DNS最新動向～BIND9～

神明 達哉
(株)東芝 研究開発センター/KAMEプロジェクト
jinmei@{isl.rdc.toshiba.co.jp, kame.net}

はじめに

アンケートへのご協力をお願い致します

対象

- DNSの運用経験があり、プロトコル上の背景を深めたい人
- BIND 9未経験者で、BIND 9に関心を持っている人
  - とに、　BIND 4または8の運用経験者
- DNSの最近の話題に関心のある人
- IPv6とDNSの関係に興味のある人

このチュートリアルの内容

- □DNSプロトコルの基礎
  - ○ドメイン名空間、名前解決手順、リソースレコード
- □BIND 9の導入
  - ○設定方法、移行上の注意点
- □DNSプロトコルの最近の話題
  - ○動的更新、差分ゾーン転送、IPv6関係、DNSSECなど
  - ○BIND 9における運用方法
- □BIND 9に関するその他の話題
  - ○管理ツール、view、lwres、dig
- □範囲外の話題
  - ○多言語ドメイン

DNSプロトコルの基礎

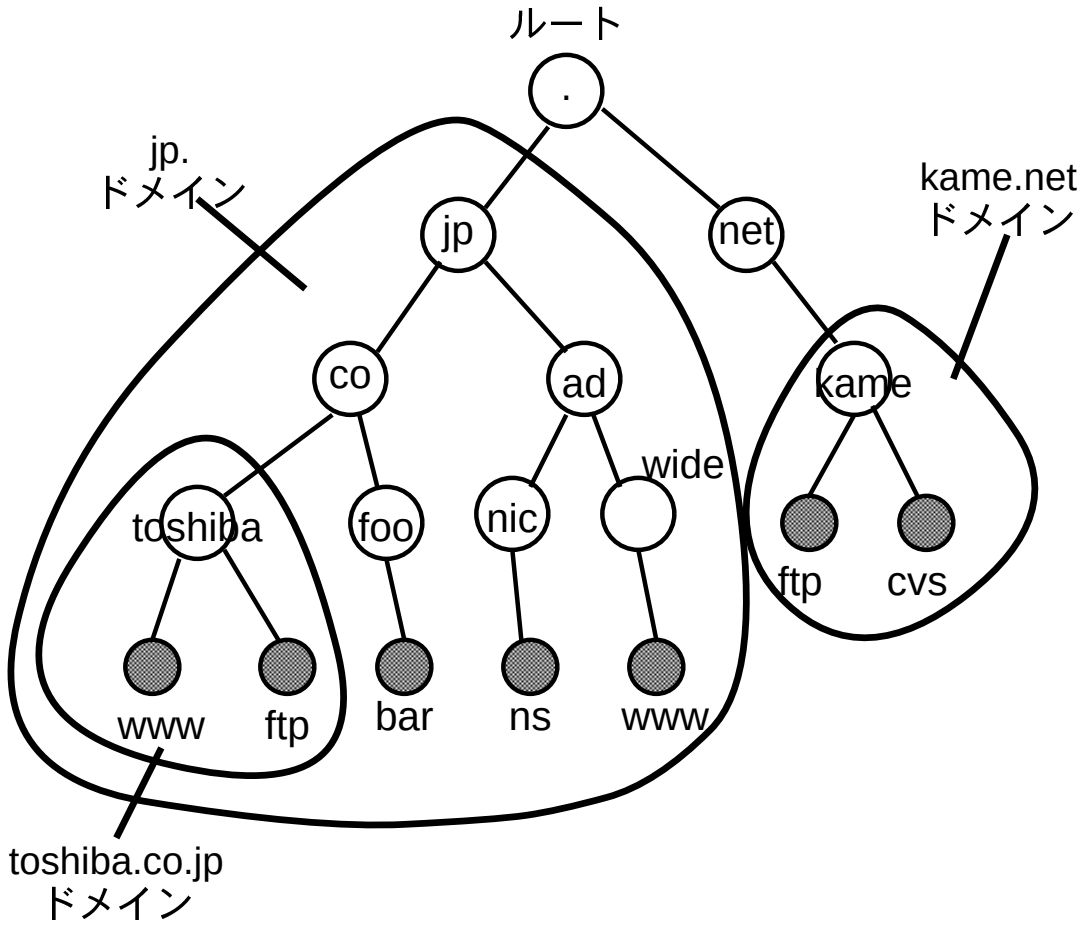
- ドメイン、ゾーン
- データ検索手順
- リソースレコード

DNS (Domain Name System)とは

- ホスト名とIPアドレス間の変換機能
  - www.toshiba.co.jp. <-> 211.7.133.18
- 階層構造を持つ分散データベース
  - 一貫性を保ちつつ負荷の集中を避ける
  - キャッシュによる検索効率の向上

ドメイン名空間

□「ルート」を頂点とする木構造



ドメインに関する用語

- ラベル
  - 英数字とハイフン(-)からなる任意の文字列
    - アルファベットの大文字小文字は無関係
  - 木構造の各ノードに付随

- ドメイン
  - あるノードを頂点とする部分木
  - ドメイン名=頂点ノードからルートまでのラベルをピリオドで区切ってつなげた文字列
    - toshiba.co.jp.
    - kame.net.
  - いわゆる「ホスト名」もドメイン名の一種
    - www.toshiba.co.jp.
    - kame202.kame.net.
    - foo.bar-baz.com.

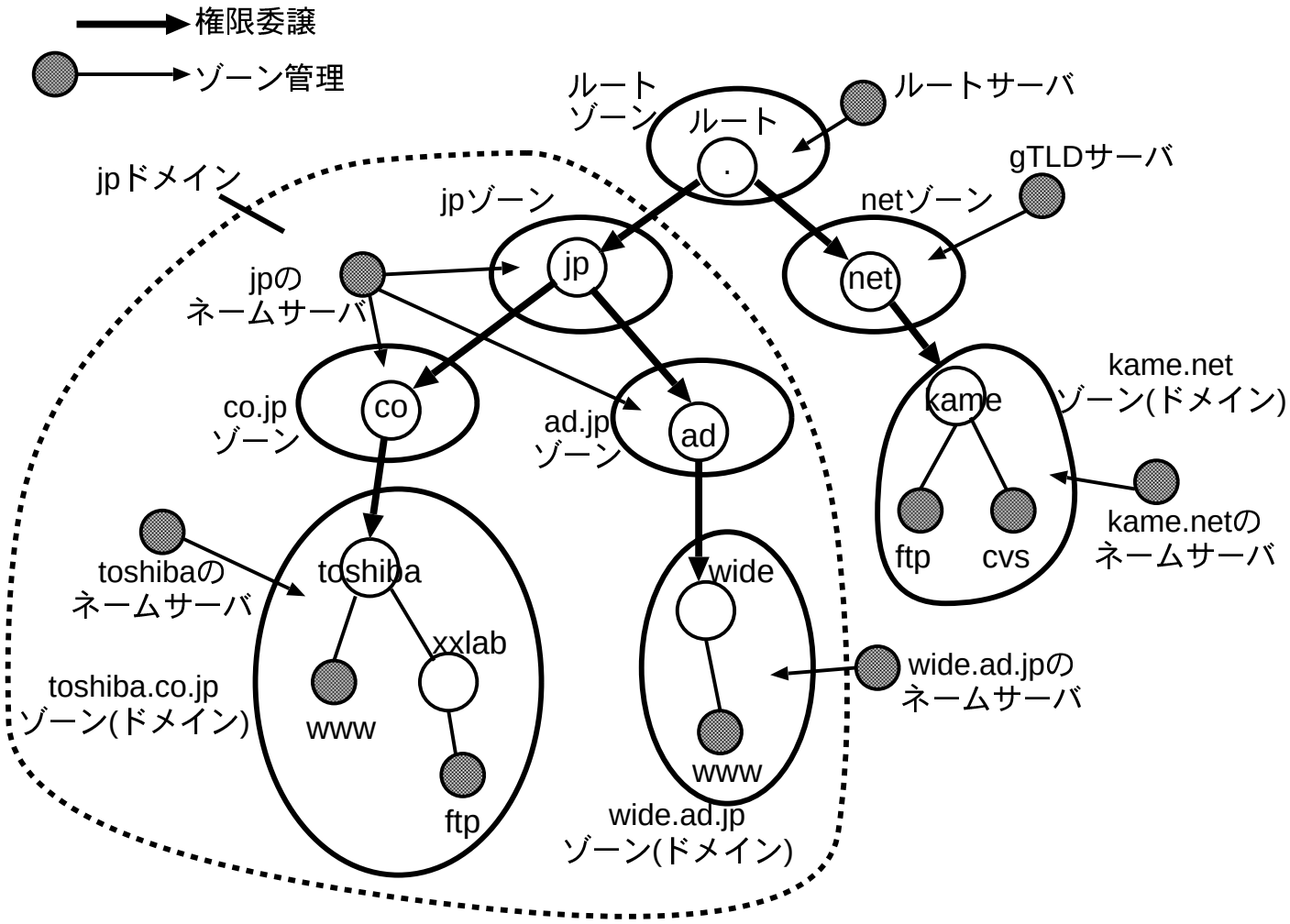
トップレベルドメイン(TLD)

- ルートドメイン
  - ドメイン名の空間全体
- gTLD (generic TLD)
  - com, net, orgなど
- ccTLD (country code TLD)
  - 国別、基本的にISO3166の国名コード(除イギリス)
  - jp(日本), kr(韓国), de(ドイツ)など

ゾーン

- ドメインの連結部分集合
- 管理単位で分類
- 必ずしもドメインとは一致しない
  - jpゾーン != jpドメイン
- 上位ゾーンから下位ゾーンへ管理権限を委譲 (delegation)
- ネームサーバ
  - 各ゾーン内のデータを管理するサーバ
    - 一台のサーバで複数のゾーンを管理する場合もある
    - 各ゾーンに対し、複数台のネームサーバを立てるのがふつう
    - ネームサーバ自身が属するゾーンと管理するゾーンが異なっていてもよい
  - とくに重要なネームサーバ
    - ルートサーバ、　gTLDサーバ

ゾーンの権限委譲(delegation)



DNSのデータ検索(名前解決)

- 正引き: ホスト名からIPアドレスを求める
- 逆引き: IPアドレスからホスト名を求める
- サーバ・クライアントモデル
  - サーバ: ネームサーバ
  - クライアント: リゾルバ
  - UDP (またはTCP)/IP、 53番ポート

