

詳解IPv6プロトコルの拡張機能 ～ IPv6実導入に向けて～

NTT情報流通プラットフォーム研究所

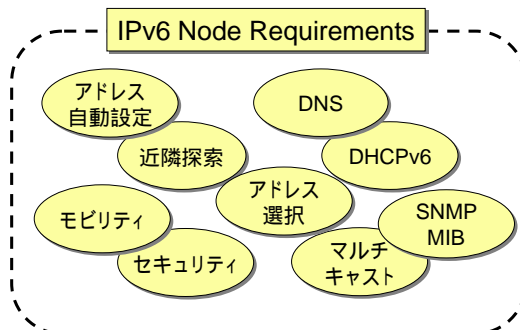
加藤淳也 <kato@syce.net>

2005/12/06

1

チュートリアルのアウトライン



- IPv6技術の概要と標準化経緯 (quick review)
 - IPv6 Node Requirements を概観
 - IPv6ノードの機能項目を網羅したドキュメント
 - MUST～MAY機能まで網羅
 - 入門書では扱われることが少ない拡張機能の動向を紹介
 - その他のトピック
 - 標準化最新動向
 - アプリケーション構造
- 実環境への導入に必要な周辺技術の洗い出しに役立てて下さい



2

IPv6 quick review - 特徴 -

NTT Information Sharing Platform Laboratories

- 広大なアドレス空間の活用
 - IPv4:IPv6 = バケツ1杯:太陽の体積
 - 経路集成による経路膨張の抑止  :
 - グローバルアドレスによりエンド間のP2P通信 
- プラグアンドプレイ
 - アドレス自動設定機能が標準
- セキュリティ機能 (IPsec) 搭載のMUST化
- QoS / マルチキャスト
- 経路最適化をサポートしたモビリティ
- より洗練されたプログラミングAPI

3

IPv6 quick review - 標準化の経緯 -

NTT Information Sharing Platform Laboratories

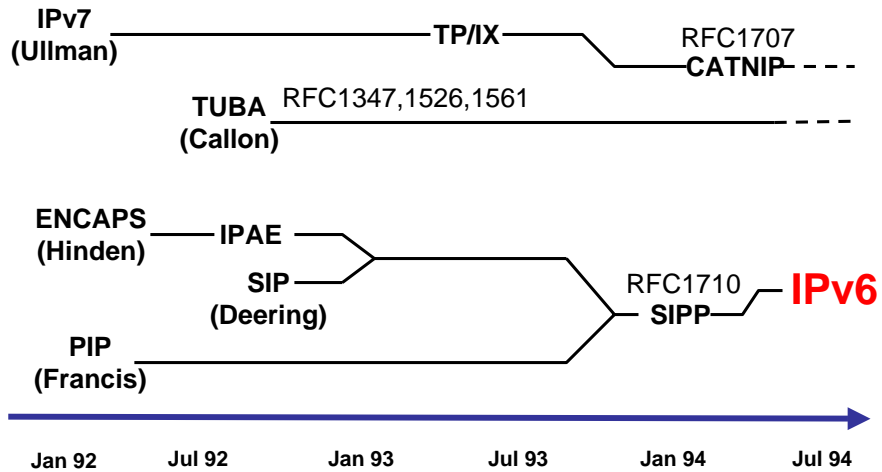
- 1991年7月
 - IPv4アドレスが不足するという研究報告に対するIETFの調査開始
- 1992年11月
 - RFC1380, “IESG Deliberations on Routing and Addressing”において調査結果報告
 - 次世代インターネットプロトコル (IPng) 検討開始
- 1993年12月
 - RFC1550, “IP: Next Generation (IPng) White Paper Solicitation”において要求条件提案
- ~ 1994年
 - さまざまなプロトコルが提案され、廃案、マージを繰り返し
- 1995年1月
 - RFC1752, “The Recommendation for the IP Next Generation Protocol”においてIPv6へと改名

4

IPngの候補たち (RFC1752)

NTT Information Sharing Platform Laboratories

• 4つの系列



5

Internet Protocolのバージョン番号

<http://www.iana.org/assignments/version-numbers>

NTT Information Sharing Platform Laboratories

Assigned Internet Version Numbers

Decimal	Keyword	Version
0		Reserved
1-3		Unassigned
4	IP	Internet Protocol
5	ST	ST Datagram Mode
6	IPv6	Internet Protocol version 6
7	TP/IX	TP/IX: The Next Internet
8	PIP	The P Internet Protocol
9	TUBA	TUBA
10-14		Unassigned
15		Reserved

6

IPv6標準化の経緯(基本仕様確定まで)

NTT Information Sharing Platform Laboratories

- 1995年12月
 - RFC1884, “IP Version 6 Addressing Architecture”において128bitアドレス空間の利用方法提案。最新版RFCが発行間近(draft-ietf-ipv6-addr-arch-v4-04.txt)
- 1997年4月
 - RFC2133, “Basic Socket Interface Extensions for IPv6”においてプログラミングAPIが定まる(現在は、RFC3493(2003))
- 1998年12月
 - RFC2460, “Internet Protocol, Version 6 (IPv6) Specification”などにおいて標準仕様決定
 - この時点でIPv6の格子がほとんど固まる。
- これ以降
 - 基本仕様の改定と拡張機能の標準化が主となる

7

IPv6基本仕様の改定状況

NTT Information Sharing Platform Laboratories

- 1995年12月～1996年8月にかけて Proposed Standard (PS)で仕様化
 - RFC1883 IPv6プロトコル仕様
 - RFC1885 ICMPv6
 - RFC1970 近隣探索
 - RFC1971 ステートレスアドレス自動設定
- 1998年12月 各仕様が Draft Standard (DS)化する
 - RFC2460 IPv6プロトコル仕様
 - RFC2461 近隣探索
 - RFC2462 ステートレスアドレス自動設定
 - RFC2463 ICMPv6
- 現在はマイナーアップデートが進行中(DSとしてRFC化予定)
 - draft-ietf-ipv6-2461bis-05.txt 近隣探索
 - draft-ietf-ipv6-rfc2462bis.txt ステートレスアドレス自動設定
 - draft-ietf-ipngwg-icmp-v3.txt ICMPv6

FULL STANDARD化に向けてのステップアップ

8

IPv6 Node Requirements draft-ietf-ipv6-node-requirements-11.txt の概要

NTT Information Sharing Platform Laboratories

- IPv6ノードが満たすべき要件を定義
 - 機能項目が列挙され、それぞれ詳細仕様を参照
 - MUST～MAYまで網羅的に記載されている
 - 機能リファレンスとしてよく整理されている
- RFCとしての発行が近いが
 - 近隣探索などの周辺仕様の改訂待ち
- 本セッションは入門書では解説が少ない
 - 実用のためより踏み込んだ内容
 - 標準化途中の内容に注目します

9

IPv6 Node Requirements の大まかな節構成

NTT Information Sharing Platform Laboratories

- Sub-IPレイヤ
 - データリンクとの連携
- IPレイヤ
 - 基本プロトコル仕様
 - 近隣探索
 - ICMPv6
 - アドレス自動設定
 - マルチキャスト
 - アドレス選択方式
- DNS / DHCPv6
- IPv4から移行技術
- モビリティ
- セキュリティ
- ルータ機能
- ネットワーク管理

10

IPv6 Node Requirements アドレス自動設定に 関するトピック

11

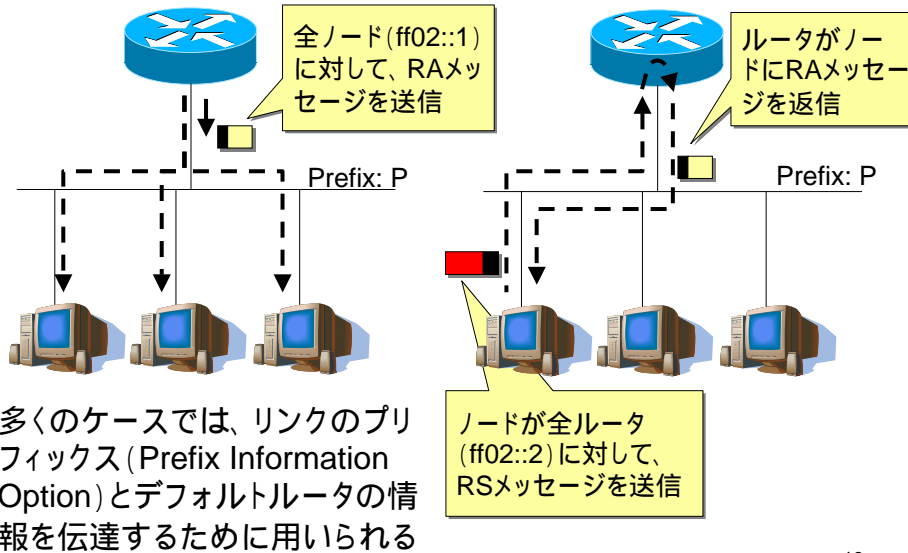
アドレス自動設定

- アドレス自動設定のための2つのモード
 - Stateless Address Autoconfiguration
 - 特徴
 - ノードのアドレスを知ることができない
 - アドレスの割り当てを制限できない
 - 実装例
 - Router Advertisement (RA)
 - Stateful Address Autoconfiguration
 - 特徴
 - ノードごとに状態を管理できる
 - アドレスの割り当てを制御できる
 - 実装例
 - DHCPv6
- ステートレス, ステートフル間の連携方法も議論中
 - RAにM, Oの2つのフラグを増設し、RAとDHCPv6を連動

12

Router Advertisement (RA) と Router Solicitation (RS)

NTT Information Sharing Platform Laboratories



13

Router Advertisement メッセージのフォーマット ManagedFlag (M) と OtherConfigFlag (O)

NTT Information Sharing Platform Laboratories

0				1				2				3											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
Type				Code				Checksum															
Cur Hop Limit				M O Reserved				Router Lifetime															
				Reachable Time																			
				Retrans Timer																			
Options ...																							

- M/OフラグフィールドはRFC2461, 2462で規定されている

14

M flag: ManagedFlag
Managed address configuration

NTT Information Sharing Platform Laboratories

- RAのPrefix Informationオプションを用いず、ステートフル手法でアドレスの割当てを要求
 - ステートフルな手法としてDHCPv6を想定する
- 連携方法
 - ノードがRAを受信後、Mフラグを検査。オンならばDHCPv6クライアントを起動しアドレス割り当てを要求する
- RAとの比較
 - ステートフルである(網側からノード管理できる)
 - RAでは渡せない情報(NTP, STPサーバ等)をノードに伝達できる

15

O flag: OtherConfigFlag
Other configuration

NTT Information Sharing Platform Laboratories

- アドレス以外の情報(DNS, NTPサーバなど)をステートレスDHCPv6により割り当てる
- 連携方法
 - ノードがOフラグ付きのRAを受信後、DHCPv6クライアントを起動しサーバからアドレス以外の情報を取得
 - アドレスを割り当てる点以外は動作はMフラグとほぼ同じ

16

ステートレス (RA) とステートフル (DHCPv6) 設定方式の周辺動向

NTT Information Sharing Platform Laboratories

- 詳細な使用方法についての議論
 - draft-ietf-ipv6-ra-mo-flags-01.txt
 - 2ビットのフラグに対して状態が多く、動作が未整理だった
 - 第64回のIETF会合で方向性が決着
- ルータ側の実装状況
 - 多くの実装でM/OフラグのON/OFFの設定は可能
- ノード側の実装状況
 - 実装はほとんど進んでいない
 - KAMEでは、RSに回答したOフラグつきRAを受信時に任意のスクリプトを起動可能

```
rtsol -O [script_name]
```

17

“Privacy Extensions for Address Configuration in IPv6”, RFC3041

NTT Information Sharing Platform Laboratories

- [標準化背景] MACアドレスからEUI-64で生成したインターフェイス識別子では、ホストが容易に特定されてしまう
 - Temporary (一時) アドレス
 - Anonymous (匿名) アドレスとも呼ばれる
- インターフェイス識別子をランダムに決定し、IPv6アドレスを一定時間で使い捨てる方式
- 短期間でアドレスが変わるのでサーバの運用には不向き
- IPv6 Node Requirementsでは[SHOULD]と規定
 - Windows XP ではデフォルトで有効化

18

匿名アドレスの有効期限の扱い

NTT Information Sharing Platform Laboratories

- RFC3041でのデフォルト値
 - 推奨有効期限が24時間
 - 最大有効期限が7日間

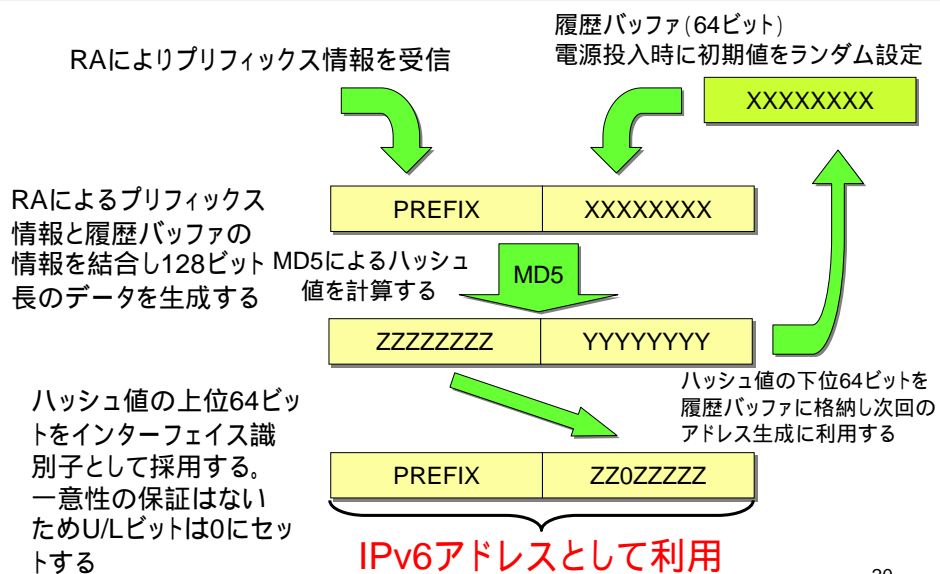
期限の延長は不可
- 参考: EUI-64によるアドレスのデフォルト値
 - 推奨有効期限が7日間
 - 最大有効期限が30日間

期限の延長が可能

19

匿名アドレス生成のアルゴリズム

NTT Information Sharing Platform Laboratories



20

匿名アドレスの効果

NTT Information Sharing Platform Laboratories

- 履歴バッファとMD5ハッシュを使うことで
 - 現在のアドレスから次のアドレスの予想が困難
 - 現在のアドレスから過去のアドレスが推測が困難
- 履歴バッファを持たない組み込み機器などは単純なランダム生成でもよい
- 改訂版の議論が継続
 - draft-ietf-ipv6-privacy-addrsv2-04.txt

21

NTT Information Sharing Platform Laboratories

IPv6 Node Requirements アドレス選択に関するトピック

22

Default Address Selection for Internet Protocol version 6 (IPv6), RFC3484

NTT Information Sharing Platform Laboratories

- IPv6ではひとつのI/Fに複数のアドレスを付与できる
 - 多くのケースでマルチアドレスを構成
 - リンクローカルアドレス
 - グローバルアドレス
 - = (自分自身を見たとき) 複数のソースアドレス(A)の中からどれを通信に使用するか選択基準が必要
 - = (外から見たとき) 複数のあて先アドレス(B)の中からどこに向けて通信を開始するか選択基準が必要
- RFC3484の概要
 - ホストにおけるデフォルトのアドレス選択ルールを定義
 - IPv6 Node Requirementsでは実装を[MUST]とする

23

(A) ホストでのソースアドレス選択ルール

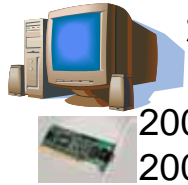
NTT Information Sharing Platform Laboratories

1. スコープの比較
 - 例: あて先がリンクローカルアドレスならば、リンクローカルアドレスをソースとして選択する
2. deprecated (推奨有効期限切れ) なアドレスは使用しない
3. ホームアドレスを使用 (Mobile IPv6)
4. ユーザが設定したポリシー(ルール)で選択
5. Longest Match ルール
 - アドレスをビット列とみなして一致する長さで選択
6. I/Fの持つアドレスの並び順 (実装依存)

24

デフォルトのソースアドレス選択ルール Longest Match

NTT Information Sharing Platform Laboratories



ホストは1つのインターフェイスを持ち、
2つのアドレスが付与されている

2001:db8:aaaa::a/64 ...

2001:db8:bbbb::b/64 ...

- 「2001:db8:abcd::ef」があて先するとき、のどちらが
ソースアドレスになるか？
プリフィックスの最長一致(longest match)で決定する

2001:db8: {
ab(hex): 1001 1010 ... あて先アドレス
aa(hex): **1001 1001** ... → ソースアドレ
bb(hex): **1010 1010** ... スとして選択

25

(B) あて先アドレスの選択について

NTT Information Sharing Platform Laboratories

- サーバのホスト名 “www.example.com” を解
決したところ、合計4つのアドレスが得られた
– IPv6
 - 2001:db8:aaaa:80, 2001:db8:bbbb:80
- IPv4
 - 192.16.178.80, 192.16.179.80
- このとき、Webブラウザは ~ のどのアド
レスへアクセスを試みるのか？

26

(B) あて先アドレスの選択について (cont.)

