BroadBand Tower, Inc.

DNSのよくある間違い

伊藤 高一

DNS Summer Days 2012

イントロダクション

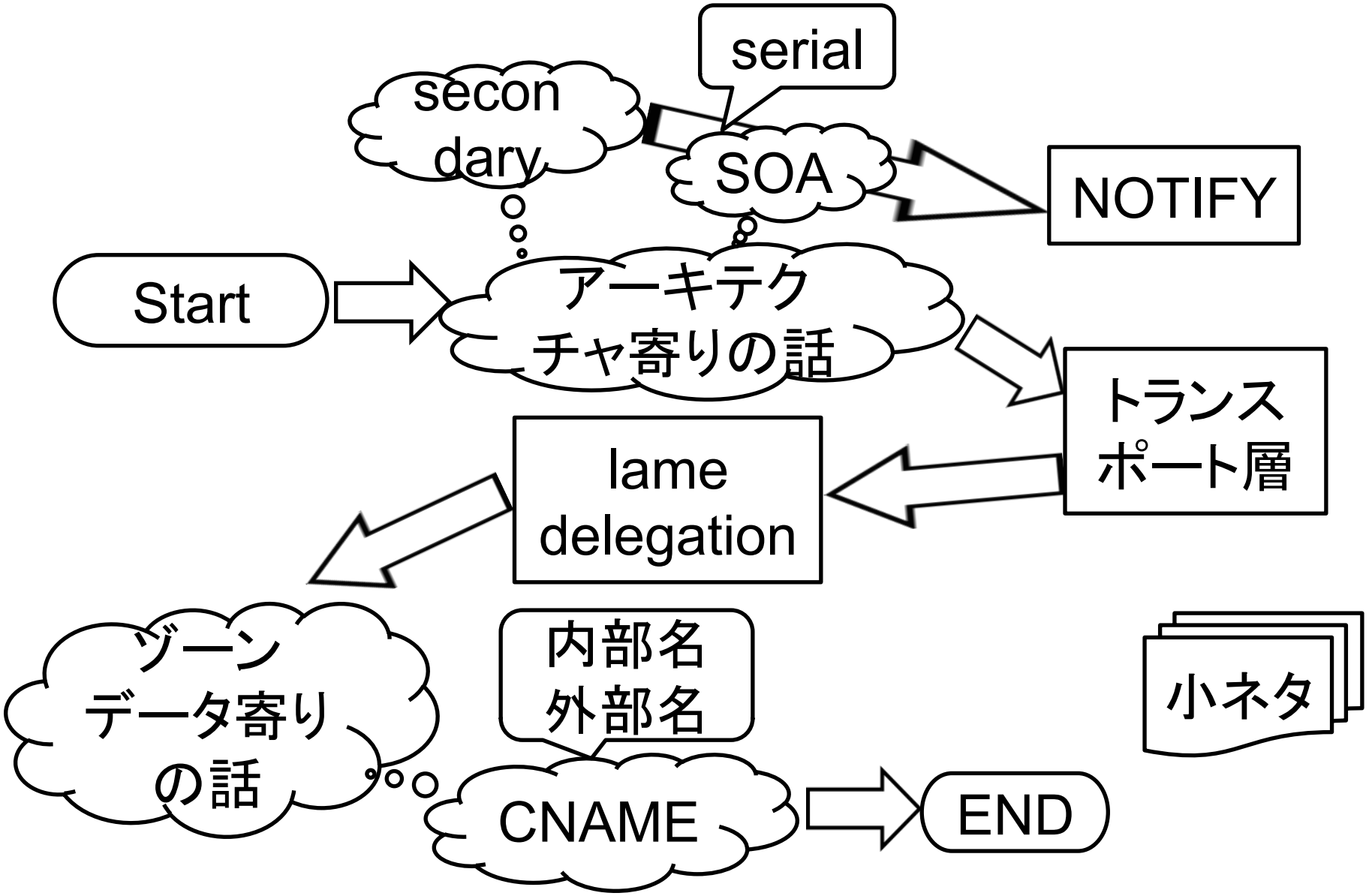
BroadBand Tower, Inc.

- DNSは「世界で唯一成功した分散データベース」って言われています。
- おかげでjust put it in the DNSなる風潮も。
- なんで成功したのかって?
- だってユルいんだもん :-)
- 多少いい加減でも、何となくそれっぽく動く。
- 成功の影に累々たる結果オーライあり。

イントロダクション(続き)

- 今日の黒幕から言い渡されたお題の「よくある間違い」は、ある程度、散りばめておきました。
- 「よくわかんないけど、設定例でこうなっているから」で流しがちな事とか、逆に設定例では取り上げられる機会が少ない事もちょっと追及してみました。
  - カルトネタではなく、実用的な範囲で。
- 午前中の話は理解されていることを前提にさせていただきました。

Agenda


serial
secon
dary
SOA
NOTIFY
Start
アーキテク
チャ寄りの話
トランス
ポート層
lame
delegation
ゾーン
データ寄り
の話
内部名
外部名
小ネタ
CNAME
END

BroadBand Tower, Inc.

