日本で普及した IPv6 閉域網におけるフォールバック問題の検出手法と緩和手法の実装

大石光† 廣海緑里† 永見健一†

2011 年に IPv4 アドレスの在庫が枯渇し，IPv6 への移行の過程でフォールバック問題が表面化している．この問題は，アプリケーションレイヤのソフトウェアにとって他人事ではない．なぜなら，この問題は，ソフトウェア開発のテスト工程での検出が難しいからである．本稿では，これを検出するために，接続性を考慮した通信シナリオを提案する．また，フォールバック問題は，接続先のサーバへの接続時間が約 1 秒遅延する．それに対して，RFC6555 の Happy Eyeballs というアルゴリズムによる緩和手法は，接続時間を数百ミリ秒の遅延に低減する効果がある．本稿では，プログラミング言語 Java で実験的に Happy Eyeballs を実装した際に得た知見を紹介する．

Detection method of the fallback issue in closed IPv6 network that is popular in Japan and an implementation of mitigation method

HIKARI OISHI† RURI HIROMI† KENICHI NAGAMI†

The pools of unallocated IPv4 addresses were exhausted in 2011. The fallback issue comes into the open in the process of migrating to IPv6. This issue is relevant for the application layer software. Because, this issue is difficult to detect during the testing phase of software development. In this paper, we propose communication scenarios considering the connectivity in order to detect this issue. In the fallback issue, connection time to the server is about 1 second delay. By contrast, mitigation method called Happy Eyeballs algorithm standardized in RFC6555 has the effect of reducing connection time in a few hundred milliseconds delay. In this paper, we introduce the knowledge obtained when implementing the Happy Eyeballs experimentally in Java programming language.

1. はじめに

2011 年の IPv4 アドレス在庫枯渇から 1 年が経過し，インターネットは, IPv4 と IPv6 が混在している. つまり, IPv4 アドレスの在庫がなくなっても, 使用中の IPv4 アドレスを使い続けることができ[1]，インターネットが一気に IPv6 に変わるわけではない.

インターネットの IPv4 と IPv6 の混在化の進展[a]により，日本国内では「フォールバック問題」が表面化している[b][2]．これは，IPv6 に対応したクライアントから，IPv6 に対応した接続先のサーバへの接続に，日本国内で普及した IPv6 閉域網内で遅延が発生するという問題で，「フォールバック」という仕組みによって発生する．本稿では，この問題を「フォールバック問題」と呼ぶ.

この問題に対するソフトウェアで実装できる緩和手法として，Happy Eyeballs がある．Happy Eyeballs は，2012 年 4 月にインターネットの標準化団体の IETF(Internet Engineering Task Force) が，RFC(Request for Comments) 6555[3]として標準化した.

2. フォールバック問題とは

フォールバック問題のメカニズムを説明する（図 1 参照）．まず，接続先のサーバが IPv6 に対応した場合，クライアントは IPv6 アドレスで接続を試みる[c]．ところが，IPv6 閉域網には，この IPv6 アドレスへの接続性がないため接続は失敗する．つぎに，クライアントは IPv4 アドレスで接続を試みる．この結果，接続先のサーバへの接続時間が約 1 秒遅延する（日本国内で普及した IPv6 閉域網の場合）.

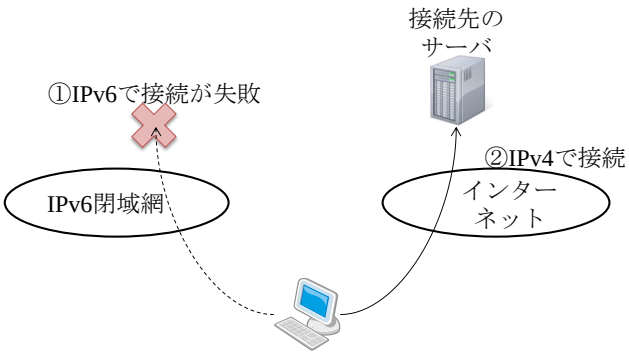

図 1 フォールバックのメカニズム

Figure 1 The fallback mechanism

† 株式会社インテック
INTEC Inc.

a) 2012 年 6 月 6 日に始まった「World IPv6 Launch」では，Google, Yahoo などの大手コンテンツ事業者が参加して，恒久的に IPv6 に対応した.

b) 本稿執筆時点では，ネットワーク側の一時的な対策により，大きな問題は起こっていない.

c) IPv6 閉域網につながったクライアントは，IPv6 閉域網が割り当てた IPv6 アドレスが，IPv6 閉域網のアドレスなのか，インターネットのアドレスなのか区別できない.