NTT Information Sharing Platform Laboratories

- 標準状態での接続試行の順序
 - step 1. すべてのIPv6アドレスに接続を試行 (,)
 - step 2. すべてのIPv4アドレスに接続を試行 (,)
 - step 3. 接続失敗

リゾルバが返却したアドレスのリスト

2001:db8:aaaa:80 2001:db8:bbbb::80

↓

192.16.178.80 192.16.179.80
- 同じプロトコルでの順序はリゾルバに依存。ラウンドロビンで入れ替え
 - 名前解決 1回目
2001:db8:aaaa:80, 2001:db8:bbbb::80
 - 名前解決 2回目
2001:db8:bbbb::80, 2001:db8:aaaa:80
 - 名前解決 3回目
2001:db8:aaaa:80, 2001:db8:bbbb::80

27

ユーザが設定可能なアドレス選択のための ポリシーデータベース

NTT Information Sharing Platform Laboratories

- ソースアドレス(A)、あて先アドレス(B)ともにユーザが優先させるアドレスを明示的に指定可能
IPv6スタック内部に下表の構造のデータベースを構成

Prefix	Precedence	Label	Prefixの意味
::1/128	50	0	ループバックアドレス
:::0	40	1	IPv6アドレス
2002::/16	30	2	6to4アドレス
::/96	20	3	互換アドレス
::ffff:0:0:/96	10	4	IPv4アドレス

- あて先アドレス
 - 優先度の高いものからマッチ
- ソースアドレス
 - あて先アドレスと同じラベルのものとマッチ

RFC3484で規定されるデフォルトポリシーデータベース

28

ポリシーデータベースの実装状況

NTT Information Sharing Platform Laboratories

- Windows XP以降, KAME/BSD, Solaris 10に実装
- USAGI/Linuxは未実装(2005年12月現在)

Windows XP でのポリシーデータベースの確認

```
C:¥>netsh interface ipv6 show prefixpolicy
アクティブ状態を照会しています...
```

Precedence	Label	Prefix
10	4	::ffff:0:0/96
20	3	::/96
30	2	2002::/16
40	1	::/0
50	0	:::1/128

29

ポリシーデータベースの運用例(1) あて先アドレス選択の制御

NTT Information Sharing Platform Laboratories

IPv4 IPv6の順で接続させたい

リゾルバが返すアドレスリストの順序を制御

Prefix	Precedence	Label	Prefixの意味
:::1/128	50	0	ループバックアドレス
::/0	40	1	IPv6アドレス
2002::/16	30	2	6to4アドレス
::/96	20	3	互換アドレス
::ffff:0:0/96	60	4	IPv4アドレス

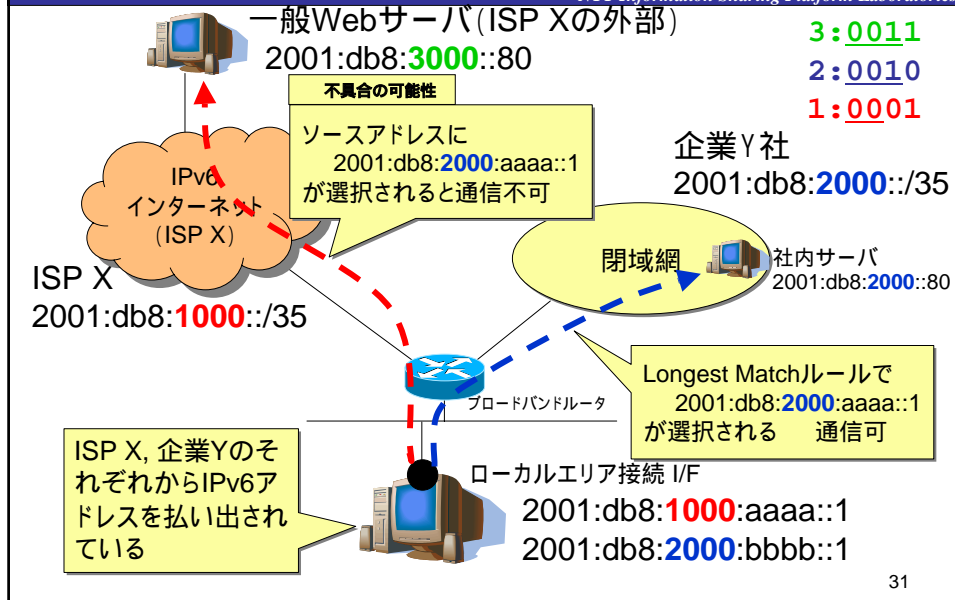
```
www.example.com IN AAAA 2001:db8:aaaa::80 ..
                   IN A    192.16.178.80  ..
```

リゾルバがアプリケーションに返すアドレスの順序
192.16.178.80, 2001:db8:aaaa::80

30

ポリシーデータベースの運用例(2) 特定ソースアドレスの使用を抑制する

NTT Information Sharing Platform Laboratories



ポリシーデータベースの運用例(2) 特定ソースアドレスの使用を抑制する(cont.)

NTT Information Sharing Platform Laboratories

- インターネットと通信するときは、ISP Xのアドレス (2001:db8:1000::) を必ず用いる
- = インターネットと通信するときは、企業Yのアドレス (2001:db8:2000::) を絶対使わない

通信相手	ソースアドレス
2001:db8:3000::80 · A	2001:db8:1000:aaaa::1 · B
	2001:db8:2000:bbbb::1 · C

Prefix	Precedence	Label
::1/128	50	0
:::0	40	1 A B
2002::/16	30	2
:::/96	20	3
2001:db8:2000::/35	45	99 C
::ffff:0:0:/96	10	4

Step 1:
あて先Aのラベルを求める

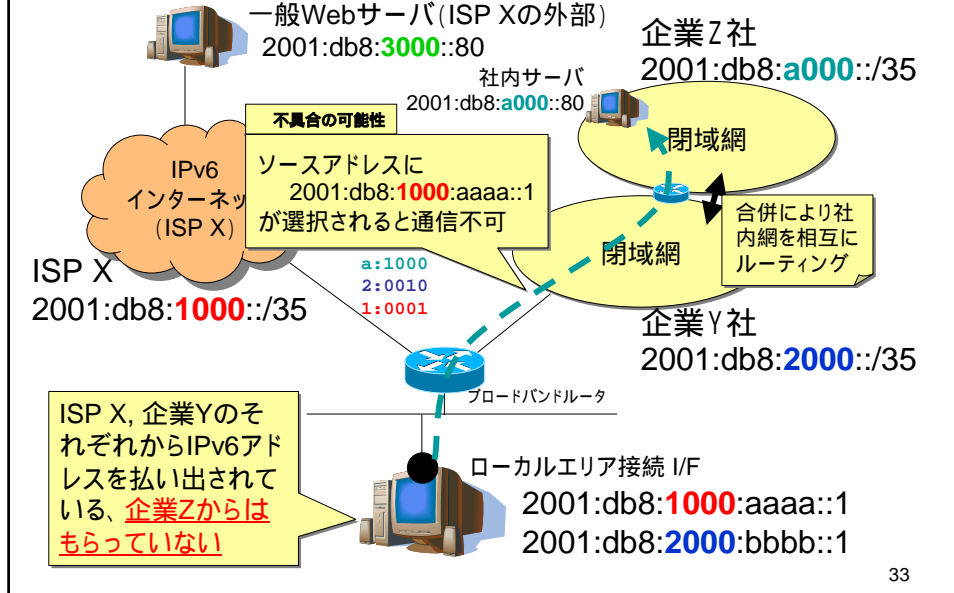
Step 2:
ソースアドレスB,Cのラベルを求める

Step 3:
Step 1.とStep 2.で求めラベルを比較して、同じラベルのソースアドレスを選択する

32

ポリシーデータベースの運用例(3) あて先アドレスに応じて特定ソースアドレスを使用する

NTT Information Sharing Platform Laboratories



33

ポリシーデータベースの運用例(3) あて先アドレスに応じて特定ソースアドレスを使用する (cont.)

NTT Information Sharing Platform Laboratories

- 企業Zと通信するときは、企業Yのアドレス (2001:db8:2000::) を必ず用いる
- インターネットと通信するときは、ISP Xのアドレス (2001:db8:1000::) を必ず用いる

通信相手 2001:db8:a000::80 ·· A
ソースアドレス 2001:db8:1000:aaaa::1 ·· B
2001:db8:2000:bbbb::1 ·· C

Prefix	Precedence	Label
::1/128	50	0
::/0	40	1 B
2002::/16	30	2
::/96	20	3
2001:db8:2000::/35	45	99 C
2001:db8:a000::/35	45	99 A
::ffff:0:0:/96	10	4

Step 1:
あて先Aのラベルを求める

Step 2:
ソースアドレスB,Cのラベルを求める

Step 3:
Step 1.とStep 2.でもとめたラベルを比較して、同じラベルのソースアドレスを選択する

34

IPv6 Node Requirements DNS & DHCPv6 に関するトピック

35

IPv6 DNSの現状

- IPv6に対応したDNSは標準化も完了済
 - “DNS Extensions to Support IP Version 6”, RFC3596 (Draft Standard)
 - 実装もほぼ出揃っている
 - ただしWindows XP SP2はIPv6トランスポートでのDNSクエリは未実装
 - Windows Server 2003は対応済
 - Windows Vista (Longhorn)も対応予定
- 最近までホストがDNS情報を取得する方法 (DNSディスカバリ) が議論されていた

36

ホストにおけるDNS情報の取得法

NTT Information Sharing Platform Laboratories

- IETFで長らく議論が続き3つの方式に集約される
 1. Router Advertisement (RA)
 2. DHCPv6 option
 3. Well-known Anycast Addresses
- 各手法の長短所を比較したドキュメントが Informational RFCとして発行間近
 - “IPv6 Host Configuration of DNS Server Information Approaches”, draft-ietf-dnsop-ipv6-dns-configuration-04.txt
 - どの方式が標準的に利用されるかは不明。機器・OSベンダーのデファクトにゆだねられる

37

1. Router Advertisement (RA) による DNS情報の取得

NTT Information Sharing Platform Laboratories

- 特徴
 - Router Advertisementにオプション(RDNSS)を増設し、ルータとブリッジの通知と同時にDNSサーバの通知を行う
- 長所
 - 既存のRA/NDの方式の上に構築(DNS通知はオプションのひとつ)
 - NTP, SIPなど他のパラメータも(原理的には)通知可能
 - RAが動作するリンクなら、どのメディアでも動作可
 - オーバヘッドが小さい
- 短所
 - カーネル内部の実装を必要とする
 - レイヤ3がNTP, SIPなどアプリケーションの設定を扱うべきか疑問視
 - RAを送信するルータの存在を仮定している

38

2 . DHCPv6 optionを用いたDNS情報の取得

NTT Information Sharing Platform Laboratories

- 概要
 - DHCPv6-liteによりDNSサーバの情報を取得する。DHCPv6-liteはサーバ情報の取得のため軽量化したDHCPv6
- 長所
 - DHCPv6とそのDNS通知オプションは標準化済。相互接続可能な実装が多く存在する
 - NTP, SIPのような他のパラメータも通知可能
- 短所
 - 他の2つの手法に比べオーバーヘッドが大きい
 - DNSのためだけにDHCPv6は重過ぎるという意見もある
- 実装について
 - ルータベンダー各社の実装が増えている
 - Linux, BSD, Windows向けにフリーの実装が存在
 - Windows Vista(beta1)ではDHCPv6を利用する設定項目がある
 - ただし動作は未確認

39

3 . Well-known Anycast Addresses による DNSサーバへのアクセス

NTT Information Sharing Platform Laboratories

- 概要
 - 規定の (well-known) アドレスをDNSサーバとする
- 長所
 - あらかじめ機器にDNSのアドレスを埋め込める
 - DNS情報取得のためのオーバーヘッドがない
- 短所
 - 指定したアドレスで、DNSサーバへの到達性を確保(ルーティング)をする必要がある
 - IP層(L3)の設定が煩雑になる

40

Well-knownサイトローカルアドレスによる DNSとの通信(参考)

NTT Information Sharing Platform Laboratories

- サイトローカルアドレスの廃止で、本方式もすでに廃止
- draft-ietf-ipv6-dns-discovery-08.txt
 “Well known site local unicast addresses to communicate with recursive DNS servers”
- DNS用のサイトローカルアドレスをあらかじめ予約する
 – fec0:0:0:ffff::1, fec0:0:0:ffff::2, fec0:0:0:ffff::3
- Windows XPなどに残骸が見られる (SP2でも動作不可)

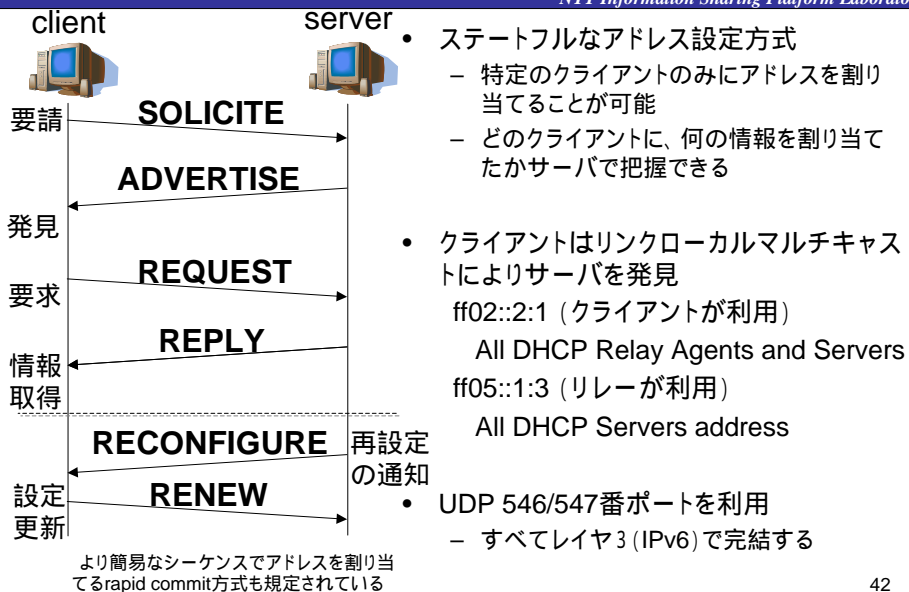
```

C:\>ipconfig /all
Ethernet adapter ローカル エリア接続:
. . . . .
DNS Servers . . . . : 192.16.178.230
                    fec0:0:0:ffff::1%1
                    fec0:0:0:ffff::2%1
                    fec0:0:0:ffff::3%1
    
```

41

“Dynamic Host Configuration Protocol for IPv6 DHCPv6”, RFC3315

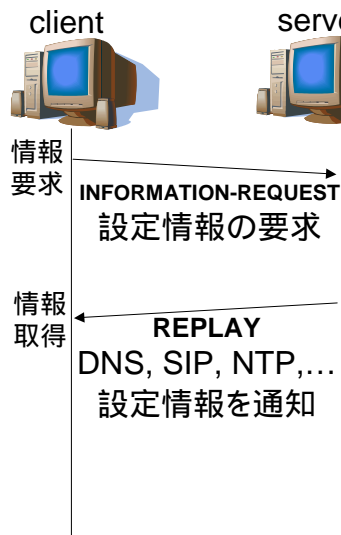
NTT Information Sharing Platform Laboratories



42

“Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6”, RFC3736

NTT Information Sharing Platform Laboratories



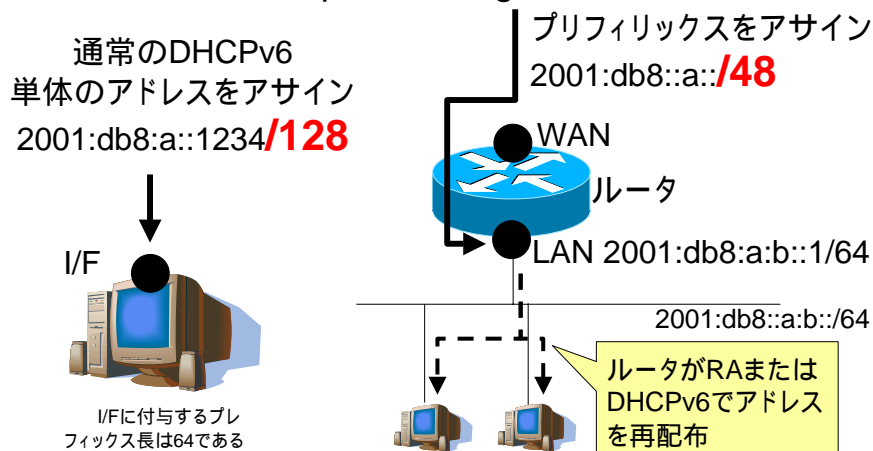
- サーバがクライアントの状態を管理しない方式
- 主にノードに設定情報 (DNS, SIP, NTP) を渡すことを想定
 - DNS情報取得法の議論で、軽いDHCPv6が必要とされた
 - DHCPv6-lite とも呼ばれる
- INFORMATION-REQUEST, REPLAYメッセージだけで情報の授受を行う

43

“IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6”, RFC3633

NTT Information Sharing Platform Laboratories

- ホスト(ルータ)にプリフィックスを払い出す
 - DHCPv6-PD (prefix delegation) と呼ばれる



44

DHCPv6-PDを利用したIPv6 ISPのサービス例 OCN ADSLサービス IPv6デュアル(A)

NTT Information Sharing Platform Laboratories

- 概要
 - NTTコム・アッカネットワークスによるIPv4/IPv6コネクティビティが提供されるデュアルサービス
- 顧客の宅内設備(ブロードバンドルータ)に対して
 - IPv4:PPPoE(IPCP)を用いて、IPv4アドレス(ダイナミック)をひとつ払い出す
 - IPv6:PPPoE(IPV6CP)+DHCPv6-PDを用いて、/48のプリフィックス(固定)をひとつ払い出す
- 仕様書
 - 「OCN ADSLサービスIPv6デュアル(A)用 ユーザ網インタフェース仕様書 第2.3版」,
http://www.ocn.v6.ntt.net/dual/pdf/ipv6if_ver2.3.pdf

