアルスタックとなるため、IPv4 アドレス変換が必要
  - nftables を使用

```
ubuntu@raspi:~$ sudo nft add table ip mostly-nat ↩
ubuntu@raspi:~$ sudo nft add chain ip mostly-nat postrouting \
             '{ type nat hook postrouting priority srcnat; }' ↩
ubuntu@raspi:~$ sudo nft add rule ip mostly-nat \
            postrouting ip saddr 192.168.200.0/24 \
            oif eth0 masquerade ↩
```

```
ubuntu@raspi:~$ sudo nft list ruleset ↩
...
table ip mostly-nat {
  chain postrouting {
    type nat hook postrouting priority srcnat; policy accept;
    ip saddr 192.168.200.0/24 oif "eth0" masquerade
  }
}
...
```

- 設定の永続化
  - 現在のルールセットを設定ファイルに上書きする

```
ubuntu@raspi:~$ cat /etc/sysctl.d/99-sysctl.conf ↩
...
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
...
ubuntu@raspi:~$ sudo sysctl -p ↩
```

```
ubuntu@raspi:~$ sudo bash -c \
             'nft list ruleset > /etc/nftables.conf' ↩
```

※ 説明は省略

手順4. Kea を用いた DHCPv4 の設定

- Kea のインストール
  - ISC が提供するリポジトリを使用することで簡単に取得できる
    - stable 版である kea-3.0 を使用
    - kea-3.0 から IPv6-Only Preferred（RFC8925）を正式サポート[参考]
- 設定ファイルの準備
  - data は、V6ONLY_WAIT を表す
    - クライアントに DHCPv4 設定プロセスを停止させる時間

      → 無駄な DHCPv4 Discover を削減


```
ubuntu@raspi:~$ curl -1sLf \
  'https://dl.cloudsmith.io/public/isc/kea-3-0/setup.deb.sh' \
  | sudo -E bash ↵
...
   OK: The repository has been installed successfully ...
ubuntu@raspi:~$ sudo apt install isc-kea ↵
...
ubuntu@raspi:~$ kea-dhcp4 -V ↵
3.0.1 (3.0.1 (isc20250909094157 deb))
...
```

```
ubuntu@raspi:~$ cat /etc/kea/kea-dhcp4.conf ↵
{
  "Dhcp4": {
    ...
    "subnet4": [
      {
        ...

        "option-data": [
          ...
          // (108) IPv6-Only Preferred: RFC8925
          {
            "name": "v6-only-preferred",
            "data": "1800" // 1800秒 → 30分
          },
        }
    ],

    ...
  }
}
```

【補足】V6ONLY_WAIT によるクライアントの挙動

- RFC8925 によると..
  - V6ONLY_WAIT: サーバが通知するタイマー
    - クライアントが DHCPv4 設定プロセスを停止すべき時間
      デフォルト値: 1800秒
  - MIN_V6ONLY_WAIT: クライアントがあらかじめ持つタイマー
    - V6ONLY_WAIT の下限値
      値: 300秒
- Macbook Air (macOS 15.5) では..

data = 0 にした場合、 300秒後に DHCPv4 Discover が送信される

No.	Date	Src Hw	Dst Hw	Source	Destination	Protocol	Info
5	2025-10-27 17:48:53.586110	02:56:98:95:cf:e2	ff:ff:ff:ff:ff:ff	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xc2508704
7	2025-10-27 17:48:53.598711	e4:5f:01:40:e0:75	02:56:98:95:cf:e2	192.168.200.1	0.0.0.0	DHCP	DHCP Offer -
13	2025-10-27 17:53:53.604742	02:56:98:95:cf:e2	ff:ff:ff:ff:ff:ff	0.0.0.0	255.255.255.255	DHCP	DHCP Discover -
14	2025-10-27 17:53:53.639686	e4:5f:01:40:e0:75	02:56:98:95:cf:e2	192.168.200.1	0.0.0.0	DHCP	DHCP Offer -