クライアントは，名前解決で得た複数の IP アドレスで，順番に接続を試みることでフォールバックする（図 2 参照）．すなわち，まず，クライアントは，接続先のサーバの名前から，名前解決により接続先のサーバの IPv6 アドレスと IPv4 アドレスを取得する．つぎに，クライアントは，IPv6 アドレスで接続を試みて，接続が失敗する．そして，クライアントは，IPv6 アドレスで接続が失敗した後に IPv4 アドレスで接続を試みる．

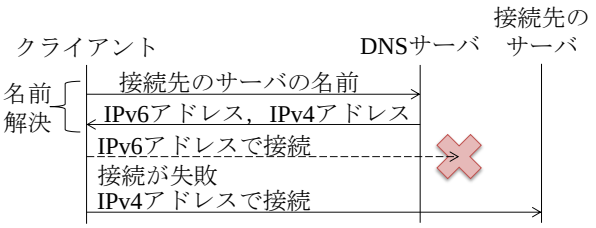

図 2　フォールバックの流れ

Figure 2 The fallback flow

3. フォールバック問題をテスト工程で検出する通信シナリオ

3.1 一般的な通信シナリオと問題点

プログラミング言語 Java を使う開発者が IPv6 に対応する際の参考文献として，「ネットワーク IPv6 ユーザーガイド」[4]がある．本稿では，このガイドにある「通信のシナリオ」を一般的な通信シナリオとする．一般的な通信シナリオに従った場合の組合せは，図 3 に示す 7 通りになる．この通信シナリオは，クライアントとサーバ間の正常系テストに利用できる．

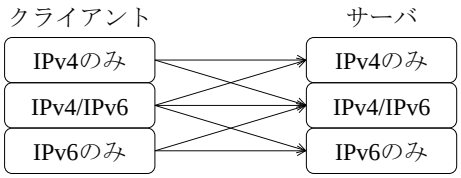

図 3　一般的な通信シナリオ

Figure 3 Typical communication scenarios

しかし，IPv6 から IPv4 に遷移するフォールバック問題は，この一般的な通信シナリオでは検出できない．このため，ソフトウェアを出荷してから，フォールバック問題に気づくという問題点がある．

3.2 接続性を考慮した通信シナリオの提案

われわれは，フォールバック問題の検出が可能な通信シナリオを提案する（図 4 参照）．提案する通信シナリオでは，IPv4/IPv6 対応のクライアントとサーバ間の IPv4 と IPv6 の接続性に着目し，片方のみの通信ができる状態を考慮した．OS やアプリケーションが IPv6 を優先して動作する場合，IPv4 のみの接続性のシナリオで，フォールバック問題が検出できる．OS やアプリケーションが IPv4 を優先して動作する場合，IPv6 のみの接続性のシナリオで，フォールバック問題が検出できる．なお，OS やプログラミング言語には，優先する IP アドレスを決める仕組みがある．

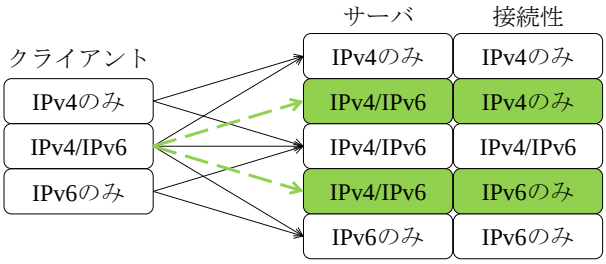

図 4　接続性を考慮した通信シナリオ

Figure 4 Communication scenarios considering the connectivity

3.3 優先する IP アドレスを決める仕組み

OS やプログラミング言語の優先する IP アドレスを決める仕組みを説明する．これらの仕組みと提案する通信シナリオを組み合わせて，クライアントの条件を加味したテスト項目を作ることができる．