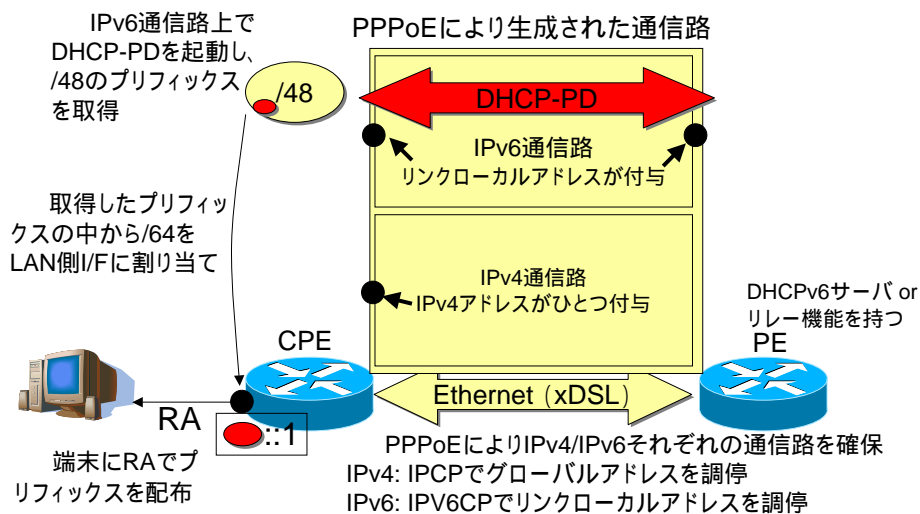
45

DHCPv6-PDによるプリフィックス割り当てから 通信開始のまでの流れ

NTT Information Sharing Platform Laboratories

PE: Provider Edge

CPE: Customer Premises Equipment



46

IPv6 Node Requirements Network Management に 関するトピック

47

IPv6 Node Requirementsでの SNMPについての言及

- 概要
 - IPv6 Node Requirementsで実装は[MAY]とされる
 - IPv6対応に必要なこと
 - トランスポートのIPv6対応 容易
 - IPv6 MIBの標準化
 - MIBはIPv4 / IPv6をマージしたものを標準化
 - RFC4022, “MIB for TCP”
 - RFC4133, “MIB for UDP”
 - draft-ietf-ipv6-rfc2011-update.txt, “MIB for IP”
 - draft-ietf-ipv6-rfc2096-update.txt, “IP Forwarding Table MIB”
- 後者の2つのI-DはRFCがまもなく発行される

48

ローカルアドレスに関するトピック

49

サイトローカルアドレス (fec0::/16) の状況

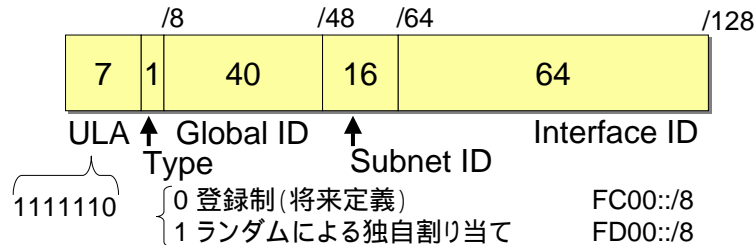
- サイトローカルアドレス (fec0::/16) は廃止が決定
 - 規格は存在するが使用は推奨されない
 - “Deprecating Site Local Addresses”, RFC3879
- サイトローカルアドレスの問題
 - スコープ(サイト境界)の定義があいまいになる
 - 他のスコープ(グローバルなどの)空間へのリーク
 - DNSクエリ・ソースアドレスがサイトローカルのパケットの発生
 - サイトの併合時に重複が発生する可能性
 - IPv4のプライベートアドレスでもほぼ同じ問題が指摘されている
 - 同一のアドレス空間を利用していた企業間連携・合併時の問題
 - NATを助長する恐れがある
- しかしローカルアドレスは有効かつ必要と言う認識

50

“Unique Local IPv6 Unicast Addresses” (ULA), RFC4193
 アドレス構造とGlobal IDの生成方法

NTT Information Sharing Platform Laboratories

• Unique Local IPv6 Unicast Addressの構造



• FD00::/8 をアドレス空間として使用

• Global IDを**ランダム**に生成する

- 独自に使用可能 (申請の必要はない)
- サンプルとして時刻を種にする方法が示される。
 trunc(SHA1(NTP current time + EUI-64), 40bit)

51

“Unique Local IPv6 Unicast Addresses” (ULA), RFC4193
 一意性の保証と運用上の注意点

NTT Information Sharing Platform Laboratories

• Global IDの一意性の保証について

- RFC4193では完全な一意性は保証していない
- 適切な乱数を使えば衝突の可能性は限りなく低い
- ランダム割り当ての運用状況を見て、一意割り当て (centrally assign) の検討を行う
 - この場合、FC00::/8 (type = 0) 空間を使用予定

• ULAの運用上の注意点

- グローバル空間へのリーク
 - ルーティング, DNSクエリ
- グローバル空間とのマルチホーム時の挙動
 - ローカルとグローバルの使い分けをどうすべきか?
 - アドレス選択が正しく動くか?

52

文書作成用アドレス空間 (2001:db8::/32)

NTT Information Sharing Platform Laboratories

- APNICが文書作成用にアドレス空間を予約
- マニュアルや設定サンプルへの利用を想定
- 明示的な割り当て(申請)は不要
- 通信に利用してはならない

- 参考ドキュメント
 - APNIC “IPv6 Documentation Prefix”
<http://www.apnic.net/info/faq/ipv6-documentation-prefix-faq.html>
 - RFC3849, “IPv6 Address Prefix Reserved for Documentation”

53

6bone (3ffe::/16) の廃止について

NTT Information Sharing Platform Laboratories

- 1996年から運用を開始したIPv6実験用ネットワーク
 - pTLA (3ffe::/16)のアドレス空間を利用
 - 国内でも活発な実験実施のため広く割り当てられた
 - 経路情報を観測すると現在でも利用している組織は多い

- 2006年6月6日以降は廃止が決定
 - RFC3701, “6bone (IPv6 Testing Address Allocation) Phaseout”
 - 利用組織はsTLAを利用したサービスへ移行が必要

54

IPv6アドレスの逆引きについて ip6.int 空間の廃止

NTT Information Sharing Platform Laboratories

- IPv6アドレスの逆引き空間は
 - ip6.int (2001年8月にdeprecated化)
 - ip6.arpa (" よりip6.intから移行)
に収容されている。現在は両者の併用が多い
- 2006年6月で ip6.int は廃止が決定
 - “Deprecation of ip6.int reverse DNS service in APNIC”,
<http://www.apnic.net/docs/policy/discussions/prop-030-v001.txt>

55

NTT Information Sharing Platform Laboratories

Mobile IPv6 NEMO (Network Mobility) に関するトピック

56

IPv6上でモビリティを実現するプロトコル Mobile IPv6 (RFC3775, RFC3776)

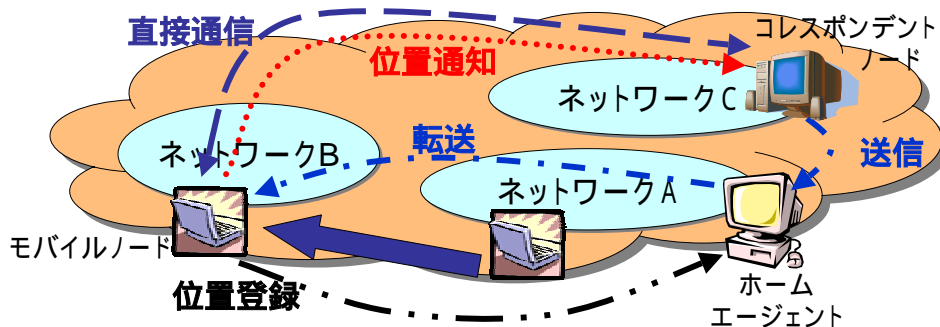
NTT Information Sharing Platform Laboratories

- IPv6 Node Requirementsでは[MAY]とされる
 - 位置透過性を提供
 1. 接続リンクが変更になっても同一アドレスでノードを参照できる
 2. 接続リンクが変更になっても進行中の通信(セッション)を維持できる
 - IPv6レイヤのみで実現
 - アプリケーションおよびL2のサポートが不要
(アプリケーションの改造もL2の改造も不要)
- ➡ IPv6コネクティビティさえあれば位置透過な通信が可能
- Mobile IPv4からの改善点
 - 経路最適化が標準で定義
 - Foreign Agentの廃止
 - ソースアドレス詐称が無い
ファイヤウォールのingress filterに整合

57

Mobile IPv6 の動作の概略

NTT Information Sharing Platform Laboratories



移動端末はホームエージェントに**位置登録**

相手端末は、通信開始時はホームエージェント経由でパケットを**送信**

ホームエージェントが移動端末にパケットを**転送**する

移動端末は相手端末にも**位置通知**する

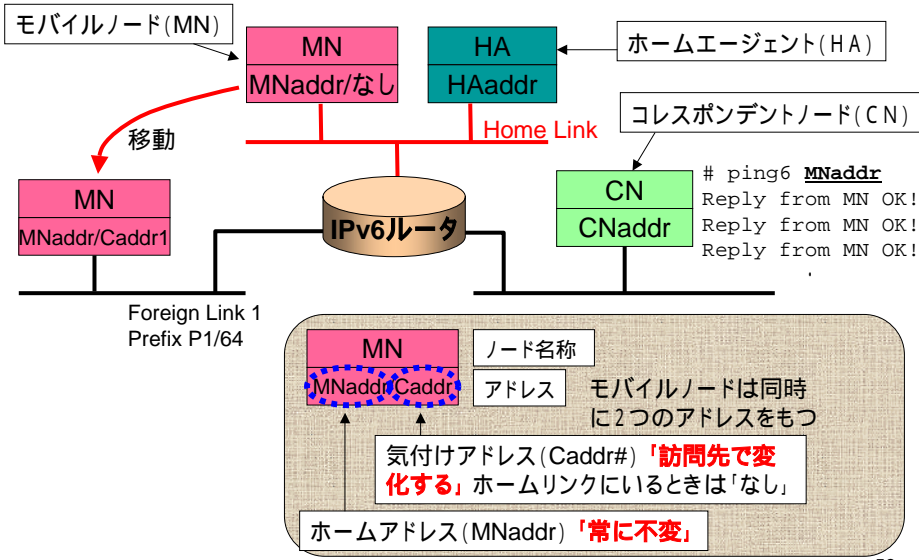
通信開始後は、移動端末と相手端末は**直接通信**

Mobile IPv6の特徴

58

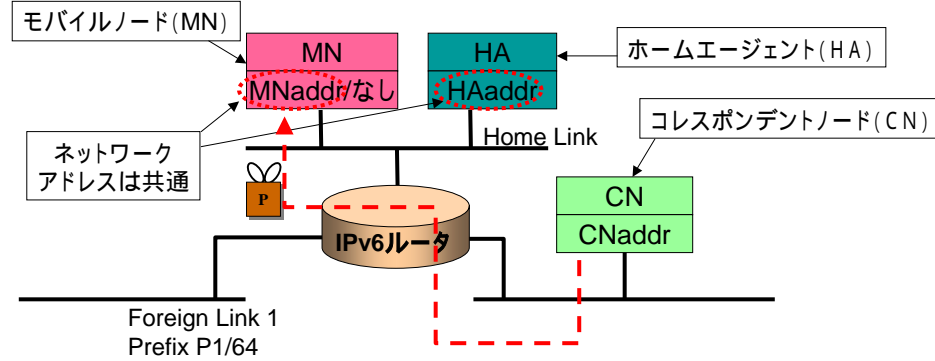
Mobile IPv6を用いた移動通信

NTT Information Sharing Platform Laboratories



MNがHome Linkに存在

NTT Information Sharing Platform Laboratories

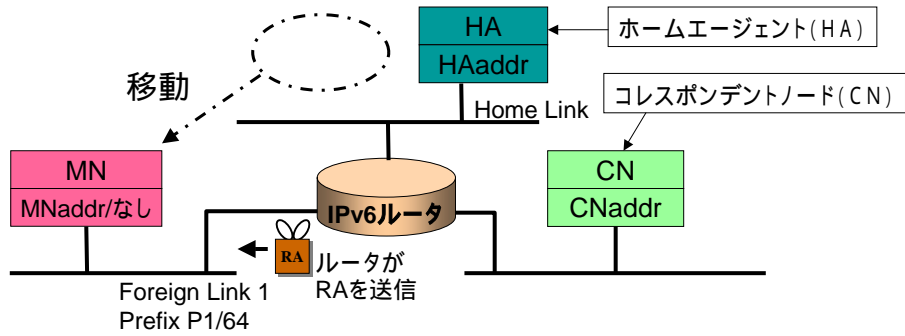


通常のIPv6通信が行われる

60

MNは自分が移動したことを検出する

NTT Information Sharing Platform Laboratories



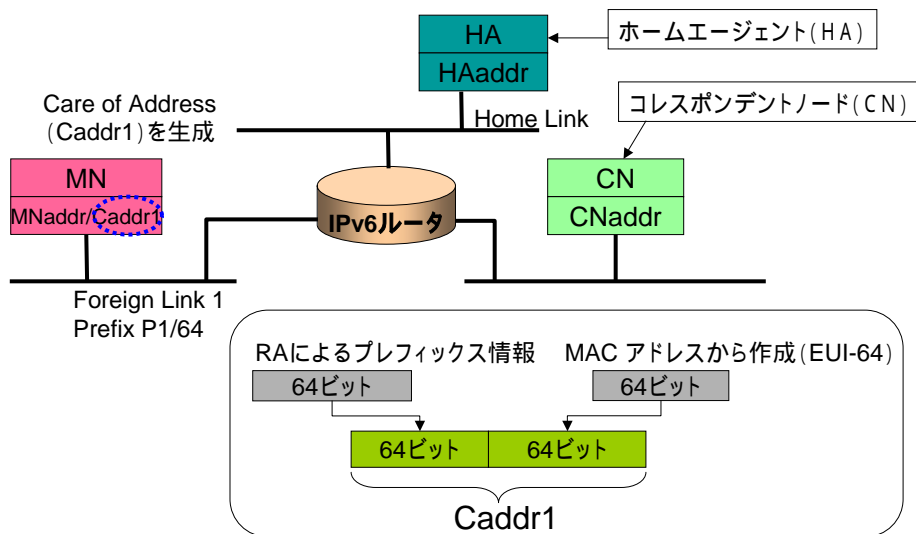
- RAで受信したプリフィックスがホームアドレスのものと一致しない
- 直前に受け取ったプリフィックスと一致しない

➡ MNは移動したと判断

61

MNは気付けアドレスを生成する

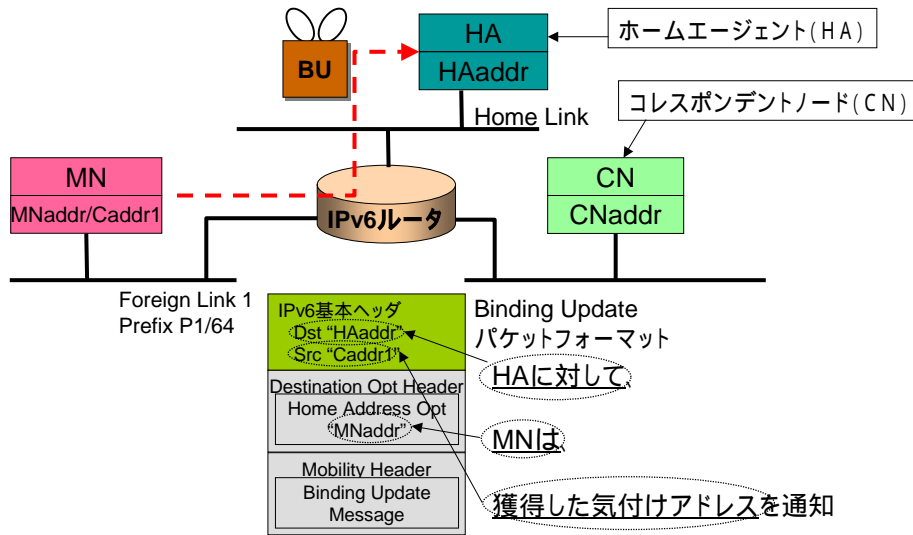
NTT Information Sharing Platform Laboratories



62

MNが気付けアドレスをHAに通知する(位置登録) HAへのBinding Update

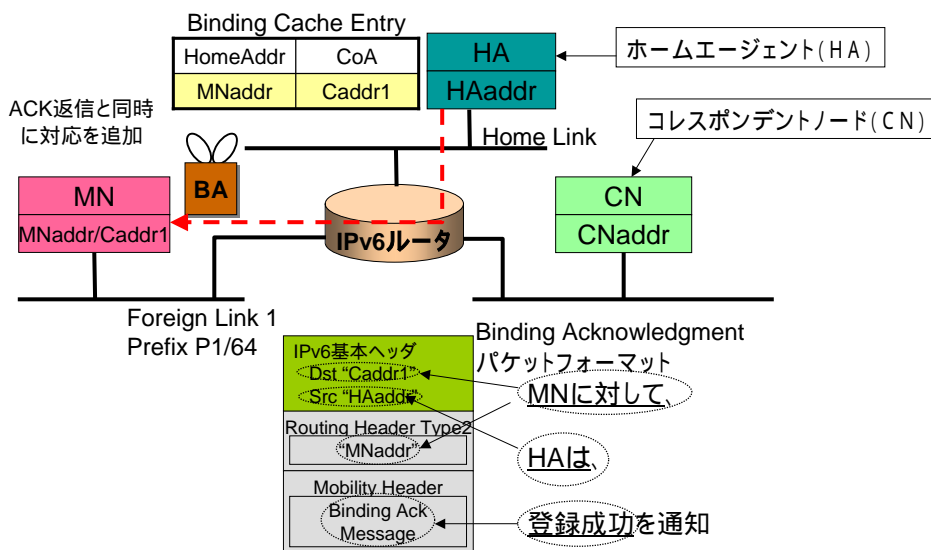
NTT Information Sharing Platform Laboratories