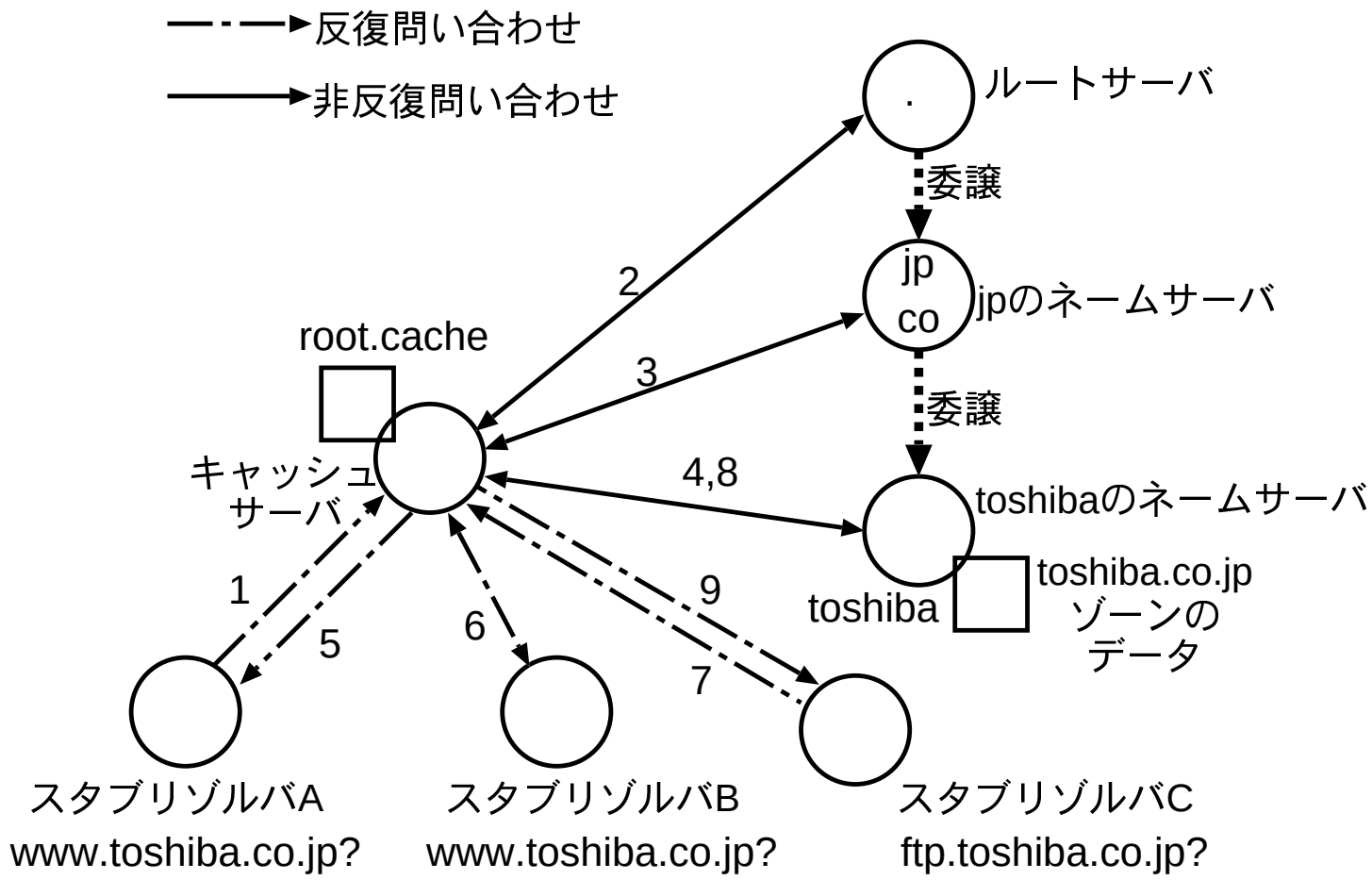
キャッシュ

- 検索結果をキャッシュして再利用できる
  - 応答速度の向上、上位ゾーンのサーバ負荷を軽減
- キャッシュの有効期限
  - データのTTL(Time To Live)値
- キャッシュサーバ
  - 検索、キャッシュ専用
  - 自分ではゾーンのデータを管理しない(例外も多い)
  - 一般ユーザから見える「DNSサーバ」
    - アプリケーションのリゾルバ (スタブリゾルバ)はキャッシュサーバに問い合わせる

キャッシュサーバを用いた検索手順

- スタブリゾルバからキャッシュサーバへの問い合わせ
  - 反復問い合わせ(recursive query): 全検索手順の代行を要求
- キャッシュサーバによる検索
  - 常にルートサーバへ問い合わせからはじまる
    - ルートサーバのアドレスは何らかの方法で入手
    - コードへの埋め込みやヒントファイルなど
  - 下位ゾーンの検索は、委譲先のサーバに対して継続
  - 目的のゾーンに到達するまで繰り返す
  - 各問い合わせは非反復的
    - ルートサーバやgTLDサーバは反復問い合わせにも非反復的に応答する
- キャッシュされた名前に対する問い合わせ
  - キャッシュされたデータをそのまま返す
  - TTLが少しずつ減っていく

キャッシュサーバへの反復問い合わせ



プライマリとセカンダリ

- プライマリサーバ
  - データベースのマスターファイルを管理
  - ファイルの編集はプライマリの持つデータに対して行う
- セカンダリサーバ
  - プライマリサーバのデータをコピーとして持つ
  - プライマリサーバのバックアップ
  - 冗長性確保・負荷分散
    - 分散配置が重要
- データのコピー: ゾーン転送 (zone transfer)
  - TCPを利用
  - コピーのタイミング
    - 定期的に or プライマリからの通知による
- 問い合わせる側(リゾルバ)にとっては違いはない
  - どのサーバに聞いても同じ応答が返る

リソースレコード(RR)

- DNSデータベースの構成要素(レコード)

- <name> <ttl> <class> <type> <data>
  - name: ドメイン名、データベース検索の鍵
  - ttl: キャッシュの有効期限、秒単位
  - class: ふつうIN (Internet Domain)
  - type: レコードの種類(A, NS, MX, PTR, SOAなど)
  - data: class, type依存の値
    - 例: Aレコードの場合はIPv4アドレス
    - 複数のフィールドからなる場合もある

- RRの記述例

  www.kame.net. 3600 IN A 203.178.141.220

ゾーンファイル

- ゾーンごとのデータベースファイル
  - ゾーン内の名前に対するRRの列
  - そのままBINDの設定ファイルになる
- 記法
  - 上位ドメインの省略
    - 設定間違いに注意

(kame.net.のゾーンファイル)

```
www 3600 IN A 10.0.0.1 ;; for www.kame.net.
ns  3600 IN NS ns.kogame.org ;; おそらく最後の"."忘れ
```

  - "$ORIGIN": 上位ドメインのデフォルト値

```
$ORIGIN kame.net.
www 3600 IN A 10.0.0.1 ;; for www.kame.net.
```

  - @: ゾーン自体の名前
  - TTL: BINDでは時間の単位を指定できる
    - 例: 1D (1日), 1H30M (1時間30分)

SOA (Start of Authority) RR (1/3)

```
@ 1D IN SOA  <プライマリサーバ名> <管理者メールアドレス> (
        2001101505      ; シリアル番号
        3600            ; リフレッシュ間隔
        600             ; リトライ間隔
        2419200         ; ゾーン有効期限 (この例では4週間)
        1200            ; 最小TTL
        )
```

- □すべてのゾーンファイルに必要
- □SOA RRのパラメータ
  - ○<プライマリサーバ名>
    - ▹ns.wide.ad.jp.など
    - ▹動的更新を利用するなら重要
  - ○<管理者メールアドレス>
    - ▹@のかわりに". "を使う
    - ▹jinmei@kame.net → jinmei.kame.net.

SOA RR (2/3)

- SOA RRのパラメータ(続き)
  - シリアル番号
    - ゾーン転送が必要かどうかの判定に用いる
    - 32ビット整数、値は単調に増加していれば何でもよい
    - YYYYMMDDxx方式なら4294年までOK
    - 更新忘れに注意
  - リフレッシュ間隔
    - 定期的なゾーン転送の間隔
    - 推奨値: 環境に依存するが、数時間程度 (RFC 1912)
  - リトライ間隔
    - ゾーン転送に失敗した後のリトライまでの時間
    - 推奨値: リフレッシュ間隔の数分の一程度
  - ゾーン有効期限
    - ゾーン転送に失敗し続けた後、ゾーンを無効とみなすまでの期間
    - 推奨値: 2-4週間

SOA RR (3/3)

- 最小TTLパラメータ
  - RFC 1035: ゾーンのデフォルトTTL
    - 推奨値: 数日程度
  - RFC 2308: ネガティブキャッシュのTTL
    - 推奨値: 数10分程度
- TTLパラメータの意味変更に注意
  - デフォルトTTLとして使う場合、あまり小さいと不便
    - キャッシュの効率が下がる
  - ネガティブキャッシュのTTLとして使うなら、あまり大きいと危険
    - 新規に追加した名前が長時間牽けなくなる
- どう設定すべきか
  - ゾーンのデフォルトTTLを明示的に指定する ($TTL)
  - 最小TTLはネガティブキャッシュのTTLとして設定する
    - 小さめの値にする

A (Address) RR

- ドメイン名からIPv4アドレスへの対応を与える

  www.kame.net. 1D IN A 203.178.141.220

- 一つのドメイン名に複数のアドレスが対応することもある

  www.kame.net. 1D IN A 203.178.141.220
  www.kame.net. 1D IN A 203.178.141.221

NS (Name Server) RR

□下位ゾーン名に対して、権威を持つネームサーバ名を与える

○権限委譲のために必要

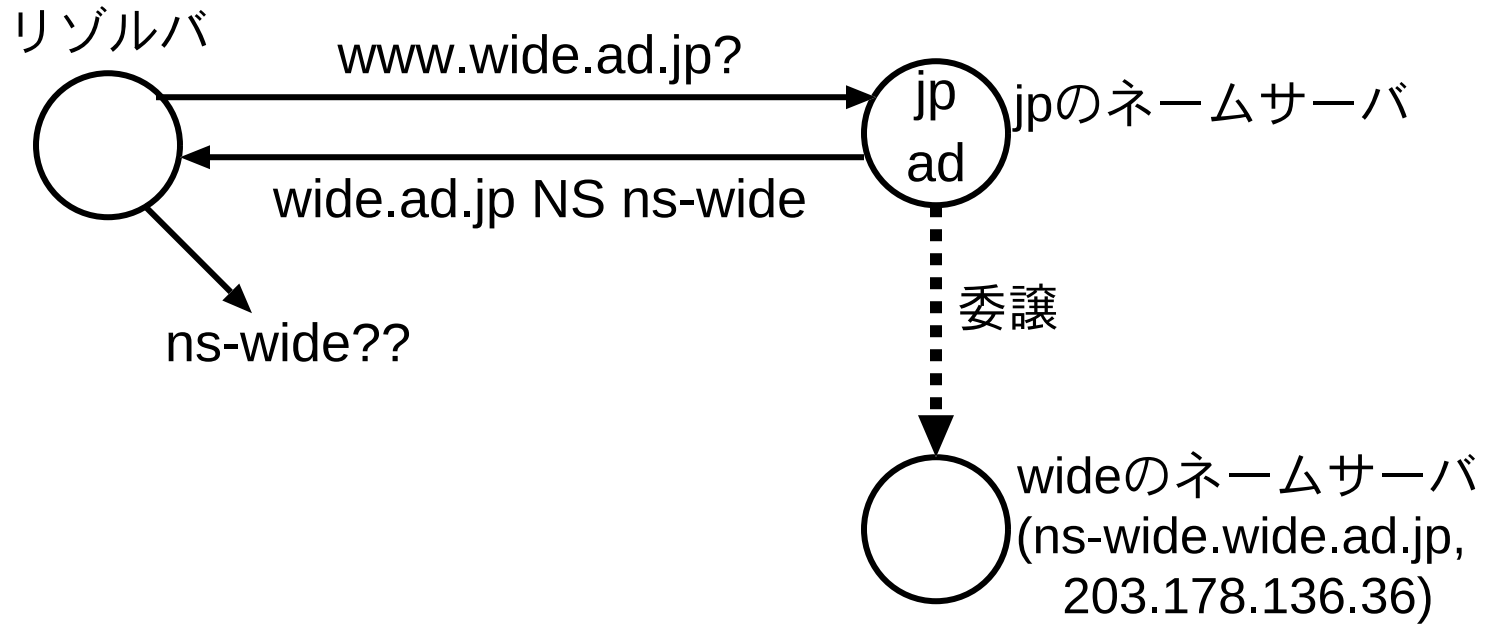
○検索のために、上位ゾーンのゾーンファイルにも登録する

wide.ad.jp. 1D IN NS ns-wide.wide.ad.jp.

グルー(glue) RR

- ○NS RRが与えるネームサーバのアドレスを得るためのRR
  - ▹ふつうA RR
- ○上位ゾーンのデータベースに登録
  - ▹グルーがなければ下位ゾーンにアクセスできない

ns-wide.wide.ad.jp. 1D IN A 203.178.136.36



PTR (Pointer) RR

□IPアドレスからドメイン名への対応を与える

□IPv4の場合

- ○全アドレスに共通の上位ドメイン: in-addr.arpa.
- ○1バイトずつを1ラベルとし、下位バイトから逆順に4階層で表現

203.178.141.220 = www.kame.net.
220.141.178.203.in-addr.arpa. 1D IN PTR www.kame.net.

□IPv6にも適用できる(後述)

MX (Mail Exchanger) RR

□ドメイン名に対し、そのドメイン行きのメールを処理するサーバ名を与える

kame.net. 1D IN MX 10 orange.kame.net.