data = 600 にした場合、 600秒後に DHCPv4 Discover が送信される

No.	Date	Src Hw	Dst Hw	Source	Destination	Protocol	Info
1	2025-10-27 18:03:00.001723	02:56:98:95:cf:e2	ff:ff:ff:ff:ff:ff	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xc2508707
3	2025-10-27 18:03:00.057562	e4:5f:01:40:e0:75	02:56:98:95:cf:e2	192.168.200.1	0.0.0.0	DHCP	DHCP Offer - Transaction ID 0xc2508707
16	2025-10-27 18:13:00.068686	02:56:98:95:cf:e2	ff:ff:ff:ff:ff:ff	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xc2508708
17	2025-10-27 18:13:00.111789	e4:5f:01:40:e0:75	02:56:98:95:cf:e2	192.168.200.1	0.0.0.0	DHCP	DHCP Offer - Transaction ID 0xc2508708

手順5. Jool を用いた Stateful NAT64 の設定

- Jool のインストール
  - Kernel modules / Userspace tools
  - カーネルモジュールのロード
- 設定ファイルの準備
  - 複数のパラメータを一度に設定可能な Atomic Configuration を使用
    - もちろん、joolコマンドで設定することも可能
  - Pref64::/n は、 64:ff9b::/96 を使用
- 今回は Stateful NAT64 を利用
  - SIIT を使用する場合には、jool_siit.service を使用

```
ubuntu@raspi:~$ sudo apt install \
              jool-dkms jool-tools \
              linux-headers-$(uname -r) \
              linux-modules-extra-$(uname -r)
...
ubuntu@raspi:~$ sudo modprobe jool
ubuntu@raspi:~$ jool -V
4.1.11.0
```

```
ubuntu@raspi:~$ cat /etc/jool/jool.conf ↩
{
    "instance": "nat64-minimal",
    "framework": "netfilter",

    "global": {
        "pool6": "64:ff9b::/96"
    }
}
```

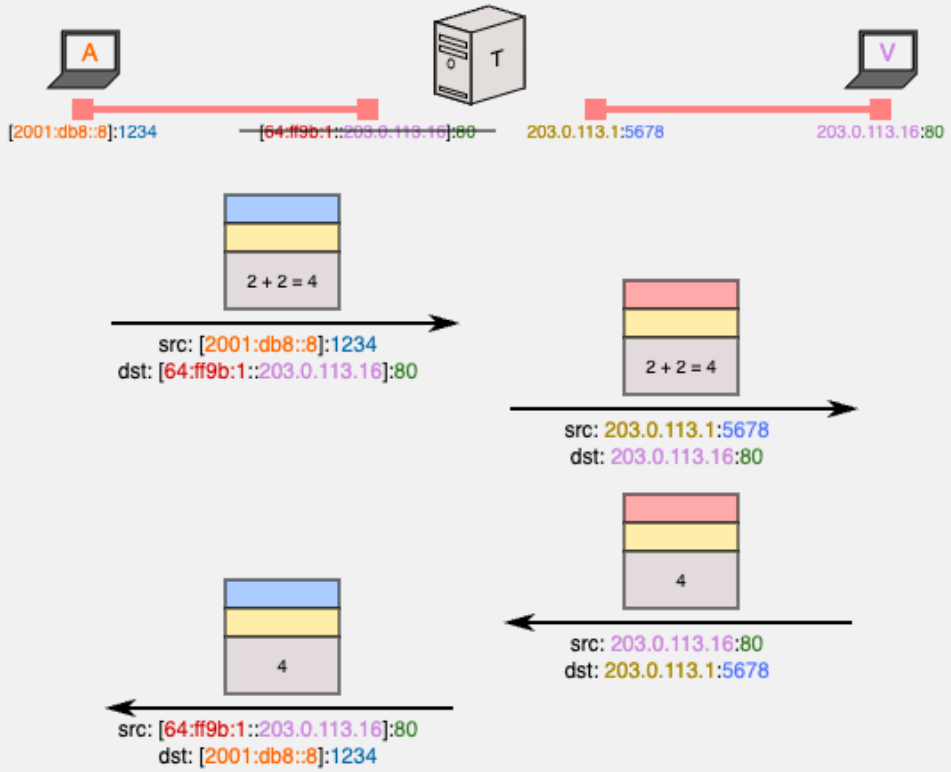
```
// joolコマンドで設定する場合(設定は永続化されない)
ubuntu@raspi:~$ jool instance add "nat64-minimal" \
                          --netfilter --pool6 64:ff9b::/96 ↩
```

```
ubuntu@raspi:~$ sudo systemctl start jool.service ↩
ubuntu@raspi:~$ systemctl status jool.service ↩
● jool.service - Stateful NAT64
     Loaded: loaded (/usr/lib/systemd/system/jool.service; ...
...
```