63

HAが気付けアドレスを確認し登録成功を通知 Binding Acknowledgment

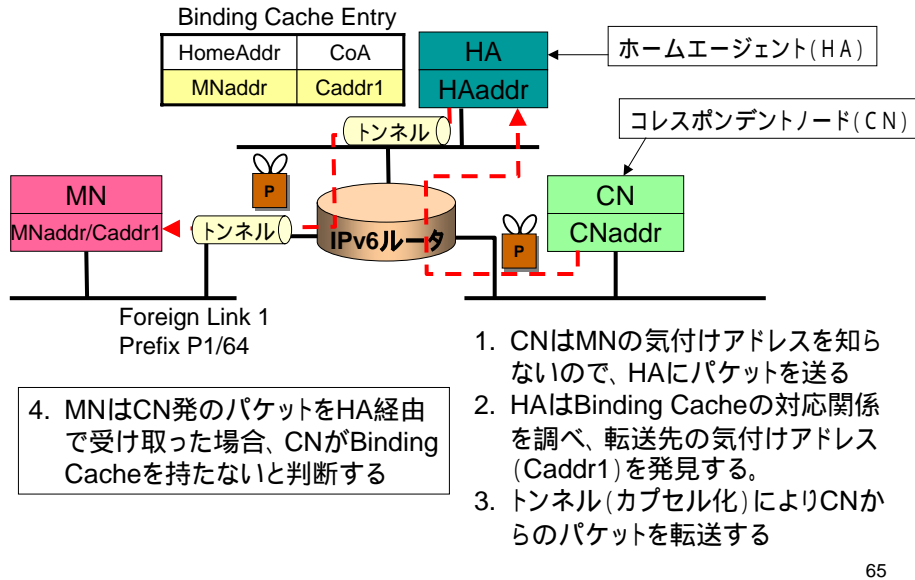
NTT Information Sharing Platform Laboratories



64

CNがMNに向かって通信を開始

NTT Information Sharing Platform Laboratories

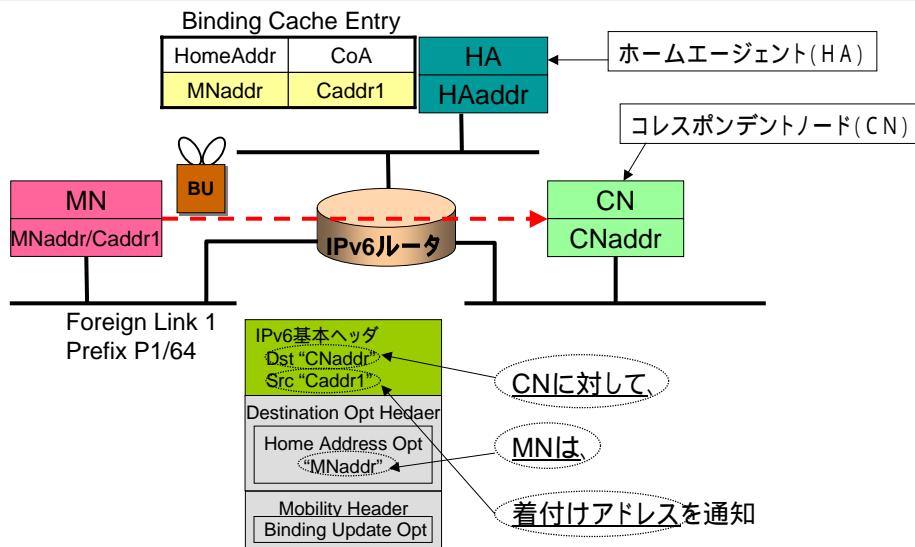


65

MNがCNに対して気付けアドレスを通知する

通知前にReturn Routability手続きが必要だが簡略化のため別途解説

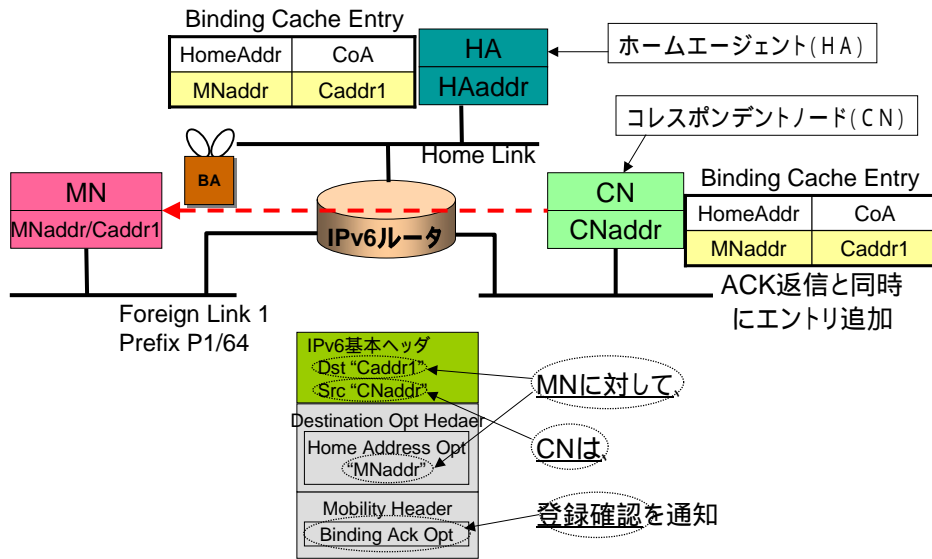
NTT Information Sharing Platform Laboratories



66

CNが気付けアドレスを確認 (Acknowledge) する Return Routability Procedureは別途解説

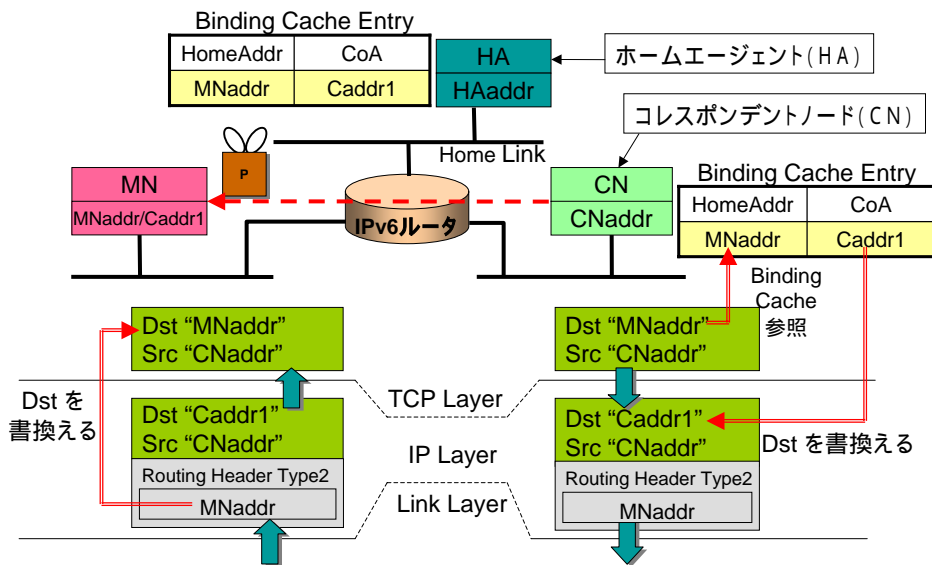
NTT Information Sharing Platform Laboratories



67

CNはBinding Cacheを使いMNにパケットを送信

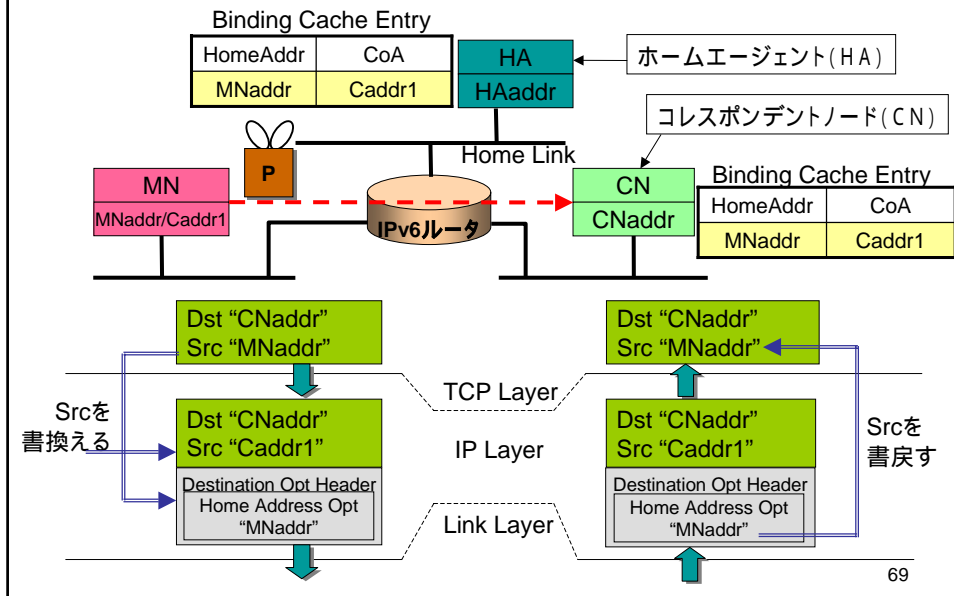
NTT Information Sharing Platform Laboratories



68

MNはHome Addressオプションを使い CNにパケットを送信

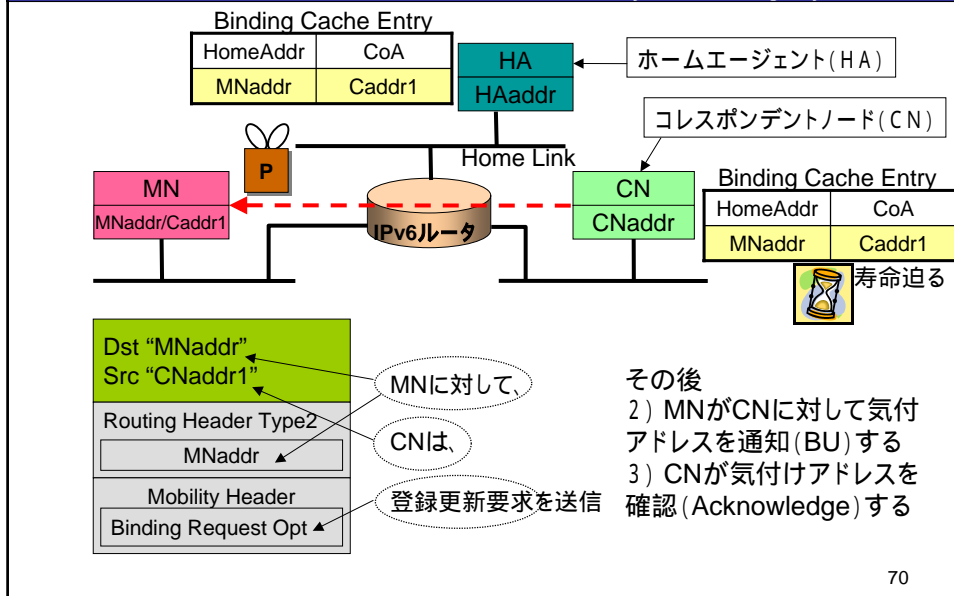
NTT Information Sharing Platform Laboratories



69

CNはBinding Cacheの寿命が切れそうになったら MNにBinding Requestを送信

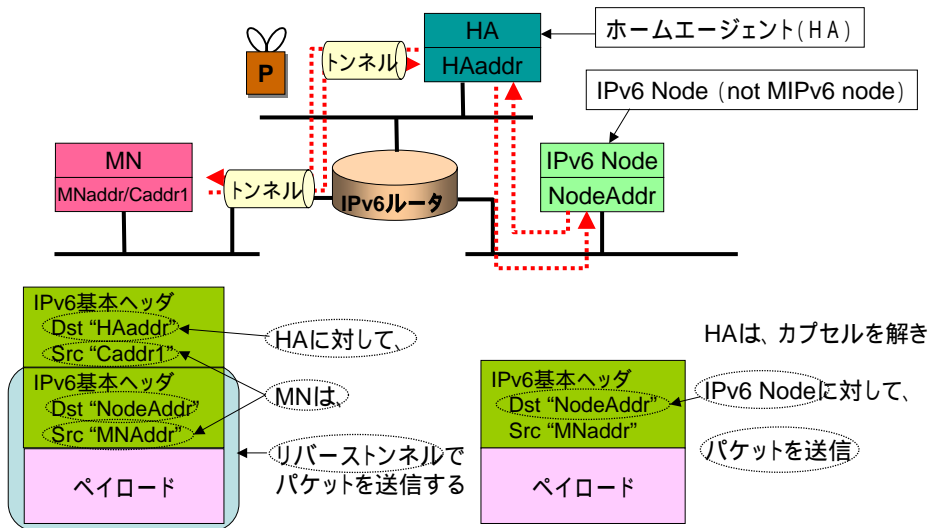
NTT Information Sharing Platform Laboratories



70

Mobile IPv6 を実装しないノード (Binding Cacheを持たないCN)との通信

NTT Information Sharing Platform Laboratories



71

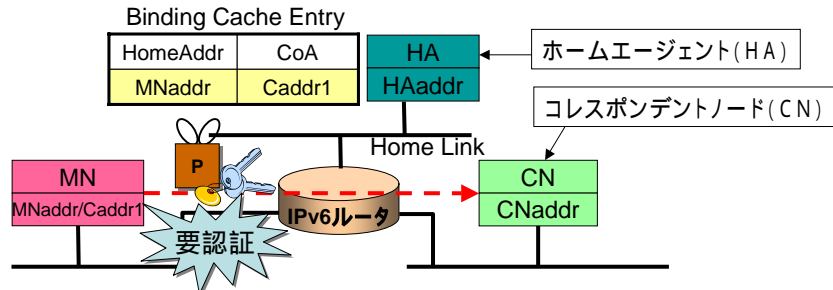
NTT Information Sharing Platform Laboratories

MN CN間の認証手続き Return Routability

72

CNによるMNの認証の必要性

NTT Information Sharing Platform Laboratories



- 成りすまし防止のためCNはMNを認証する
- (現状では) 一般にCNはMNの認証鍵を持たない
- HA MN間はIPsecにより安全という仮定を置く

Return Routability (RR) 手続きを発行

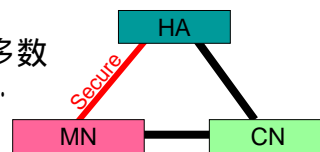
73

Return Routabilityの特徴と問題点

NTT Information Sharing Platform Laboratories

特徴

- CNにアクセスしてくるMNは不特定多数
 - CNが公開Webサーバだったとしたら...
 - あらかじめMNの鍵を持つことは困難
 - PKI + IPsec は現状では非現実的
- HAがMNを認証していることを前提とする
 - RRはHA MN間の通信が信頼できることを利用



問題点

- CNの近傍 (non-IPsec区間のCN HA通信) の盗聴には無防備

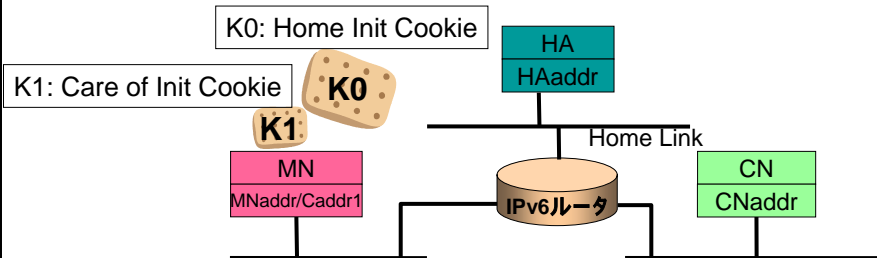
不特定多数の認証と不完全な認証の折衷案

74

Return Routability手続き STEP 1

MNはクッキー (K0, K1) を生成

NTT Information Sharing Platform Laboratories



K0: Home Init Cookie

K1: Care of Init Cookie

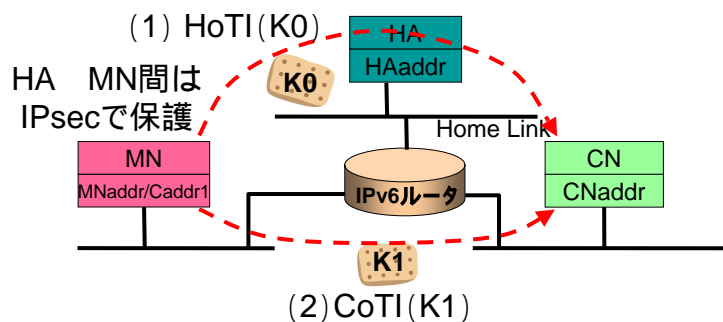
いずれも64ビット長の乱数

75

Return Routability手続き STEP 2

MNは2種類のTest Initメッセージを送信

NTT Information Sharing Platform Laboratories



MNはCNへ2種類のメッセージを同時に送信

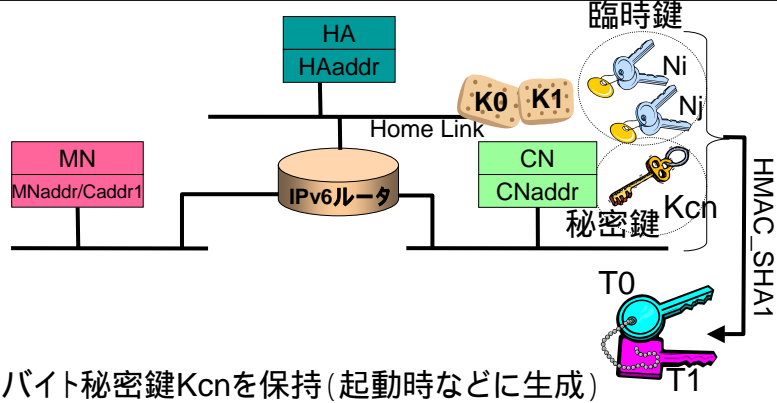
(1) Home Test Init (HoTI) (クッキー-K0を含む / ホームエージェント経由)

(2) Care of Test Init (CoTI) (クッキー-K1を含む / CNへの直接配送)