クライアントには，優先する IP アドレスを決める仕組みがある．IETF が RFC3484 [5]として標準化したアドレス選択機構で，この機構を実装する OS[d]では，接続先のサーバの名前解決結果を，OS のポリシーテーブルに基づいた IP アドレスのリストに並び替える[6]．つまり，接続先のサーバの IP アドレスは，DNS サーバの設定順序をそのまま利用するのではなく，OS のポリシーテーブルの設定で IPv6 アドレスの利用を優先する場合，IPv6 アドレスを優先する．

一方，プログラミング言語 Java [e]の場合，OS のポリシーテーブルで IPv6 アドレスの利用を優先する場合でも，既定では IPv4 アドレスの利用を優先する仕様である．なお，システムプロパティ java.net.preferIPv6Address を true にする[f]ことで，IPv6 アドレスの利用を優先することができる．

4. フォールバック問題の接続挙動

本章では，フォールバック問題の接続挙動に関して，調査で得た知見を述べる．

4.1 接続挙動を決めるソケットの実装

クライアントと接続先のサーバとの接続挙動を決めるのは，ソケットの実装である．なお，ソケットとは，IP ネットワークでは，IP アドレスとポート番号の組のことで，IPv6 アドレスでも同様である．では，ソケットの使い方を説明する．まず，ソケットは，クライアントが名前解決で

d) 主要な OS では，Windows XP/Vista/7，Linux 2.6.25 以降，BSD 系 UNIX そして Solaris 10 以降で実装している．
e) 筆者が検証した環境は，Java SE 6 Update 9（Windows7）である．
f) System.setProperty("java.net.preferIPv6Addresses","true");で設定できる．この場合，IPv4 にフォールバックしない問題が発生する．

得た IP アドレスから作る．つぎに，クライアントは，作ったソケットを使い, 接続先のサーバに接続する. 要するに，一般のアプリケーション開発者から見ると，ソケットの実装で接続挙動が決まる．そして，プログラミング言語によっては，例えば HTTP 固有の機能をサポートするような，サーバ接続で利用できるクラスを提供しており，開発者から見るとソケットの実装を隠ぺいしている場合がある．この場合，接続挙動は，プログラミング言語のソケットの実装に依存する．つまり，開発者がソケットの実装に手を入れることが難しくなる．

4.2 接続挙動の検証

提案する通信シナリオに基づく IPv4/IPv6 対応のサーバへの接続挙動を検証し，フォールバック問題の発生と将来的なリスクを検出した．表 1 の接続挙動(A)に，クライアントアプリケーションとして Internet Explorer 8 を用いた場合の検証結果を示す.そして，表 2 の接続挙動(B)は，プログラミング言語 Java の標準 API [g]を使用した検証用プログラムを用いた場合の検証結果である．なお,接続挙動(A)(B)ともに，OS のポリシーテーブルが IPv6 アドレスの利用を優先する設定で検証した.

表 1　接続挙動(A)

Table 1 connection behavior (A)

クライアント	サーバ	接続性	接続挙動
IPv4/IPv6	IPv4/IPv6	IPv4 のみ	IPv4 にフォールバックして接続(約 21 秒)
IPv4/IPv6	IPv4/IPv6	IPv4/IPv6	IPv6 で接続
IPv4/IPv6	IPv4/IPv6	IPv6 のみ	IPv6 で接続

接続挙動(A)の結果から, 提案する通信シナリオを用いるとフォールバック問題の検出ができることを検証した．また，今回の検証環境におけるフォールバック問題の検出では, 接続先のサーバへの接続遅延は約 21 秒という結果であった.

しかし, 実際の問題となっている IPv6 閉域網での接続遅延は約 1 秒である．これは，当該 IPv6 閉域網で，フォールバックにかかる遅延の緩和策として，IPv6 閉域網の中で接続性がない IPv6 アドレスに対して, クライアントから接続があった場合, IPv6 閉域網は接続を拒否する TCP リセットと呼ぶエラー応答を返す運用を行っているためである[7]. このようにネットワークでは対策を取っている場合があるため，テスト時に得た接続時間の値と実利用時に得る接続時間の値には違いがあることが予想できる．それでも，フォールバックする環境でのテスト項目の試行という意味において提案する通信シナリオは意義があると考える.

この IPv6 閉域網による緩和手法は, コネクションを確立する TCP 通信には効果がある．しかし，この手法は，コネクションを確立しない UDP 通信には効果がない. なぜなら, リセットは TCP 通信で接続を拒否するための仕組みであり，UDP 通信には，この仕組みがないからである．なお, TCP 通信であっても，障害でタイムアウトする場合には対応できない.