【補足】Stateful NAT64 と SIIT

参考: Jool, Introduction to IPv4/IPv6 Translation, https://www.jool.mx/en/intro-xlat.html

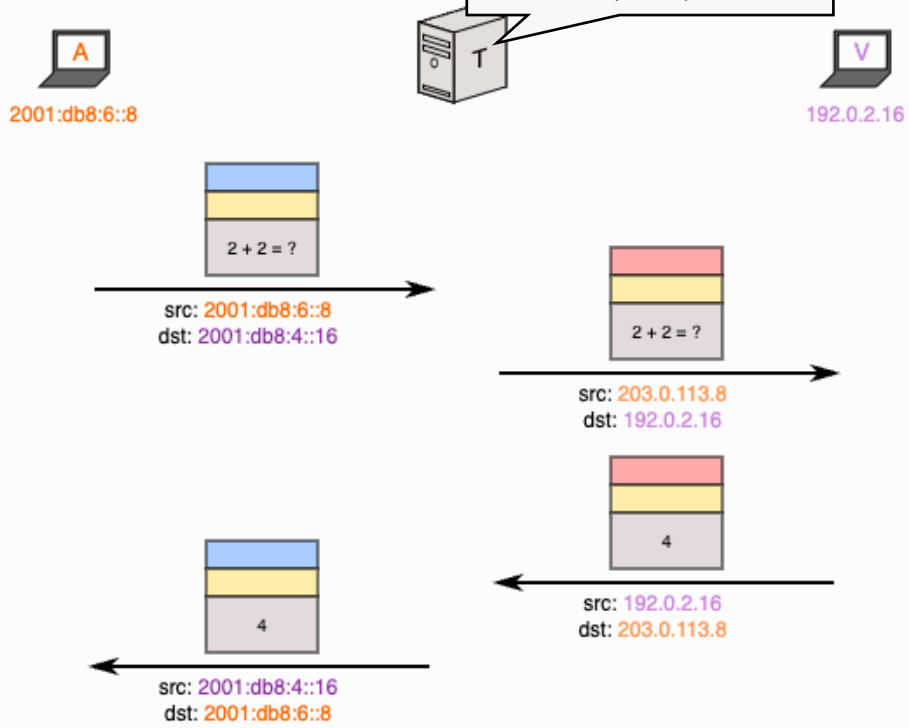
Stateful NAT64



SIIT (Stateless IP/ion)

IPv6	IPv4
2001:db8:6::8	203.0.113.8
2001:db8:6::9	203.0.113.9
2001:db8:6::a	203.0.113.10
2001:db8:6::b	203.0.113.11
2001:db8:6::c	203.0.113.12
2001:db8:4::16	192.0.2.16

Explicit Address Mapping Table (EAMT)



手順6. radvd を用いたルータ広告デーモンの設定

- radvd のインストール
  - 2.20_rc1 で Pref64（RFC8781）をサポート[参考]
    - Ubuntu 25.04（Plucky）以降は公式リポジトリから 2.20 を取得可能
    - 今回はソースコードからビルドする

```
ubuntu@raspi:~$ wget \
            https://radvd.litech.org/dist/radvd-2.20.tar.gz ↩
ubuntu@raspi:~$ tar -zxvf radvd-2.20.tar.gz && cd radvd-2.20 ↩
ubuntu@raspi:~$ sudo apt install \
            pkg-config libbsd-dev bison flex ↩
ubuntu@raspi:~$ ./configure --prefix=/usr/local \
            --sysconfdir=/etc --mandir=/usr/share/man
ubuntu@raspi:~$ make && sudo make install ↩
...
ubuntu@raspi:~$ radvd -v ↩
Version: 2.20
...
```