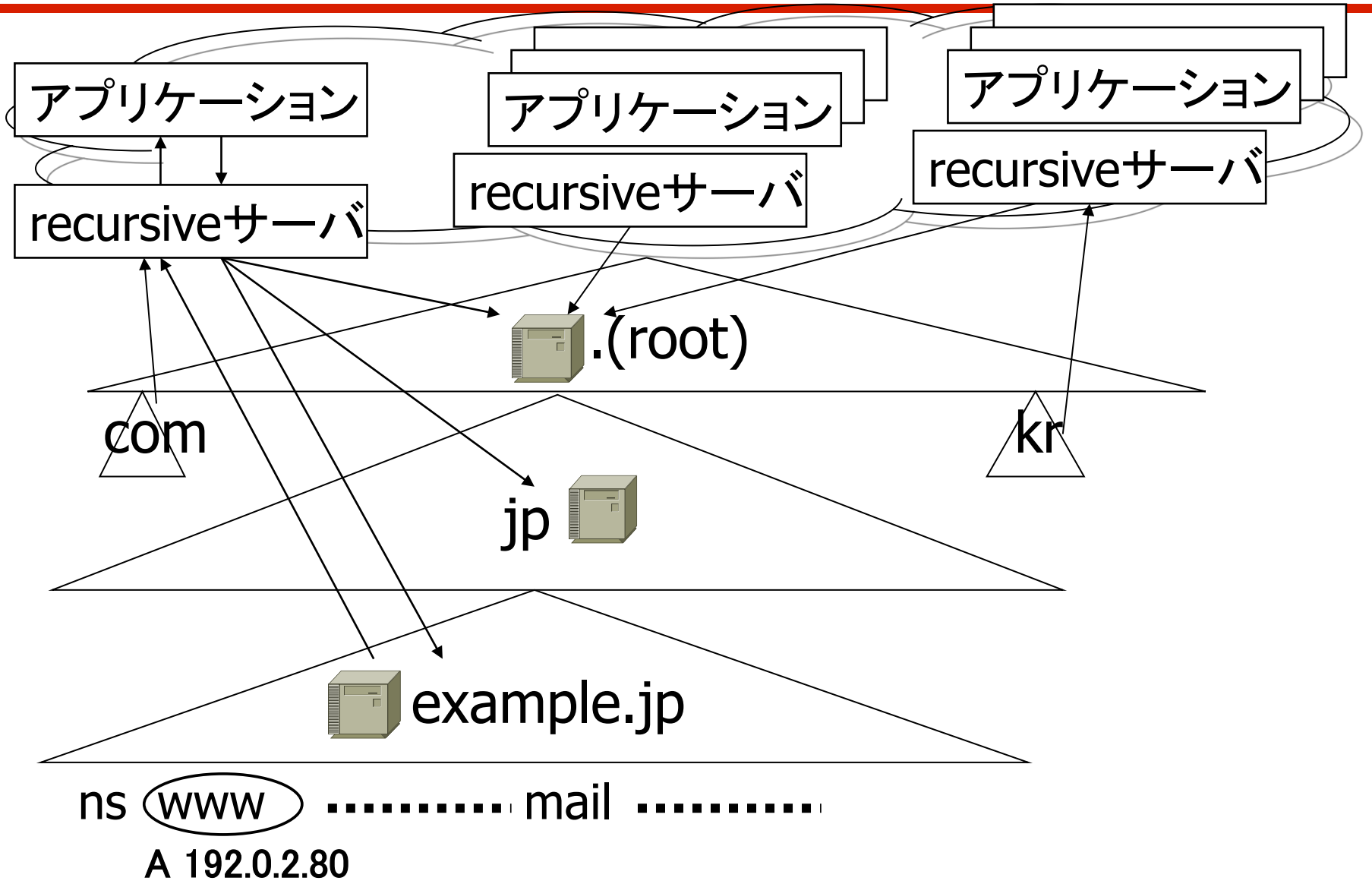
アーキテクチャ寄りの話

DNSってどんなアーキテクチャだっけ?

BroadBand Tower, Inc.

	recursive サーバ	authoritative サーバ
サービス内容	Worldwide	特定ゾーン
サービス対象	自ネットワーク内	Worldwide
authority	なし	あり/委任先の提示

DNSってどんなアーキテクチャだっけ?(続き)



関連RFC

- RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

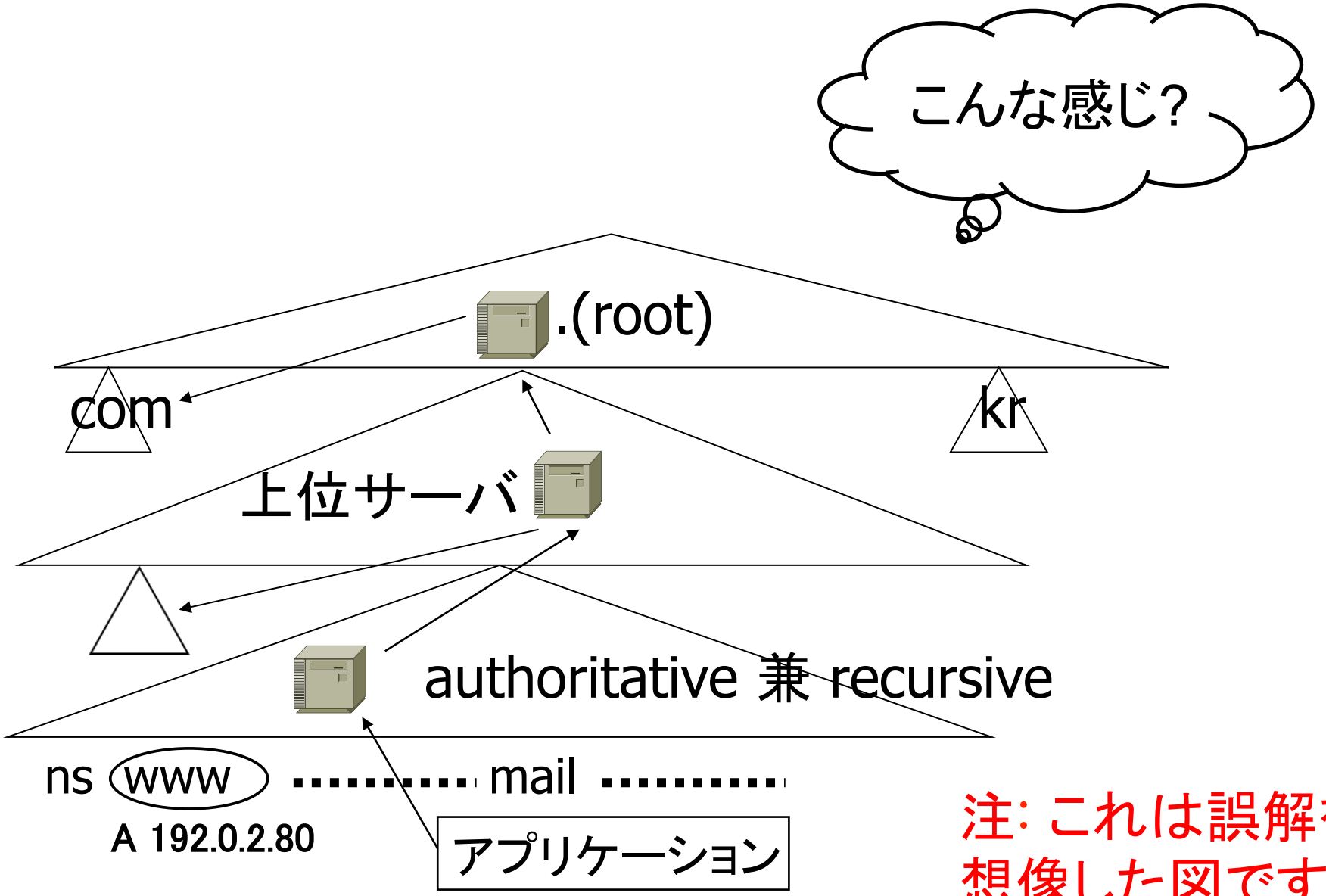
上位サーバ

- ユーザ:「上位サーバのIPアドレスを教えて下さい」
- xSP:「上位サーバって何ですか???」

上位サーバ

- recursive queryはrootサーバを起点として下降探索するという動作が理解されていない。
- ユーザのrecursiveサーバは「上位サーバ」なる物に問い合わせるんだ、という誤解。
  - かつてはauthoritativeサーバとrecursiveサーバを兼用するのが常識で、両者の分担が認識されていなかった。
- 確かにforwarderを使えば、そういうアーキテクチャも実装できるが…

上位サーバ



注: これは誤解を想像した図です!!

primary/secondary/master/slave

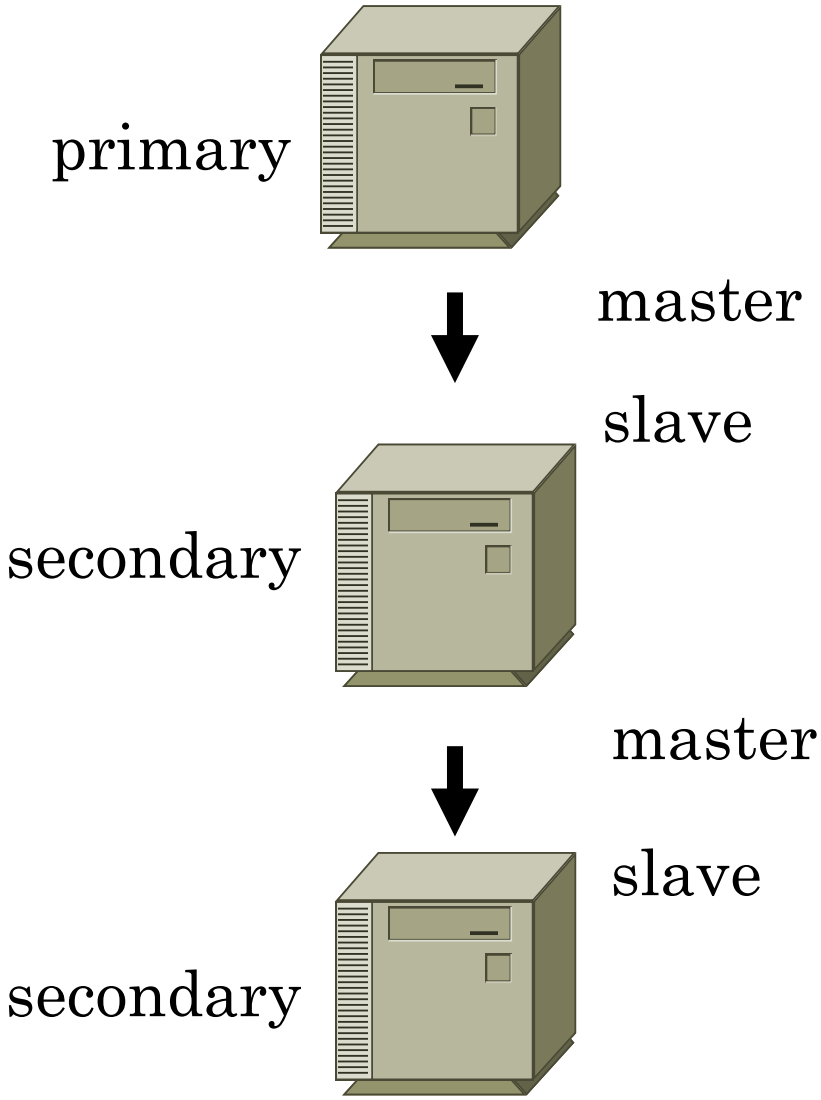
- primary v.s. secondary
  - ゾーンデータの出所に注目
- primary
  - ローカルファイルなど、ゾーン転送以外で得たゾーンデータを参照するauthoritativeサーバ。
  - masterはいない。
- secondary
  - masterからのゾーン転送により得たゾーンデータを参照するauthoritativeサーバ。