表 2　接続挙動(B)

Table 2 connection behavior (B)

クライアント	サーバ	接続性	接続挙動
IPv4/IPv6	IPv4/IPv6	IPv4 のみ	IPv4 で接続
IPv4/IPv6	IPv4/IPv6	IPv4/IPv6	IPv4 で接続
IPv4/IPv6	IPv4/IPv6	IPv6 のみ	接続できない

一方，接続挙動(B)では，フォールバックによる問題は発生しなかった．しかし，IPv6 のみの接続性のシナリオで接続できないという事象を検出した．これは，IPv4 から IPv6 にフォールバックしないことを意味する．このような環境が出現していない現状では，実環境で問題は起きない．しかし, 将来的に IPv6 のみの接続性を持つネットワーク環境が出現すると，リスクがある.

4.3 フォールバックの実装

フォールバックする場合のアプリケーションからソケット API を利用するシーケンスで，時間がかかる部分を説明する（図 5 参照）. IPv6 アドレスで接続し，ソケット API が IPv6 エラーを検出した後，アプリケーションは IPv4 アドレスで接続を試みる．つまり，IPv6 アドレスで接続してから IPv6 エラーを検出するまでにかかる時間が, フォールバック問題である．この IPv6 エラーを検出するまでの間，アプリケーションは，ブロッキング I/O により処理が停止する．なお，IPv6 エラーには，タイムアウト，ICMP エラー受信[h]および TCP リセット受信がある.

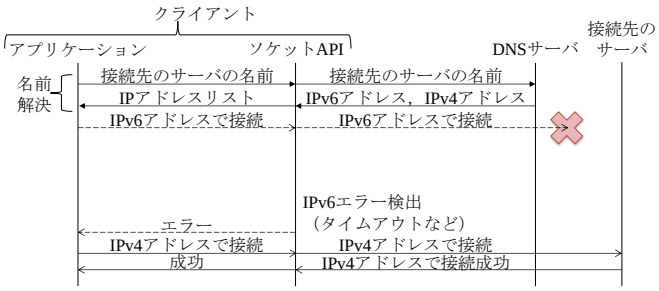

図 5　フォールバックのシーケンス

Figure 5 The fallback sequence

g) 筆者が検証した標準 API は，java.net パッケージの Socket クラスと URL クラスと HttpURLconnection クラス，そして，java.nio.channels パッケージの SocketChannel クラス（Windows 版は IPv6 に関してのバグあり bug_id:623076）である.

h) プログラミング言語 Java で，ICMP エラーメッセージ(TYPE=3,code=13)受信を検証したところ，Windows7 では，タイムアウトと同じ Connection timed out: no further information であった.

さて，このシーケンスには，つぎの 2 つの機能がある．これは IETF が RFC4038[8]で，Example of TCP Client Application として 2005 年 3 月に公知[i]した実装である．

① 名前解決で，複数のアドレスをリストとして扱うようにした．

② アドレスリストの最初のアドレスに接続できなかったら，つぎのアドレスに接続を試みるようにした．

一方，この 2 点に対応しない場合は，フォールバックしない．例えば，古いソフトウェアで，IPv6 に対応していない場合が該当する．しかし，最近のソフトウェアであっても，接続挙動(B)で示したプログラミング言語 Java（調査したバージョンのリリース日は 2011 年 10 月）は，接続挙動から見て，上記の 2 点を満たしていない．なお，プログラミング言語 Java は，IPv6 のみに対応したサーバには接続できる．

5. Happy Eyeballs の実装に関する知見

本章では，Happy Eyeballs の実装実験から得た知見を述べる．

5.1 Happy Eyeballs のアルゴリズムの例

Happy Eyeballs は，フォールバックによる接続時間を低減するソフトウェアで実装できる緩和手法である．そして，Happy Eyeballs のアルゴリズムの例は，RFC6555 で，つぎのように記載している．

① getaddrinfo()（名前解決）を呼び出す．ホストのアドレス優先ポリシーによってソートした IP アドレスのリストが返る．

② そのリスト内の最初のアドレスで接続を試みる．(例えば IPv6 アドレス)

