DNS入門

2012年8月31日

DNS Summer Days 2012

株式会社日本レジストリサービス（JPRS）

森下 泰宏

講師自己紹介

- 氏名: 森下 泰宏(もりした やすひろ)
  - 1965年9月21日生まれ(46歳)男性
  - 勤務先:株式会社日本レジストリサービス
  - 肩書: 技術広報担当
- 通称: オレンジ
  - 呼び方は苗字でも通称でも、お好きなほうでOKです
    - ただし、勤務先に電話する際には「森下」でお願いします
    - 昔、某教授(仮名)が「オレンジいる?」と電話してきました
  - 通称の由来などは機会を改めて
- DNS歴:1990年から
  - 初めて触ったBINDは4.8.3でした

今日の目的

- DNSの「初学技術者」を主なターゲットとし、DNSに関する以下の項目の理解促進を図ることにより、受講者が自学自習を進める際の手がかりとなる知識の習得をめざす
  - 基本的な成り立ち
  - 基本構造と基本動作
  - 構造上の弱点とその対策
  - 持って生まれた「宿命」にどう立ち向かうか

今日の内容

1. HOSTS.TXTからドメイン名・DNSまでの道のり
2. DNSの基本構造と名前解決の基本動作
3. 構造に由来するDNSの美点・弱点と
   弱点克服のためのさまざまな工夫
4. 持って生まれた悲しい宿命と
   それに立ち向かうための必要事項
5. ここまでのまとめ・Q&A

今日話さない内容①

- ゾーンにまつわるいくつかの話（入れてもよかったかも）
  - 例1：ゾーン／委任／権威の関係
  - 例2：ゾーン分割、ゾーンカット
  - 例3：ドメイン名とゾーンの違い
- 逆引き実現のしくみ
  - 例1：in-addr.arpa、ip6.arpaの作られ方
  - 例2：クラスレス逆引きのための設定
- DNS通信がUDPである理由
- （主な）リソースレコードに関するより具体的な説明
  - 例1：SOAの各パラメーターの意味
  - 例2：CNAMEで指定されるのは別名ではなくて正式名である
  - 例3：間接参照はトラブルの元、な話（MX、NSなど）

今日話さない内容②

- ドメイン名の長さや文字種別などの細かな仕様
  - 例1：なぜFQDNの最大長は253文字なのか
  - 例2：DNSではなぜ大文字と小文字を区別しないのか
  - 例3：255文字を超えるTXTレコードはRFC違反らしい、など
- 各種周辺プロトコルの説明、用語の説明
  - 例1：IXFR、Notify、Dynamic Update、TSIGなど
  - 例2：絶対名、相対名、FQDN、
- 512バイトを超える応答への対応
  - 例：切り詰めとTCPフォールバック、EDNS0
- キャッシュの取り扱い全般
  - 例：キャッシュとネガティブキャッシュ
- 各サーバーソフトウェアにおける設定例
  - 例：プライマリサーバーのためのnamed.conf設定例

今日話さない内容③

- ドメイン名とDNSの運用体制
  - 例1：ICANN/IANAの体制、レジストリ・レジストラモデル
  - 例2：gTLDとccTLDの違い
  - 例3：新gTLD
- 国際化ドメイン名（IDN）全般
- DNS上に構築されたプロトコル・サービス全般
  - 例：SPF、DKIM、DANE、ENUM
- DNSとIPv6の関係全般
- DNSキャッシュポイズニング全般
- DNSSEC全般

ということで、はじまりはじまり・・・

最後までお付き合いくださいませ

1. HOSTS.TXTから
ドメイン名・DNSまでの道のり

おさらい：
インターネットにおける通信のしくみ

- インターネットにおける通信では、IPアドレスという番号で相手を指定・識別している
  - 送信側：受信側のIPアドレスを指定してデータを送る
  - 受信側：送信元のIPアドレスにより、どの相手からデータが届いたのかを知る（識別する）

おさらい：IPアドレスとは？

- 現在使われているのは、IPv4とIPv6の2種類
  - IPv4：0から4,294,967,295まで（2^{32}=約43億個）
  - IPv6：0から340,282,366,920,938,463,463,374,607,431,768,211,455まで（2^{128}=約340澗（かん）個）
- IPアドレスの表記方法
  - IPv4：8ビットずつ4つに区切り、それぞれを10進で記述し、ドットでつなげて表記する
    - つまり、0.0.0.0から255.255.255.255まで
  - IPv6：16ビットずつ8つに区切り、それぞれを16進で記述し、コロンでつなげて表記する
    - つまり、0:0:0:0:0:0:0:0からffff:ffff:ffff:ffff:ffff:ffff:ffff:ffffまで

番号ではなく名前で指定したい

- IPアドレスは人間が記憶・指定するのにはちょっと大変
  - IPv4（例：192.0.2.1）ならまだ何とかなるかもしれないが、IPv6（例：2001:db8:10:8f01:face:b00c:0:25）となると、もう無理
- しかも、IPアドレスは不変ではない
  - 何らかの事情により相手（自分）のIPアドレスが変わった場合、覚え直す（皆に教え直す）必要がある
- そのため、いにしえの昔から番号（IPアドレス）ではなく、より記憶・指定・提示しやすい名前を使えるようにするための工夫が行われてきた

必要なこと:
名前とIPアドレスの対応付け

- インターネットにおいて名前による相手の指定・識別を実現するためには、名前とIPアドレスを対応付ける何らかのしくみが必要
- 対応付けは双方向ともあり得る
  - 送信側:名前に対応づけられているIPアドレスを調べ、そのIPアドレスを指定してデータを送る

  - 受信側:IPアドレスに対応づけられている名前を調べ、データがどこから送られてきたのかを識別する

基本的な2つの方法

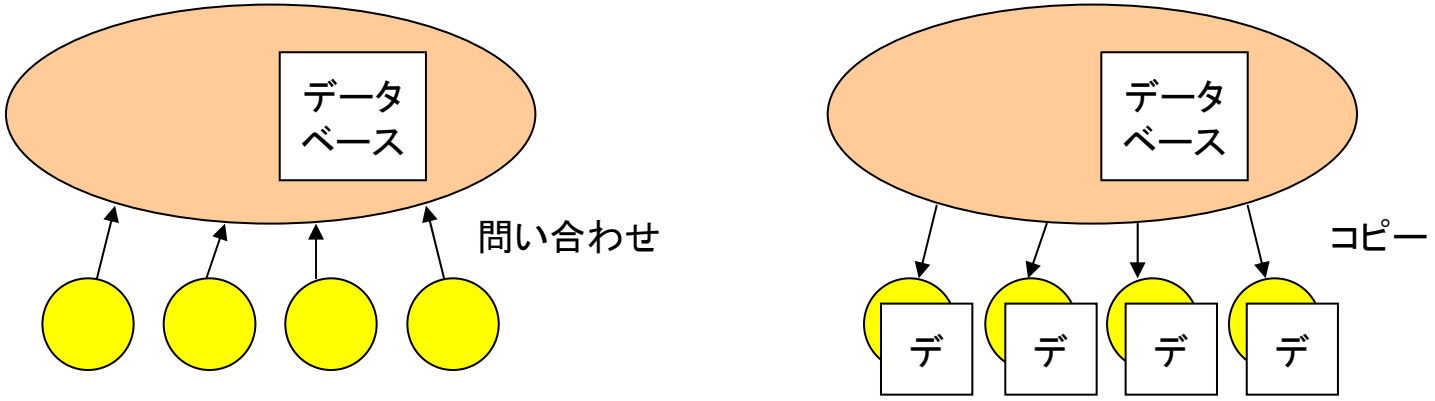
- ドメイン名とIPアドレスの対応付けを、どのように実現するか
  - それには、何らかのデータベースを作るのがよさそう
  - では、どんな方法で作るのがよさそうか？
- 基本的な2つの方法
  - 方法1：それぞれのユーザーが特に連携することなく別個にデータベースを作り、使用する
  - 方法2：一つのデータベースをみんなで共有する

方法1：別個にデータベースを作る

- 携帯電話の「アドレス帳」と同様のしくみ
- 同じ名前（ドメイン名）であっても別のコンピュータ（別のデータベースを持っているかもしれない）では、同じIPアドレスに対応づけられているかどうか保証できない
  - ネットワークでつながっているのに、名前を共有できない⇒不便
- 何らかの方法でデータベースを共有できないか
  - データベースを共有することにより、名前を共有できるようになる⇒方法2へ

方法2: 一つのデータベースを共有する

- 2つの基本的な考え方（またはその組み合わせ）
  1. ネットワーク上のどこかにデータベースを持つ「サーバー」を置いて、名前はすべてそのサーバーに問い合わせる
  2. ある「サーバー」にデータベースを置いておき、ネットワーク経由でそのサーバーからデータベースを入手（配布）し、それぞれのコンピューターで使う