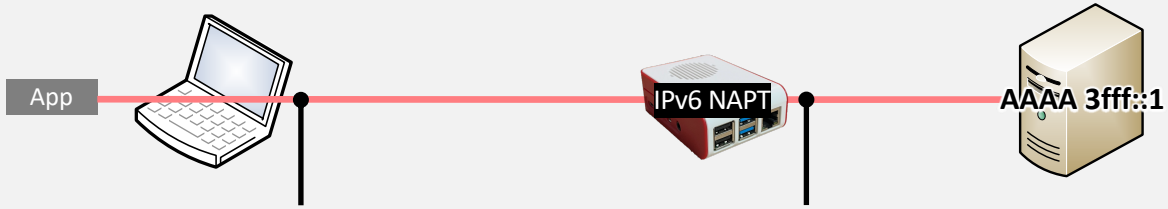
- 設定ファイルの準備
  - Jool で設定した Pref64::/n と同じプレフィックスを設定する

```
ubuntu@raspi:~$ cat /etc/radvd.conf ↩
interface eth0
{
    AdvSendAdvert on;
    ...

    prefix fc00:192:168:200::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime infinity;
        AdvPreferredLifetime infinity;
    };

    ...
    nat64prefix 64:ff9b::/96 {
        AdvValidLifetime 1800;
    };
};
```

動作確認（IPv6 接続性）

- クライアント OS ごとの検証に先立って、構築した実証環境でインターネット接続性が得られるかを確認する
  - IPv6-Only 対応ホストを接続してみる

```
yykzm@MacBook-Air ~ ❯ ifconfig en0
en0: flags=88e3<UP,BROADCAST,SMART,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 02:56:98:95:cf:e2
        inet6 fe80::149d:98be:200c:6bab%en0 prefixlen 64 secured scopeid 0xc
        inet6 fc00:192:168:200:1061:6553:f06d:7c05 prefixlen 64 autoconf secured
        inet 192.0.0.2 netmask 0xffffffff broadcast 192.0.0.2
        inet6 fc00:192:168:200:1857:ab81:4ef4:39ef prefixlen 64 clat46
        nat64 prefix 64:ff9b:: prefixlen 96
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```



IPv6 サーバとの通信

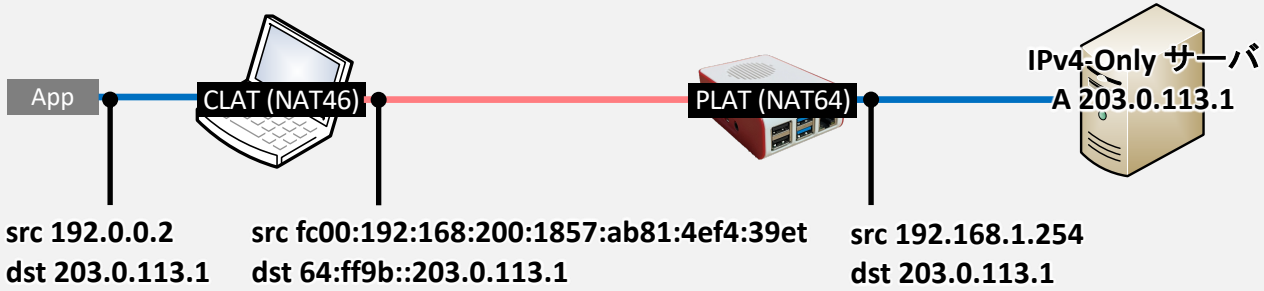
テストサイト例: https://ipv6-test.com

動作確認（IPv4 接続性）

- クライアント OS ごとの検証に先立って、構築した実証環境でインターネット接続性が得られるかを確認する
  - IPv6-Only 対応ホストを接続してみる