③ その接続が短い時間(先行事例では 300 ミリ秒)内に完了しない場合，他のアドレスファミリ（例えば IPv4）に属する最初のアドレスとの接続を試みる．

④ 確立した最初の接続を使用する．他の接続は破棄する．

それでは，③の「短い時間」が必要な理由と推奨値を説明する．短い時間が必要な理由は，RFC6555 によると，同時接続によるネットワークへのスラッシングを避けるためである．このほかの理由を考えると，接続を受け付けるサーバで，1 つの接続で良いところに，2 つ以上の接続が同時に発生するのは好ましくないからである．なお，RFC6555 によると，この短い時間は 150 ミリ秒～250 ミリ秒を推奨する．

5.2 接続挙動の改善

提案する通信シナリオに基づく Happy Eyeballs を実装した場合，IPv4/IPv6 対応のサーバへの接続挙動において，フォールバック問題を緩和する効果を検証できた（表 3 参照）．なお，この接続挙動は，OS のポリシーテーブルが IPv6 アドレスの利用を優先する設定で検証した．

表 3 Happy Eyeballs の接続挙動
Table 3 connection behavior of Happy Eyeballs

クライアント	サーバ	接続性	接続挙動
IPv4/IPv6	IPv4/IPv6	IPv4 のみ	IPv4 で接続 (短い遅延あり)
IPv4/IPv6	IPv4/IPv6	IPv4/IPv6	IPv6 で接続
IPv4/IPv6	IPv4/IPv6	IPv6 のみ	IPv6 で接続

このように，従来は，フォールバックで時間がかかっていた部分が，短い遅延になる．この短い遅延は，500 ミリ秒に実装して検証し，実際の遅延は 500 ミリ秒であった．この遅延は，日本国内で普及した IPv6 閉域網の緩和手法における約 1 秒の遅延に比べて，短くなる．なお，Happy Eyeballs は，IPv6 閉域網の緩和手法では対応できない UDP 通信[j]やタイムアウトに対応できる．

5.3 接続時間の低減

Happy Eyeballs を実装した場合のアプリケーションからソケット API を利用するシーケンスでは，アプリケーションが IPv6 エラーの検出を待たないことを説明する（図 6 参照）．ソケット API が IPv6 エラーを検出する前に，アプリケーションが IPv4 アドレスで接続を試みることで，接続時間が低減する．このように実装するには，ブロッキング I/O によりアプリケーションの処理が停止する制限を，何かしらの手法で回避する必要がある．

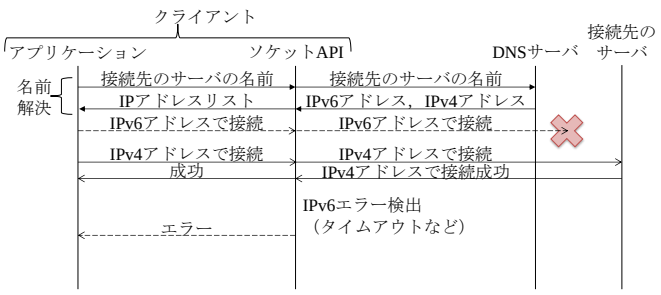

図 6 Happy Eyeballs のシーケンス
Figure 6 Happy Eyeballs sequence

5.4 実装する際の留意点

Happy Eyeballs を実装する際は，一般的な IPv6 対応コードに比べると，つぎに示すような留意点がある．

① 従来は 1 つのソケットを使いまわして実装できた．ところが，複数のソケットを同時に扱うため，非同期 I/O またはスレッドを使う．すなわち，これらが，ブロッキング I/O を回避する手法である．

② 最初の接続を試みてからつぎの接続を試みるまでの時間（タイマ）を検討する．

i) この RFC のカテゴリは Informational であり，標準ではない．

j) 要求に対して応答を待つようなアプリケーションの場合に適用できる．

③ 名前解決の際，IPv6 アドレスが複数あったり，IPv4 アドレスが複数あったりする場合を考慮し，他のアドレスファミリを 2 番目にするために，IP アドレスのリストを並べ替える．

④ IPv4 で接続が成功した後に IPv6 エラーを検出する場合の処理が，接続後の処理に悪い影響を与えないように注意する．なお，IPv6 で接続が成功した後に IPv4 エラーを検出する場合も同様である．