- ○foo@kame.net行きメールはorange.kame.net.が処理する
- ○10: プリファレンス値
  - ▹複数のサーバ間の優先順位を定める
- ○メールサーバ名に対するアドレスは別途求める必要がある

□MXの必要性

- ○メール配送の信頼性確保
  - ▹少しでも「近い」サーバまでとにかく届ける
- ○メールアドレスのドメイン名とサーバのホスト名との関係
  - ▹jinmei@orange.kame.netではなくjinmei@kame.net
  - ▹"kame.net."という「ホスト」とは別にメールサーバを立てたい

CNAME (Canonical Name) RR

□別名定義

paradise.kame.net. 1D IN CNAME kame202.kame.net.

kame202.kame.net. 1D IN A　203.178.141.202

□CNAMEの利用に関する注意

○チェインを避ける

▹検索効率、無限プの防止

○NS, MXが与える名前に対して使わない

▹悪い例:

kame.net. 1D IN NS paradise.kame.net.

paradise.kame.net. 1D IN CNAME kame202.kame.net.

kame202.kame.net. 1D IN A　203.178.141.202

○同一の名前に複数の別名を付けない(multiple CNAMEs)

▹悪い例:

dup.kame.net. 1D IN CNAME foo.kame.net.

dup.kame.net. 1D IN CNAME bar.kame.net.

CNAME (Canonical Name) RR (続き)

- CNAMEの利用に関する注意(続き)
  - 同一の名前に対してCNAMEとそれ以外のRRを同時に定義しない
    - 悪い例:

```
dup.kame.net. 1D IN CNAME foo.kame.net.
dup.kame.net. 1D IN A     1.2.3.4
```

ゾーンファイル全体のサンプル

□jinmei.org.ゾーン

```
$TTL 1D
@       1D IN   SOA     ns.jinmei.org.     root.jinmei.org. (
                        2001122403      ; serial
                        3600            ; refresh
                        600             ; retry
                        2419200         ; expire
                        1200 )          ; minimum
        IN      NS      ns
        IN      MX      10      mail

ns      IN      A       203.178.141.194
mail    IN      A       203.178.141.195

www     IN      CNAME   ns

child   IN      NS      ns.child
        IN      NS      ns.kame.net.
ns.child IN     A       203.178.141.202
```

BIND 9の初歩

- **BIND**とは
- **BIND 9**の特徴
- **BIND 9**のインストール、設定

BIND

- Berkeley Internet Name Domain
  - もっとも使われているDNSの実装
- ISC (Internet Software Consortium)
  - http://www.isc.org/
- 入手
  - ftp://ftp.isc.org/isc/bind/
  - ftp://ftp.isc.org/isc/bind9/
  - 開発中のコードは非公開

BINDのバージョン

- 3つの系列: 4, 8, 9

- どのバージョンを使うべきか
  - BIND 4ではそろそろ辛い
    - 開発は原則終了
    - 機能、互換性、標準への準拠の面でも難あり
  - BIND 8 vs BIND 9
    - BIND 9でも大体問題ない
    - IPv6やDNSSECを本格運用するなら BIND 9しかない
    - 大規模ゾーンでの性能・高度の安定性が必要ならBIND 8
    - いずれにしても、最新版を利用するのが原則
  - BIND 9.1 vs 9.2
    - 9.2がおすすめ

- このチュートリアルでは
  - 原則としてBIND 9.2を対象
  - 必要に応じて9.1や8.xにも言及する

BIND 9

- BINDの最新バージョン
- 完全な書き直し
  - 仕様への準拠にこだわったリファレンス実装
  - DNSSEC対応のためのthreadベース
- プラットフォーム
  - *BSD, Linux, Solaris 2.6-8, etc.
  - 9.2からはWindows NTでも動作
- DNSの最新仕様に対応
  - IPv6関係, DNSSEC
- ビュー(view)
  - 問い合わせ元に応じて答え方を変更
- lwres (light weight resolver)
  - DNSSEC, IPv6関係など新機能サポートのためのリゾルバ

BIND 9の配布物

- 実行バイナリ: bind-9.x.y/bin/
  - named: ネームサーバ、キャッシュサーバ
  - lwresd: ローカルキャッシュサーバ、lwresプロトコル対応
  - nsupdate: 動的更新(dynamic update)クライアント
  - rndc: named制御ツール
  - dig: デバッグツール
    - 他にhost, nslookupなど
  - dnssec-xxx: DNSSEC関係のツール
- ライブラリ: bind-9.x.y/lib/
  - libbind: BIND 8互換リゾルバライブラリ
  - liblwres: lwresライブラリ

BIND 9のインストール

- autoconf対応で、ふつうは簡単
  - configure → make でOK
- configureで気をつける点
  - opensslのpath

    --with-openssl
  - thread
    - 無効化がおすすめ

      --disable-threads
    - BIND 9.2では、*BSD, Linuxについてはデフォルトで無効
- random device (/dev/random)
  - DNSSEC, 管理ツールの設定に必要
  - FreeBSD 4.3以前やprngd経由の場合は正常に動作しない
    - キーボードやファイル経由で代替する
  - 厳密な安全性が必要なら/dev/urandomは使わないこと

BIND 9の設定 (1/2)

- コマンドラインオプション
  - -gの意味は変更
  - -b configfileは廃止
- 設定ファイルはほぼBIND 8互換
- デフォルト値が変更されたオプション
  - auth-nxdomain: デフォルトではno
  - fetch-glue: 常にno
  - multiple-cnames: 廃止
    - ゾーンファイル読み込み時にエラーになる
- 未実装のオプション
  - check-names, memstatistics-file, host-statistics, topology,
  - min-roots, rrset-order, rfc2308-type1, statistics-interval

BIND 9の設定 (2/2)

- 追加されたオプション
  - max-cache-size, recursive-clients, max-cache-ttl
  - notify, allow-notify

- ゾーン転送
  - BIND 4との間でのゾーン転送には以下が必要
    transfer-format one-answer;
  - BIND 4, 8とゾーン転送する場合は新しいRRに注意
    - 知らないRRを含むゾーンは丸ごと無視される
    - A6やDNAMEなど

- ログ機能
  - 設定方法はBIND 8と同じ: カテゴリとチャンネルの組み合わせ
    - カテゴリが若干異なるので注意

BIND 9のTTL設定 (1/2)

- 要注意: BIND 8からの移行で一番間違いやすい点

- ゾーンファイルの先頭でデフォルト値を定義するのが簡単
  - $TTL 1D

- BIND 8まで: SOA RRの最小TTL値がデフォルト
  - TTLの指定がないゾーンファイルがたくさん存在する
  - => BIND 9.1ではエラーになる

BIND 9のTTL設定 (2/2)

- BIND 9.2の挙動
  - 1. 個別のRRに対して明示されていればそれを使う
    www.kame.net. 1D IN A 203.178.141.220
  - 2. ゾーンのデフォルト値が明示されていればそれを使う
    ($TTL)
  - 3. 一つ前のRRのTTL値を流用する
    using RFC 1035 TTL semantics
  - 4. SOA RRの場合は限定最小TTL値を使う
    no TTL specified; using SOA MINTTL instead
  - 5. どれにも該当しなければエラー

DNSの最近の話題とBIND 9

- 動的更新**(dynamic update)**
- 差分ゾーン転送**(Incremental Zone Transfer)**
- **IPv6**との関係 **(**リソースレコード、トランスポート**)**
- セキュリティ

動的更新 (Dynamic Update)

- 背景: IPアドレスを自動的に割り当てる仕組みの普及
  - DHCP、IPv6の自動アドレス設定
  - 自動で割り当てたアドレスとDNSデータベースの同期を取りたい

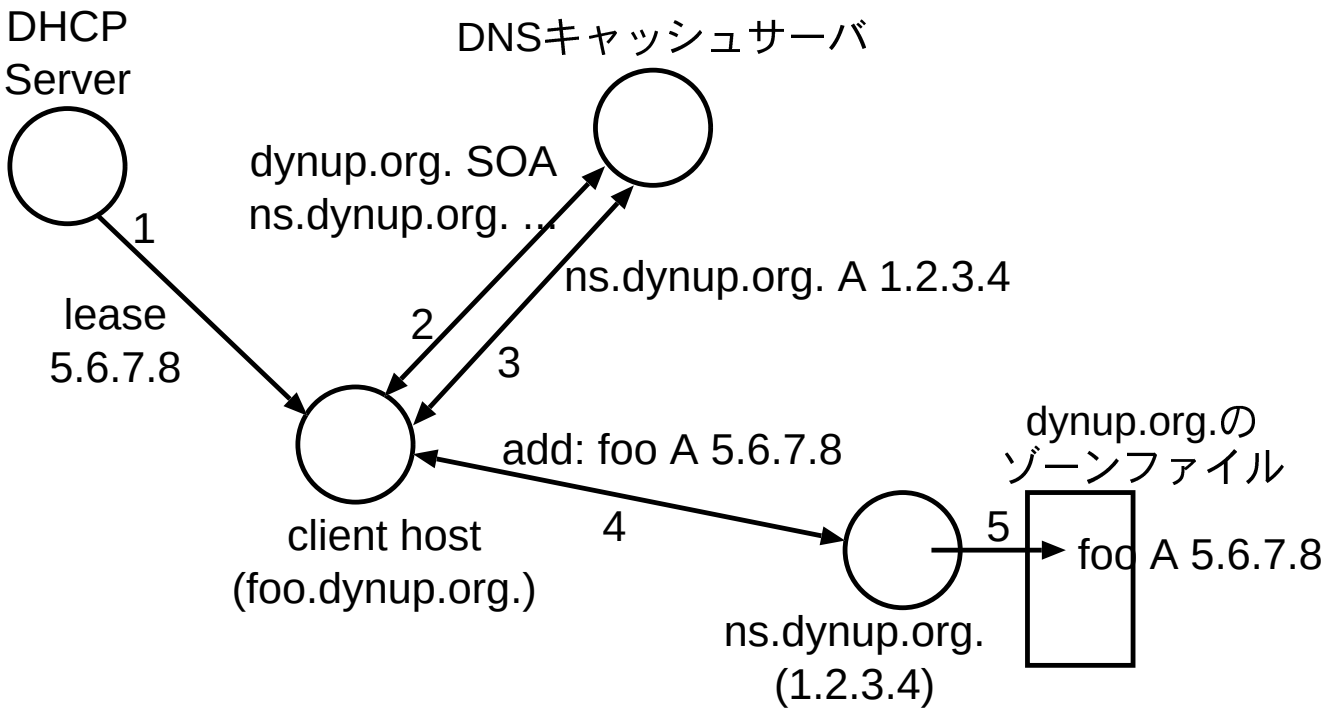
- Dynamic Update
  - RFC 2136
  - ゾーンのデータを動的に変更する枠組み
    - DNSプロトコルの拡張として実現
    - 新しいオペレーションコード　"UPDATE"
  - WindowsやMacはデフォルトでupdateしようとする
    - サーバのログがうるさくなる…

- 認証
  - RFC 3007
  - メッセージ全体に対する署名によって認証する

動的更新 **(Dynamic Update)** の動作

□更新手順(概略)

- ○該当するゾーンのSOA RRを求める
- ○SOAに記述されたプライマリサーバのアドレスを求める
- ○プライマリサーバにUPDATEのパケットを送る



BIND 9による動的更新の運用 (1/4)

- 認証
  - 共有秘密鍵による署名(TSIG RR)
  - アドレスベースの認証では不十分
    - なりすましによる攻撃を受ける
- 動的更新用のゾーン作成
  - 動的更新用のゾーンを分けて運用する方がよい
  - BINDでは動的更新と手動更新は共存できない
- ゾーンに対する更新ポリシーの設定
  - update-policyサブステートメント
    - ホスト名単位など、細かく制御できる
  - allow-updateサブステートメント
    - 古い設定用、推奨しない

BIND 9による動的更新の運用 (2/4)

□サーバ設定ファイルの例 (named.conf)

○ゾーン名: dyn.toshiba.com.

○更新ポリシー(例)

▹管理者はすべてのRRを変更できる

▹ホストjinmeiはjinmei.dyn.toshiba.com.のRRだけを更新できる

```
zone "dyn.toshiba.com" {
        type master;
        file "dyn-toshiba.zone";
        update-policy {
                grant admin.dyn.toshiba.com. wildcard
                    *.dyn.toshiba.com. ANY;
                grant jinmei.dyn.toshiba.com. self
                    jinmei.dyn.toshiba.com.;
        };
};
```

BIND 9による動的更新の運用 (3/4)

□鍵の作成: dnssec-keygenコマンド

dnssec-keygen -a hmac-md5 -b 128 -n host jinmei.dyn.toshiba.com.

