DNSミーティング: 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}

内容

- □BIND 9の概要
- □インストール・設定方法
- □BIND 9とIPv6トランスポート
- □管理ツール
- □BIND 9の性能
- □view
- □lwres

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

BIND 9

- ftp://ftp.isc.org/isc/bind9/
- **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 kit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
  - ○opensssl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
    - ▹例: # dnssec-keygen -r keyboard
  - ○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**の場合に限り、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**

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(?)

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の便利なコマンド
    - 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

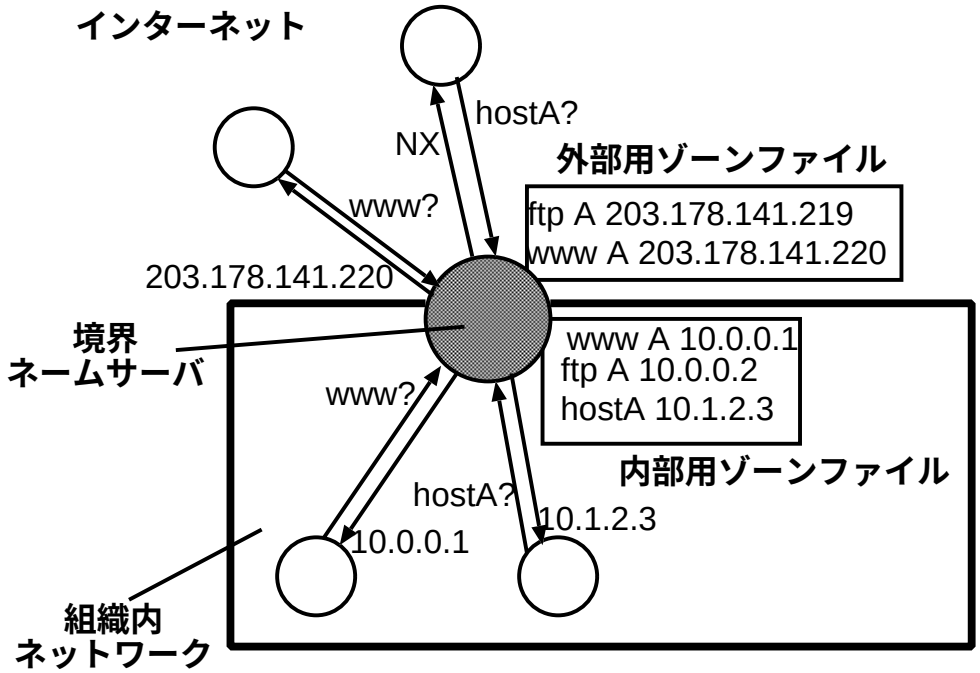
- ○単位(qps)
- ○グルーRRの数が多いほど遅くなる
  - ▹データベースの構造上の問題

	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

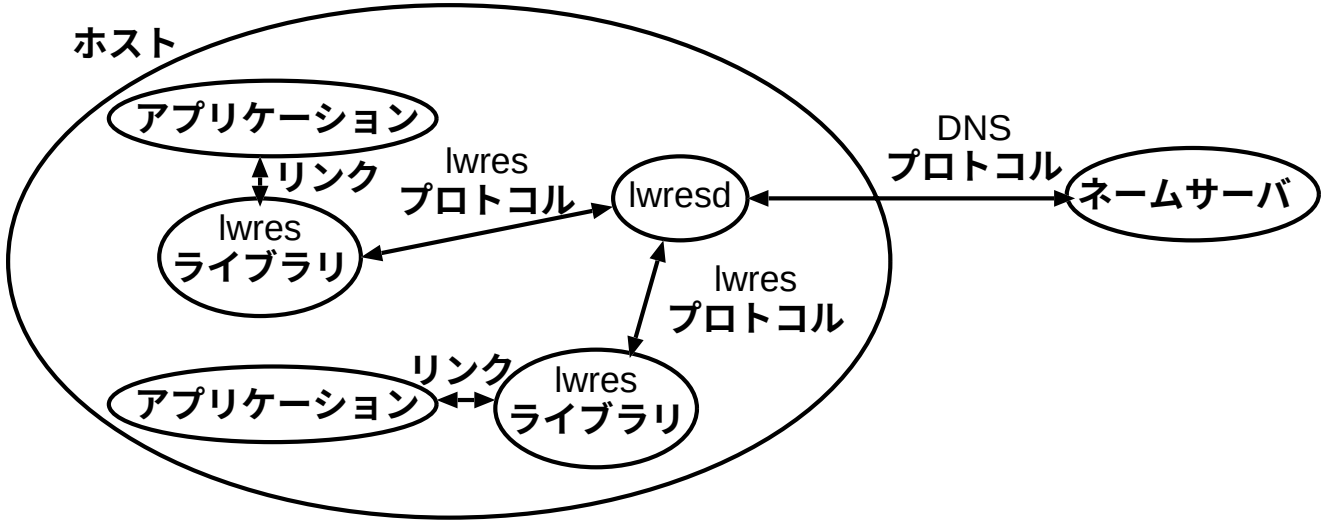
- ○viewの順序は重要: 最初にマッチしたアドレスが適用される
  - ▹"any"は最後に置かないといけない
- ○viewを一つでも定義したら、view外でのゾーン定義は不可

```
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し、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);
        }
}
```

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; };
};
```