76

Return Routability手続き STEP 3 CNは署名用トークン(T0,T1)を生成

NTT Information Sharing Platform Laboratories

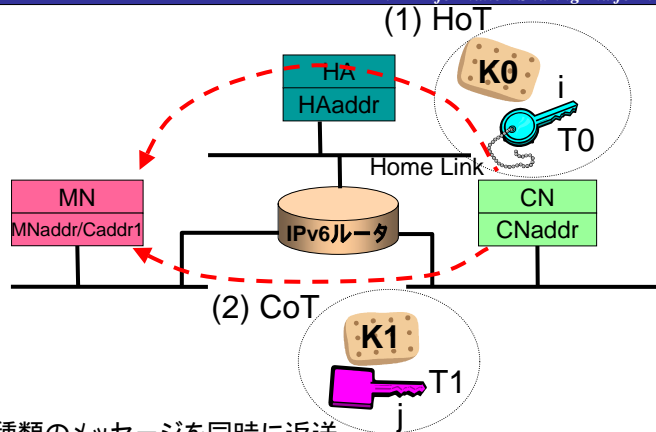


- CNは20バイト秘密鍵Kcnを保持(起動時などに生成)
 CNは臨時鍵Ni, Njを乱数で生成(i, jは鍵を区別する通し番号)
 ハッシュにより2つの署名用トークンT0, T1を生成
- (1) home keygen: $T0 = \text{HMAC_SHA1}(Kcn, (\text{MNAddr} + \text{Ni} + 0x00))$
 - (2) care of keygen: $T1 = \text{HMAC_SHA1}(Kcn, (\text{Caddr1} + \text{Nj} + 0x01))$

77

Return Routability手続き STEP 4 CNはMNへ署名用トークン(T0,T1)を返送

NTT Information Sharing Platform Laboratories

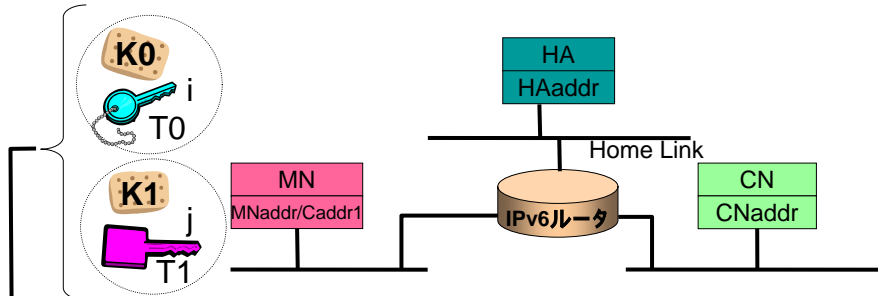


- CNはMNへ2種類のメッセージを同時に返送
- (1) Home Test (HoT)
 cookie K0, 署名用トークンT0, 臨時鍵番号iを含む / HA経由
 - (2) Care of Test (CoT)
 cookie K1, 署名用トークンT1, 臨時鍵番号jを含む / 直接配送

78


Return Routability手続き STEP 5 MNは署名鍵(T0,T1)から署名MACを生成

NTT Information Sharing Platform Laboratories



Binding Key の生成 $K_{bm} = \text{SHA1}(T0+T1)$

MNはBinding Updateメッセージを生成し、 K_{bm} で署名(MAC)を生成する

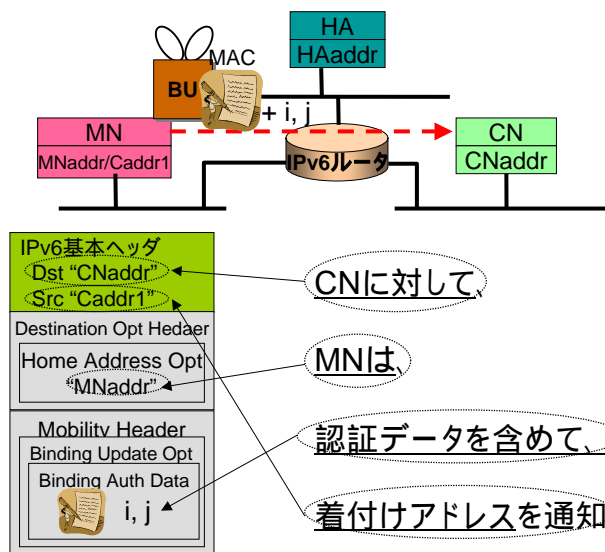
 $\text{MAC} = \text{HMAC_SHA1}(K_{bm}, (\text{Caddr1} + \text{CNAddr} + \text{BU}))$

Binding Updateメッセージ自身のデータ

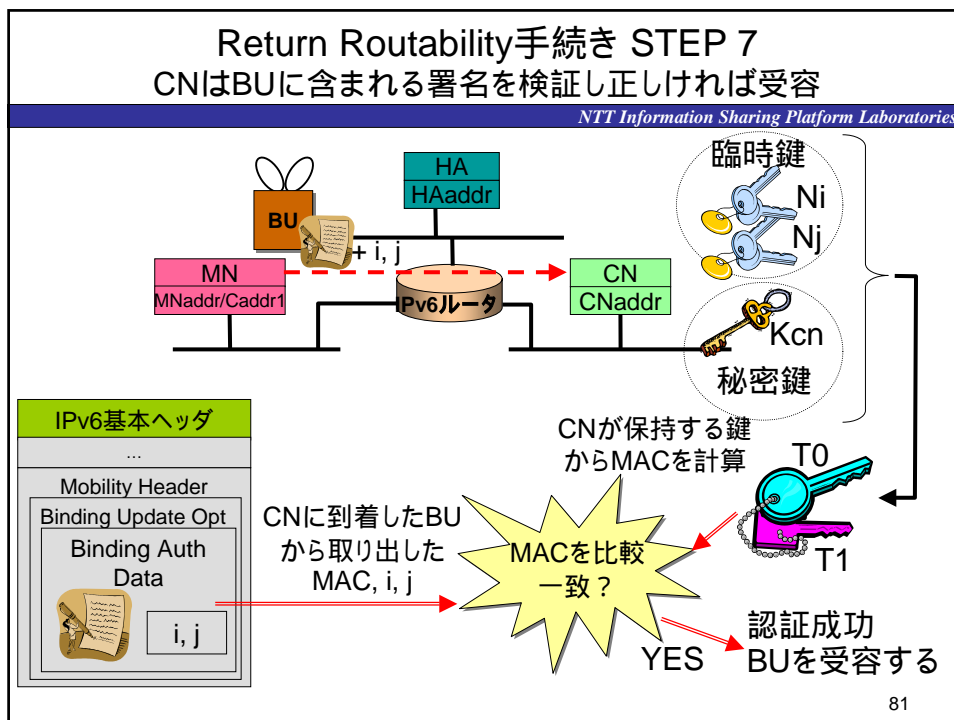
79

Return Routability手続き STEP 6 MNはCNへ署名つきBinding Updateを送信

NTT Information Sharing Platform Laboratories



80

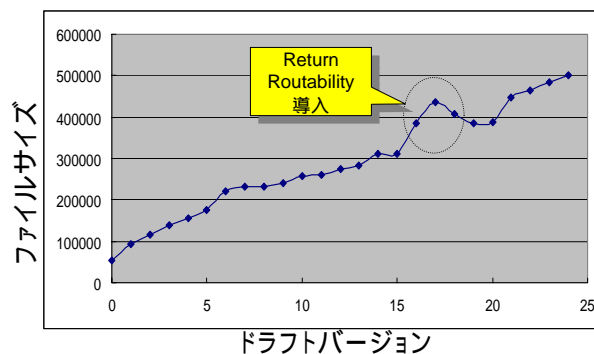


- ### Mobile IPv6 がもたらす効果
- NTT Information Sharing Platform Laboratories
- IPはメディアに非依存の性質を利用すると
 - アクセス統合 / ISP統合が実現
 - ホームエージェントサービスの可能性
 - IP層の接続性が特定アクセス網 / ISPから分離
 - ISPが変わっても同じアドレスを使い続けられる
 - 考え方はMSP (Mail Service Provider) と類似
xxxxx@hotmail.com はISPに依存しない
- 82

IETFでのMobile IPv6の標準化までの経緯

NTT Information Sharing Platform Laboratories

- '96年～'04年にかけて第24版まで長らく議論
- 第15版でRFC化が決まりかけた。しかしIESGよりセキュリティに関するコメントがつく
 - 移動ノードの認証に対してIPsecの利用は、鍵交換インフラが不十分な現状では実用性に問題がある



83

MIPv6 WGで議論されている主なトピック

NTT Information Sharing Platform Laboratories

- ホームエージェントの冗長化などの信頼性向上
- bootstrapping (初期接続時のPnP機構)
 - アドレス獲得, ホームエージェント発見, セキュリティ設定
- 認証機構やセキュリティ機能の強化
 - アクセス, HA, 認証サーバを分離した配置モデルを検討
 - Firewallとの親和性の検討
- 管理用MIB, プログラミングAPI
- IP Address Location Privacyに関する検討

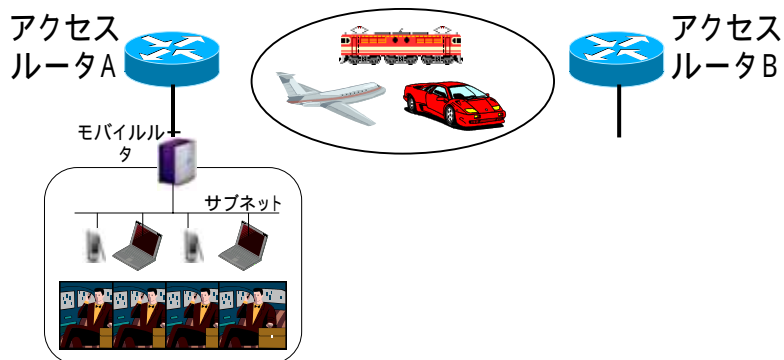
実用品として必要な周辺技術を固める段階

84

ネットワークモビリティ技術 サブネットワークが丸ごと移動

NTT Information Sharing Platform Laboratories

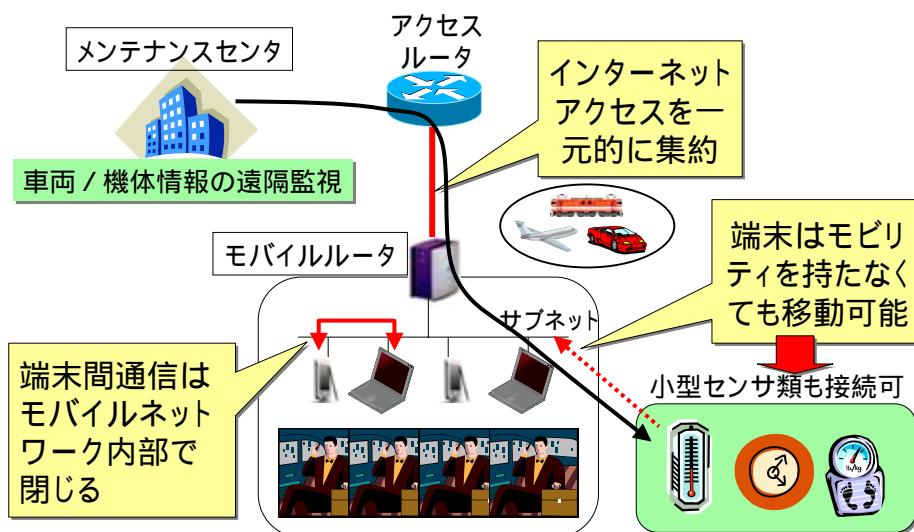
電車、自動車、飛行機など乗り物では
- 乗客 / センサーなど
複数のデバイスが同時に移動する



85

ネットワークモビリティの応用例

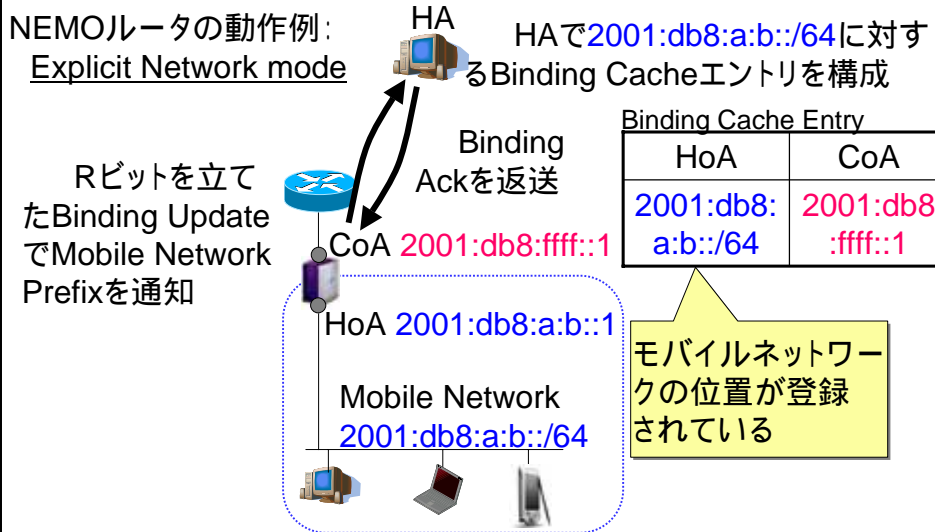
NTT Information Sharing Platform Laboratories



86

“Network Mobility (NEMO) Basic Support Protocol”, RFC3963 (1)

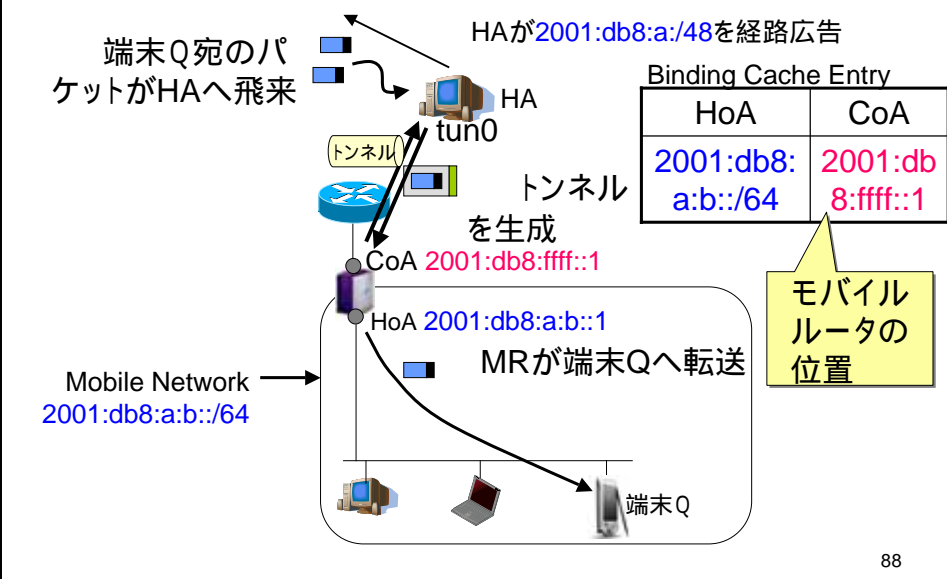
NTT Information Sharing Platform Laboratories



87

“Network Mobility (NEMO) Basic Support Protocol”, RFC3963 (2)

NTT Information Sharing Platform Laboratories



88

NEMO WGで議論されている主なトピック

NTT Information Sharing Platform Laboratories

- MobileIPv6をベースとして、移動ネットワークの実現手法を策定
- 議論中の項目
 - マルチホーミング
 - 経路最適化
 - IPv4 / IPv6 混在時の利用方法

89

NTT Information Sharing Platform Laboratories

IP security (IPsec) に関するトピック

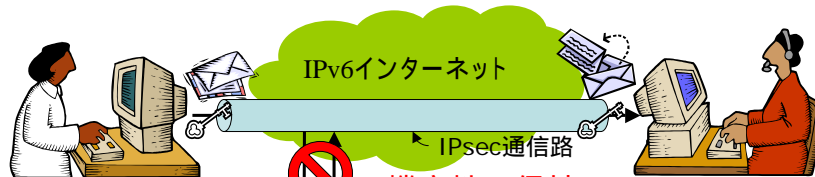
90

IP Security (IPsec)

NTT Information Sharing Platform Laboratories

- インターネット上にセキュアな通信路を確立する技術
- インターネットの基本プロトコルレベルで**4機能**を実現
- IPv6 Node Requirementsでは[MUST]と規定

1. 認証: 通信相手を確認できること



- 4. 否認の防止: 通信事実を否定できないようにすること
- 2. 機密性の保持: 第三者に内容を知られないこと
- 3. 改竄の防止: 第三者による改竄を検知できること



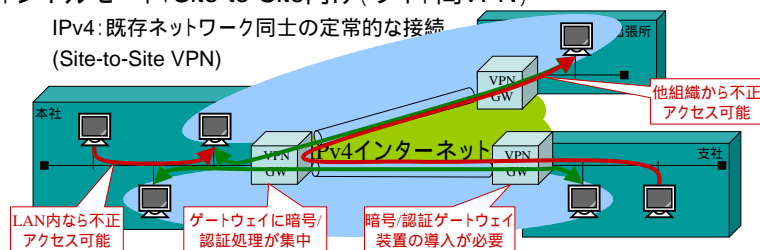
91

IPsecの2つの通信モード

NTT Information Sharing Platform Laboratories

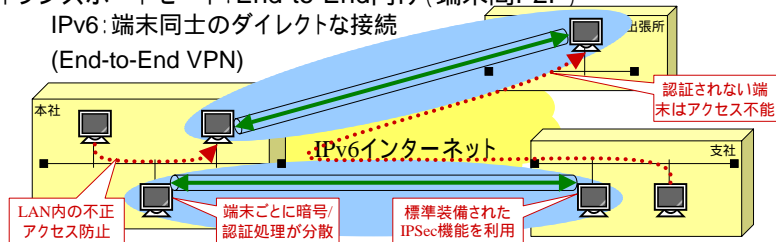
- トンネルモード: Site-to-Site向け (サイト間VPN)

IPv4: 既存ネットワーク同士の定常的な接続 (Site-to-Site VPN)



- トランスポートモード: End-to-End向け (端末間P2P)

IPv6: 端末同士のダイレクトな接続 (End-to-End VPN)