- ○2つのファイルができる (内容はほとんど同じ)
  - ▹Kjinmei.dyn.toshiba.com.+157+xxxxx.{key, private}

□鍵の配布

- ○クライアント(ホスト)側
  - ▹2つのファイルを適当な場所にコピー
  - ▹パーミッションに注意
- ○サーバ側
  - ▹設定ファイル(named.conf)に鍵を追加
  - ▹パーミッションに注意

```
key jinmei.dyn.toshiba.com {
        algorithm hmac-md5;
        // Kxxx.privateのKey:フィールドの値
        secret "JXg3xkI+mt/O0ZAbTCqzDA==";
};
```

BIND 9による動的更新の運用 (4/4)

- □更新用コマンド: nsupdate
  - ○鍵ファイルを指定して起動 (-kオプション)
    - ▹パーミッションに注意
    - ▹-yオプションは使わないこと
  - ○行単位で更新用のコマンドを入力する
    - ▹TTLには数字(秒)しか指定できないので注意

```
% nsupdate -k Kjinmei.dyn.toshiba.com.+157+30187.
> server 203.178.141.194 ←省略可能
> update add jinmei.dyn.toshiba.com. 3600 A 1.2.3.4
> send ←単なる空行でもよい
> quit
```

- □うまくいかない場合
  - ○nsupdate -dで試す
  - ○サーバ側のログを確認
  - ○サーバとクライアントでの時刻同期に注意

ジャーナルファイル **(1/2)**

- 動的更新するゾーンのための特殊なゾーンファイル
  - 更新履歴の管理
    - 差分ゾーン転送のために必要
  - バイナリフォーマット

- 最初に動的更新したときに自動生成される
  - 拡張子 "jnl"
  - dyn-toshiba.zone → dyn-toshiba.zone.jnl

ジャーナルファイル (2/2)

- ジャーナルファイルが生成された後
  - 再起同時にジャーナルファイルのみ参照される
  - 元のゾーンファイルは定期的に自動更新される

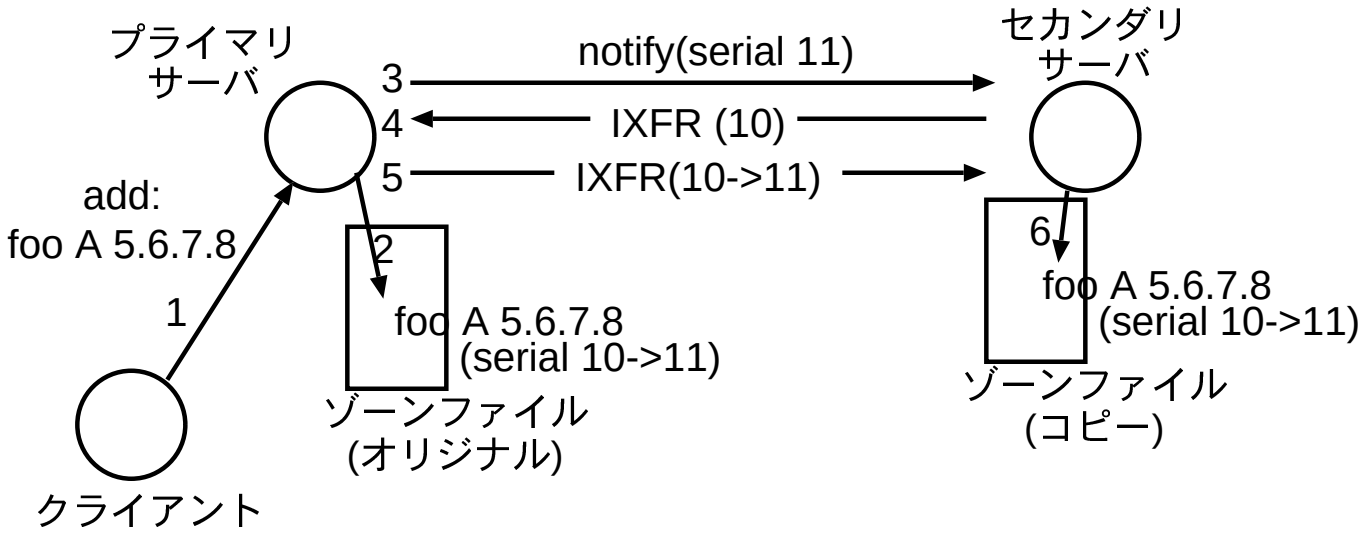
- やり直したくなったら
  - 1. rndc stop (元のゾーンファイルを更新して終了)
  - 2. ジャーナルファイルを削除
  - 3. 再起動

差分ゾーン転送

□頻繁に更新されるゾーン全体をその都度転送するのは大変
- ○サーバの負荷、ネットワーク帯域

□差分ゾーン転送: Incremental Zone Transfer (IXFR)
- ○RFC 1995
- ○プライマリは更新履歴を保存
- ○セカンダリ現在持っているデータのシリアル番号をプライマリに通知
- ○プライマリ更新履歴をもとに最新版との差分のみを転送

差分ゾーン転送の動作イメージ



BIND 9による差分ゾーン転送の運用

- 特別な設定は不要
  - 差分転送はデフォルトで有効
- サーバごとの差分転送の制御
  - provide-ixfr: プライマリ用
  - request-ixfr: セカンダリ用

IPv6アドレス

- 128ビット(16バイト)の整数
- 16ビットずつ8つのフィールドに分け、":"で区切って16進表記

  3ffe:0501:0019:2000:0000:0000:fe71:81fc
- 省略記法
  - 各フィールド内で、頭の0は省略可能

    3ffe:501:19:2000:0:0:fe71:81fc
  - 0だけから成る連続したフィールドは"::"に圧縮可能

    3ffe:0501:19:2000::fe71:81fc
  - 極端な例:

    :: (0.0.0.0に相当)

    ::1 (127.0.0.1に相当)
- プレフィクス: アドレス/プレフィクス長
  - 例: 2001:200::/32

IPv6アドレスのためのRR

□RFC 1886

□正引き: AAAA RR

www.kame.net. IN AAAA =>
3ffe:501:4819:2000:280:adff:fe71:81fc
www.kame.net. IN A 203.178.141.220

□逆引き: PTR RR

○上位ドメイン: ip6.int.

○4ビットずつを1ラベルとし、下位バイトから逆順に32階層で表現

;;2001:0200:0000:4819:0280:adff:fe71:81fcの逆引き
$ORIGIN 9.1.8.4.0.0.0.0.0.0.2.0.1.0.0.2.ip6.int.
c.f.1.8.1.7.e.f.f.f.d.a.0.8.2.0 IN PTR www.kame.net.

DNSのIPv6トランスポート

- DNSの「IPv6対応」とは
  - RRでの対応: AAAA, ip6.int.ドメイン
  - トランスポートでの対応: 問い合わせ、応答をIPv6パケットで行う

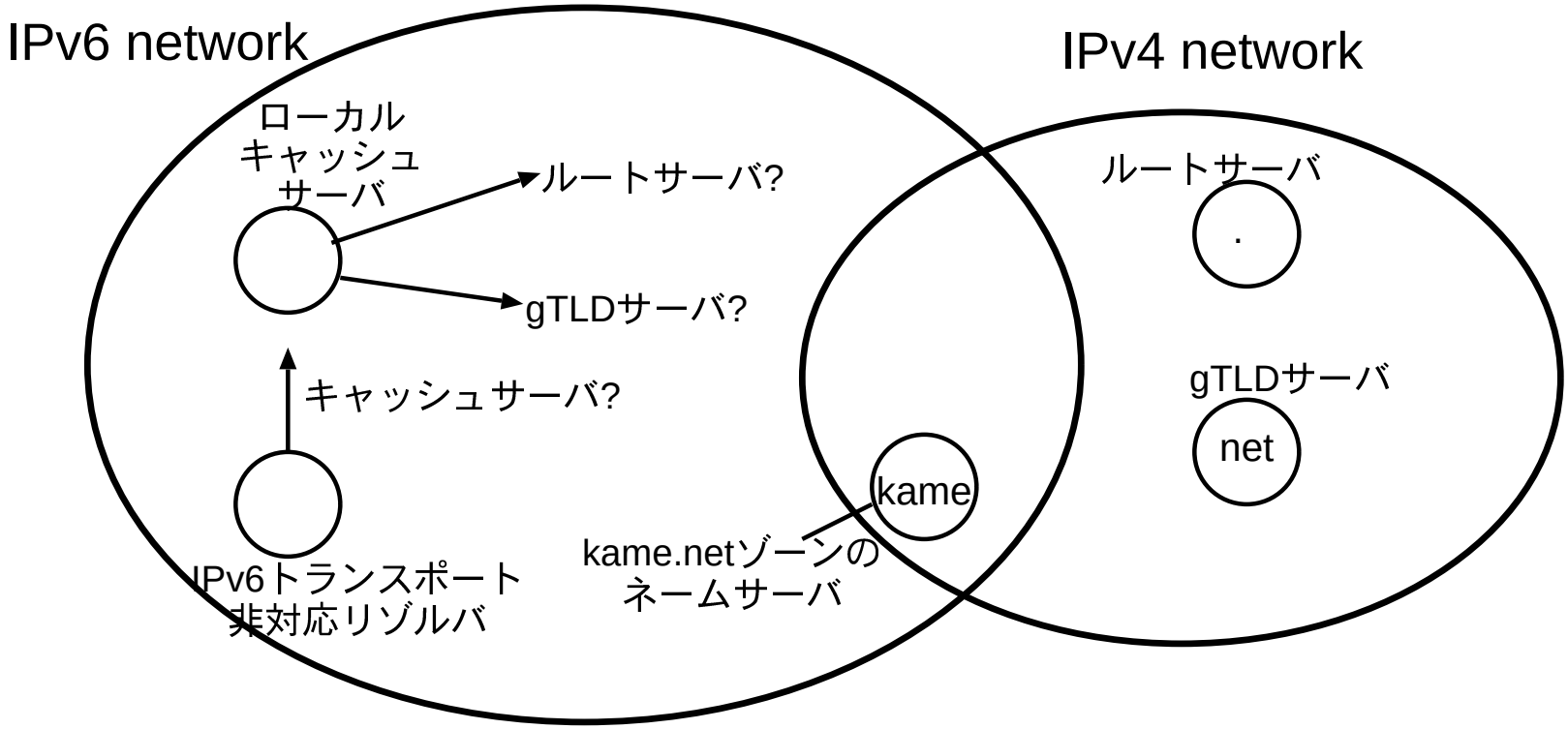
- 3つのケース
  - 1. IPv6用のRR, IPv4トランスポート
  - 2. IPv6用のRR, IPv6トランスポート
  - 3. IPv4用のRR, IPv6トランスポート
  - (IPv4用のRR, IPv4トランスポート => IPv6とは無関係)
    - 現在稼働している実装の多くは1のみ
    - 1.のみで実運用上は問題ないが、IPv4に依存しない環境構築には2.が重要
    - 2.をできる実装はふつう3.もできる

IPv6トランスポートの現状

- TLDの現状: IPv6アドレスを持つネームサーバの数
  - ルート, gTLD: 13ネームサーバ中ゼロ
    - 改善の試みが進展中
  - ccTLD: 243ゾーン、584ネームサーバ中
    - 2ゾーン、3サーバ
    - ルートゾーンからのグルーRRなし
- 実装状況
  - サーバ
    - BIND 9
  - リゾルバ
    - FreeBSD/NetBSD/OpenBSD/Linux
    - BIND9 lwres: アプリケーションの再コンパイルが必要
    - BIND8, 9 libbind: OS標準ライブラリへの統合はこれから
    - BSD/OS, Solaris, Windows...は未対応