1.集中管理したサーバーに問い合わせる　2.集中管理したサーバーから入手（配布）する

方法2: 一つのデータベースを共有（続き）

- これにより、名前情報の共有による名前の一意性が実現可能になった
  - インターネット上のどこでも、同じ名前は同じIPアドレスに
- DNSができる前（～1980年代）は、実際にこの方法でインターネット（ARPANET）が運用されていた
  - データベースは「HOSTS.TXT」という名前で、SRI-NICという団体により集中管理・公開されていた
  - この名前の名残はUNIXやWindowsなどのhostsファイルとして残っている
- 当時のSRI-NICのIPアドレス：10.0.0.73
  - このIPアドレスさえ知っていればHOSTS.TXTを入手可能
  - ユーザーは自分のHOSTS.TXTを常に最新に保てばよい

方法2（HOSTS.TXT方式）の破綻

- 登録されているコンピューターの数が少ないうちはうまく機能した
- しかし、インターネット自身の成長により登録されているコンピューターの数が増えてくるに従い、うまく機能しなくなってきた
  - SRI-NICの負荷増大
    - HOSTS.TXTの巨大化、更新頻度の増加
  - ネットワークの負荷増大
    - HOSTS.TXTをFTPで入手するユーザーの増加
  - ユーザーの負荷増大
    - 最新版の入手・設定・再配布の必要性

方法2の改良：階層構造の導入による分散管理

- 方法2の問題点：データベースを1つのファイル（HOSTS.TXT）で集中管理していたこと
  - ネットワーク自身の成長により、管理が破綻寸前に
- 名前に階層構造を導入、データベースを分散管理することにより、この欠点の克服を図った

名前の改良：
階層化をしやすくした「ドメイン名」

- 階層構造を導入しやすいように名前を改良
- ラベルと呼ばれる部分的な名前を、ドット（.）でつないだ形により構成
- 左側のラベルほど対象が狭まるように設計

www.example.co.jp

より狭い（下位）⟷より広い（上位）

- 参考：欧米における住所表記

12025 Waterfront Drive, Suite 300, Los Angeles, CA, USA

ドメイン名による階層化の実現:「DNS」

- DNS:Domain Name System
  - ドメイン名(Domain Name)を使えるようにするために開発されたシステム(System)
- HOSTS.TXTの問題点を解決
  - ドメイン名に対応させる形でデータベースを分散
  - 担当する部分のデータベースをそれぞれが管理
- HOSTS.TXTの美点(名前の一意性)は維持
  - 分散管理されたデータベースをネットワークで共有
  - 全体を1つのデータベースのように見せる

2. DNSの基本構造と名前解決の基本動作

…についての説明の前に・・・

重要な注意

重要な注意

- この節の「②：階層構造をたどる」で説明する「たどるサーバー（＝キャッシュDNSサーバー）」の動作は、現時点における実際のものとは一部異なっています。
- 特に、外部名／内部名のチェック、RFC 2181で定義されているリソースレコードの信頼度のチェックなど、本来必要となるいくつかの事項について、意図的に説明を省略しています。

重要な注意（続き）

- その結果、RFC 1034/1035が意図したと考えられるよりシンプルな（ただし、セキュリティ上の考慮が十分ではない）動作説明となっています。
- 今回、意図的に省略した部分の動作説明については、いつか何らかの形（続編（？））で実施できたらいいなと考えています。

…ということで、説明に戻ります。

おさらい：DNSがめざした大事なこと

① HOSTS.TXTの問題点を解決
  - 階層構造を構成することによる分散管理の実現

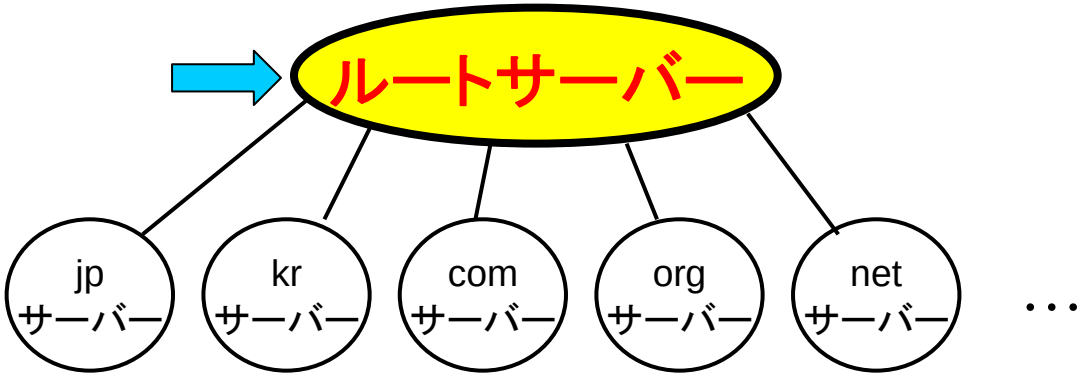
② HOSTS.TXTの美点（名前の一意性）を維持
  - 階層構造をたどることによる名前解決の提供

- ①、②ともに、とても大事なこと
- ①は大抵の教科書にきちんと書かれている
  - 「DNS＝階層構造」というのもよく知られている
- しかし、②はあまり明確に書かれていない
  - きちんと解説されていない場合が多いように思える

そこで、このチュートリアルでは②を特に手厚く説明します

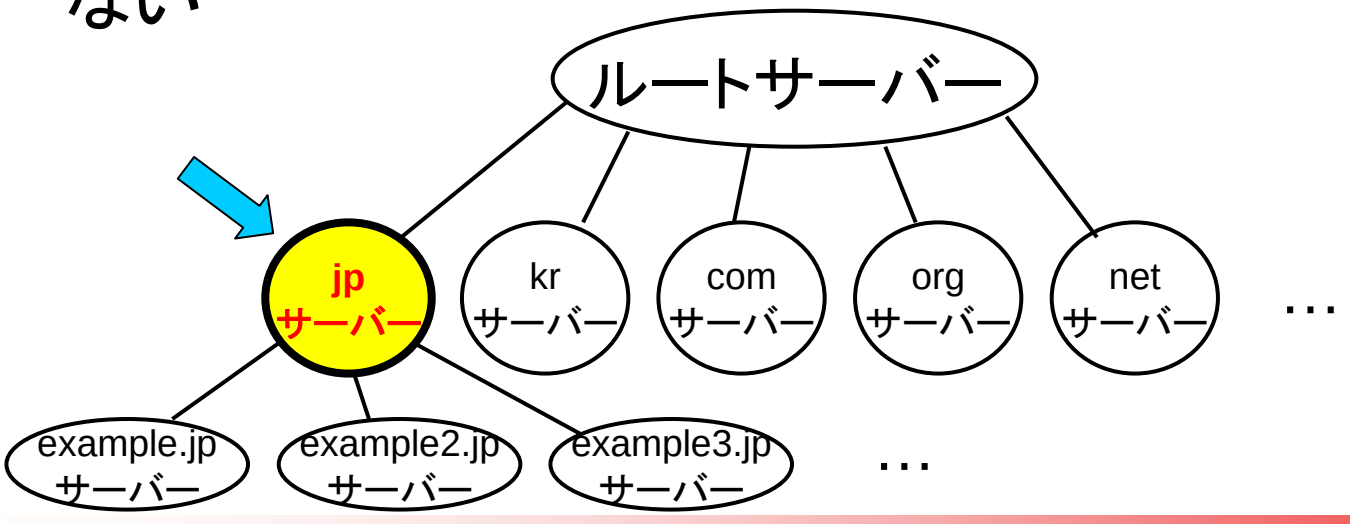
①:階層構造を構成する

- まず「根っこのサーバー」を作る
  - ルートサーバー（root server）と呼ばれる
  - ルートサーバーはjpやcomなど、各ドメイン名の1番上のドメイン名（トップレベルドメイン：TLD）のサーバーが、インターネット上のどこにあるのか（つまり、どのIPアドレスなのか）を知っている
  - 名前とIPアドレスのデータベースそのものは持っていない