```
yykzm@MacBook-Air ~ ❯ ifconfig en0
en0: flags=88e3<UP,BROADCAST,SMART,RUNNING,NOARP,SIMPLEX,MULTI(
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINV(
        ether 02:56:98:95:cf:e2
        inet6 fe80::149d:98be:200c:6bab%en0 prefixlen 64 secur(
        inet6 fc00:192:168:200:1061:6553:f06d:7c05 prefixlen 64 autoconf
        inet 192.0.0.2 netmask 0xffffffff broadcast 192.0.0.2
        inet6 fc00:192:168:200:1857:ab81:4ef4:39ef prefixlen 64 clat46
        nat64 prefix 64:ff9b:: prefixlen 96
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

192.0.0.0/29（192.0.0.2）は、アプリケーションが IPv4 アドレスを使うためのアドレス
→ IPv4サービス継続用プレフィックス [RFC7335]



IPv4 サーバとの通信

テストサイト例: https://ipv6-test.com

4. クライアント OS における動作検証

検証したクライアント OS 一覧

機器	OS	バージョン	リリース日
iPhone 13	iOS	26.0.1	2025/9/29
Macbook Air M2	macOS	Tahoe 26.0.1	2025/9/29
ThinkPad E14	Windows 11	24H2 KB5066835 (26100.6899)	2025/10/14
Pixel 7a	Android	Android 16	2025/6/10
NEC NM550/K	Ubuntu Desktop	24.04 TLS	2024/4/25

検証内容

① クライアントが送信する DHCPv4 Discover の Parameter Request List に **IPv6-Only Preferred (108)** が含まれるか

② ① を満たす場合に、ホストにおいて、ルータ広告のオプションを介して取得した **PREF64 を使った CLAT** が動作するか

検証の結果

OS	バージョン	① 108 オプションの有無	② CLAT の動作	IPv6-Mostly での動作
iOS	26.0.1	あり	OK	IPv6-Only
macOS	Tahoe 26.0.1	あり	OK	IPv6-Only
Windows 11	24H2 KB5066835 (26100.6899)	なし		デュアルスタック
Android	Android 16	あり	OK	IPv6-Only
Ubuntu Desktop	24.04 TLS	なし		デュアルスタック

- モバイル系やApple 製品（Android / iOS / macOS）は IPv6-Only 運用が可能
- Windows 11 でのサポートが期待される
  - CLAT サポートを予定しているが、~~具体的な計画はなし~~ [参考]
  - (2025/11/19 new) プライベートプレビューでのサポートを開始 [参考] ←
- Linux においては、NetworkManager で 「② CLAT」 を開発中 [参考]
  - 「① 108 オプション」はサポート済み [参考]
- IoT や Smart Home でもサポートが期待される

まとめ

- Raspberry Pi を使った IPv6- Mostly 実証環境の構築方法を具体的に紹介
- クライアント OS における動作検証
  - モバイル系やApple 製品は IPv6-Only で動作可能（IPv6-only-capable host）
    - 多くの環境で IPv4 アドレスの節約が実現可能
  - Windows 11 や Linux はデュアルスタック（IPv4-requiring host）
    - 開発は進んでいるため、今後のサポートに期待
    - Winsows は、プライベートプレビューでの CLAT サポートを開始
      - 一般提供への期待が高まる

（付録）検証の詳細

iOS 26.0.1

◆ IPv6-only-capable host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv6-Only で動作可能なホスト