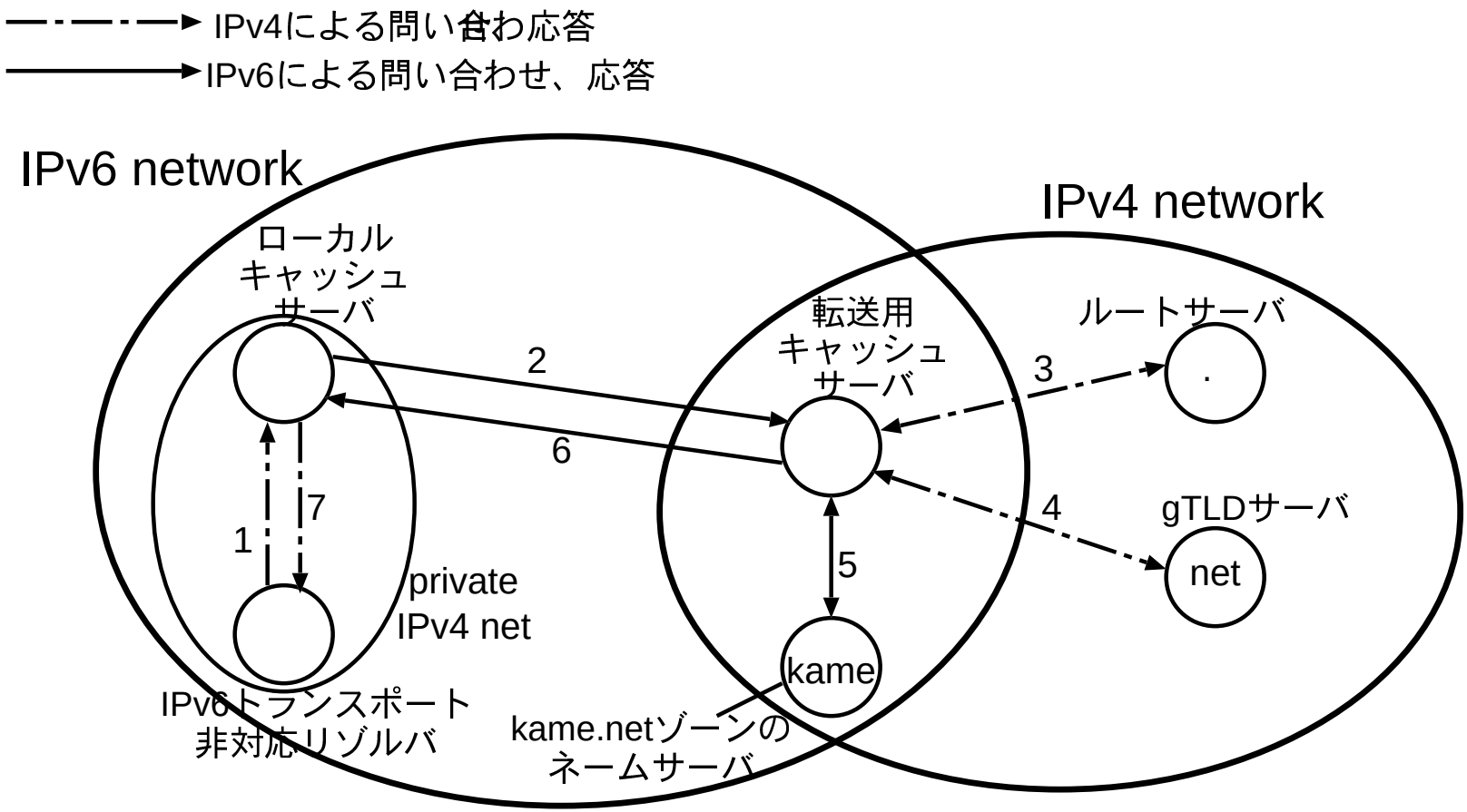
現状の問題点

- IPv6トランスポートに対応しないスタブリゾルバはキャッシュサーバへの問い合わせができない
- IPv6のみのネットワークにいるキャッシュサーバは名前解決できない



運用での対応

○名前解決用のIPv4ローカルネットワーク
○デュアルスタックの転送用キャッシュサーバ



BIND9のIPv6トランスポート(1/2)

- IPv4と同等
  - 問い合わせ、応答、ゾーン転送、ACL、制御コマンド
- IPv6トランスポートの有効化

  listen-on-v6 { any; };
  - anyまたはnoneしか設定できない
  - デフォルトがnoneであることに注意
  - wildcard bindされたソケットですべてのUDP応答を処理する
    - 2つのnamedを同時に起動するのは不可
- ソースアドレスの指定: xxx-source-v6
  - 例: 問い合わせのソースアドレスを指定する

    query-source-v6 address 2001::abcd;
- アクセス制御
  - 例: ゾーン転送元を制限

    allow-transfer { 3ffe:501::/32; };

BIND9のIPv6トランスポート(2/2)

- IPv4-mapped IPv6アドレスに注意
  - ::ffff:x.y.z.w (x.y.z.wはIPv4アドレス)の形式
  - IPv6ソケットでIPv4パケットを受信する実装がある
    - FreeBSD, Linux, Compaq tru 64など
    - IPv4アドレスを"mapped" IPv6で表現する
  - アクセス制御が複雑になる
    - 例: IPv4によるアクセスを禁止するための設定

```
listen-on { none; };
allow-query { !::ffff:0.0.0.0/96; any; };
allow-transfer { !::ffff:0.0.0.0/96; any; };
...
```

    - match-mapped-addressesオプション

```
match-mapped-addresses yes;
allow-query { !10.0.0.1; any; };
   means...
allow-query { !10.0.0.1; !::ffff:0.0.0.0/96; any; };
```

BIND9+IPv6での推奨される運用方法

- □IPv4についてアドレスでのアクセス制御をしない場合
  - ○とくに気にしなくてよい
- □IPv4についてアドレスでのアクセス制御をする場合(現在)
  - ○mapped アドレスをサポートしないOSを使う
    - ▹NetBSD(のデフォルト), OpenBSD
- □IPv4についてアドレスでのアクセス制御をする場合(近い将来)
  - ○IPV6_V6ONLY optionをサポートするOS, BIND 9を使う
    - ▹KAME snap, FreeBSD 4.5(?)
  - ○BIND 9.2.1(?), 9.3(?)

データ長の制限と**EDNS0**

- DNSデータ長(UDP)の上限=512バイト
  - 大きなデータが扱えない
  - IPv6, DNSSECへの対応には不十分
  - 例: ルートサーバにwww.kame.net.のA RRを問い合わせる
    - 応答 = gTLDサーバに対するNS RRとグルーRR
    - 現状: 13サーバ → 459バイト
    - AAAA グルーRRも付けるなら5サーバが限界 → 471バイト
    - 13サーバのままだと → 823バイト必要
- EDNS0: Extension Mechanisms for DNS
  - RFC 2671
  - OPT RR: 問い合わせ時に、応答用の受信バッファ長を通知
    - 応答には、通知された長さまで詰めてよい
- EDNS0の実装
  - BIND 9(サーバ、libbind)
  - BIND 8は、EDNS0付きの問い合わせにはエラーを返す

BIND 9とEDNS0

□新規のサーバへの問い合わせには常にOPT RRを付ける

□エラーが返ってきた場合
- ○OPT RRなしでやり直す
- ○エラーを返したサーバは記憶する
- ○次回からはOPT RRなしで問い合わせる
- ○24時間でリフレッシュ

□オプションでの制御

```
server 203.178.141.194 {
    edns no; //このサーバにはENDS0を使わずに問い合わせる
};
```

- ○通常はこの設定は不要

IPv6用の新しいRR

- 1つのアドレスを複数のRRに分割して登録する
  - マルチホーム、リナンバリング(アドレスつけかえ)対応
- 正引き
  - A6 RR (RFC 2874)
- 逆引き
  - ビットラベル (RFC 2673)
  - DNAME RR (RFC 2672)
  - 上位ドメイン: ip6.arpa.
    - 政治的な理由により導入
- 実際には採用しない方向
  - 複雑すぎ、導入コストに比べて効果が見えない
  - A6、ビットラベル、DNAMEは"experimental"に
  - ip6.arpa.は導入の方向 (RFC 3152)

BIND 9と新RR

- A6、ビットラベル、DNAMEを実装済み

- 運用上の注意
  - 通常は気にしなくてよい
    - A6の問い合わせを勝手に出す場合があるが、無害
  - allow-v6-synthesis
    - A6をサポートしないリゾルバ用
    - 使用しない方が安全
  - どうしても試してみたい場合は
    - プライマリ・セカンダリともにすべてBIND 9にする
    - A6とAAAAの両方を登録する
  - 一部管理ツールのデフォルト値に注意(後述)

ip6.arpa.への移行(1/2)

- リゾルバ側: ip6.arpa.とip6.int.の両方を試す
  - 実装待ち
- サーバ側: ip6.arpa.とip6.int.の両方を管理
- プライマリサーバの設定例 (BIND 8, 9)
  - 単一のゾーンファイルを二つのゾーンで共有
  - $ORIGINによる上位ドメイン変更に注意

(in named.conf)

```
zone "9.1.8.4.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.int." {
    type master;
    file "2001:200:0:4819::.zone";
};
zone "9.1.8.4.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa." {
    type master;
    file "2001:200:0:4819::.zone"; //ip6.int.と共有
};
```

(in 2001:200:0:4819::.zone)

```
;;$ORIGIN 9.1.8.4.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.int. →うまくいかない
c.f.1.8.1.7.e.f.f.f.d.a.0.8.2.0 IN PTR www.kame.net.
```

ip6.arpa.への移行(2/2)

□セカンダリサーバの設定例 (BIND 8, 9)

```
zone "9.1.8.4.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.int." {
    type slave;
    file "bak/2001:200:0:4819::.int.zone";
    masters ...
};
zone "9.1.8.4.0.0.0.0.0.0.0.0.2.0.1.0.0.2.ip6.arpa." {
    type slave;
    file "bak/2001:200:0:4819::.arpa.zone"; //共有しない
    masters ...
};
```

DNSのセキュリティ

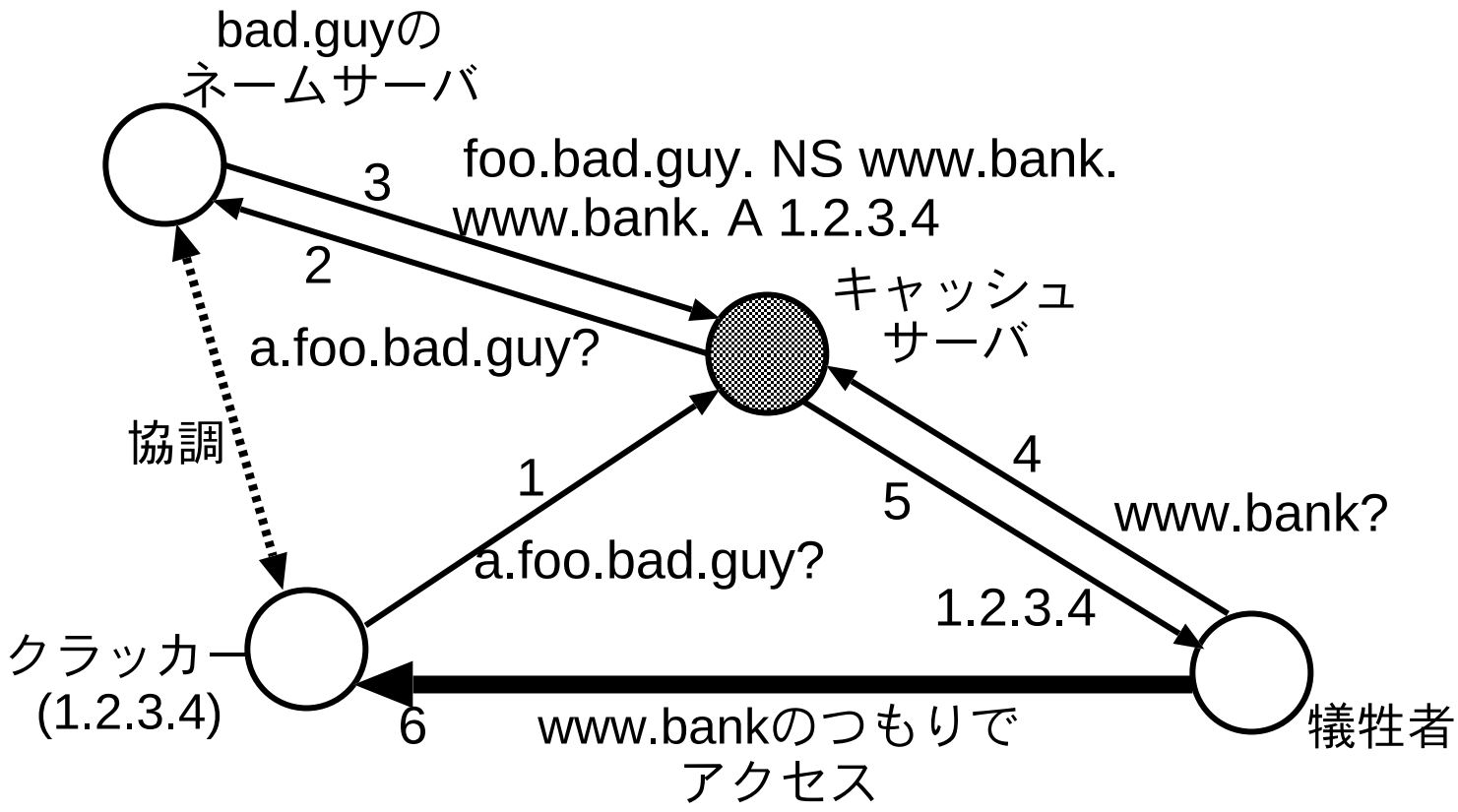
- **DNS**におけるセキュリティ
- トランザクションのセキュリティ **(TSIG, SIG(0))**
- **DNSSEC**
  - 動作原理
  - リソースレコード
  - **BIND 9**での運用
  - **DNSSEC**は必要か？