primary/secondary/master/slave(続き)

BroadBand Tower, Inc.

- master v.s. slave
  - ゾーン転送の送出側/受け側に注目
- master
  - あるゾーン転送に注目したときの送出側
- slave
  - あるゾーン転送に注目したときの受け側

primary/secondary/master/slave(続き)



関連RFC

- RFC 1034: DOMAIN NAMES - CONCEPTS AND FACILITIES
- RFC 1996: A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)

secondaryのIPアドレス

- ユーザ:「secondaryのIPアドレスを教えて下さい。」

secondaryのIPアドレス: 昔

- 昔はsecondaryの名前に関してゾーン外のデータを書こうとする間違いだった。

secondaryのIPアドレス: 昔(続き)

```
$ORIGIN example.jp.
@   IN SOA ....
       NS  ns.example.jp.
       NS  ns.example9.ad.jp.
ns     A   192.0.2.53
ns.example9.ad.jp. A 198.51.100.1
```

- これを書こうとしている。
- authoritativeデータにならない。

secondaryのIPアドレス: 昔(続き)

- 昔はsecondaryの名前に関してゾーン外のデータを書こうとする間違いだった。
  - 不要な設定であることをご説明した。
- 一度開示したIPアドレスは独り歩きしてしまい、renumberするときにトラブルが起こるのが目に見えている。

secondaryのIPアドレス: 今

BroadBand Tower, Inc.

- 今どきは、正当な設定をするためにsecondaryのIPアドレスが必要なケースもある。
- アクセス制限
  - primaryのallow-transfer{} (に相当する設定)
- 内部名でのNS

- もはやsecondaryのIPアドレスは死守すべきマジックナンバ。

secondaryって?

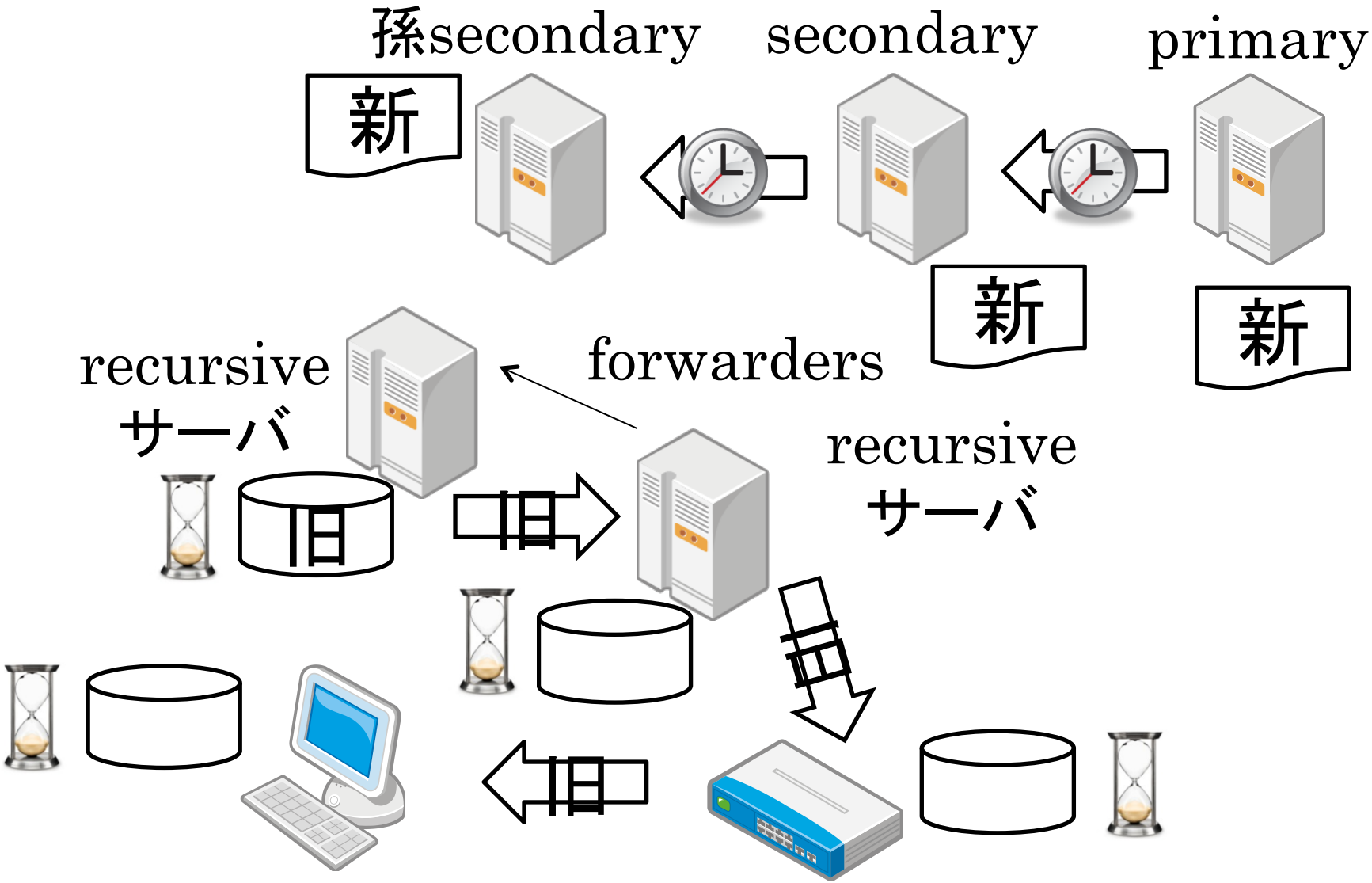
- 誤: recursiveサーバは、まずprimaryに問い合わせて、ダメならはじめてsecondaryに問い合わせる。
- 正: recursive queryの動作では、primaryとsecondaryは対等。
  - recursiveサーバが参照するのはNSだが、NSを見てもprimaryかsecondaryかはわからない。
  - SOAのmnameを見るのは、NOTIFYとdynamic update。
- 正: primaryが正常に動いていても、secondaryからデータを受け取ることもある。
  - 順序は規定されていない。