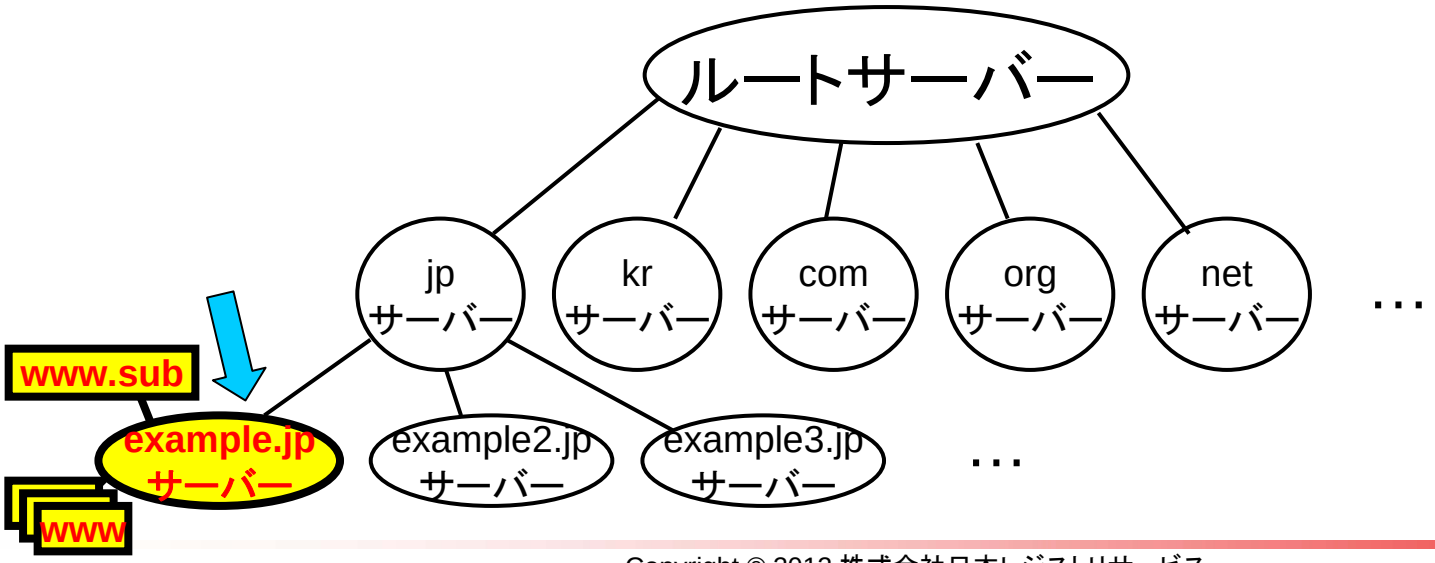
①:階層構造を構成する(続き)

- 次に「TLDサーバー」を作る(例:jp)
  - 自分のTLD内の情報のみを管理する
  - jpのサーバーはjpのひとつ下、つまりexample.jpやexample2.jpなどのサーバーがインターネット上のどこにあるかを知っている
  - 名前とIPアドレスのデータベースそのものは持っていない



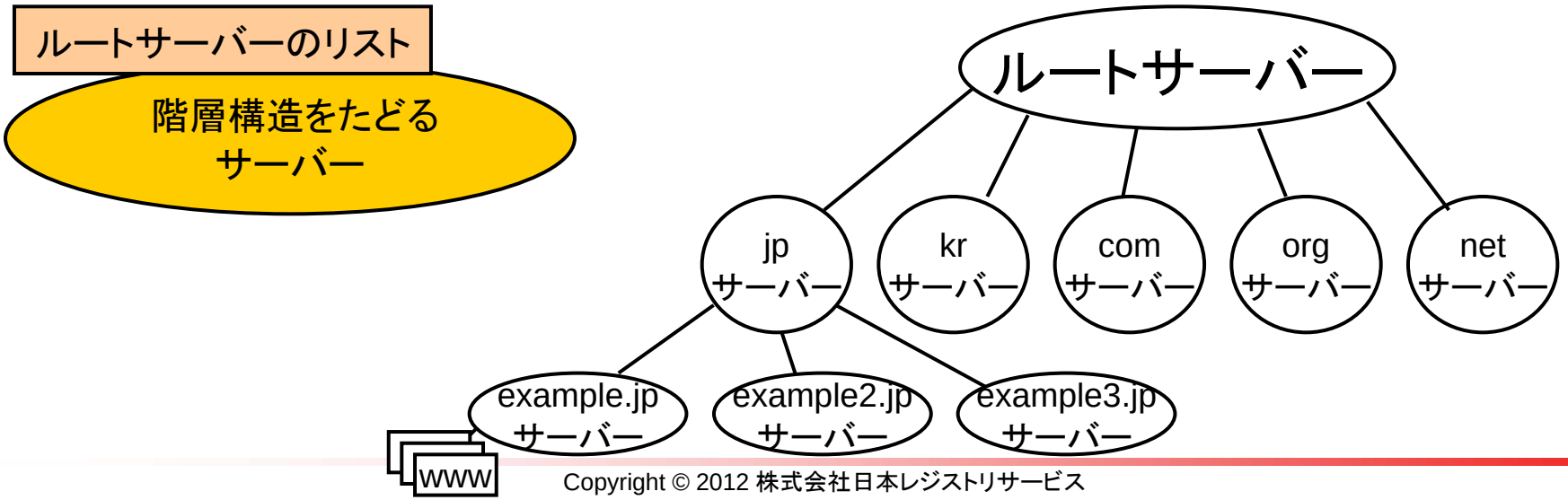
①:階層構造を構成する(続き)

- さらに「2番目のレベルのサーバー」を作る(例:example.jp)
  - そのドメインの内部の名前のデータベースを管理する
  - 必要に応じ、example.jpの内部にwww.sub.example.jpのようなサブドメインを作ることも可能
- これで「www.example.jp」という、TLDまでの名前に対する階層構造が完成
  - 「完全修飾ドメイン名(FQDN)」と呼ばれる



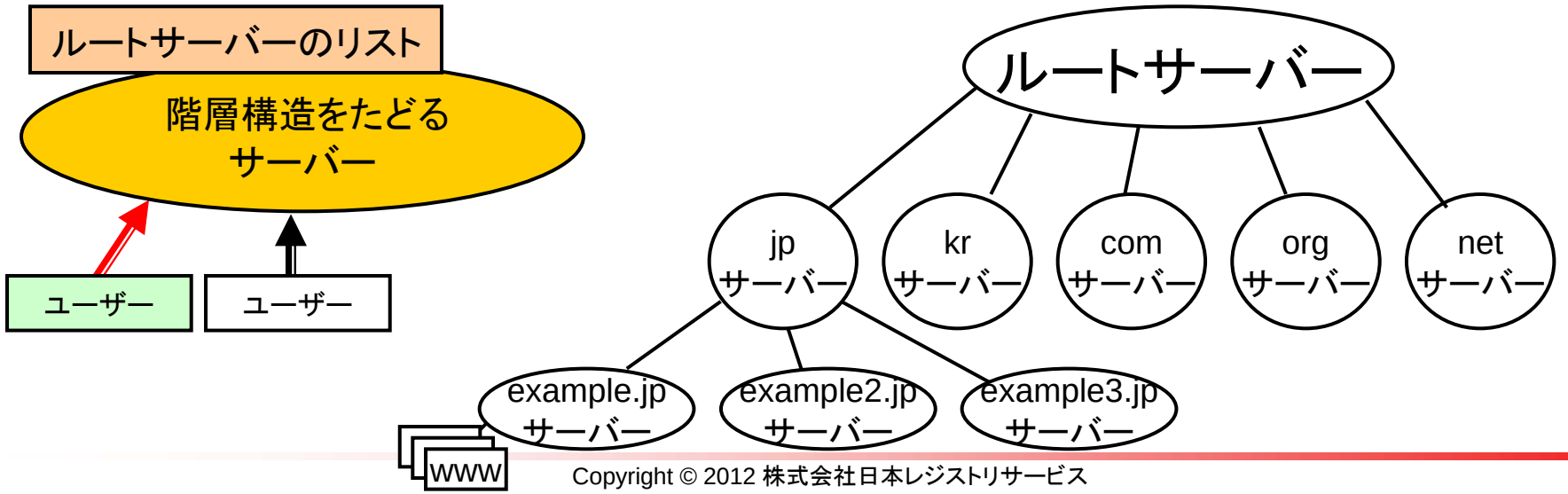
②:階層構造をたどる

- ①の完成後、階層構造をたどるサーバーを①とは別に準備する
- 階層構造をたどるサーバーには、ルートサーバーのリストだけを持たせる
  - 他のサーバーのリストを持つ必要はない
    - たどる際の流れの中で、示された他のサーバーから手がかりを入手
  - 階層構造の構成には参加しない（役割が違う）