iPhone が送信する DHCPv4 Discover に
IPv6-Only Preferred (108) オプション
が含まれることを確認

```
> Option: (53) DHCP Message Type (Discover)
v Option: (55) Parameter Request List
    Length: 10
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (108) IPv6-Only Preferred
    Parameter Request List Item: (114) DHCP Captive-Portal
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (162) Unassigned
    Parameter Request List Item: (252) Private/Proxy autodiscovery
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

② CLAT の動作確認

iOS 26.0.1

◆ IPv6-only-capable host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv6-Only で動作可能なホスト

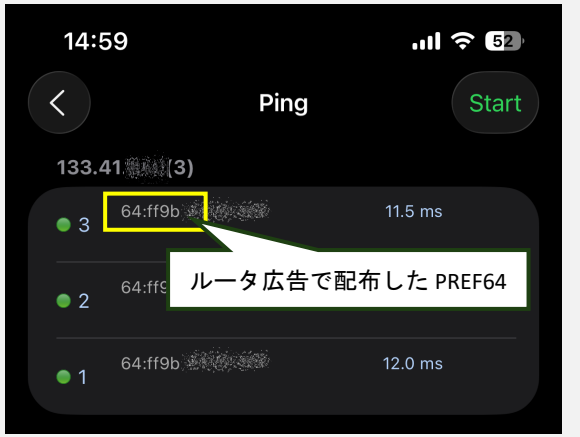
iPhone が送信する DHCPv4 Discover に
IPv6-Only Preferred (108) オプション
が含まれることを確認

```
> Option: (53) DHCP Message Type (Discover)
v Option: (55) Parameter Request List
    Length: 10
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (108) IPv6-Only Preferred
    Parameter Request List Item: (114) DHCP Captive-Portal
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (162) Unassigned
    Parameter Request List Item: (252) Private/Proxy autodiscovery
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

PREF64 と clat46 アドレスを使って IPv4 宛先と
通信ができていることを確認



Network Analyzer を使用して ping を実行している様子

clat46 アドレス

```
fc00:192:168:200:6c:7614:3c15:68df   64:ff9b::                    ICMPv6   Echo (ping) request
64:ff9b::                            fc00:192:168:200:6c:7614:3c15:68df   ICMPv6   Echo (ping) reply id
```

上記の ping の実行を Raspberry Pi でキャプチャした様子

② CLAT の動作確認

macOS 26.0.1

- IPv6-only-capable host であることを確認
  - IPv6-Mostly ネットワークにおいて、IPv6-Only で動作可能なホスト

Macbook Air が送信する DHCPv4 Discover に IPv6-Only Preferred (108) オプション が含まれることを確認

```
> Option: (53) DHCP Message Type (Discover)
v Option: (55) Parameter Request List
    Length: 13
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (108) IPv6-Only Preferred
    Parameter Request List Item: (114) DHCP Captive-Portal
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (162) Unassigned
    Parameter Request List Item: (252) Private/Proxy autodiscovery
    Parameter Request List Item: (95) LDAP [TODO:RFC3679]
    Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
    Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

PREF64 と clat46 アドレスを使って IPv4 宛先と通信ができていることを確認

```
[yykzm@Yokoos-MacBook-Air ~ % ifconfig en0
en0: flags=88e3<UP,BROADCAST,SMART,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether ca:
        inet6 fe8                                          secured scopeid 0xb
        inet6 fc00:192:                  :834:bb11:787f:43a8 prefixlen 64 autoconf secured
        inet 192.0.0.2 netmask 0xffffffff broadcast 192.0.0.2
        inet6 fc00:192:168:200:1c22:b583:75ef:8ffc prefixlen 64 clat46
        nat64 prefix 64:ff9b:: prefixlen 96
        nd6 options=201<PER                MNUD,DAD>
        media: autoselec
        status: active
```

IPv4サービス継続用プレフィックス

ルータ広告で配布した PREF64

clat46 アドレス

clat46 アドレス

```
fc00:192:168:200:1c22:b583:75ef:8ffc  64:ff9b::                    ICMPv6   Echo (ping) request
64:ff9b::                             fc00:192:168:200:1c22:b583:75ef:8ffc  ICMPv6   Echo (ping) reply i
```

ping の実行を Raspberry Pi でキャプチャした様子

② CLAT の動作確認

Windows 11 24H2

◆ IPv4-requiring host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv4 アドレスを必要とするホスト

Windows が送信する DHCPv4 Discover に
IPv6-Only Preferred (108) オプション
が**含まれていないこと**を確認