Summer Daysだし怪談「浸透するDNS」

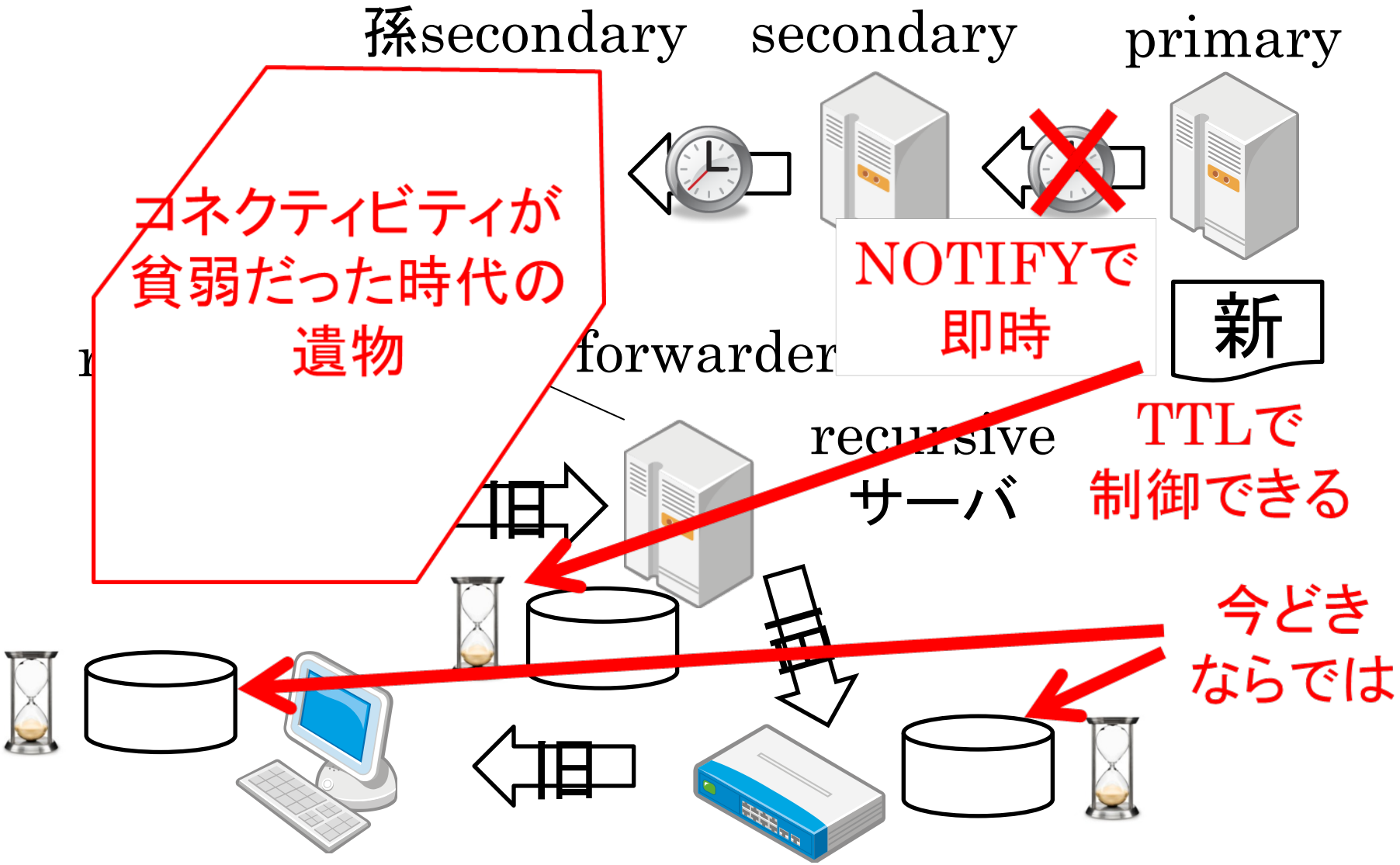
BroadBand Tower, Inc.

- 「浸透」って?
- 確かに伝搬遅延を伴う場面はある。
  - 誤: 人事を尽くして浸透を待つ。
  - 正: 主に自分が世間様のキャッシュたちにバラ撒いてしまった旧データの賞味期限切れ待ち。
- DNS以外の原因による遅延。

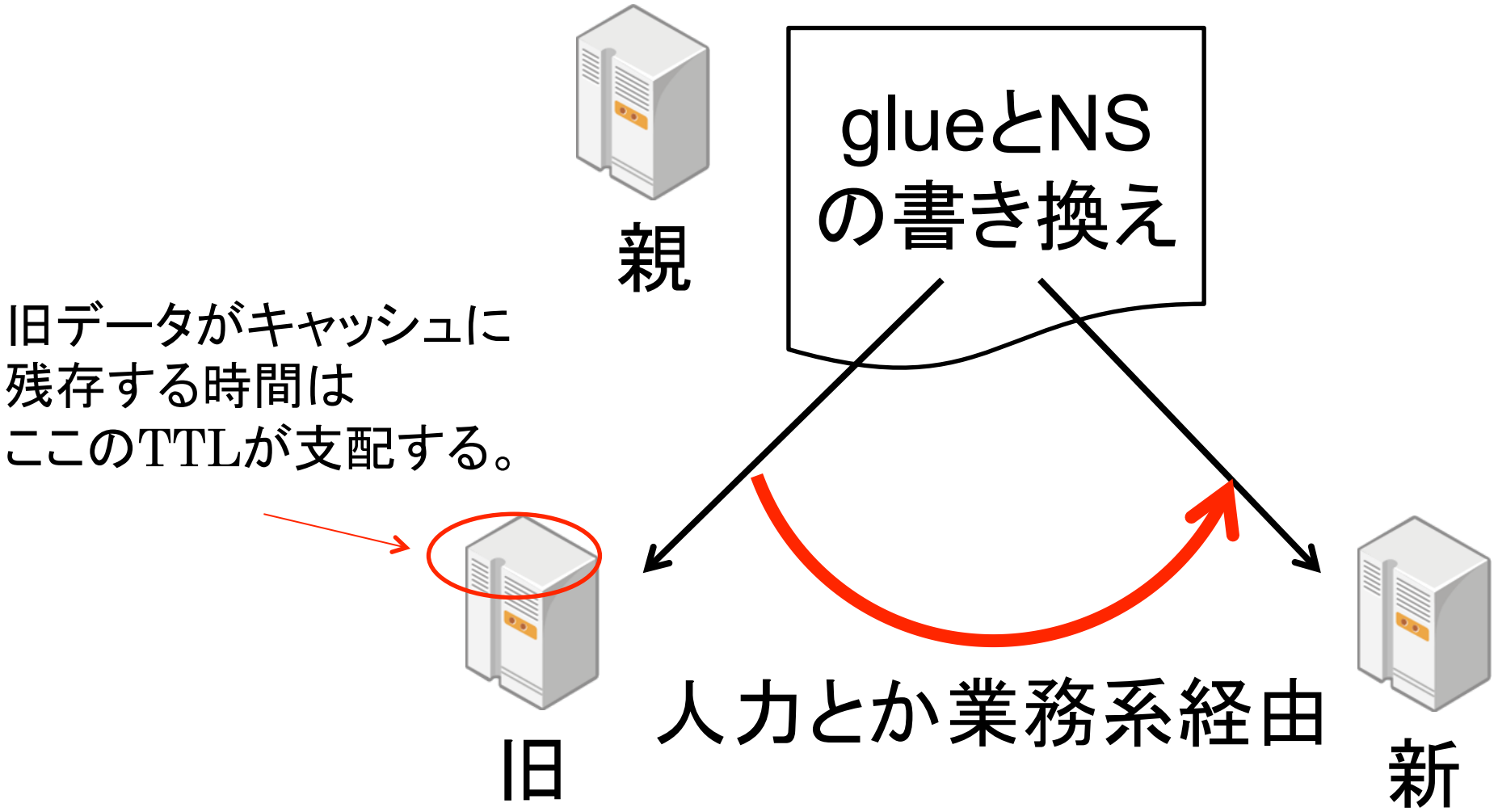
確かに伝搬遅延を伴う場面はあるが…



伝搬遅延を斬る!