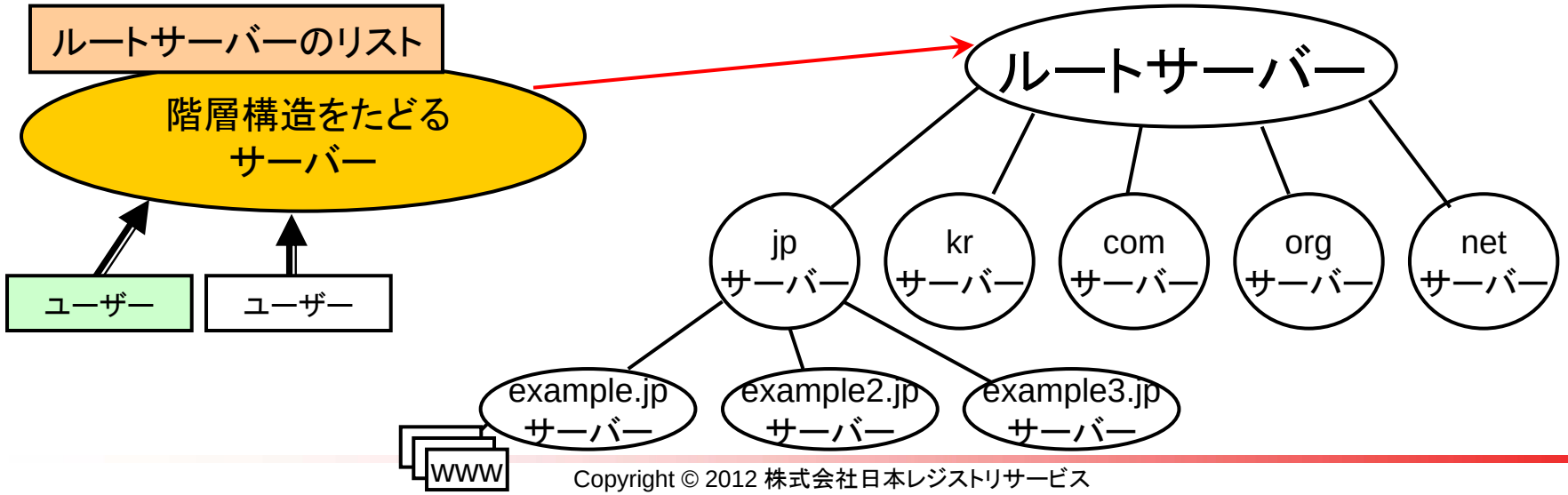
②：階層構造をたどる（続き）

- ユーザーはたどるサーバー（長いので省略）に対し「www.example.jpのIPアドレスは？」などの問い合わせ（名前解決要求）を送信する
  - 大事なポイント：ユーザーはたどるサーバーとのみ通信
  - たどるサーバーは複数のユーザーからの問い合わせを同時に処理できるように作られている（図参照）
- ユーザーからの問い合わせには「あなたが階層構造をたどってください」という特別な印（RDと呼ばれる）がつけられている
  - RDつきの問い合わせを2本線矢印（⇒）で表す



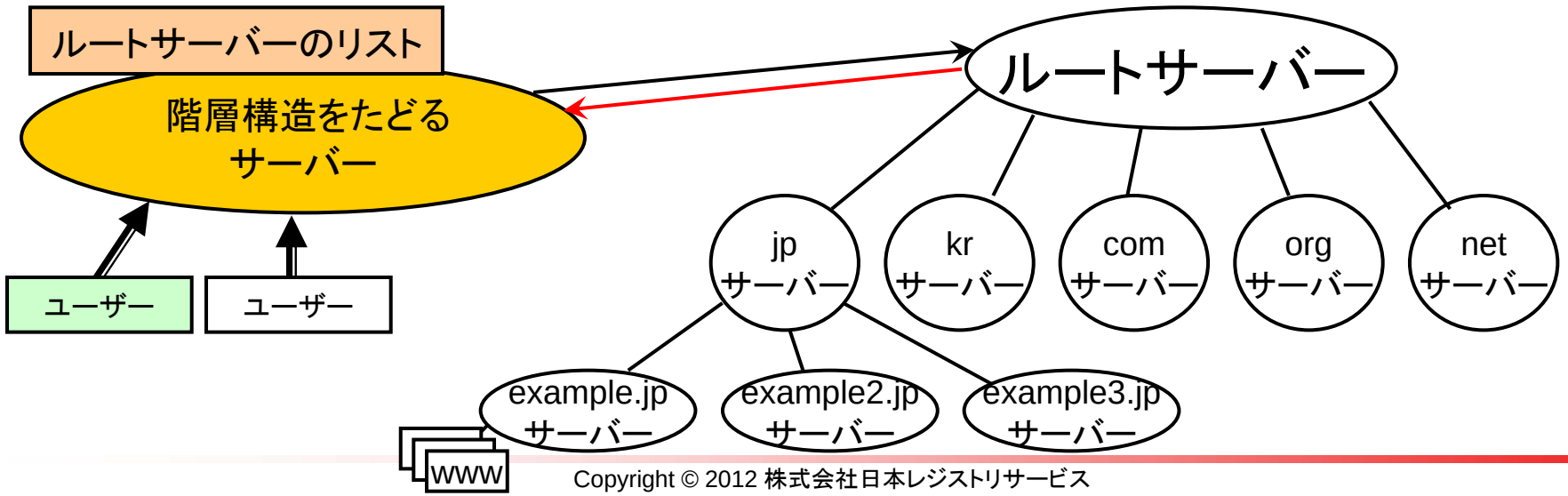
②：階層構造をたどる（続き）

- たどるサーバーはまずルートサーバーに問い合わせる
  - 最初の状態ではルートサーバーしか知らないため
- 問い合わせの内容としてユーザーから受け取った「www.example.jpのIPアドレスは？」をそのまま流用する
- （オプション）なお、最近のたどるサーバーは用心深く作られており、サービスを始める前に「私の持っているルートサーバーのリストは本当にあってる？」とルートサーバーに一度だけ問い合わせ、得られた結果を本番の問い合わせで使う
  - この行動のことを「プライミング」と呼ぶ



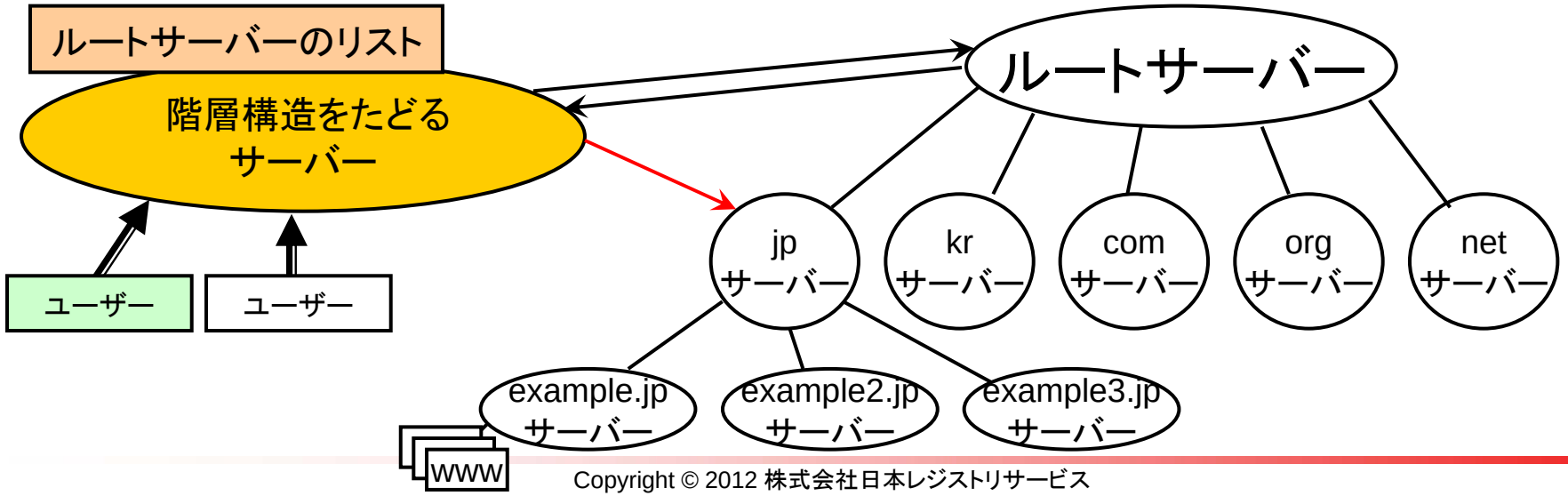
②:階層構造をたどる(続き)

- 階層構造を構成するサーバーには階層構造をたどる機能がない(必要ないので有効にされていない)ため、問い合わせに印(RD)はつけない(以降、RDなしの問い合わせを1本線矢印(→)で表す)
- ルートサーバーはjpで終わっているドメイン名はjpのDNSサーバーに管理を委任しており自分では管理していないので、「jpのサーバーに委任しています」という応答を返す



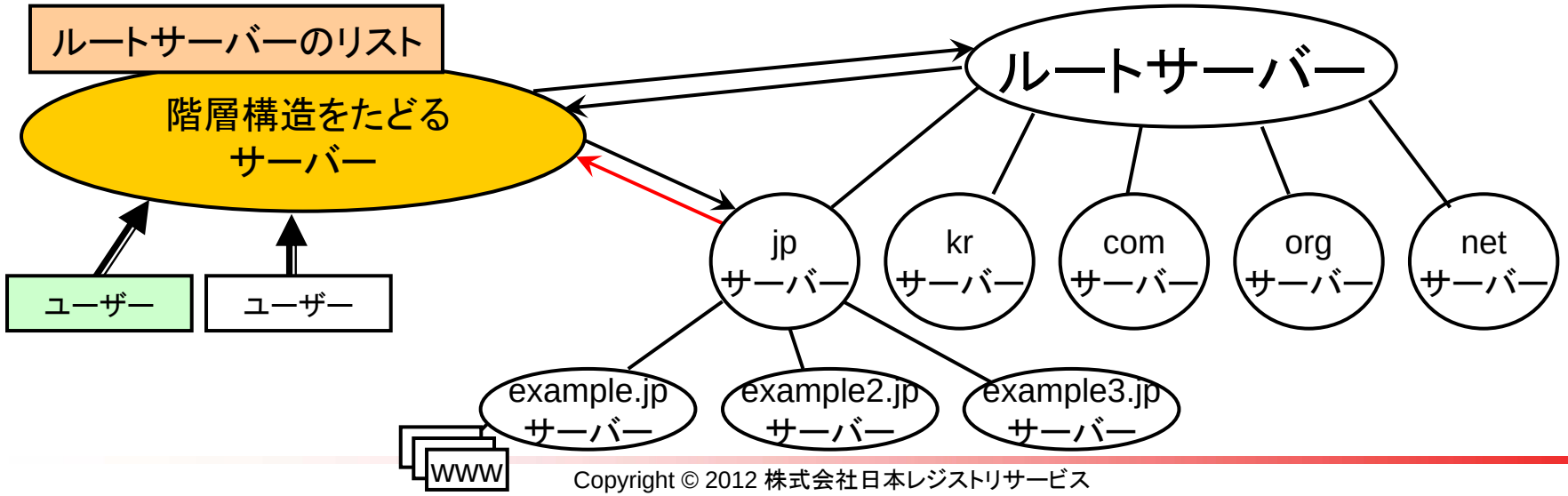
②:階層構造をたどる(続き)