DNSにおけるセキュリティ

- □サーバ・リゾルバ間、サーバ・サーバ間の通信(transaction)を認証する
  - ○動的更新、ゾーン転送
- □データベース検索結果の正当性
  - ○偽者サーバへの誘導を防ぎたい
    - ▹キャッシュ汚染
    - ▹乗っ取られたネームサーバからの応答
    - ▹応答メッセージの改竄、なりすまし
- □範囲外の話題
  - ○秘匿性
  - ○大量の問い合わせによるサービス妨害攻撃(DoS)
  - ○DNSの実装そのものに対するセキュリティホール

キャッシュ汚染

□古いBINDの実装に対しては簡単に実現できる



TSIG: transaction signatures

- RFC 2845
- サーバ同士、またはサーバ・リゾルバ間の通信を認証
  - 用途: 動的更新、ゾーン転送など
- 共通秘密鍵でDNSメッセージ全体に署名
- TSIG RR
  - メタRR: ゾーンファイルには現れない
    - メッセージの最後に自動的に追加される
  - DNSメッセージ全体の一方向ハッシュ値を格納
  - 繰り返し攻撃防止のために時刻同期が必要
- BIND 9の実装
  - ハッシュ関数: HMAC-MD5
  - 鍵の生成: dnssec-keygenコマンド
  - 動的更新での利用(動的更新の項を参照)

ゾーン転送での利用例

□設定上の注意

○鍵の名前はプライマリとセカンダリで揃えること

□プライマリ側の設定

```
key kame-transfer {
        algorithm hmac-md5;
        secret "VbWW3pKDonY8axAqpC/ArA==";
};
zone "kame.net" {
        type master;
        file "kame.zone";
        allow-transfer { key kame-transfer; };
};
```

□セカンダリ側の設定

○keyステートメントはプライマリと同じものを使う

```
zone "kame.net" {
        type slave;
        file "bak/kame.zone";
        masters { 10.0.0.1 key kame-transfer; };
};
```

SIG(0): Request and Transaction Signatures

- RFC 2931

- 動機・用途はTSIGと同じ
  - SIG RRを使った公開鍵方式

- SIG(0)
  - フォーマット・タイプなどはSIG RRと同じ
  - メタRR: ゾーンファイルには現れない
  - 時刻同期が必要

- BIND9の実装
  - 不完全: 受けて検証はできるが、送信する手段がない

DNSSEC (1/2)

- RFC 2535

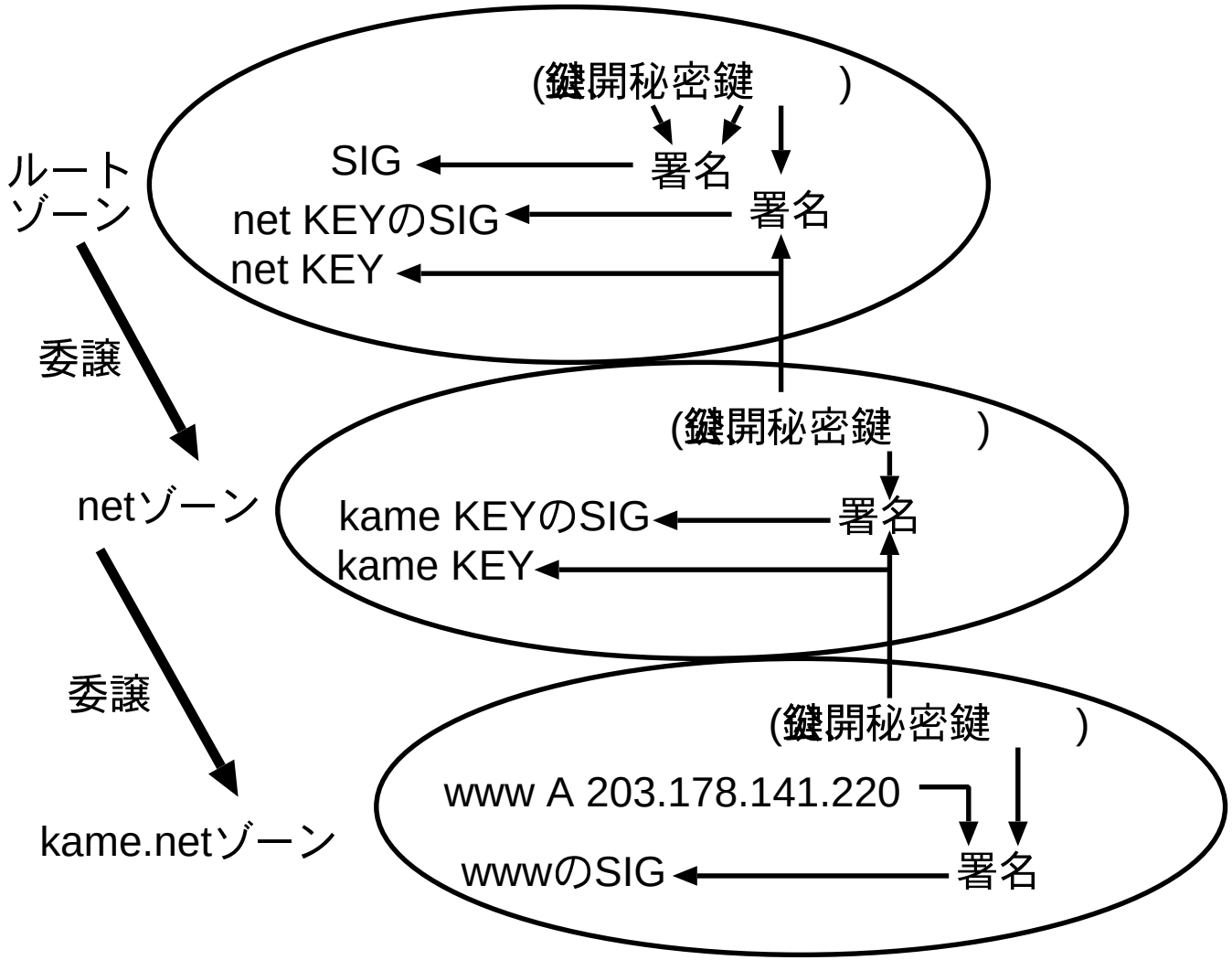
- 目的: DNSのデータベース全体の正当性を保証する
  - 検索時のなりすまし、キャッシュ汚染からの防御
  - "NXDOMAIN(名前が存在しない)"の正当性も保証
    - NXDOMAINによるDoSからの防御

- 方法: 公開鍵方式による署名
  - 鍵はゾーンごとに管理
  - 公開鍵をKEY RRとして作る
  - ゾーン内の各RRを秘密鍵で署名(SIG RR)

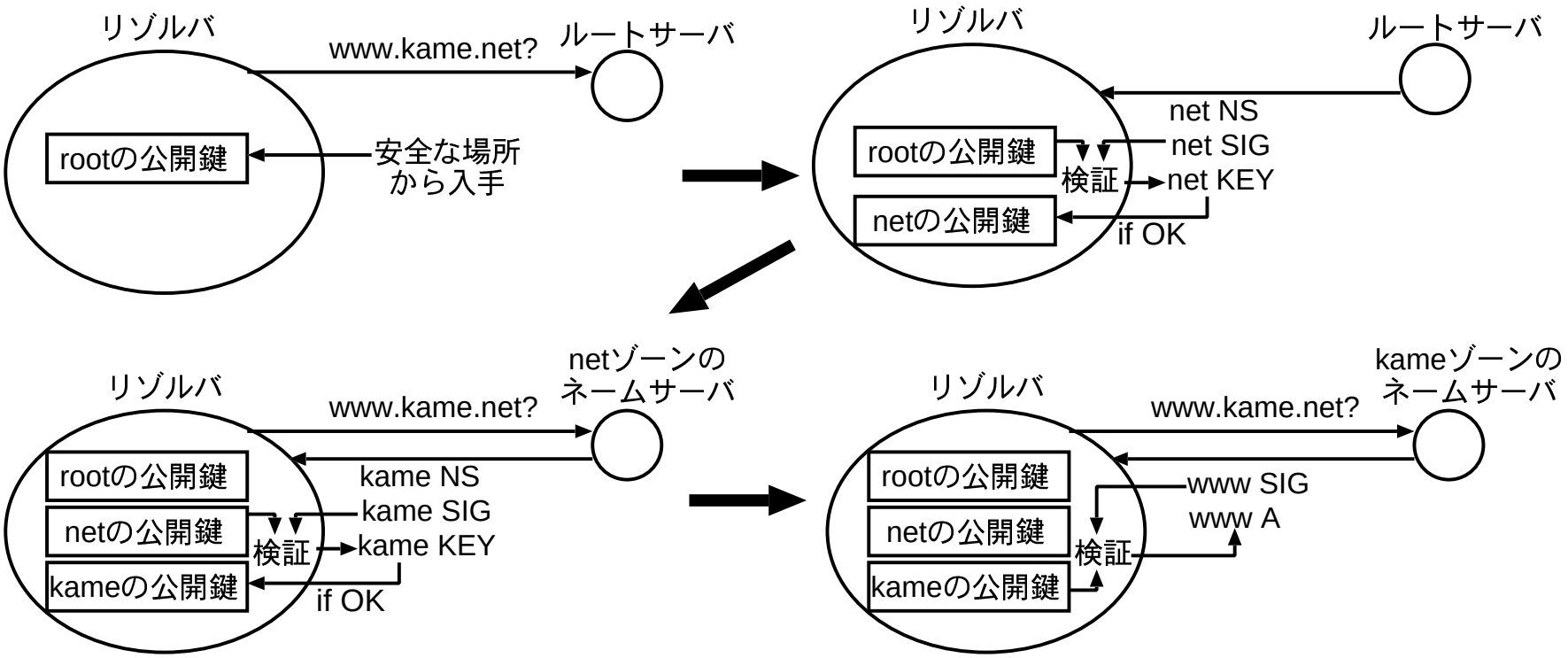
DNSSEC (2/2)

- 署名の検証
  - 応答のRRに対応するSIG RRを入手
  - そのゾーンのKEY RRによって検証
  - KEY RR自体の正当性は上位ゾーンの署名によるSIG RRで保証
    - 最終的にはルートゾーンによる署名に帰着する
    - ルートゾーンの公開鍵は既知であると仮定

DNSSECの署名


(鍵開秘密鍵　　)
SIG
署名
net KEYのSIG
署名
net KEY
ルート
ゾーン
委譲
(鍵開秘密鍵　　)
netゾーン
kame KEYのSIG
署名
kame KEY
委譲
(鍵開秘密鍵　　)
www A 203.178.141.220
wwwのSIG
署名
kame.netゾーン

DNSSECの署名検証



KEY RR

□KEY RRのデータ: フラグ、プロトコル、アルゴリズム、公開鍵

- ○フラグ(2バイト): DNSSECではふつう256 = 認証と秘匿
- ○プロトコル(1バイト), 3=DNSSEC, 4=IPSEC, etc.
- ○アルゴリズム(1バイト): 1=RSA/MD5, 3=DSA, etc.
- ○公開鍵: base 64でエンコーディング

```
kame.net. 1D IN KEY 256 3 1 (
        AQOX3l7BJVz1AnDGuzRokxKPD7Jceae
        7Y9MCgnK2KCfZcu6WpkZx0ThkAJHfFf
        kIlCA2F6mveu3Hy3wFudreND8N)
```