孫secondary
secondary
primary
コネクティビティが
貧弱だった時代の
遺物
NOTIFYで
即時
新
forwarder
recursive
サーバ
TTLで
制御できる
旧
旧
今どき
ならでは

DNS以外の原因による遅延



親ゾーンでのglueとNSの書き換え

- 自営サーバがオフィスなどにあり、対外接続を並行運用なしで切り替える場合
  - 昔はよくあったが、今ではレアケース?
  - 技巧をこらし、かつsecondaryが協力的なら、かなり詰められる。
  - 「secondaryが協力的」の部分が望みにくい。
- レンタルサーバなどで並行運用できる場合
  - Web等は、DNSの切り替えが終わるまで、旧でも均質なサービスを提供し続ける。
  - 新が動き出したら、旧でも新のA/AAAAを提供する。→親での書き換え。
    - 旧のアドレスをキャッシュに撒かない。

zone cut、2種類のNS

- zone cutに現れる2種類のNS RR
  - 親側
    - 子ゾーンのauthorityを委任する先のネームサーバの提示。
    - 暗にそのサブドメインが独立したゾーンであることも表現している。
    - '.'から、そのNS RRのRDATAへの連鎖を確立するために必要であれば、glueとしてA/AAAAを伴う。
    - 親側のNSとglueはauthoritativeデータではない。
  - 子側
    - zone apexにNS RRを対応付けることにより、そのゾーンのauthorityの所在を表現している。
    - authoritativeデータである。

zone cut、2種類のNS(続き)

BroadBand Tower, Inc.

- rootから順に委任をたどるとき
  - 子が返すauthoritativeデータに出会うまでは止まらない。
  - authoritative answerに由来するデータがキャッシュに載っていたら、親が返してきたnon-authoritativeデータに出会っても捨てる。
    - 親ゾーンでNSやglueを書き換えても、キャッシュに残存している旧データは強い。
- 詳しくはRFC 2181 5.4.1. Ranking dataに載っている。

関連RFC

BroadBand Tower, Inc.

- RFC 2181: Clarifications to the DNS Specification

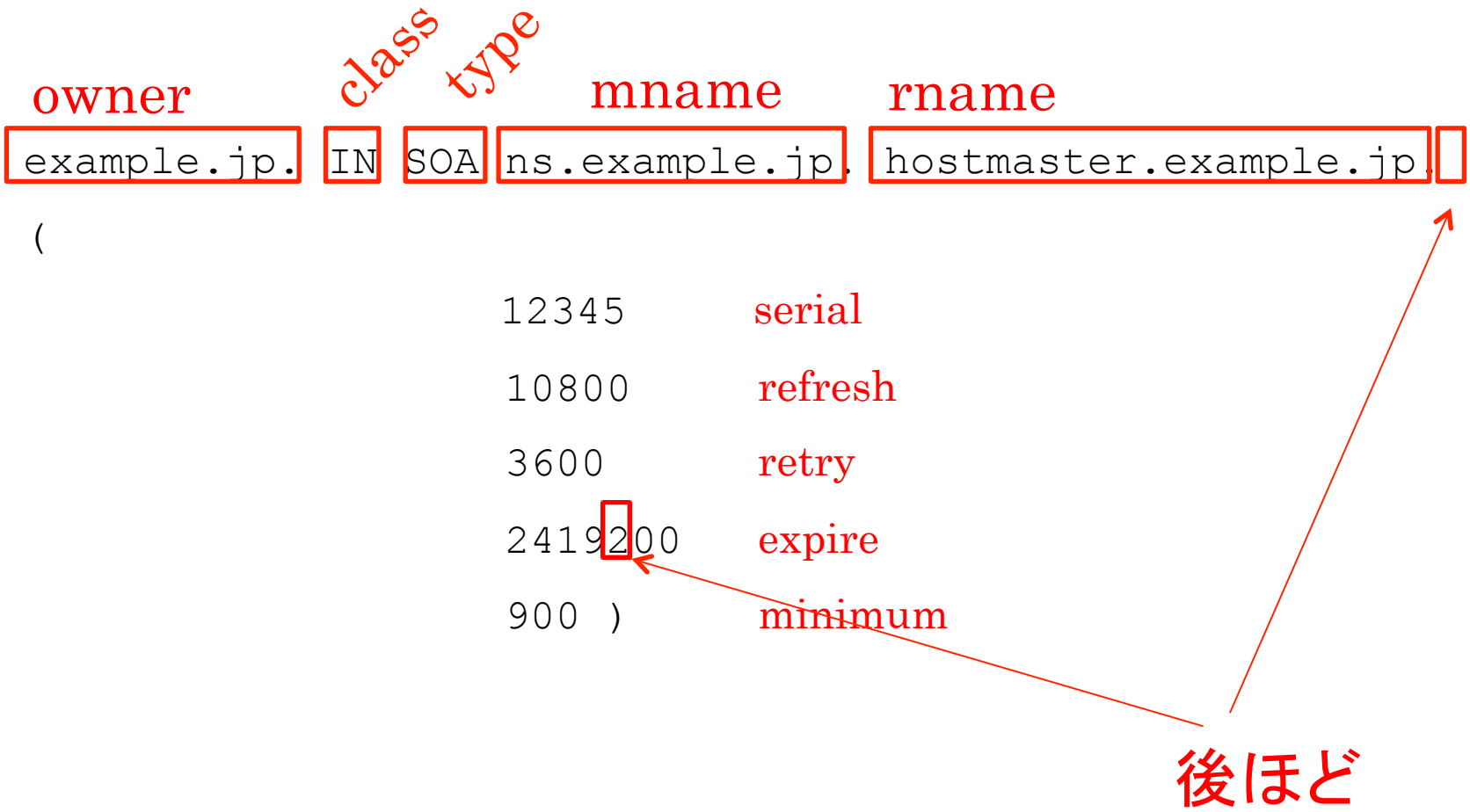
dual stack

- dual stackのrecursiveサーバでは、クライアントから受けたqueryと、それを解決するためにrootから順にrecursive queryするときのアドレスファミリとは、必ずしも一致しない。
- Aやv4のPTRがv6トランスポートでやりとりされることもあるし、AAAAやv6のPTRがv4トランスポートでやりとりされることもある。
  - BINDのAAAA filterは特別な仕様あり。