```
Option: (55) Parameter Request List
    Length: 14
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (31) Perform Router Discover
    Parameter Request List Item: (33) Static Route
    Parameter Request List Item: (43) Vendor-Specific Information
    Parameter Request List Item: (44) NetBIOS over TCP/IP Name Server
    Parameter Request List Item: (46) NetBIOS over TCP/IP Node Type
    Parameter Request List Item: (47) NetBIOS over TCP/IP Scope
    Parameter Request List Item: (119) Domain Search
    Parameter Request List Item: (121) Classless Static Route
    Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
    Parameter Request List Item: (252) Private/Proxy autodiscovery
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

デュアルスタックで動作していることを確認

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : fc00:192:168:200:bb93:c36d:9bc3:7c6d
   Temporary IPv6 Address. . . . . . : fc00:192:168:200:7d82:5b13:42f3:2924
   Link-local IPv6 Address . . . . . : fe80::41e6:768a:4d2:293f%14
   IPv4 Address. . . . . . . . . . . : 192.168.200.150
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::212:e2ff:
                                       fe80::e65f:1
                                       192.168.200.
```

DHCPv4 の アドレスプールから
IPv4 が割り当てられている

Android 16

◆ IPv6-only-capable host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv6-Only で動作可能なホスト

Android が送信する DHCPv4 Discover に
IPv6-Only Preferred (108) オプション
が含まれていることを確認

```
Option: (55) Parameter Request List
    Length: 12
    Parameter Request List Item: (1) Subnet Mask
    Parameter Request List Item: (3) Router
    Parameter Request List Item: (6) Domain Name Server
    Parameter Request List Item: (15) Domain Name
    Parameter Request List Item: (26) Interface MTU
    Parameter Request List Item: (28) Broadcast Address
    Parameter Request List Item: (51) IP Address Lease Time
    Parameter Request List Item: (58) Renewal Time Value
    Parameter Request List Item: (59) Rebinding Time Value
    Parameter Request List Item: (43) Vendor-Specific Information
    Parameter Request List Item: (114) DHCP Captive-Portal
    Parameter Request List Item: (108) IPv6-Only Preferred
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

しかし..

今回構築した実証環境に接続できない..
→ いくつか工夫が必要なことを確認

Android 16: 工夫①

- ユニークローカルアドレスではなく、<u>グローバルユニキャストアドレス</u>を使用する
- Android における IPv6 では、
  - グローバルユニキャストアドレス
  - デフォルトルート
  - DNS サーバ

  が揃っている必要がある

```
ubuntu@raspi:~$ cat /etc/radvd.conf ↵
interface eth0
{
    ...
    # prefix fc00:192:168:200::/64
    prefix 3fff:192:168:200::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
        AdvValidLifetime infinity;
        AdvPreferredLifetime infinity;
    };
    ...
};
```

```
/**
 * Returns true if this link is provisioned for global IPv6 connectivity.
 * This requires an IP address, default route, and DNS server.
 *
 * @return {@code true} if the link is provisioned, {@code false} otherwise.
 * @hide
 */
@SystemApi
public boolean isIpv6Provisioned() {
    return (hasGlobalIpv6Address()
            && hasIpv6DefaultRoute()
            && hasIpv6DnsServer());
}
```

Android 16: 工夫②

- 各種ライフタイムを短くしすぎない
  - Android には、Android Packet Filter（APF）という機能が存在し、電力消費を抑える目的で、ハードウェアでパケットをドロップすることがある
- 例えば、ルータ広告の RDNSS オプションのライフタイムは..

```
yykzm@MacBook-Air ~ ❯ adb shell dumpsys network_stack | grep "Minimum RDNSS lifetime:"
    Minimum RDNSS lifetime: 180
```