SIG RR

□SIG RRのデータ(抜粋)

- ○type covered: 署名対象のRRタイプ
- ○アルゴリズム
- ○オリジナルTTL: ゾーンファイルに登録されたRRのTTL
- ○署名の有効期間(はじまりと終り)
- ○鍵のタグ: 複数の鍵を同時に使用している場合の識別子
- ○署名を付けたゾーン名
  - ▹ふつうはRRの属するゾーン
  - ▹KEY RRに対するSIG RRの場合は上位ゾーン
- ○署名: base 64でエンコーディング

```
foo.kame.net. 1D IN SIG A 1 3 3600 20011028062225 (
                20010929062225 13375 kame.net.
                gYHHmpMfRkspRKK2lgz5F02lAn82+AKwe546
                aSQQN0tpkWHH+lszgFOcWdtMkcaGYtM694y7
                i5uRRaimC5jGUA== )
```

NXT RR

- □"NXDOMAIN"の正当性を保証するための仕掛け
  - ○単純に"NX"を署名したのでは再利用される
  - ○後前の名前を示して、その中間がないことを保証すればよい
    - ▹RRの正規化と順序付けが必要
- □NXT RRのデータ: 「次の名前」と「そのRRのタイプのリスト」
- □応答データの構成
  - ○コード"NXDOMAIN"と一緒にNXT RRが返る
    - ▹NXT RRのkey: 「一つ前」のRR
    - ▹next name: 「一つ後」のRR
  - ○NXT RRは、その中間には何もデータがないことを主張する
  - ○NXT RR自体の正当性はSIG RRで保証する
- □欠点: あるゾーン内の全RRが芋蔓式に漏れる

署名の手順

- □初回の手順
  - ○1. ゾーン用の鍵を生成する
  - ○2. 公開鍵(KEY RR)を上位ゾーンに署名してもらう
    - ▹KEY RRに対するSIG RRを得る
  - ○3. 秘密鍵でゾーンの各RRを署名する
    - ▹それぞれ対応するSIG RRが生成される
    - ▹各RRに対応するNXT RRと、それに対するSIG RRも生成される
- □更新手順
  - ○4. 新しい鍵を生成する
  - ○5-1. 新旧両方のKEY RRを上位ゾーンに署名してもらう
  - ○5-2. 新しいKEY RRだけを上位ゾーンに署名してもらう
  - ○6. 新しい秘密鍵でゾーンの各RRを署名する。古いKEY RRは残す
  - ○(古い鍵で署名したSIG RRの有効期限が切れる)
  - ○7. 古いKEY RRを削除する

DNSSEC関係の用語

- □ゾーンの状態
  - ○Secure: 信用できるKEY RRを持つゾーン
  - ○Unsecured: "Secure"でないゾーン
- □検索したRRの分類
  - ○Authenticated: 署名の検証に成功したRR
  - ○Pending: 署名検証中のRR
  - ○Insecure: "Unsecured"なゾーンのRR
  - ○Bad: 検証に失敗したRR
    - ▹"Bad"のRRをリゾルバに返してはいけない
- □DNSSECに関係するフラグビット
  - ○AD (authenticated data) ビット
    - ▹Authenticated または InsecureなRRにだけ立てる
  - ○CD (checking disabled) ビット
    - ▹「自力で検証しますのでそのまま返してください」
    - ▹Pending data返ってくる

拡張仕様

- 全体像: draft-ietf-dnsext-dnssec-roadmap-05.txt

- RFC2538 (CERT RR)
- RFC3008 (DNSSEC signing authority)
- RFC2930 (TKEY)
  - TSIG用の鍵交換プロトコル
- draft-ietf-dnsext-dnssec-okbit-03.txt
  - EDNS0の追加トで　　DNSSECするresolverを識別する
- NO RR: draft-ietf-dnsext-not-existing-rr-02.txt
  - NXTの改良。ハッシュを取ってから比較するので漏洩がない。
- draft-ietf-dnsext-ad-is-secure-03.txt
  - AD bitはauthorized dataにだけ立てる
- その他いろいろ

BIND 9によるDNSSECの運用

□鍵を作る: dnssec-keygen

dnssec-keygen -a rsa -b 512 -n zone kame.net.

Kkame.net.+001+52522

○Kkame.net.+001+52522.{key, private}ができる

□ゾーンファイルに鍵を取り込む

$INCLUDE Kkame.net.+001+52522.key

□ゾーンに署名する: dnssec-signzone

dnssec-signzone -o kame.net. kame.zone

kame.zone.signed

○NXT RRは自動生成される

ゾーンファイルの署名 **(1/2)**

□元のゾーンファイル (kame.netゾーン)

```
$TTL 1D
@  IN SOA orange.kame.net. root.kame.net. (1 3600 600
                                2419200 1200)
        IN      NS      ns
ns      IN       A      10.0.0.1
foo     IN       A      10.0.0.2
        IN      AAAA    2001:ffff::2
```

ゾーンファイルの署名 **(2/2)**

□署名後のゾーンファイル

```
kame.net.     86400   IN SOA  orange.kame.net. root.kame.net. (
          ...
              86400   KEY     256 3 1 (
                        AQO/ruxOPls/OnqlzMiuD5fSQ2DqEQsRMtMw
                        +S8Dp2ibh1B3DHgfBV6hCxRoVsJIqciApuht
                        ca9pFzBnTFxLzSr9 ) ; key id = 52522
foo.kame.net. 86400   IN A    10.0.0.2
              86400   SIG     A 1 3 86400 20011117072241 (
                        20011018072241 52522 kame.net.
          ...
              86400   AAAA    2001:ffff::2
              86400   SIG     AAAA 1 3 86400 20011117072241 (
                        20011018072241 52522 kame.net.
          ...
                        aY35FZGAh7OqHA== )
              86400   NXT     ns.kame.net. A SIG AAAA NXT
              86400   SIG     NXT 1 3 86400 20011117072241 (
          ...
ns.kame.net.  86400   IN A    10.0.0.1
```

署名されたゾーンに対する問い合わせの例 **(1/2)**

- BIND9 digの+dnssecオプション
  - 存在する名前に対する問い合わせ

```
% dig @127.0.0.1 foo.kame.net a +dnssec
;;; ANSWER SECTION:
foo.kame.net.  86400  IN  A    10.0.0.2
foo.kame.net.  86400  IN  SIG  A 1 3 86400 20011117072241 20011018072241 =>
  52522 kame.net. ZcEJRbDKj9AvTtqb2/OLWRFoIHTecATC0uVaoNm9sxg4rp6 =>
                FRsDVNr1s qG6ue5A3fJbZqYDGY0Wy3Dcqd9T3HA==
...
```

署名されたゾーンに対する問い合わせの例 **(2/2)**

○存在しない名前に対する問い合わせ

```
% dig @127.0.0.1 hoo.kame.net a +dnssec

;;
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 58381
;; AUTHORITY SECTION:
;;
foo.kame.net.  86400 IN  NXT  ns.kame.net. A SIG AAAA NXT
foo.kame.net.  86400 IN  SIG  NXT 1 3 86400 20011117072241 20011018072241 =>
  52522 kame.net. Om4du45EnQEuAf1fu2qnWvKujNdARUJk9mBaxLqDdECg2Uv =>
                3ZKTzA5pc aSgfakIShbzJFoJXoM8zGNrqW19p5g==
```

上位ゾーンによる鍵の署名 **(1/2)**

- 上位ゾーンに送る鍵セットを作る:

  dnssec-makekeyset

  - できファイルを上位ゾーンの管理者に送る

```
# dnssec-makekeyset -t 172800 Kkame.net.+001+52522.key
keyset-kame.net.

$ORIGIN .
$TTL 172800 ; 2 days
kame.net  IN KEY 256 3 1 (
                AQO/ruxOPls/OnqlzMiuD5fSQ2DqEQsRMtMw+S8Dp2ib
                h1B3DHgfBV6hCxRoVsJIqciApuhtca9pFzBnTFxLzSr9
                ) ; key id = 52522
        SIG   KEY 1 2 172800 20011117075356 (
                20011018075356 52522 kame.net.
                Rlns5wL6HA3P+1D/XTzeWM6rxeelLkg2VOWDFxedspc3
                sF+pMp2R4VJLvdjIT2IFQvYud5tSHsnWDw+xcis6Cg== )
```

上位ゾーンによる鍵の署名 **(2/2)**

- 下位ゾーンの鍵に署名する: dnssec-signkey
  - できファイルを上位のゾーンファイルに取り込む

```
# dnssec-signkey keyset-kame.net. Knet.+001+43574.private
signedkey-kame.net.

$ORIGIN .
$TTL 172800     ; 2 days
kame.net IN KEY  256 3 1 (
                AQO/ruxOPls/OnqlzMiuD5fSQ2DqEQsRMtMw+S8Dp2ib
                h1B3DHgfBV6hCxRoVsJIqciApuhtca9pFzBnTFxLzSr9
                ) ; key id = 52522
        SIG     KEY 1 2 172800 20011117075356 (
                20011018075356 59646 net.
                aUkkSJlsZ1QG6g5KRlALZKVykPnrIywp+QInbB7/KhpS
                5AAUYb319DB3cf4PHrktQWIh6qrKLmoNxfC+FUWk6A== )
```

Security Root

- ルートゾーン以下が全部Secureであるという仮定は厳しい
  - 現にルートゾーンはSecureでない(KEY RRを持っていない)

- 既知の公開鍵に対応するゾーンを信用するところからはじめる

- BIND9 trusted-keys

```
trusted-keys {
        kame.net. 256 3 1 "AQPEZ6sMe9KwMOBAQ =>
        SQ3qHjeW51Zr7WGljSP =>
        we3BVMISL2p45BbfpoZ =>
        R 1HOJsh5fXCnYWioRs =>
        IdQxL/hq/iAd901";
};
```

DNSSECは使える道具か？

- □いまひとつ流行ってないのが現状
  - ○普及が難しい
    - ▹とくにリゾルバ側 - 簡単には入れ替えられない
    - ▹キャッシュサーバに頼る場合、その負荷が未知数
    - ▹パケット長の問題: 受信バッファサイズの制限、path MTU
  - ○脅威が薄れている
    - ▹最近のネームサーバではキャッシュ汚染を起こしにくい （「外部」のグルーRRはキャッシュされない)
    - ▹応答のなりすましは現実的には簡単でない
  - ○所詮完全な解ではない
    - ▹アドレスが正しくても、そのノードが信用できるかどうかは別問題
    - ▹結局、アプリケーションレベルでの認証が必要
  - ○検証結果の利用方法が難しい
    - ▹UnsecuredなゾーンのRRをどう扱うべきか？
- □保険としての使い道
  - ○「自分の組織には問題がない」ことを主張する手段として

BIND 9に関するその他の話題

- 管理ツール
- **BIND 9**の性能
- **view**
- **lwres**
- **dig**

管理ツール (1/2)

- rndc
  - BIND 8のndcに相当
  - TCPでnamedと通信する
    - 遠隔ホストからの操作も可能
    - シグナルによる制御は廃止
  - 認証
    - アドレスによる認証と共通秘密鍵によるメッセージ署名
  - rndcの便利コマンド
    - reload: namedの再起動
    - dumpdb: キャッシュデータをファイルへダンプする
    - querylog: log出力の切り替え
    - trace [level]: logレベルの指定
    - flush: キャッシュデータを消去
  - 設定ファイル: rndc.conf
    - 自動生成コマンド(rndc-confgen)が便利
    - rndc-confgenの出力をrndc.confとして利用
    - コメント部分をnamed.confにコピー

管理ツール (2/2)

- rndc設定上の注意
  - BIND 9.1と9.2ではプロトコルが異なる
    - 9.1のrndcと9.2のnamedという組合わせは動作しない
  - 認証用のアドレスにはホスト名ではなく、アドレスを指定
    - ホスト名を指定した場合、複数アドレスに対応できない
    - IPv4とIPv6のデュアルスタックの場合に困る
    - 悪い例:

```
options {
        default-key "rndc-key";
        default-server localhost; // "localhost"でなく"127.0.0.1"や"::1"を使うこと
};
```