関連RFC

- RFC 4472: Operational Considerations and Issues with IPv6 DNS

SOAのおさらい

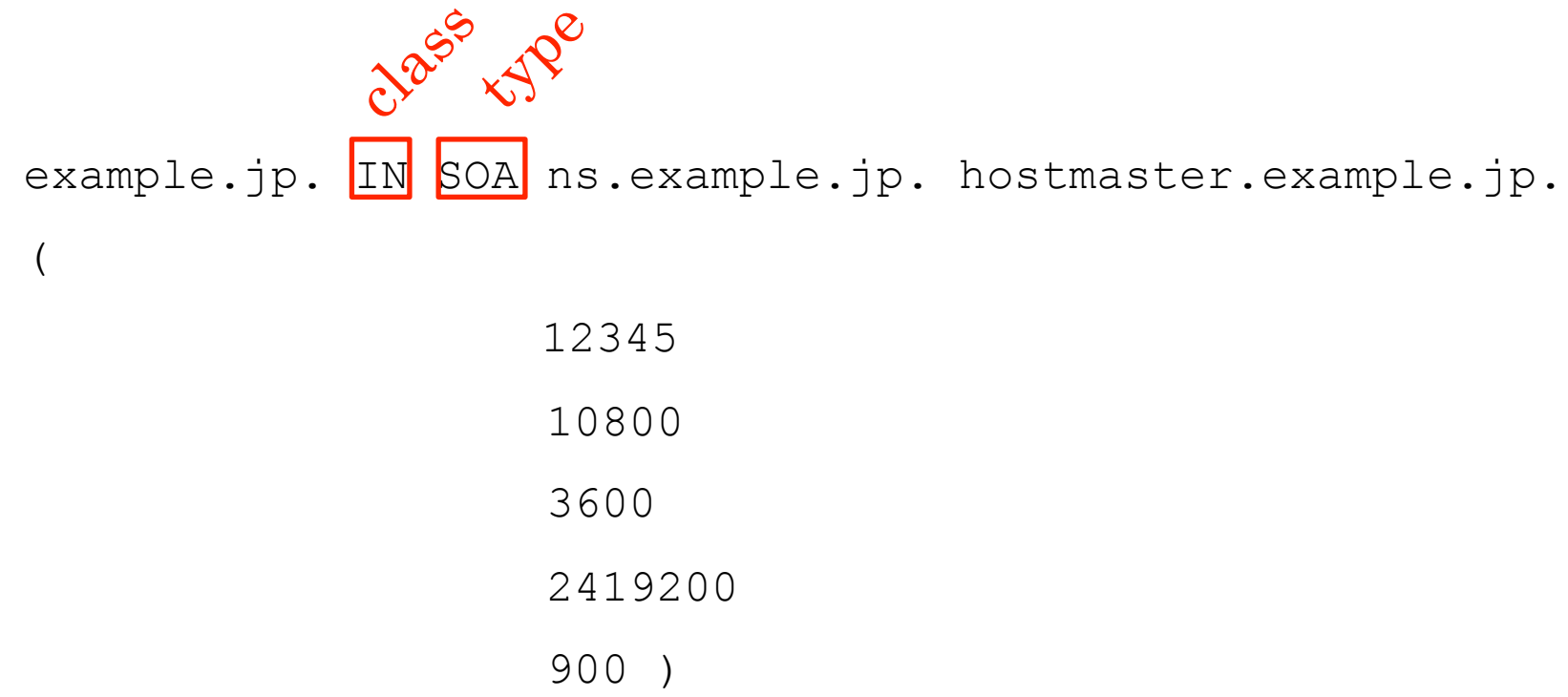

owner
class
type
mname
rname
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
12345 serial
10800 refresh
3600 retry
2419200 expire
900) minimum
後ほど

SOAのおさらい(続き)

BroadBand Tower, Inc.

- owner
  - ownerはRRの一部ではない。
  - RRが対応付けられる名前。
  - SOAはzone apexに対応付いていなければならない。
  - たいていの設定例で'@'が書いてあるが、zone apexを指せば表現は自由。

SOAのおさらい(続き)


class
type
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
12345
10800
3600
2419200
900)

SOAのおさらい(続き)

- class
  - IN
  - HSとかCHとかを使う機会は、まずないでしょう。
- type
  - SOA…の話をしてるところですよね、今。

SOAのおさらい(続き)

mname

```
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
                        12345
                        10800
                        3600
                        2419200
                        900 )
```

SOAのおさらい(続き)

- mname
  - そのゾーンデータの原本のありか = primary
  - dynamic updateはmnameめがけて飛んでくる。
    - dynamic updateを使っているつもりはなくても、DHCPとの連携で飛んでいることもある。
    - jpゾーンのz.dns.jpは、不正なdynamic updateを吸い込むためにNS RRには登場しないサーバが意図的に設定されている。
  - NOTIFYはmnameには送らない。

SOAのおさらい(続き)

rname

```
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
                        12345
                        10800
                        3600
                        2419200
                        900 )
```

SOAのおさらい(続き)

- rname
  - そのゾーンに関する連絡先
  - ちゃんと届くアドレスを書くべき…だったが、harvestingのことを考えると、どうしたものか…
  - メールアドレスには必ず'@'が登場するが、ゾーンデータでは'@'はoriginを意味する→'.'に書き換える。
    - '@'の前に登場する'.'は'\.'に書き換える。

Erai.Hito@example.jp

↓

Erai\.Hito.example.jp.

末尾の'.'を忘れずに or 相対表記

SOAのおさらい(続き)

BroadBand Tower, Inc.

- hostmaster
  - RFC 2142に載っています。

7. DOMAIN NAME SERVICE ADMINISTRATION MAILBOX

In DNS (see [RFC1033], [RFC1034] and [RFC1035]), the Start Of Authority record (SOA RR) has a field for specifying the mailbox name of the zone's administrator.

:

:

For simplicity and regularity, it is strongly recommended that the well known mailbox name HOSTMASTER always be used <HOSTMASTER@domain>.

関連RFC

- RFC 2142: MAILBOX NAMES FOR COMMON SERVICES, ROLES AND FUNCTIONS

SOAのおさらい(続き)

```
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
                        12345       serial
                        10800
                        3600
                        2419200
                        900 )
```

SOAのおさらい(続き)

BroadBand Tower, Inc.

- serial
  - secondaryが、masterのゾーンデータが更新されたかどうかを判断する材料。
  - secondaryはrefreshの間隔でmasterにSOAをqueryし、masterからの応答のserialが自分の手許にあるゾーンデータより大きければ、ゾーン転送を要求して新しいゾーンデータを得る。
    - NSDはいきなりIXFRを要求する。
  - 大事なのは増えること。*YYYYMMDDXX*という流儀をよく見るが、技術的必然性はない。
  - ツールでゾーンデータを自動生成するときには、unixtimeもよく使われる。

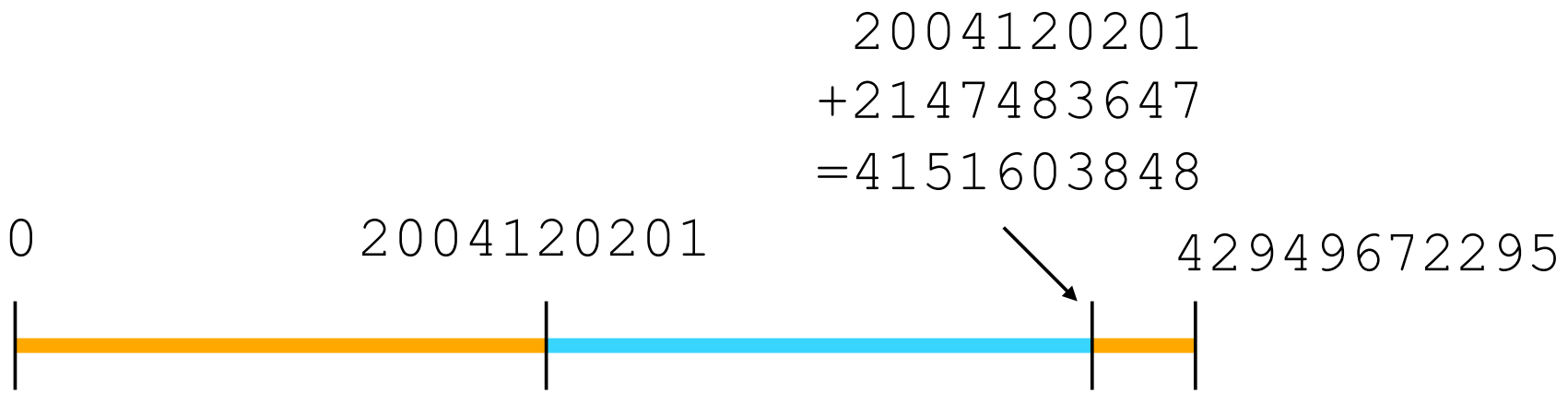
SOAのおさらい(続き)

BroadBand Tower, Inc.