- 応答を受け取ったたどるサーバーはjpのサーバーに対し、改めてルートサーバーに聞いたのと同じ「www.example.jpのIPアドレスは？」という内容を問い合わせる
  - ルートサーバーの時と同様、印(RD)はつけない



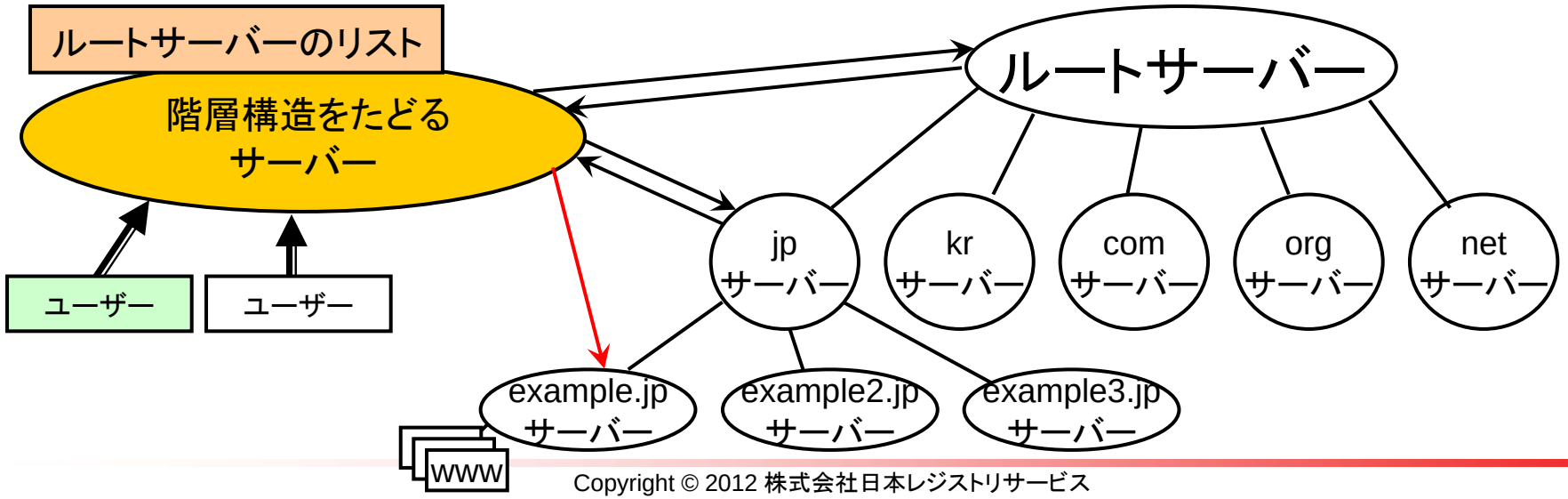
②:階層構造をたどる(続き)

- jpのサーバーはルートサーバーと同様「example.jpのサーバーに委任しています」という応答を返す



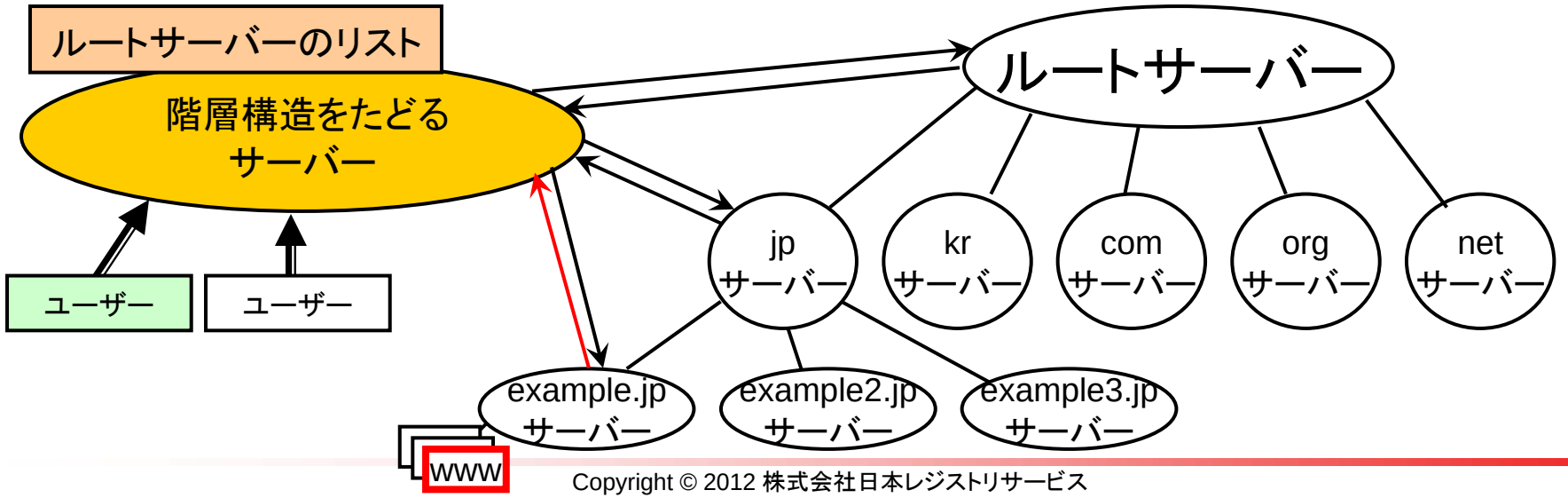
②：階層構造をたどる（続き）

- 応答を受け取ったたどるサーバーはexample.jpのサーバーに対し、改めて「www.example.jpのIPアドレスは？」という内容を問い合わせる
  - ルート、jpのサーバーの時と同様、印（RD）はつけない



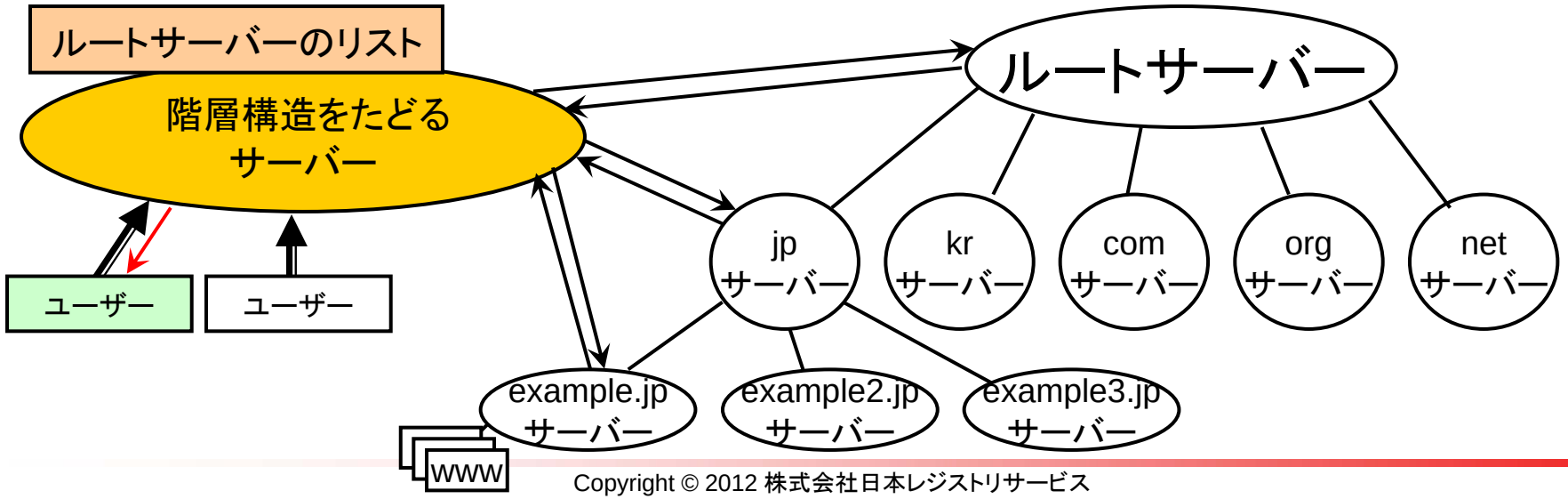
②：階層構造をたどる（続き）

- example.jpのサーバーはwww.example.jpのIPアドレスを管理しているので「www.example.jpのIPアドレスは192.0.2.1です」という応答を返す
- 実際には「その名前は存在しません」「その名前は存在しますがIPアドレスがつけられていません」「その名前は実は別名で、本名はwww.example.comです」など、状況により応答の内容は変化する