- 設定ファイルのチェックツール
  - 再起動前の確認に便利
  - named-checkconf: named.confのチェック
  - named-checkzone: ゾーンファイルのチェック
    - ゾーンファイルのあるディレクトリで実行すること

BIND 9の性能 (1/2)

- BIND 8の半分程度
  - ただし、多くの環境では実運用上の問題はない
  - 毎秒数千の問い合わせを受けるクラスのサーバでは少し厳しい
- threadの影響
  - DNSSECを使わない限り、性能面ではむしろマイナス
  - 通常は無効化しておくべき

BIND 9の性能 (2/2)

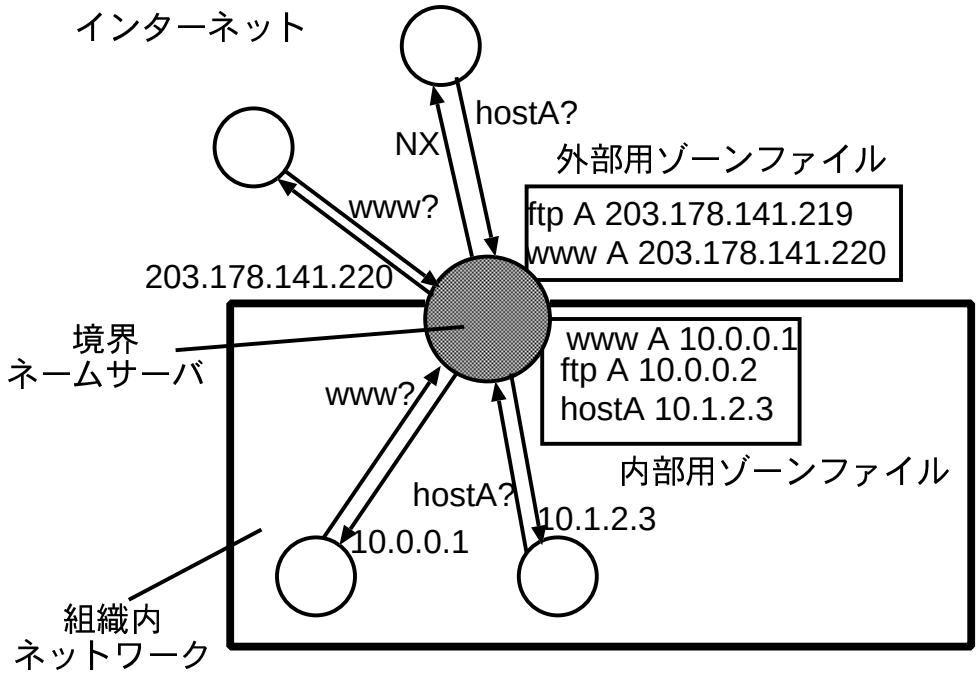
- ルートゾーンファイルでのベンチマーク
  - 4種類のTLDに対して、それぞれ同じ問い合わせを連続して送る
    - 同時に発する問い合わせ: 最大20
  - FreeBSD 4.4, Pentium III 866MHz
  - BIND 9はthreadなしで構築

- ベンチマーク結果
  - 単位(qps)
  - グルーRRの数が多いほど遅くなる
    - データベースの構造上の問題

	com	yu	gu	hoge
bind 8.2.5	3024	4322	5790	6482
bind 9.1.3	1328	1870	2762	3927
bind 9.2.0rc6	1504	2177	3275	4698

ビュー(View)

□問い合わせ元のアドレスに応じて応答を制御
- ○ゾーンファイル、転送の有無、転送先などをviewごとに定義できる
- ○防火壁の境界にいるサーバなどで利用する



Viewの設定例

□設定上の注意

○viewの順序は重要: 最初にマッチしたアドレスが適用される

▹"any"は最後に置かないといけない

○viewを一つでも定義したら、view外でのゾーン定義は不可

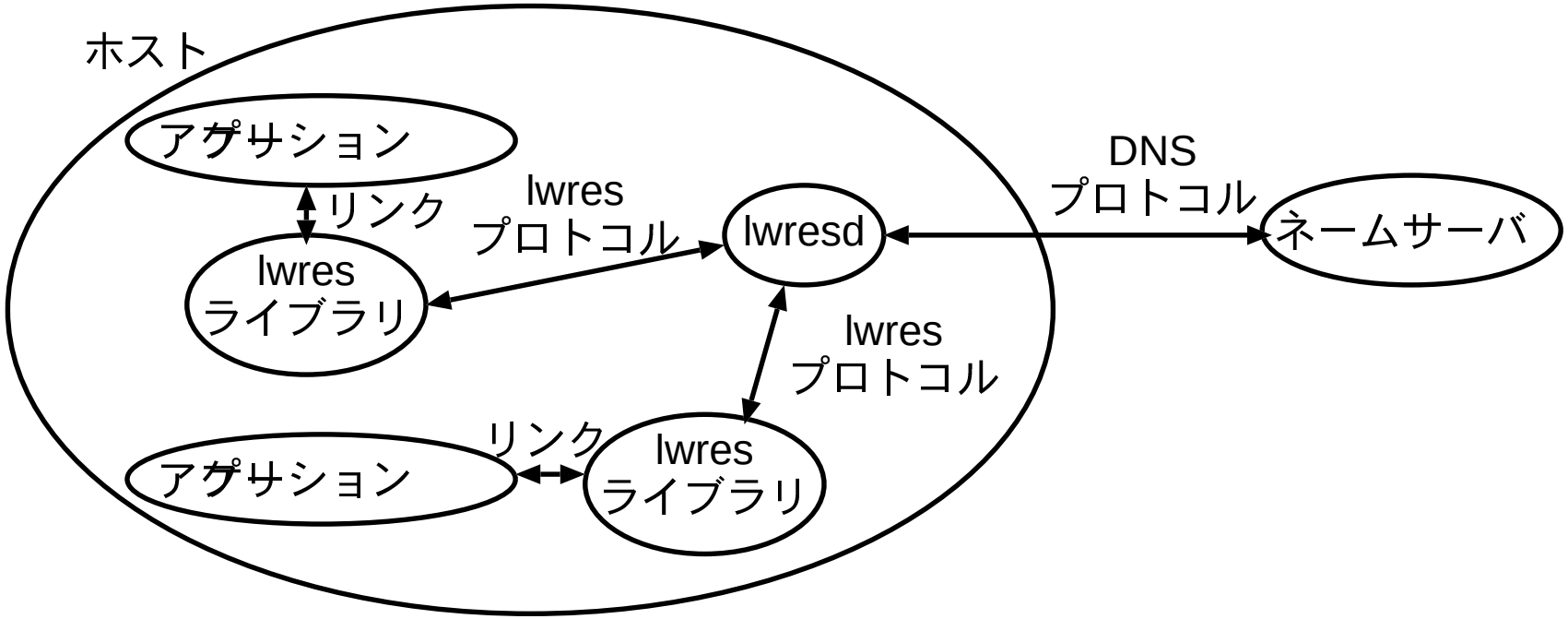
```
view "internal" {
        match-clients { 133.196.0.0/16; };
        zone "toshiba.co.jp" {
                type master;
                file "toshiba-internal.zone";
        };
};
view "external" {
        match-clients { any; };
        zone "toshiba.co.jp" {
                type master;
                file "toshiba-external.zone";
        };
};
```

lwres: Light Weight Resolver

- DNSの新機能を基本ライブラリとして実装することの限界
  - DNSSEC, A6

- lwresライブラリとlwresデーモン(lwresd)
  - lwresライブラリ
    - アプリケーションへのインタフェース
    - インタフェースは既存のDNS用ライブラリ関数に合わせる
    - ヘッダファイルだけ変えればソースレベルで互換
  - lwresデーモン:
    - DNSによる名前解決を担当
    - 実体はnamed: デーモンは"light weight"ではない
    - DNSの機能拡張をデーモンで吸収ライブラリを軽くする
  - ライブラリとデーモンは独自のプロトコル(UDP)で通信

lwres環境のアーキテクチャ



lwresの利用法

- lwresdの起動
  - 特別な設定は不要
  - /etc/resolv.confを見る
    - サーバの指定があればそのサーバに問い合わせ
    - サーバの指定がなければ自力でルートサーバに問い合わせる
    - IPv6トランスポートもサポート
- アプリケーション側
  - lwresライブラリのヘッダファイルをincludeする
    - コードは変える必要なし
  - liblwresをリンクする

```
#include <lwres/netdb.h>
main(int argc, char *argv[])
{
        struct hostent *ent;
        if ((ent = gethostbyname(argv[1])) == NULL) {
                herror("gethostbyname failed");
                exit(1);
        }
}
```

dig

- Domain Information Groper
  - ネームサーバへの問い合わせツール
  - nslookupからdigへの移行を推奨
  - ネームサーバからの応答を加工せずに表示する
- digによる問い合わせの基本型
  - dig [@server_address] [RR_type] domain_name
    - オプションの順序はかなり柔軟
  - server_address
    - 省略すると/etc/resolv.confを見る
    - ホスト名でなくアドレスを指定すること
  - RR_TYPE
    - デフォルトはA RR
    - IPv6アドレスを検索するなら"aaaa"を指定する
- 出力のどこを見るべきか
  - "ANSWER SECTION"
  - HEADERの"status"と"flag"

digの出力例

```
; <<>> DiG 9.2.0rc1 <<>> @203.178.141.194 kame202.kame.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55547
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 11

;; QUESTION SECTION:
;kame202.kame.net.              IN      A

;; ANSWER SECTION:
kame202.kame.net.       86400   IN      A       203.178.141.202
;; AUTHORITY SECTION:
kame.net.               86400   IN      NS      orange.kame.net.
kame.net.               86400   IN      NS      coconut.itojun.org.
kame.net.               86400   IN      NS      tiger.hiroo.oshokuji.org.

;; ADDITIONAL SECTION:
tiger.hiroo.oshokuji.org. 1180  IN      A       210.145.33.242
orange.kame.net.        86400   IN      A       203.178.141.194
orange.kame.net.        86400   IN      AAAA    3ffe:501:4819:2000:220:18ff:fe58:adca
(...省略)
```

digの便利な使い方

□-xオプション: 逆引き用のショートカット

dig -x 203.178.141.194

=> dig 194.141.178.203.in-addr.arpa. ptr

▹IPv6アドレスの場合には-nも必要

dig -n -x 2001:200:0:4819:280:adff:fe71:81fc

=> dig c.f.1.8.…1.0.0.2.ip6.int. ptr

□hostコマンド: 入出力をシンプルにしたdig

% host www.wide.ad.jp

www.wide.ad.jp has address 203.178.136.57

○-tオプション: RRを指定

% host -t aaaa www.wide.ad.jp

www.wide.ad.jp has AAAA address 2001:200:0:1001::6

○-nオプション: IPv6アドレスの逆引き用

% host -n 2001:200:0:1001::6

6.0.0.0.…1.0.0.2.ip6.int domain name pointer sh.wide.ad.jp.

付録: 典型的な**named.conf**の例

```
options {
        directory "/etc/namedb";
        max-cache-size 16M; //キャッシュサーバの場合
        listen-on-v6 { any; }; // IPv6トランスポートを使う場合
};
key "rndc-key" {
        algorithm hmac-md5;
        secret "xxxxxxxxxxxxxxxxxxxxxxxx";
};
controls {
        inet 127.0.0.1 port 953
                allow { 127.0.0.1; } keys { "rndc-key"; };
};
logging {
        channel namedlog {
                file "/var/log/named.log" versions 5 size 1M;
                severity dynamic;
                print-severity yes;
                print-time yes;
        };
        category default { default_syslog; namedlog; };
};
```

参考文献、リンクなど

- DNSのプロトコル仕様
  - http://www.ietf.org/rfc.html
  - http://www.ietf.org/ID.html
  - IETF dnsext wg
    - http://www.ietf.org/html.charters/dnsext-charter.html
    - メーリングリスト: namedroppers@ops.ietf.org

- BIND関係
  - DNS & BIND (O'Reilly)
    - 第4版が最新 (BIND 9に対応)
    - 第3版まで邦訳あり
  - BIND 9メーリングリスト
    - bind9-users, bind9-workers
    - http://www.isc.org/products/BIND/bind9.html
  - ARM (Administrator Reference Manual)
    - bind-9.x.y/doc/arm/Bv9ARM.html
  - BIND 9付属のドキュメント
    - bind-9.x.y/doc/misc/