- よくある失敗
  - ゾーンデータを変更したのにserialを増やすのを忘れた。
    - primaryのデータを更新したのにsecondaryに伝搬しない→食い違い
  - エディタでゾーンファイルを開いたら、とにかく最初にserialを増やす習慣付け。

serial, comin' back!

- serial
  - 32bit
  - 0～4,294,967,295=(2^32)−1
  - ある数nから次の2,147,483,647(=(2^31)−1)個の数はnより大きい。
  - nの前2,147,483,647個の数はnより小さい。
  - 4,294,967,295の次は0。



serial, comin' back!(続き)

BroadBand Tower, Inc.

- 4294967295より大きな値だとエラーになる。

```
Sep  4 14:43:18 ns named[35341]: general: error: dns_rdata_fromtext:
    localhost:3: near '4294967297': out of range
Sep  4 14:43:18 ns named[35341]: general: error: zone localhost/IN: loading
    master file localhost: out of range
```

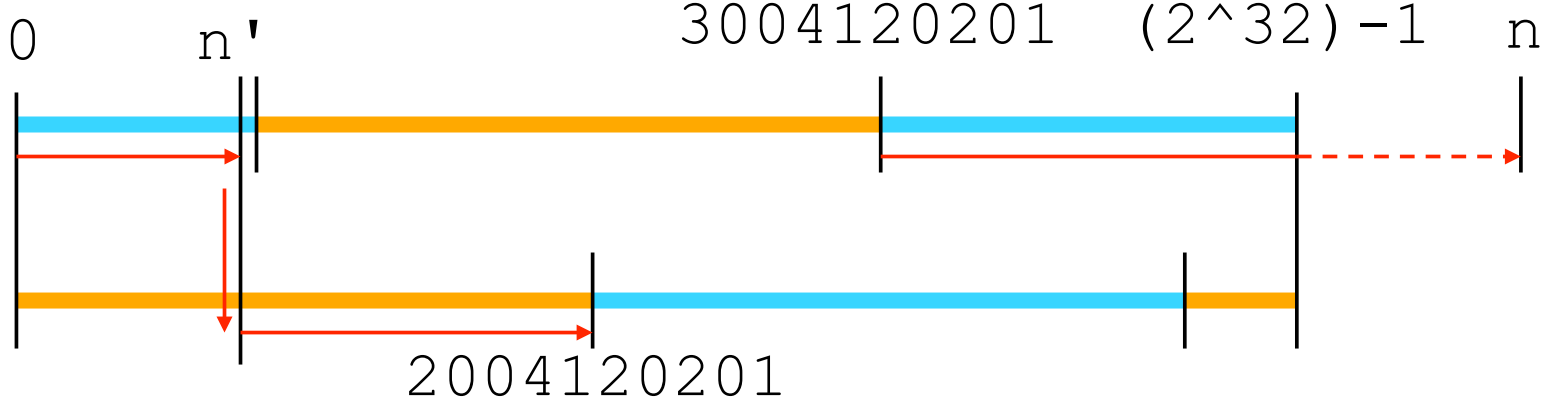
- 2004120201に設定したかったのに3004120201とタイプしてしまい、reloadしてから気付いた ;_;
  - 3004120201より「大きく」2004120201より「小さな」番号に設定
  - secondaryにゾーン転送
  - 2004120201に設定
  - secondaryにゾーン転送

serial, comin' back!(続き)

- n=3004120201+(2^31)−1=5151603848>2^32
- n'=n−2^32=856636552

  3004120201 < n'

  n' < 2004120201



関連RFC

BroadBand Tower, Inc.

- RFC 1912: Common DNS Operational and Configuration Errors
- RFC 1982: Serial Number Arithmetic

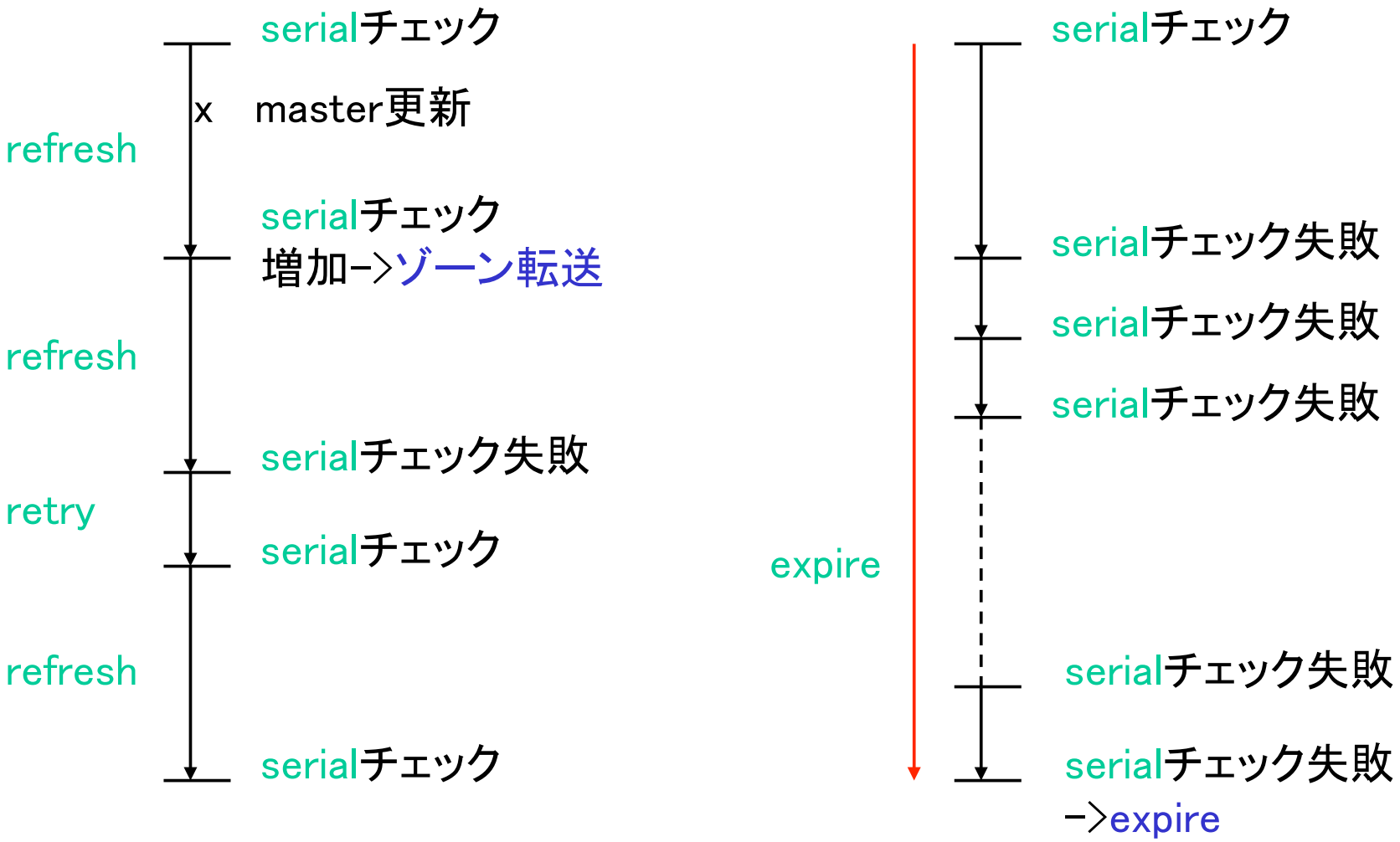
SOAのおさらい(続き)

```
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
                        12345
                        10800       refresh
                        3600        retry
                        2419200     expire
                        900 )
```

SOAのおさらい(続き)