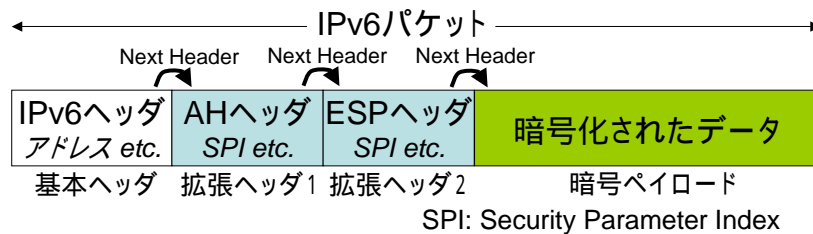


92

IPsecの基本機能

NTT Information Sharing Platform Laboratories

- 認証ヘッダ (AH)
 - パケット全体の改ざんの防止、認証、否認の防止
 - アルゴリズム: MD5, SHA1 など
- 暗号化ペイロードヘッダ (ESP)
 - 機密性の保持
 - ペイロードの改ざん防止
 - アルゴリズム: DES, 3DES, AES など



93

IPv6でのIPsec標準搭載 (MUST) の効果

NTT Information Sharing Platform Laboratories

- End-to-End (P2P) で通信可能
 - ホスト単位でVPN参加者を制御できる
 - ゲートウェイの配置に依存しない柔軟なVPNを構成
 - IPv6 P2Pアプリケーション発展の期待
 - VoIP、IP携帯、テレビ会議、インスタントメッセージなど
- NATの排除
 - NATによりIPsec機能が制約されない
 - NAT traversalなどの補助機能の追加も不要
- 高価なゲートウェイ装置が不要
 - だがIPv4と同様の構成をとることも可能

94

IPv6 IPsecの現状と問題点

NTT Information Sharing Platform Laboratories

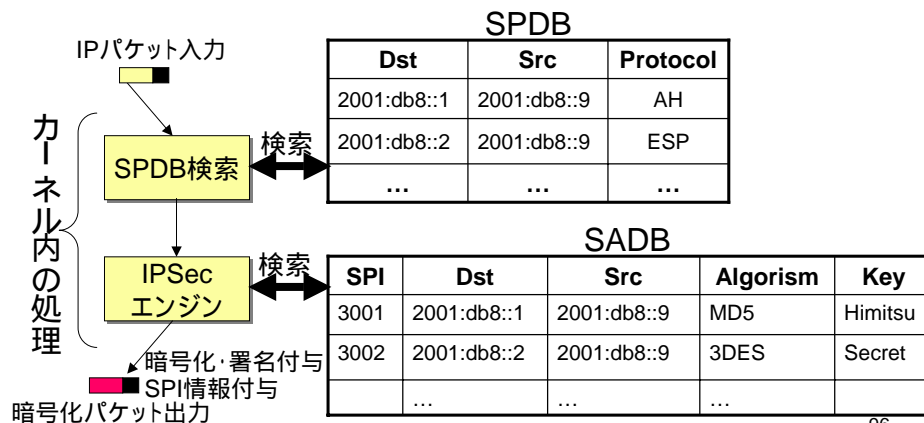
- 現状
 - IPsec(AH, ESP)は安定した実装が多い
 - 処理の大部分はIPv4とほぼ同じ
 - 上位層オフローディング機能搭載のNIC登場
 - IPsecを処理するハードウェア機構が載りつつある
- 問題点
 - MUSTだが実装していないIOSがある
 - 組み込み系にとってIPsecはまだ重い
 - トランスポートモードはホスト毎の設定が必要
 - インターネット鍵交換プロトコルが不十分

95

IPsec動作のためのデータベース ホストが持つ2つのDB

NTT Information Sharing Platform Laboratories

- SP(セキュリティポリシー)
 - IPsec通信の適用/不適用を決定するポリシーDB
- SA(セキュリティアソシエーション)
 - IPsec通信の暗号化・復号化に必要な鍵DB

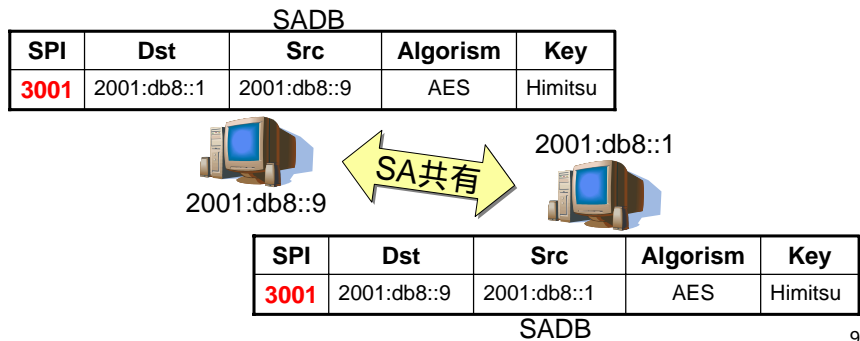


96

セキュリティアソシエーションの共有

NTT Information Sharing Platform Laboratories

- 両端でのSA(符号化方式・鍵)の共有が必要
- 事前設定(pre-shared)ではアプリケーションに限界
 - 公開Webサーバ
 - P2Pアプリケーション(不特定多数から着信するVoIP)
 - IPsec通信ではSPIだけが運ばれる

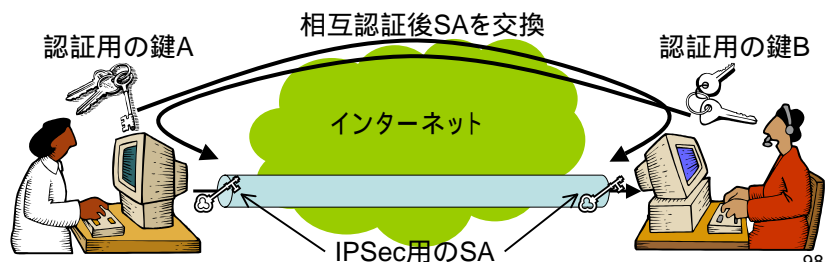


97

インターネット鍵交換プロトコル(IKE)

NTT Information Sharing Platform Laboratories

- 目的
 - 不特定多数のホストと自動的にSAを交換しIPsec通信を行う
- IKEでの相互認証方式(IPsec SA交換用の通信路を確保)
 - 事前共有秘密鍵(これはスケールしない)
 - デジタル署名方式(PKIを利用)
 - 公開鍵を使った暗号文で認証(RSAの仕組みを利用)
- 現状
 - 仕様も設定も非常に複雑。改善のためIKEv2が存在
 - IPv6に対応した実装は多くない

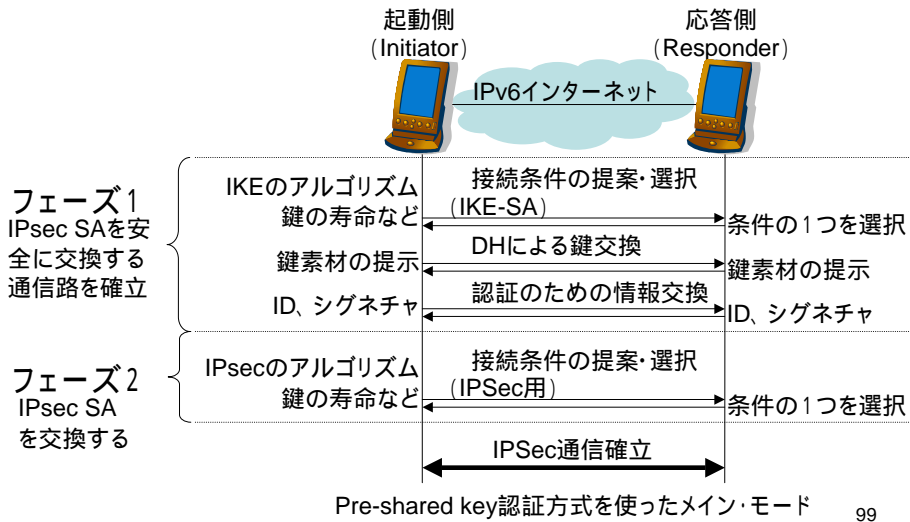


98

インターネット鍵交換の動作

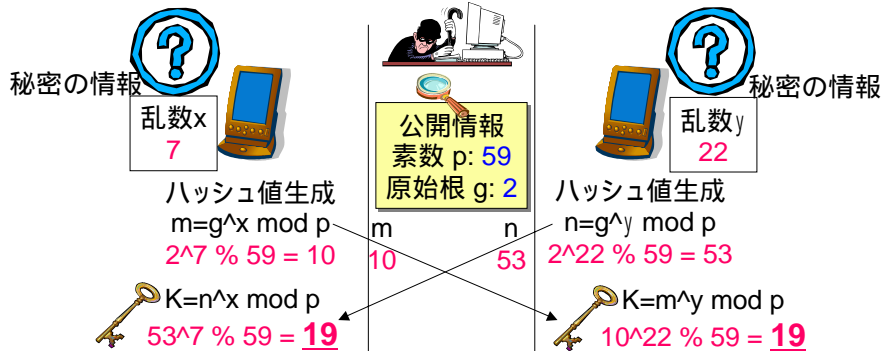
NTT Information Sharing Platform Laboratories

IKEの2つのフェーズ



Diffie-Hellman法による鍵交換アルゴリズム

NTT Information Sharing Platform Laboratories



素数がpが大きくなると p,g,m,n からx,yの計算は数学的に困難

FFFFFFFF	FFFFFFFF	C90FDAA2	2168C234	C4C6628B	80DC1CD1	29024E08
8A67CC74	020BBEA6	3B139B22	514A0879	8E3404DD	EF9519B3	CD3A431B
302B0A6D	F25F1437	4FE1356D	6D51C245	E485B576	625E7EC6	F44C42E9
A63A3620	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

グループ1
768ビット

グループ2
1024ビット

IKEで利用される素数p
原始根 g = 2

IPv6アプリケーションの構造

101

アプリケーション構造の解説目的

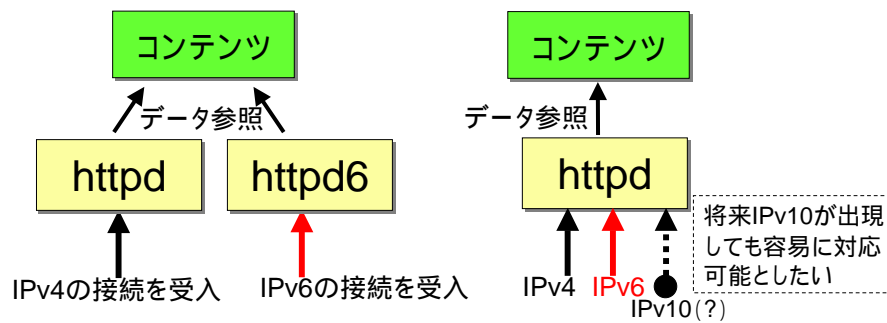
- IPv6アプリケーションの基本的な動作を知る
 - アプリケーション開発の基本パラダイムを理解
 - IPv6アプリの多くは類似する挙動を示す
- 既存のIPv4アプリケーションをIPv6対応に書き直すアプローチをとり、比較を行う
- アプリケーションはWebを想定
 - ブラウザ(TCPクライアント)
 - Webサーバ(TCPサーバ)

102

IPv6対応アプリケーションとは？

NTT Information Sharing Platform Laboratories

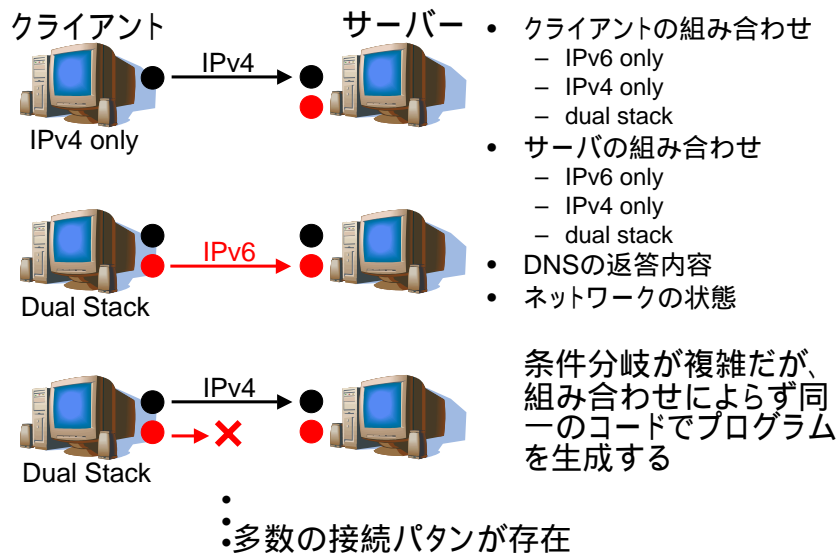
- IPv6専用のアプリケーションではなくIPv4, IPv6いずれでも接続できるものが一般的
 1. IPv6専用のアプリケーションと、IPv4専用のアプリケーションを同時に動作(並存)させる
 2. プロトコルに非依存なアプリケーション
- IPv6対応のWebサーバの動作例



103

プロトコル非依存のアプリケーションとは？

NTT Information Sharing Platform Laboratories



104

ネットワークアプリケーションのAPI BSD (バークレー)ソケット

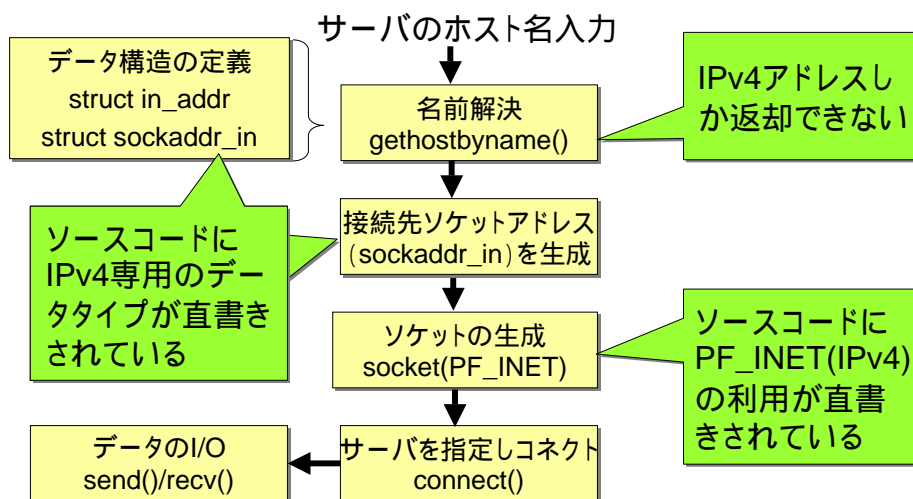
NTT Information Sharing Platform Laboratories

- 概要
 - 4.2BSDへTCP/IP搭載時に規定
 - Windows Socket (WinSock) はBSDソケットから派生
- 特徴
 - 仮想的な通信の端点(ソケット)を提供、そのソケットを指定するだけで通信可
 - 個別のプロトコル(TCPのSYNやACKなど)の通信手順を隠蔽
 - 通信内容が「ソケットアドレス」により表現される
 - アドレスやポート番号などを詰め込む構造体を定義
 - `sockaddr_in`(IPv4), `sockaddr_in6`(IPv6), ... など
 - マルチプロトコルでの利用を前提とした設計
 - IPX, AppleTalk, OSI などIP以外のプロトコルでも利用される

105

BSDソケットによるIPv4 TCPクライアントの処理フロー

NTT Information Sharing Platform Laboratories



106

従来のIPv4アプリケーション サンプルコード(TCPクライアント編)

NTT Information Sharing Platform Laboratories

```

/*
 * サーバに接続し受信した文字列を表示するクライアント
 * IPv4バージョン
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main(int argc, char *argv[])
{
    char *hostname; /* サーバのホスト名 */
    char *servname; /* ポート名 */
    struct hostent *hp; /* IPv4アドレスの解決結果 */
    struct servent *sp; /* ポート名の解決結果(ポート番号) */
    unsigned short port; /* ポート番号を格納 */
    struct sockaddr_in sin4; /* ソケットアドレス */
    int s;

    /* コマンドライン引数のチェック */
    if (argc != 3) {
        fprintf(stderr, "usage: v6-tcp-client <hostname>
        <port><N>");
        exit(1);
    }

    /* コマンドラインの第一引数(ホスト名)と第二引数(ポート名)をセット */
    hostname = argv[1];
    servname = argv[2];

    /* ホスト名からIPv4アドレスを解決 */
    if ((hp = gethostbyname(hostname)) == NULL) {
        fprintf(stderr, "cannot resolve hostname: %s\n",
        hostname);
        exit(1);
    }

    /* ポート名からポート番号を解決 */
    if ((sp = getservbyname(servname, "tcp")) == NULL) {
        port = atoi(servname);
        if (port == 0) {
            fprintf(stderr, "cannot resolve servname: %s\n",
            servname);
            exit(1);
        }
    }

    /* ネットワークバイトオーダーへ変換 */
    port = htons(port);
} else
    port = sp->a_port;

/* sockaddr_in 構造体を初期化 */
memset(&sin4, 0, sizeof(struct sockaddr_in));
/* アドレスファミリーとしてIPv4を指定 */
sin4.sin_family = AF_INET;
/* ソケットアドレスの長さを設定 BSD系UNIXではコメントを外す */
sin4.sin_len = sizeof(struct sockaddr_in);
/* IPv4アドレスを設定 */
memcpy(&sin4.sin_addr, hp->h_addr, sizeof(struct in_addr));
/* ポート番号を設定 */
sin4.sin_port = port;

/* IPv4ソケットを生成する */
s = socket(AF_INET, SOCK_STREAM, 0);
if (s < 0) {
    perror("socket");
    exit(1);
}

/* sin4で指定したサーバのポートへ接続 */
if (connect(s, (struct sockaddr *)&sin4, sizeof(struct
sockaddr_in)) < 0) {
    perror("connect");
    exit(1);
}

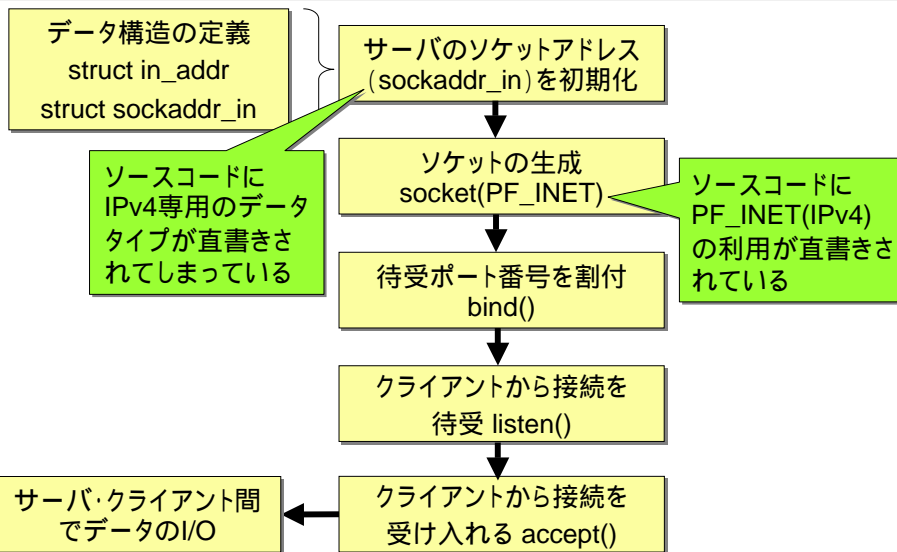
/* サーバとのデータのI/O */
while (1) {
    /* サーバからデータを受け取り表示 */
    int n;
    char buf[4096];
    n = read(s, buf, sizeof(buf));
    if (n == 0)
        exit(0);
    fputs(buf, stdout);
}
}