②:階層構造をたどる(続き)

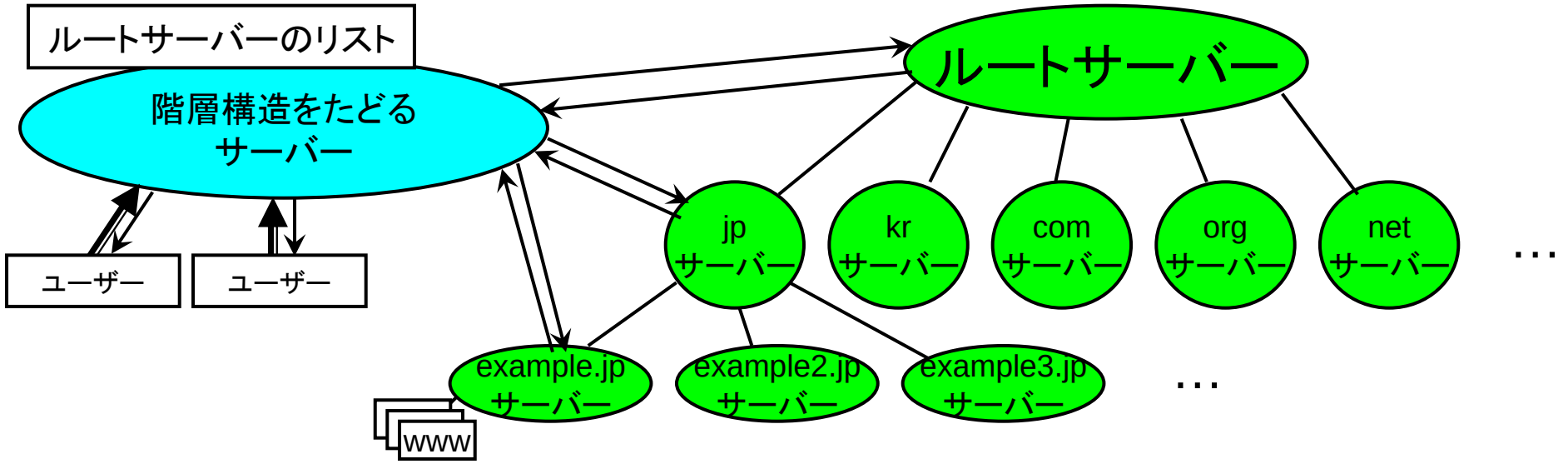
- 応答を受け取ったサーバーが結果をユーザーに返す
  - 実際には別名だった場合に本名として指定されたドメイン名に対する名前解決を行うなど、状況に応じた追加作業が発生する場合がする
  - その場合、たどるサーバーはたんたんと作業を続ける
- これで「www.example.jp」に対する名前解決が完了



DNSは目的・動作が異なる2種類の「サーバー」により構成

- 階層構造を構成することによる分散管理
- 階層構造をたどることによる名前解決

いずれが欠けてもDNSは動作しない



3. 構造に由来するDNSの美点・弱点と弱点克服のためのさまざまな工夫

DNSの美点

- 管理権限の委任（delegation）を実現できる
  - 指定したドメイン名（*1）の管理を別のサーバー（組織）に任せることができる

  （*1）委任した時点で別のゾーンとなる
- サーバー1台が持つデータベースのデータ量を減らすことができる
  - それぞれの委任先にデータを分けて持たせることができる
  - サーバー1台あたりの負荷を減らすことができる

DNSの美点（続き）

- データベースの更新がやりやすくなる
  - データベースは委任先のサーバーが分散して持つ
  - それぞれのデータベースはそれぞれのサーバーが更新する
- 名前空間の一意性が損なわれない
  - 集中管理方式の美点が維持される

DNSの美点（続き）

- これらの美点はとても大きなものであった

ドメイン名とDNSの導入による分散管理の実現により、
従来のHOSTS.TXTによる集中管理方式において
問題となった事項を解決することができた

- 1980年代に存在したさまざまな広域ネットワークのうちインターネットのみが生き残り、かつ成長できた（成長を支えられた）大きな要因

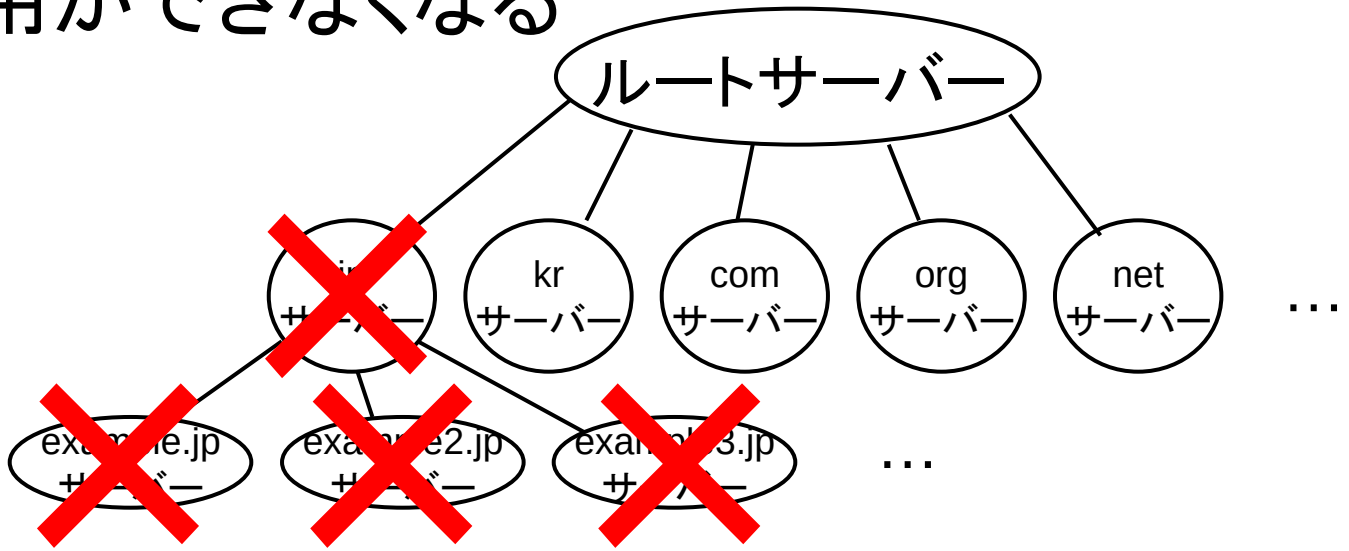
DNSの弱点

- しかし、DNSにはその構造に由来する弱点も存在する
- 今回取り上げる弱点
  - 特定サーバーへの負荷／リスクの集中
  - ルートサーバーのIPアドレスの固定化（変更困難）
- 以降ではこれらの弱点の詳細と、その克服のために採用された技術について説明
- 上記以外の弱点については私の以下の資料も参照

  DNSをあえてdisってみる（#dnstudy02 2011年10月29日発表資料）
  <http://www.slideshare.net/OrangeMorishita/20111029-part1dnsdis>

DNSの弱点①:
特定サーバーへの負荷／リスクの集中

- 上位のサーバーほど負荷が集中する
  - ルートサーバーやTLDのサーバーなど
- 万一、上位のサーバーのサービスがすべて停止すると、そこから下のすべてのドメイン名の利用ができなくなる



弱点①の克服(1):
サーバーの冗長化・分散化

- サーバーを複数にすることで負荷を分散し、かつサービスダウンが起きにくくする
  - 複数のサーバーを設置した場合、データは1つのプライマリサーバーから他のセカンダリサーバーにコピー(ゾーン転送)され、すべてのサーバが同じデータを持つ
  - このためDNSでは当該サーバーを外から見た場合、プライマリサーバーとセカンダリサーバーは基本的に同一視されることに注意