最低でも 180 秒必要であることがわかる

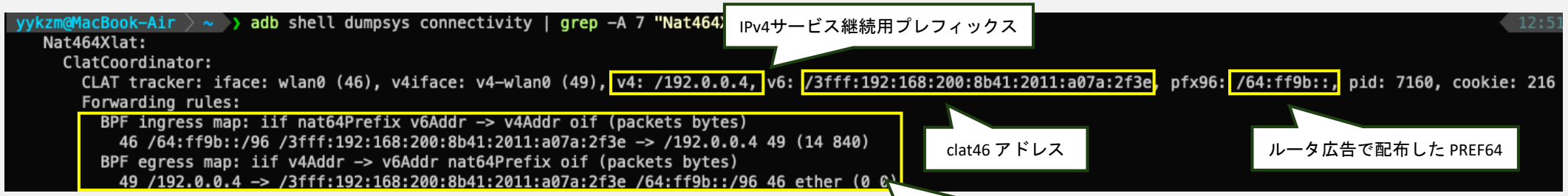
```
ubuntu@raspi:~$ cat /etc/radvd.conf ↵
interface eth0
{
    ...
    RDNSS 2001:db8:1000:1::1
    {
        AdvRDNSSLifetime 270;
    };
    ...
};
```

Android 16

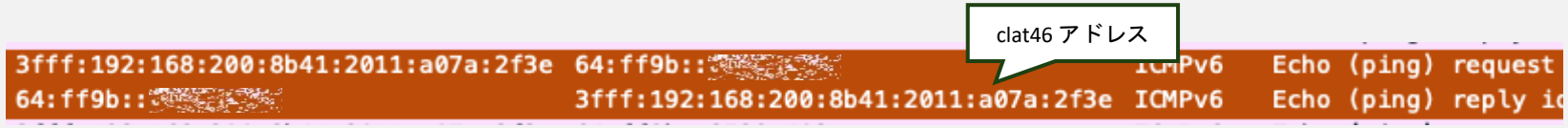
◆ IPv6-only-capable host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv6-Only で動作可能なホスト

PREF64 と clat46 アドレスを使って IPv4 宛先と
通信ができていることを確認



※ ユーザにはこれらの情報は隠蔽されており、
設定画面などからは確認できなかった



ping の実行を Raspberry Pi でキャプチャした様子

② CLAT の動作確認

Ubuntu Desktop 24.04 TLS

◆ IPv4-requiring host であることを確認

❑ IPv6-Mostly ネットワークにおいて、IPv4 アドレスを必要とするホスト

Ubuntu Desktop が送信する DHCPv4 Discover に
IPv6-Only Preferred (108) オプション
が**含まれていないこと**を確認

```
Option: (55) Parameter Request List
   Length: 17
   Parameter Request List Item: (1) Subnet Mask
   Parameter Request List Item: (2) Time Offset
   Parameter Request List Item: (6) Domain Name Server
   Parameter Request List Item: (12) Host Name
   Parameter Request List Item: (15) Domain Name
   Parameter Request List Item: (26) Interface MTU
   Parameter Request List Item: (28) Broadcast Address
   Parameter Request List Item: (121) Classless Static Route
   Parameter Request List Item: (3) Router
   Parameter Request List Item: (33) Static Route
   Parameter Request List Item: (40) Network Information Service Domain
   Parameter Request List Item: (41) Network Information Service Servers
   Parameter Request List Item: (42) Network Time Protocol Servers
   Parameter Request List Item: (119) Domain Search
   Parameter Request List Item: (249) Private/Classless Static Route (Microsoft)
   Parameter Request List Item: (252) Private/Proxy autodiscovery
   Parameter Request List Item: (17) Root Path
```

DHCPv4 Discover パケットを Raspberry Pi でキャプチャした様子

① IPv6-Only Preferred Option の確認

デュアルスタックで動作していることを確認

```
2: wlp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 70:c9:4e:7c:de:a9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.151/24 brd 192.168.200.255 scope global dynamic noprefixroute wlp1s0
       valid_lft 2956sec preferred_lft 2956sec
    inet6 fc00:192:168:200:a3d8:6dfe:e649:8326/64
       valid_lft 604690sec preferred_lft 862
    inet6 fc00:192:168:200:b669:9c81:c946:a9
       valid_lft forever preferred_lft forev
    inet6 fe80::3f73:2fa:82ee:ce23/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

DHCPv4 の アドレスプールから
IPv4 が割り当てられている

- ✓ サードパーティ製のソフトウェアを追加で
  インストールすることで実現はできそう
- ✓ 現状、自動構成ではデュアルスタックとなる