```

107

BSDソケットによるIPv4 TCPサーバの処理フロー

NTT Information Sharing Platform Laboratories



108

従来のIPv4アプリケーション サンプルコード(TCPサーバ編) (1 / 2)

NTT Information Sharing Platform Laboratories

```
/*
 * 接続してきたクライアントに対して文字列を送信するサーバ
 * IPv4バージョン
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *servname; /* ポート名 */
    struct servent *sp; /* ポート名の解決結果(ポート番号) */
    unsigned short port; /* ポート番号を格納 */
    struct sockaddr_in sin4; /* ソケットアドレス */
    int s; /* ソケット */
    fd_set rfd, rfd0;

    /* コマンドライン引数のチェック */
    if (argc != 2) {
        fprintf(stderr, "usage: v6-tcp-server <port>\n");
        exit(1);
    }

    /* コマンドラインの第一引数(ポート名)をセット */
    servname = argv[1];

    /* ポート名からポート番号を解決 */
    if ((sp = getservbyname(servname, "tcp")) == NULL) {
        port = atoi(servname);
        if (port == 0) {
            fprintf(stderr, "cannot resolv servname: %s\n",
                servname);
            exit(1);
        }
        /* ネットワークバイトオーダーへ変換 */
        port = htons(port);
    } else
        port = sp->s_port;

    /* sockaddr_in 構造体を初期化 */
    memset(&sin4, 0, sizeof(struct sockaddr_in));
    /* アドレスファミリーとしてIPv4を指定 */
    sin4.sin_family = AF_INET;
    /* ソケットアドレスの長さを設定 BSD系UNIXではコメントを外す */
    /* sin4.sin_len = sizeof(struct sockaddr_in); */
    /* IPv4アドレスを設定(バイトオーダーに注意) */
    sin4.sin_addr.s_addr = htonl(INADDR_ANY); /* ポート番号を設
*/
    sin4.sin_port = port;

    /* IPv4ソケットを生成する */
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s < 0) {
        perror("socket");
        exit(1);
    }

    /* ポート番号をソケットに結びつける */
    if (bind(s, (struct sockaddr *)&sin4, sizeof(struct
sockaddr_in)) < 0) {
        perror("bind");
        close(s);
        exit(1);
    }

    /* クライアントからの接続の要求へ待機する */
    if (listen(s, 5) < 0) {
        perror("listen");
        close(s);
        exit(1);
    }

    /* ソケットへのクライアントからの接続要求をselectにより監視 */
    FD_ZERO(&rfd0);
    FD_SET(s, &rfd0);

    /* サーバとのデータのI/O */
    while (1) {
        int as;
        struct sockaddr_in csin4; /* クライアントのソケットアドレス */
        int len; /* ソケットアドレスの長さ */

```

109

従来のIPv4アプリケーション サンプルコード(TCPサーバ編) (2 / 2)

NTT Information Sharing Platform Laboratories

```
int n;
int pid;

/* クライアントから接続が来るまでselectで待つ */
rfd = rfd0;
n = select(FD_SETSIZE, &rfd, NULL, NULL, NULL);
if (n < 0) {
    printf("n = %d\n", n);
    perror("select");
    close(s);
    exit(1);
}

/* クライアントからの接続を受け入れる */
/* クライアントを示すソケットアドレスがcsin4に書き込まれる */
as = accept(s, (struct sockaddr *)&csin4, &len);
if (as < 0) {
    perror("accept");
    close(s);
    exit(1);
}

/* サブプロセスを生成し、クライアントとのI/Oを処理させる */
if ((pid = fork()) < 0) {
    perror("fork");
    exit(1);
}

if (pid == 0) {
    /* サブプロセスの処理 */
    char *text = "send string: Vserver to clientV\n";
    write(as, text, strlen(text));
    close(as);
    exit(0);
} else
    close(as);
}
}
```

110

IPv4アプリケーションをIPv6対応にするために

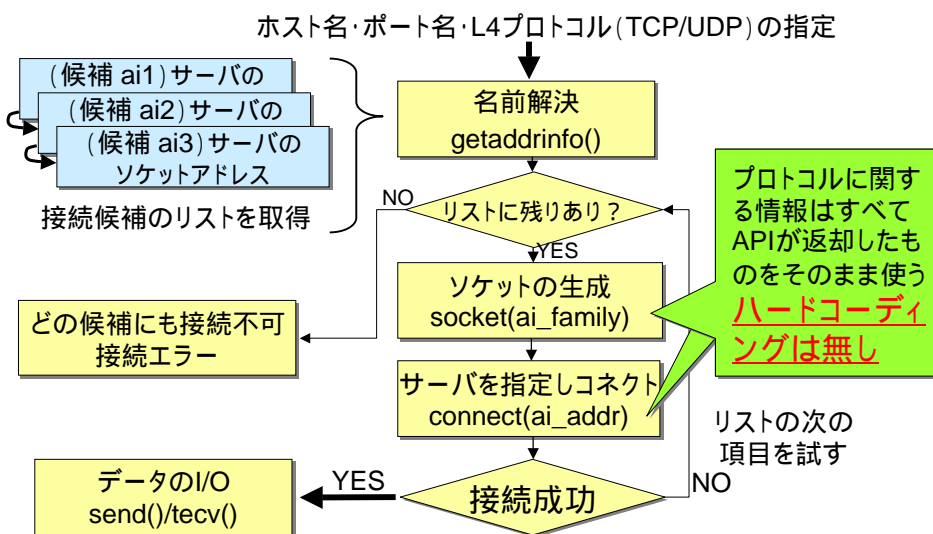
NTT Information Sharing Platform Laboratories

- 基本的な方針
 - IPv4のコードに(やっつけの)IPv6の機能を付け加えない
 - IPv4/IPv6だけに限らずマルチプロトコルが扱えるアプリケーションに書き換える
- プロトコル非依存なプログラミングの推奨
 - APIの規定とその利用方法の解説書
 - RFC3493, "Basic Socket Interface Extensions for IPv6"
 - RFC3542, "Advanced Sockets Application Program Interface (API) for IPv6"
 - 「IPv6ネットワークプログラミング」荻野. 2003. アスキー.
 - ポイントは新たなリゾルバの導入
- IPv4のプログラムも非常に簡潔なコードになります

111

プロトコル非依存TCPクライアントの処理フロー

NTT Information Sharing Platform Laboratories



112

新しい名前解決API getaddrinfo(3) 関数

NTT Information Sharing Platform Laboratories

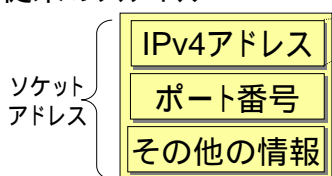
- 入力
 - ホスト名
 - ポート番号
 - L3プロトコル(IPv4, IPv6 or 問わない)
 - L4プロトコル(TCP, UDP or RAW)
- 出力
 - ソケットアドレスと接続に必要な付加情報の **リスト**
 - 接続の候補となりうるアドレスがリストとして列挙される
- 概念的には
getaddrinfo() gethostbyname() /* ホスト名解決 */
 + getservbyname() /* ポート名解決 */
 + ソケットアドレスの初期化
 + バイトオーダの隠蔽
- 標準化されているためほとんどのOSで利用可能

113

getaddrinfo(3)が生成する情報

NTT Information Sharing Platform Laboratories

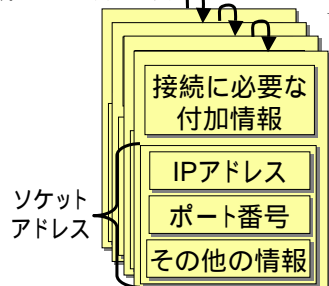
従来のスタイル



gethostbyname()

APIが生成する情報はIPv4アドレスだけで残りの部分はプログラマが組み立てる

新しいスタイル



getaddrinfo()

IPv4/IPv6どちらのアドレスファミリーに対してもAPIがソケットアドレス全体を組み立てる。接続に必要な付加情報も提供

114

異なるプロトコルのソケットアドレスを効果的に扱う addrinfo 構造体 (1)

NTT Information Sharing Platform Laboratories

- このデータ構造の役割
 - getaddrinfo()の振舞いを決める「ヒント」を格納する
 - ホスト名解決の結果を格納する

addrinfoのデータ構造

ai_flags
ai_family
ai_socktype
ai_protocol
ai_addrlen
ai_canonical (ポインタ)
ai_addr (ポインタ)
ai_next (ポインタ)

```

AI_PASSIVE /* サーバ用のソケットアドレスを生成 */
AI_CANONNAME /* 別名を格納する */
AI_NUMERICHOST /* ホスト名は生アドレスで与えられる */
PF_UNSPEC /* IPv6, IPv4 両方を対象とする */
PF_INET /* IPv4のみ対象とする */
PF_INET6 /* IPv6のみ対象とする */
SOCK_STREAM /* socketの第2引数 */
SOCK_DGRAM /* TCP or UDP or any other */
    
```

```

/* wwwのポートhttp(80)にTCPで接続したい */
/* IPv4, IPv6は問わない */
memset(&hints, 0, sizeof(hints));
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
/* 名前解決 */
g = getaddrinfo("www", "http", &hints, &ai0);
    
```

115

異なるプロトコルのソケットアドレスを効果的に扱う addrinfo 構造体 (2) - 結果の格納方法 -

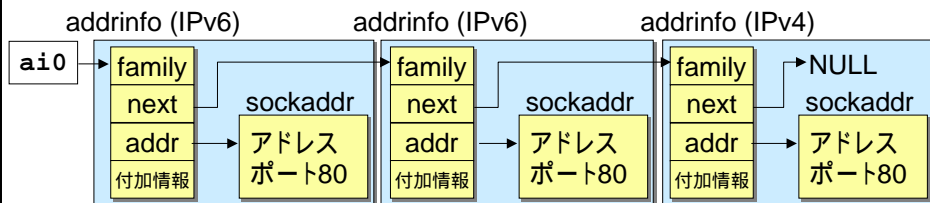
NTT Information Sharing Platform Laboratories

- Webサーバには下記のアドレスが付与されている

```

www.example.com IN A      192.16.178.80 ... IPv4
                  IN AAAA  2001:db8::80 ... IPv6
                  IN AAAA  2001:db8:aaaa:80 ... IPv6
    
```

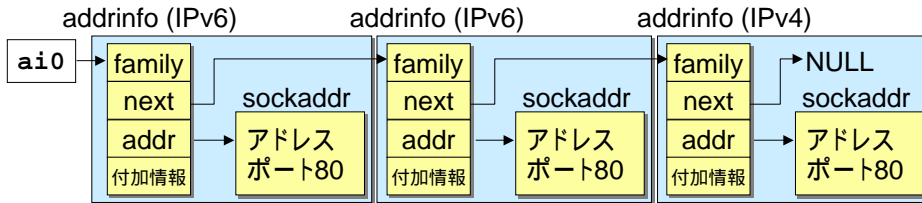
- すべてのアドレスを含んだリスト(ai0)を返す。明示的にアドレスの種類を指定できる
- リストは使い終わったら freeaddrinfo(3)により開放する



116

プロトコルに依存しないTCPクライアント

NTT Information Sharing Platform Laboratories



- ai0からnextを順にたどり、成功するまで試行を続ける
- すべてのaddrinfoが失敗したら接続失敗
- IPv6の接続が失敗しても、IPv4に自動的にフォールバックする

```
for (ai = ai0; ai; ai = ai->ai_next) {
    s = socket(ai->ai_family,
              ai->ai_socktype, ai->ai_protocol);
    connect(s, ai->ai_addr, ai->ai_addrlen); /* 接続試行 */
    if (接続成功)
        break; /* ループ脱出:接続試行はこれで終了 */
}
```

ソースコードにIPv4/IPv6のプロトコル情報を埋め込む必要がない

117

プロトコル非依存に書き換えたアプリケーション サンプルコード(TCPクライアント編)

NTT Information Sharing Platform Laboratories

```
/*
 * サーバに接続し受信した文字列を表示するクライアント
 * IPv6/IPv4デュアルバージョン
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int main(int argc, char *argv[])
{
    int s; /* クライアントと通信を行うソケット */
    struct addrinfo hints; /* getaddrinfoの動作を指定するヒント情報 */
    struct addrinfo *ai0; /* addrinfoリストの先頭の要素 */
    struct addrinfo *ai; /* 処理中のaddrinfoリストの要素 */
    int g; /* 名前解決の結果 */
    char *nodename, *servname;

    /* コマンドライン引数のチェック */
    if (argc != 3) {
        fprintf(stderr, "usage: dual-tcp-client <hostname>
        <port>\n");
        exit(1);
    }

    /* コマンドラインの第一引数(ホスト名)と第二引数(ポート名)をセット */
    nodename = argv[1];
    servname = argv[2];

    /* ヒント情報の初期化 */
    memset(&hints, 0, sizeof(hints));
    /* プロトコルは指定しない: IPv4 or IPv6 */
    hints.ai_family = PF_UNSPEC;
    /* ストリーム型(tcp)による通信を指定 */
    hints.ai_socktype = SOCK_STREAM;

    /* サーバ名とポート名の解決を行い、addrinfoリスト(ai0)を取得 */
    g = getaddrinfo(nodename, servname, &hints, &ai0);
    if (g) {
        fprintf(stderr, "%s", gai_strerror(g));
        exit(1);
    }

    /* addrinfoリストの要素を先頭から接続できるまで順に試行する */
    for (ai = ai0; ai; ai = ai->ai_next) {
        /* ソケットの生成 */
        s = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);

        /* ソケットの生成に失敗したらリストの次の項目を試す */
        if (s < 0)
            continue;

        /* ソケットが生成できたらサーバへ接続を試みる */
        /* 失敗したらリストの次の項目を試す */
        if (connect(s, ai->ai_addr, ai->ai_addrlen) < 0) {
            close(s);
            s = -1;
            continue;
        }

        /* ここですでに接続に成功しているので、forループを脱出 */
        break;
    }

    /* リストのすべての項目が失敗していないかチェック */
    if (s < 0) {
        fprintf(stderr, "cannot connect %s\n", nodename);
        exit(1);
    }

    /* サーバとのデータのI/O */
    while (1) {
        /* サーバからデータを受け取り表示 */
        int n;
        char buf[4096];

        n = read(s, buf, sizeof(buf));
        if (n == 0)
            exit(0);
        fputs(buf, stdout);
    }
}
```

118

IPv6対応サーバの基本的な考え方

NTT Information Sharing Platform Laboratories

- サーバはIPv4, IPv6いずれでも、クライアントからの接続も受け入れる
 - IPv4, IPv6両方の接続を待ち受ける必要がある
 - = 複数のソケットを開いて待ち受ける

[比較] クライアントが通信に使うソケットは
1つだけ

接続が成功するまでaddrinfoリストを順に試行するが、成功した時点で以降の試行は打切っていた

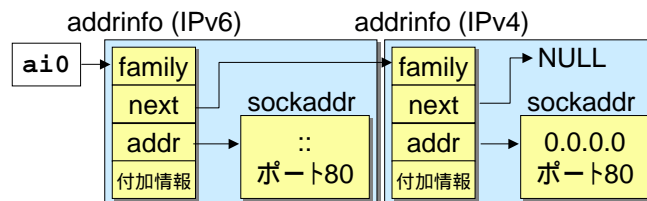
119

AI_PASSIVEフラグつきのgetaddrinfo(3)

NTT Information Sharing Platform Laboratories

- ヒント情報にAI_PASSIVEを設定すると
 - OSのプロトコル数に応じて、addrinfoのリストを生成
 - 通常、DualStackホストなら [IPv6] [IPv4] の2項目

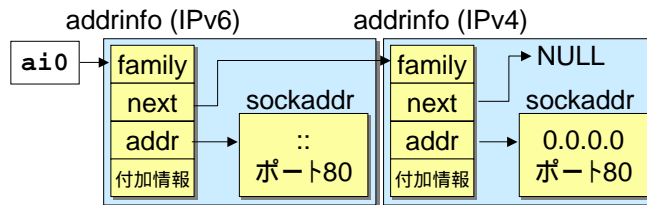
```
memset(&hints, 0, sizeof(hints));  
hints.ai_family = PF_UNSPEC;  
hints.ai_flags = AI_PASSIVE;   ワイルドカードアドレスを指定  
hints.ai_socktype = SOCK_STREAM;  
  
g = getaddrinfo(NULL, "http", &hints, &ai0);
```



120

TCPサーバの基本ロジック

NTT Information Sharing Platform Laboratories



すべてのプロトコルについてソケットを生成し待受ける (listen)

```
int s[MAXSOCKS];
for (smax = 0, ai = ai0; ai; ai = ai->ai_next) {
    s[smax] = socket(ai->ai_family, ai->ai_socktype,
                    ai->ai_protocol);

    if (bind(s[smax], ai->addr, ai->addrlen) < 0)
        continue; /* bindに失敗したら次のプロトコルを試す */
    if (listen(s[smax], 5) < 0)
        continue; /* listenに失敗したら次のプロトコルを試す */
    smax++;
}
```