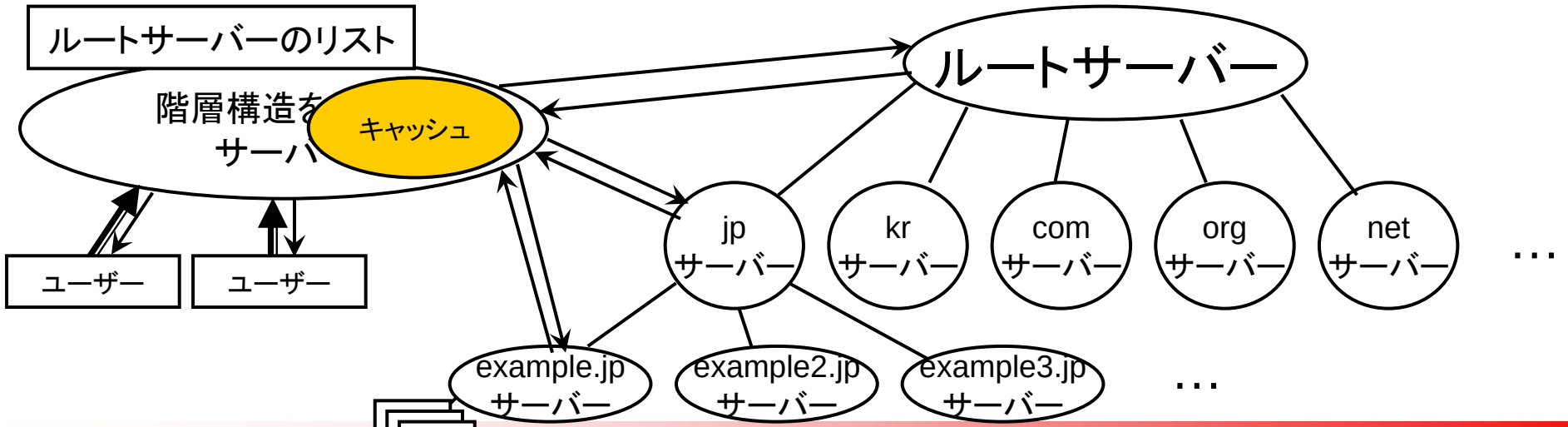
サーバーの冗長化・分散化（続き）

- 2012年8月25日時点においてルートサーバー340台、JPサーバー23台が世界中で稼働中
  - IP Anycastという技術が使われている
  - 「JPRSトピックス＆コラム No.005」を参照
    ■DNSのさらなる信頼性向上のために～IP Anycast技術とDNS

<http://www.root-servers.org/map/> より引用

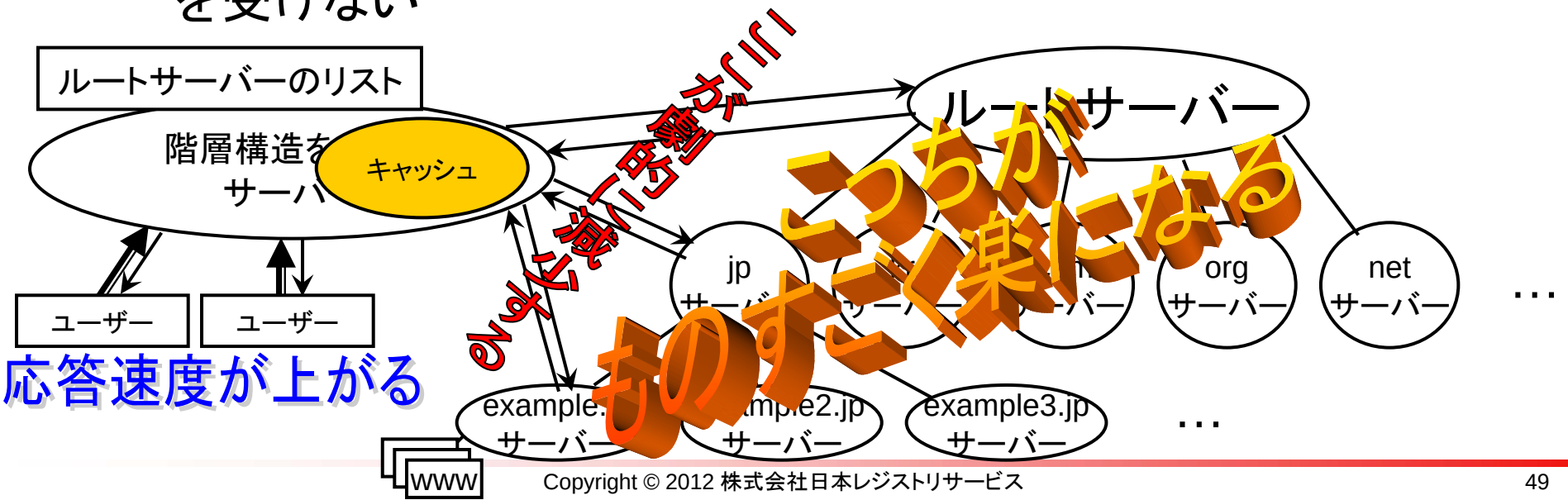
弱点①の克服(2):キャッシュの導入

- 「階層構造をたどるサーバー」内部にキャッシュを導入し、得られた応答をしばらくの間記憶させる
  - ユーザー側のクライアント(Webブラウザなど)に導入する場合もある
- キャッシュが有効である間に発生した同内容の問い合わせに対しては、キャッシュの内容を参照して応答する



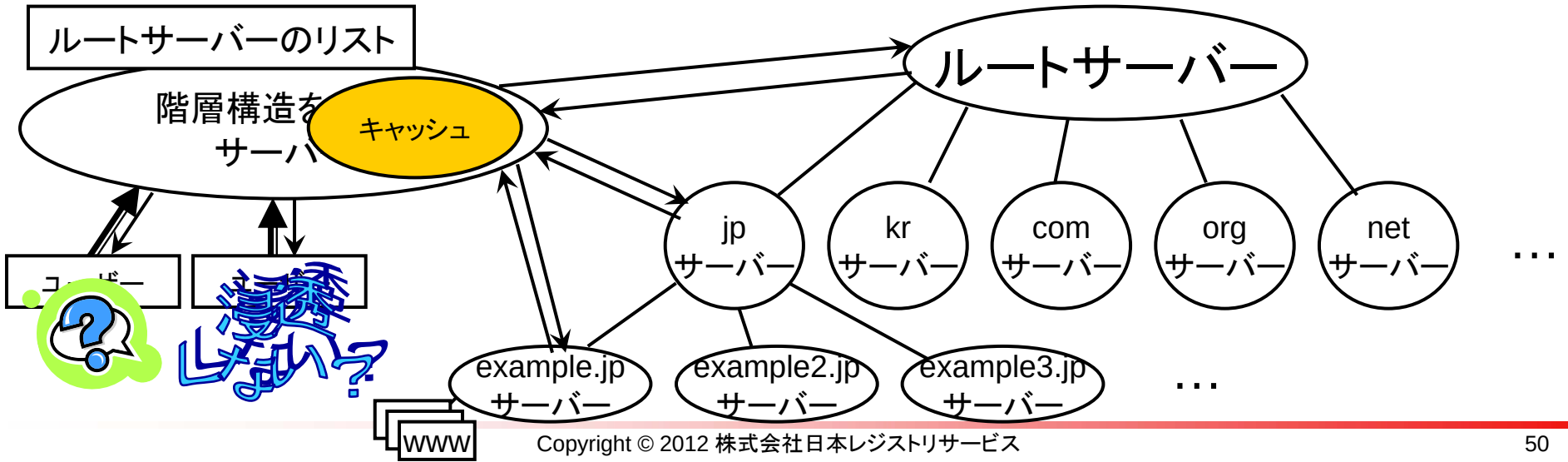
キャッシュの導入（続き）

- キャッシュの導入により階層構造を構成するサーバーへの問い合わせ回数を劇的に減少させ、サーバーの負荷を減らすことができる
  - ユーザーに対する応答速度の向上にもつながる
- キャッシュはサーバーダウンの悪影響軽減にも有用
  - キャッシュが有効である間は関係するサーバーダウンの影響を受けない



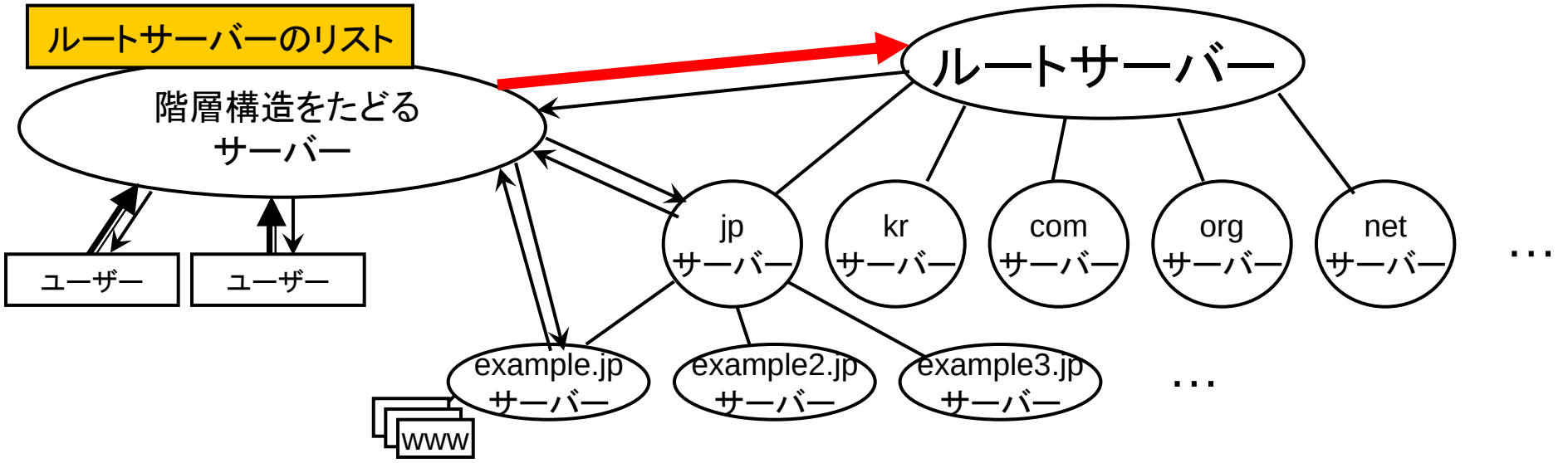
キャッシュの導入（続き）

- キャッシュの導入はDNSを大成功に導いた
  - 負荷軽減、信頼性向上、応答性能・応答速度向上
- 一方、キャッシュはDNSのしくみを複雑化し、DNSにおけるデータの取り扱いを直観的に理解しづらくした
  - いわゆる浸透や伝播など、DNSデータの取り扱いに対する誤った概念がはびこる要因の一つともなった