それでは，②のタイマの考え方を述べる．タイマの値は，接続を開始してから接続が成功するまでの平均時間より長く設定するのが良いと思う．しかし，この平均時間は，クライアントから接続先のサーバまでの接続環境によって異なり，事前に決めることが難しいのである．これは，今後の課題である．

6. おわりに

フォールバック問題は，日本のインターネット利用者がターゲットのソフトウェアにとって，検討課題である．なぜなら，2012 年 3 月末時点で約 1,656 万人[9]が，日本国内で普及している IPv6 閉域網を利用しているからである．

このフォールバック問題の起こる環境で，ソフトウェアの接続挙動がどうなるかを調べるには，提案する通信シナリオを使えばよい．そして，ソフトウェアで，Happy Eyeballs を実装するかの判断は，ネットワーク側の対策の状況[k]が検討材料になる．

さて，ソケットレベルのクラスを使ったプログラミングをしたことのない一般のアプリケーション開発者が，それぞれのアプリケーションで，Happy Eyeballs を実装するのは，現実的ではない．プログラミング言語やフレームワークで Happy Eyeballs を実装することにより，一般のアプリケーション開発者が，Happy Eyeballs を簡単に利用できるようになることが望ましいのである．例えば，プログラミング言語 Java の場合，従来では Socket(String host, int port)といったコンストラクタに，タイマ値(long)を追加すると，Socket(String host,int port,long timer)のようなコンストラクタを作ることができると思う．IPv6 対応とフォールバック問題の対策は，Java に限らず他のプログラミング言語に共通した問題である．本稿がプログラミング言語における IPv6 対応の一助となれば幸いである．

謝辞 フォールバック問題の検出手法と緩和手法の調査に，ご協力頂いた皆様に，謹んで感謝の意を表する．

参考文献

1) 日本ネットワークインフォメーションセンター：IPv4 アドレスの在庫枯渇に関して，日本ネットワークインフォメーションセンター（オンライン），入手先（http://www.nic.ad.jp/ja/ip/ipv4pool/ ）（参照 2012-04-26）．
2) IPv4 アドレス枯渇対応タスクフォース：World IPv6 Launch の実施と IPv6 対応の促進について(周知)，IPv4 アドレス枯渇対応タスクフォース（オンライン），入手先（http://www.kokatsu.jp/blog/ipv4/news/120416-v4exhTF-press.pdf）（参照 2012-07-20）．
3) D.Wing and A.Yourtchenko: Request for Comments: 6555: Happy Eyeballs: Success with Dual-Stack Hosts, IETF (online), available from (http://www.ietf.org/rfc/rfc6555.txt) (accessed 2012-04-26).
4) ORACLE：ネットワーク IPv6 ユーザーガイド，ORACLE（オンライン），入手先（http://docs.oracle.com/javase/jp/6/technotes/guides/net/ipv6_guide/index.html ）（参照 2012-04-26）．
5) R. Draves: Request for Comments: 3484: Default Address Selection for Internet Protocol version 6 (IPv6), IETF (online), available from(http://www.ietf.org/rfc/rfc3484.txt)(accessed 2012-04-26).
6) 加藤 淳也：プロトコル非依存のソケットプログラミングの基礎，T5 IPv4 アドレス枯渇時代のアプリケーション開発，pp.39-44，Internet Week 2011（2011）．
7) IPv4 アドレス枯渇対応タスクフォース：World IPv6 Day への対応について(第２版)，IPv4 アドレス枯渇対応タスクフォース（オンライン），入手先（http://www.kokatsu.jp/blog/ipv4/news/IPv4TF-W6D-v2.pdf ）（参照 2012-04-26）．
8) M-K.Shin,Ed., Y-G.Hong, J.Hagino, and P.Savola, E.M.Castro: Request for Comments: 4038: Application Aspects of IPv6 Transition, IETF (online), available from (http://www.ietf.org/rfc/rfc4038.txt) (accessed 2012-07-20).
9) 社団法人日本インターネットプロバイダー協会(JAIPA)：World IPv6 Launch に向けた ISP の対応，IPv6 によるインターネットの利用高度化に関する研究会説明会資料，総務省（オンライン），入手先（http://www.soumu.go.jp/main_content/000159750.pdf）（参照 2012-07-20）．

k) ネットワーク側の対策として，名前解決の際に IPv4 アドレスだけを見せてフォールバック問題が起きないようにした．