121

クライアントからの接続受け入れ処理

NTT Information Sharing Platform Laboratories

- IPv4, IPv6どちらの接続が来るか不明
 - IPv4, IPv6 2つのソケットを開いて待受ける時、どちらかに接続が来たら受入れ処理を開始
 - 「どちらか」を`select()`により判定する
- `select()`システムコールの役割
 - ソケットの集合を渡し、いずれか一つが読み込み可能になるまで待つ
 - `listen()`により待ち受けているソケットを`select()`に渡し、接続があるまで待つ

122

クライアントからの接続を受け入れるaccept()

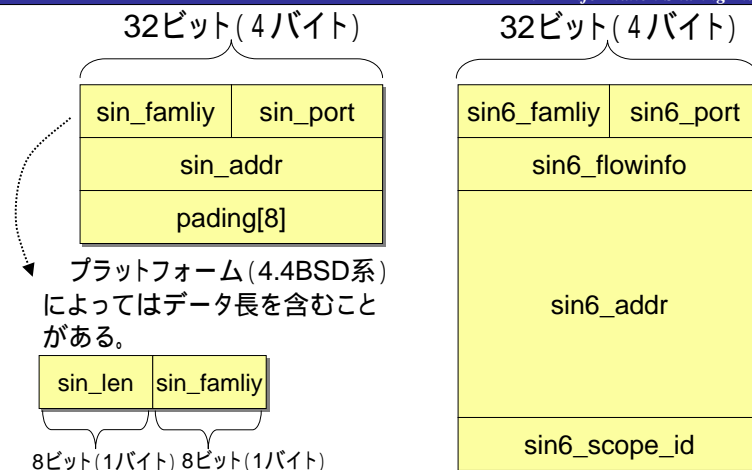
NTT Information Sharing Platform Laboratories

- `int accept(int s, /* listen()で待つソケット */
 struct sockaddr *saddr, /* OUT */
 int *len /* OUT */) /* OUTはaccept()が情報を渡すという意味 */`
- 接続要求が来たクライアントとの通信(I/O)用ソケットを返す
- 接続してきたクライアントを示すソケットアドレスを返す
 - saddrにはプロトコル種別に応じたソケットアドレスが設定される
 - IPv4の通信の場合 struct sockaddr_in
 - IPv6の通信の場合 struct sockaddr_in6
- プログラマがIPv4, IPv6どちらが返ってきても対応できるように、saddrに十分に大きな領域を確保しておく必要がある

123

2つの構造体の比較 sockaddr_in (IPv4), sockaddr_in6 (IPv6)

NTT Information Sharing Platform Laboratories

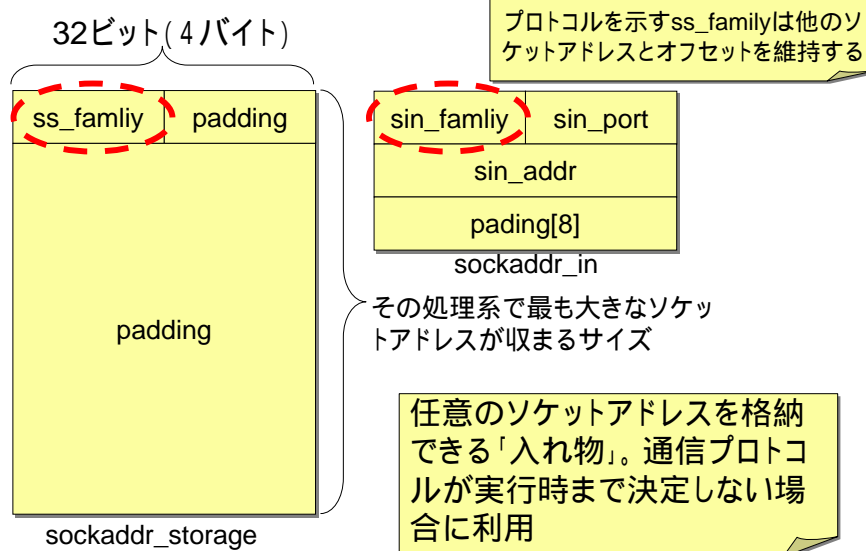


sizeof(struct sockaddr_in) < sizeof(struct sockaddr_in6)

124

任意のプロトコルのソケットアドレスを格納する sockaddr_storage 構造体

NTT Information Sharing Platform Laboratories



125

select()とaccept()による接続要求処理の多重化

NTT Information Sharing Platform Laboratories

```

struct sockaddr_storage ss;
int sslen = sizeof(ss);

while (1) {
    /* IPv4, IPv6のいずれかの接続が来るまで待つ */
    select(FD_SETSIZE, &rfd, NULL, NULL, NULL);
    s = 接続が来たソケットをrfdから取り出す;

    /* 接続を受け入れる。 ss にはクライアントの情報が書き込まれる */
    as = accept(s,
               (struct sockaddr *)&ss,
               &srlen);
    スレッド or サブプロセス生成(as);
}
    
```

スレッド or サブプロセス(int as){
 スレッド or サブプロセス(int as){
 スレッド or サブプロセス(int as){
 ソケットas に対してI/Oを行う
 recv(as, ...);
 send(as, ...);
 }
 }
 }

IPv4, IPv6どちらのソケットアドレスでもすっぽり格納できる領域を確保

126

プロトコル非依存に書き換えたアプリケーション サンプルコード(TCPサーバ編) (1 / 2)

NTT Information Sharing Platform Laboratories

```
/*
 * 接続してきたクライアントに対して文字列を送信するサーバ
 * IPv6/IPv4デュアルバージョン
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

int main(int argc, char *argv[])
{
    int s[64]; /* クライアントと通信を行うソケット */
    struct addrinfo hints; /* getaddrinfoの動作を指定するヒント情報 */
    struct addrinfo *ai0; /* addrinfoリストの先頭の要素 */
    struct addrinfo *ai; /* 処理中のaddrinfoリストの要素 */
    int g; /* getaddrinfoの処理結果 */
    char *servername; /* ホスト名 */
    int smax; /* 接続を受け入れるプロトコルの数 */
    fd_set rfd, rfd0;
    int i;

    /** マンドライン引数のチェック */
    if (argc != 2) {
        fprintf(stderr, "usage: dual-tcp-server <port>\n");
        exit(1);
    }

    /** コマンドラインの第一引数(ポート名)をセット */
    servername = argv[1];

    /** ヒント情報の初期化 */
    memset(&hints, 0, sizeof(hints));
    /* プロトコルは指定しない: IPv4 or IPv6 */
    hints.ai_family = PF_UNSPEC;
    /* ストリーム型(TCP)による通信を指定 */
    hints.ai_socktype = SOCK_STREAM;
    /* PASSIVE側のソケット生成を指定 */
    hints.ai_flags = AI_PASSIVE;

    /* 名前とポート名の解決 & 接続候補のリストを取得 */
    g = getaddrinfo(NULL, servername, &hints, &ai0);
    if (g) {
        fprintf(stderr, "%s", gai_strerror(g));
        exit(1);
    }

    smax = 0;
    /** addrinfoリストの要素を先頭から接続できるまで順に試行する */
    for (ai = ai0; ai; ai = ai->ai_next) {
        /* ソケットの生成 */
        s[smax] = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);

        /* ソケットの生成に失敗したらリストの次の項目を試す */
        if (s[smax] < 0)
            continue;
        /* ポート番号をソケットに結びつける */
        if (bind(s[smax], ai->ai_addr, ai->ai_addrlen) < 0) {
            perror("bind");
            close(s[smax]);
            continue;
        }
        /* クライアントからの接続の要求へ待機する */
        if (listen(s[smax], 5) < 0) {
            perror("listen");
            close(s[smax]);
            continue;
        }
    }
    smax++;

    /** forループで待受けソケットを生成できない場合はエラーとする */
    if (smax == 0) {
        fprintf(stderr, "cannot create server socket\n");
        exit(1);
    }

    /** s[xx]へのクライアントからの接続要求をselectにより監視 */
    FD_ZERO(&rfd0);
```

127

プロトコル非依存に書き換えたアプリケーション サンプルコード(TCPサーバ編) (2 / 2)

NTT Information Sharing Platform Laboratories

```
for (i = 0; i < smax; ++i)
    FD_SET(s[i], &rfd0);

    /** サーバとのデータのI/O */
    while (1) {
        int as;
        struct sockaddr_storage as; /* クライアントのソケットアドレス */
        int salen; /* ソケットアドレスの長さ */
        int n;
        int pid;

        /* クライアントが接続して(るまでselect())で待つ */
        rfd = rfd0;
        n = select(smax + 1, &rfd, NULL, NULL, NULL);
        if (n < 0) {
            perror("select");
            exit(1);
        }

        /* 接続を受け付けたソケットを求める */
        for (i = 0; i < smax; ++i) {
            if (FD_ISSET(s[i], &rfd))
                break;
        }

        /* クライアントからの接続を受け入れる */
        /* クライアントを示すソケットアドレスがasに書き込まれる */
        as = accept(s[i], (struct sockaddr *)&sa, &salen);
        if (as < 0) {
            perror("accept");
            exit(1);
        }

        /* サブプロセスを生成し、クライアントとのI/Oを処理させる */
        if ((pid = fork()) < 0) {
            perror("fork");
            exit(1);
        }

        if (pid == 0) {
            /* サブプロセスの処理 */
            char *text = "send string: V*server to client\n";
            write(as, text, strlen(text));
            close(as);
            exit(0);
        } else
            close(as);
    }
```

128

プロトコルごとに処理を分ける例

NTT Information Sharing Platform Laboratories

- トランスポート層の中にIPアドレスに関する情報が埋め込まれているアプリケーション
– FTP, SIP, ... など

IPv4でのFTP (send-port)

```
USER ftp
PASS xxxx@nttv6.net
PORT 192.168.0.21,14,248
RETR index.txt
```

IPv6でのFTP (send-port)

```
USER ftp
PASS xxxx@nttv6.net
EPRT |2|2001:db8:aaaa::21|3832|
RETR index.txt
```

- ログ生成・アクセスリスト処理など

```
192.16.178.192 - - [01/Nov/2005:23:19:20 +0900] "GET /index.html HTTP/1.1" 301
255 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
2001:db8:aaaa::1 - - [01/Nov/2005:23:19:49 +0900] "GET /index.html HTTP/1.1"
301 255 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)"
```

129

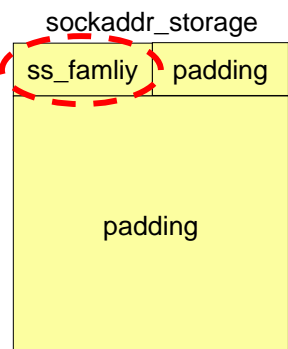
プロトコル別の処理が必要になったら

NTT Information Sharing Platform Laboratories

```
struct sockaddr_storage ss;
struct sockaddr_in *sin4;
struct sockaddr_in6 *sin6;

switch (ss.ss_family) {
case AF_INET:
    sin4 = (struct sockaddr_in *)&ss;
    ...
    break;
case AF_INET6:
    sin6 = (struct sockaddr_in6 *)&ss;
    ...
    break;
}
```

場合分け



sockaddr_storageのポインタをキャストすれば各プロトコルのソケットアドレスとして参照可能

130

IPv6アプリケーション構造のまとめ

NTT Information Sharing Platform Laboratories

- 設計の基本方針
 - プロトコル依存性を排除する
 - 依存性のある部分は極力APIに隠蔽させる
 - プロトコルごとの場合分けは必要最小限に
- IPv4のアプリケーションでも新しいAPIは有効
 - 名前解決近辺のコードが非常に簡潔に記述できる
 - 多くの場合、OSがIPv4 onlyのシステム上でも稼動する
 - IPv4の利用を強制するフラグもある
 - 利用プロトコルを指定できるようにしておくとい
 - ssh -6 : IPv6の利用を強制する
 - ssh -4 : IPv4の利用を強制する
 - ssh : IPv6,IPv4両方の接続を試す
IPv4のみの動作としたい場合は“-4”をデフォルトにする

131

チュートリアルまとめ

NTT Information Sharing Platform Laboratories

- IPv6の基本機能はほぼFIXした
- しかし周辺の拡張機能は今も発展中
- IPv6の実利用のためには、目的に応じた拡張機能を適材適所で利用することが不可欠

132

参考文献(1)

NTT Information Sharing Platform Laboratories

- IPv6の標準化経緯
 - “The Recommendation for the IP Next Generation Protocol”, 1995/1, RFC1752.
 - 「WIDEプロジェクト 1995年度研究報告書」, 第6部 IP version 6, <http://www.wide.ad.jp/document/reports/pdf1995/index.htm>.
- IPv6基本仕様
 - “Internet Protocol Version 6 (IPv6) Addressing Architecture”, 2003/4, RFC3513. (Draft Standard改訂版発行予定)
 - “Internet Protocol, Version 6 (IPv6) Specification”, 1998/12, RFC2460.
 - “Neighbor Discovery for IP Version 6 (IPv6)”, 1998/12, RFC2461.
 - “IPv6 Stateless Address Autoconfiguration”, 1998/12, RFC2462. (Draft Standard改訂版発行予定)
 - “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, 1998/12, RFC2463. (Draft Standard 改訂版発行予定)
 - “IPv6 Node Requirements”, 2004/8, draft-ietf-ipv6-node-requirements-11.txt.

133

参考文献(2)

NTT Information Sharing Platform Laboratories

- DNS関連
 - “IPv6 Host Configuration of DNS Server Information Approaches”, 2005/11, draft-ietf-dnsop-ipv6-dns-configuration-04.txt
 - “Deprecation of ip6.int reverse DNS service in APNIC”, 2005/8, <http://www.apnic.net/docs/policy/discussions/prop-030-v001.txt>
- DHCPv6
 - “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2003/7, RFC3315.
 - “DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)”, 2003/12, RFC3646.
 - “IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6”, 2003/12, RFC3633.
 - “Requirements for IPv6 Prefix Delegation”, 2004/6, RFC3769.
 - “Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6”, 2004/4, RFC3736.
 - “Simple Network Time Protocol Configuration Option for DHCPv6”, 2003/12, draft-ietf-dhc-dhcpv6-opt-sntp-00.txt

134

参考文献(3)

NTT Information Sharing Platform Laboratories

- アドレス選択方式
 - “Default Address Selection for Internet Protocol version 6 (IPv6)”, 2003/2, RFC3484.
- 匿名アドレス
 - “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”, 2001/1, RFC3041.
 - “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”, 2005/5, draft-ietf-ipv6-privacy-addr-v2-04.txt. (RFC3041の改定版)
- ネットワーク管理
 - “Management Information Base for IP Version 6: Textual Conventions and General Group”, 1998/12, RFC2465.
 - “Management Information Base for the Internet Protocol (IP)”, 2004/5, draft-ietf-ipv6-rfc2011-update-10.txt.
 - “IP Forwarding Table MIB”, 2004/8, draft-ietf-ipv6-rfc2096-update-07.txt.
 - “Management Information Base for the Transmission Control Protocol (TCP)”, 2005/3, RFC4022.
 - “Management Information Base for the User Datagram Protocol (UDP)”, 2005/6, RFC4113.
 - “IP Tunnel MIB”, 2005/6, RFC4087.

135

参考文献(4)

NTT Information Sharing Platform Laboratories

- アドレス関連
 - “Deprecating Site Local Addresses”, 2004/9, RFC3879.
 - “Unique Local IPv6 Unicast Addresses”, 2005/10, RFC4193.
 - “6bone (IPv6 Testing Address Allocation) Phaseout”, 2004/3, RFC3701.
- MobileIPv6 / NEMO
 - “Mobility Support in IPv6”, 2004/6, RFC3775.
 - “Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents”, 2004/6, RFC3776.
 - “Network Mobility (NEMO) Basic Support Protocol”, 2004/6, RFC3963.

136

参考文献(5)

NTT Information Sharing Platform Laboratories

- IPsec関連
 - IPセキュリティ・アーキテクチャ全般
 - “Security Architecture for the Internet Protocol”, 1998/11, RFC2401
 - “IP Security Document Roadmap”, 1998/11, RFC2411
 - IPsecの認証機能、暗号化機能
 - “IP Authentication Header”, 1998/11, RFC2402
 - “The Use of HMAC-MD5-96 within ESP and AH”, 1998/11, RFC2403
 - “The Use of HMAC-SHA-1-96 within ESP and AH”, 1998/11, RFC2404
 - “The ESP DES-CBC Cipher Algorithm With Explicit IV”, 1998/11, RFC2405
 - “IP Encapsulating Security Payload (ESP)”, 1998/11, RFC2406
 - “The NULL Encryption Algorithm and Its Use With IPsec”, 1998/11, RFC2410
 - “The ESP CBC-Mode Cipher Algorithms”, 1998/11, RFC2451
 - 鍵交換機能
 - “The Internet IP Security Domain of Interpretation for ISAKMP”, 1998/11, RFC2407
 - “Internet Security Association and Key Management Protocol (ISAKMP)”, 1998/11, RFC2408
 - “The Internet Key Exchange (IKE)”, 1998/11, RFC2409
 - “The OAKLEY Key Determination Protocol”, 1998/11, RFC2412
 - “Internet Key Exchange (IKEv2) Protocol”, 2004/09, draft-ietf-ipsec-ikev2-17.txt

137

参考文献(6)

NTT Information Sharing Platform Laboratories

- プログラミング関係
 - “Basic Socket Interface Extensions for IPv6”, 2003/2, RFC 3493.
 - “Advanced Sockets Application Program Interface (API) for IPv6”, 2003/5, RFC3542.
 - 「IPv6ネットワークプログラミング」, 萩野純一郎 著、小川彩子 訳, 2003年2月, アスキー.
 - “Unix Network Programming Volume 1., 3rd Edition, The Sockets Networking API”, W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, 2003/11, Prentice Hall.
 - 「Winsock 2.0プログラミング - Window Socket APIによるネットワークプログラミングのすべて」, ルイスナッパー 著, 江村豊 訳, 1998年8月, ソフトバンクパブリッシング.

138