DNSの弱点②:
ルートサーバーのIPアドレスの固定化

- ルートサーバーのIPアドレスを変えることが困難になる
  - インターネット上に存在する「階層構造をたどるサーバー」すべての設定内容に影響を及ぼす



弱点②の克服：運用でカバー

- できるだけ変えなくても済むようにする
  - 専用のIPアドレス（ブロック）やAS番号を準備し、それを使う
- 大幅な変更を避ける
  - 同時に複数台変更するなど
- IPアドレスが変わる場合の事前周知を徹底する
  - 最近の主な変更
    - 2002年11月5日: J-root
    - 2004年1月29日: B-root
    - 2007年11月1日: L-root
- 変更後、旧IPアドレスでも一定期間サービスを継続する
- その後も、旧IPアドレスでは異なったDNSサービスをすることをできる限り避ける

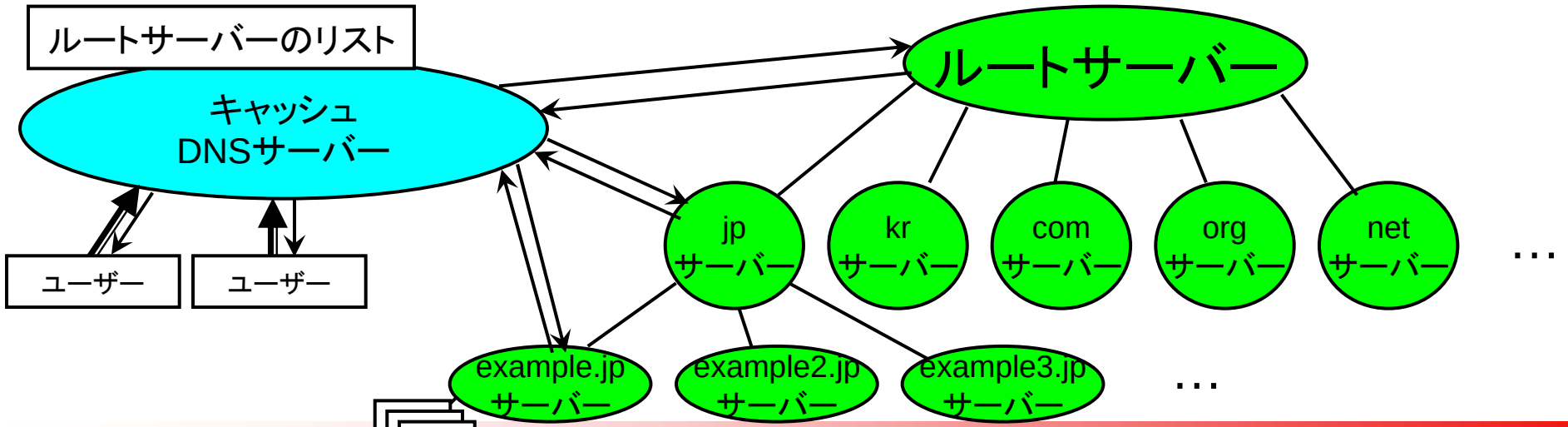
4. 持って生まれた悲しい宿命とそれに立ち向かうための必要事項

持って生まれた「悲しい宿命」

- DNSには生来の構造に起因する悲しい宿命とも言うべきものが存在する
- 以降ではその紹介と、それに立ち向かうために必要となる事項について説明

持って生まれた悲しい宿命：役割が異なる2種類のサーバー

- 「階層構造を構成するサーバー」と「階層構造をたどるサーバー」の2種類のサーバーを別々に準備する必要がある
  - 階層構造を構成する「**権威DNSサーバー**」（緑色）
  - 階層構造をたどる「**キャッシュDNSサーバー**」（水色）



…の存在が、あまりにも認識・理解されていないことによる不幸

- これがDNSが持って生まれた「悲しい宿命」
- これら2種類のサーバーは役割が異なっている
  - 管理手法や設定内容なども当然異なっている
- これらが異なっているにもかかわらず、いずれのサーバーも「**DNSサーバー**」と呼ばれている
- しかも、DNSのもっともメジャーな実装の一つであるBINDでは、この2種類のサーバーを兼用できてしまう
  - このことが多くの混乱・DNSに対する理解の阻害の原因となっている

宿命に立ち向かう(1):
役割・動作の正しい理解

- それぞれのサーバーの役割をきちんと理解する
- 「DNSサーバー」という用語を単独で使うのをできる限り避け、どちらのサーバーのことを示しているかを常に明確にする
  - 階層構造を構成する「権威DNSサーバー」
    - 英語では「authoritative DNS server」
  - 階層構造をたどる「キャッシュDNSサーバー」
    - 英語では「caching DNS server」

役割・動作の正しい理解（続き）

- 「権威DNSサーバー」「キャッシュDNSサーバー」を示す用語は、日本語・英語ともに複数の流儀があるので注意
  - 構成する：「権威サーバー」「コンテンツサーバー」など
  - たどる：「キャッシングリゾルバ」「フルリゾルバ」など
- 使う用語を統一しておくことで、混乱を避けられる
  - 少なくとも、一つの文書内では統一すること
- 他人の文章を読む場合、使われている用語がどちらのサーバーのことを示しているか、きちんと理解する必要がある

宿命に立ち向かう(2):機能ごとにサーバーを分割(BIND 9)

- 兼用をやめ、権威DNSサーバーとキャッシュDNSサーバーに分割する
  - 10年以上前からBINDの開発元(ISC)も分割を推奨している
    ISC Technical Note: Running An Authoritative-Only BIND Nameserver
    <http://ftp.isc.org/isc/pubs/tn/isc-tn-2002-2.html>
- 分割は別のハードウェアを準備しなくても可能
  - サービス用のIPアドレスが別にあればOK
  - その機器でしか使われないキャッシュDNSサーバーなら、127.0.0.1や::1(ループバックI/F)を割り当て可能
- 詳細については「JPRSトピックス&コラム No.020」なども参照

■DNSの安全性・安定性向上のためのキホン

5. ここまでのまとめ・Q&A

ここまでのまとめ①

- インターネットでは相手をIPアドレスという番号で指定・識別している
- 番号ではなくより使いやすい名前で相手を指定・識別するためのしくみとして、ラベルをドット（.）でつないだ階層構造を持つドメイン名が考案された
- ドメイン名を分散管理するためのしくみ（手段）として、DNSが開発された

ここまでのまとめ②

- DNSはルートサーバーを頂点とした階層構造を構成する部分と、その階層構造をたどる部分の2つの要素により構成されている
- DNSには上記に対応する2種類のサーバーが存在している
  - 階層構造を構成する「権威DNSサーバー」
  - 階層構造をたどる「キャッシュDNSサーバー」

ここまでのまとめ③

- DNSの最大の美点は、委任の導入による分散管理の実現である
- DNSの構造に由来する弱点を克服するため、サーバーの冗長化・分散化やキャッシュの導入が実施された
- キャッシュの導入はDNSを大成功に導いたが、同時にDNSのしくみを複雑化し、DNSにおけるデータの取り扱いを理解しづらくした

ここまでのまとめ④

- BIND 9では権威DNSサーバーとキャッシュDNSサーバーを兼用できてしまうため、多くの混乱・DNSに対する理解阻害の原因となっている
- DNSに関する理解促進のためにもキャッシュDNSサーバーと権威DNSサーバーの兼用は避けるべきであり、開発元のISCも両機能の分離を推奨している

ここまでのまとめ⑤

- 「権威DNSサーバー」「キャッシュDNSサーバー」を示す用語には複数の流儀があり統一されていないため、文書を読む場合などにはどちらのサーバーのことを示しているか、きちんと理解する必要がある

Q&A