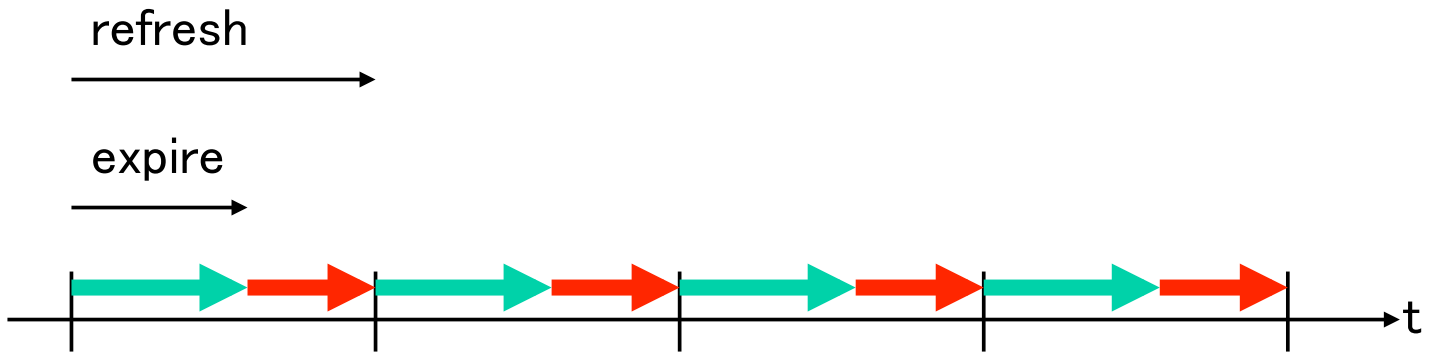
- refresh
  - secondaryがmasterにserialをqueryする間隔。
- retry
  - 前回のrefresh後のqueryが失敗したときにretryするまでの間隔。
- expire
  - retryして、retryして、retryして…も成功しないときに、賞味期限切れとして扱うまでの猶予。

SOAのおさらい(続き)


serialチェック
x master更新
refresh
serialチェック
増加->ゾーン転送
refresh
serialチェック失敗
retry
serialチェック
refresh
serialチェック
expire
serialチェック
serialチェック失敗
serialチェック失敗
serialチェック失敗
serialチェック失敗
serialチェック失敗
->expire

SOAに関する失敗例

- expire < refresh にしてしまった。
  - slaveはrefreshの間隔でmasterの更新をチェックするが、その間に必ずexpireしてしまう。
  - 時の運でslaveが正常に応答したりエラーになったり。



SOAのおさらい(続き)

```
example.jp. IN SOA ns.example.jp. hostmaster.example.jp.
(
                        12345
                        10800
                        3600
                        2419200
                        900 )       minimum
```

SOAのおさらい(続き)

BroadBand Tower, Inc.

- minimum
  - 昔と今とで意味が変わりました!
  - 昔は明示的にTTLを指定していないRRに対するデフォルトのTTLだった。
    - www 86400 IN A 192.0.2.80
  - 今はnegative cacheのTTL。
    - そのゾーンの名前空間の中の名前だが、RRが定義されていないowner/typeを索いてしまったとき。
      - wwww.example.jp (wが4つ!)
      - www.example.jp のAはあるがAAAAがないときにAAAAを索いた。
    - recursiveサーバは、minimumの間はauthoritativeサーバにqueryせずにクライアントに「ないよ」と答えていい。

SOAのおさらい(続き)

BroadBand Tower, Inc.

- 注意
  - SOAのminimumに86400と書くのは、今日では不適切。
    - 86400と書くべきは$TTL
    - RFC 2308には、特に推奨値はないが、例では1200=20分となっている。
    - BIND 9のmax-ncache-ttlのデフォルト値は3h。
  - $TTLを忘れずに。
    - デフォルトのTTLは、今では$TTLディレクティブに書く。
    - ゾーンデータの先頭には必須。SOAより前。
    - ゾーンの途中にも書ける。そこからデフォルト値が変わる。

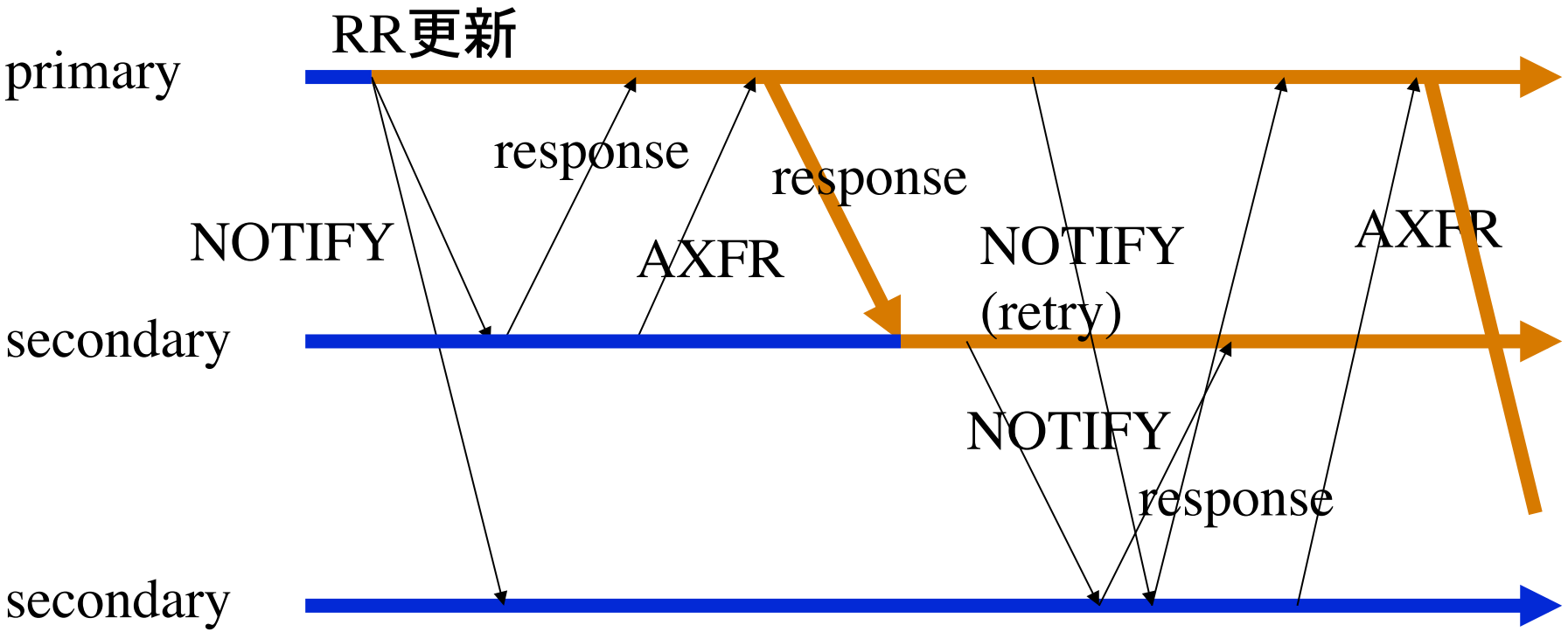
関連RFC

- RFC 1033: DOMAIN ADMINISTRATORS OPERATIONS GUIDE
- RFC 1912: Common DNS Operational and Configuration Errors
- RFC 2308: Negative Caching of DNS Queries (DNS NCACHE)

NOTIFY

- ゾーンデータ中のRRが更新されたときに、そのことをmasterからslaveに能動的に通知する。
- secondaryがrefreshを待たずにゾーン転送を要求することを期待する。
- とりこぼす可能性もあるので、refreshが要らなくなるわけではないが、念押しの意味合いが強くなった。

NOTIFYを導入すると



retryのdefault: 60秒間隔/max 5回

関連RFC

- RFC1996: A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)

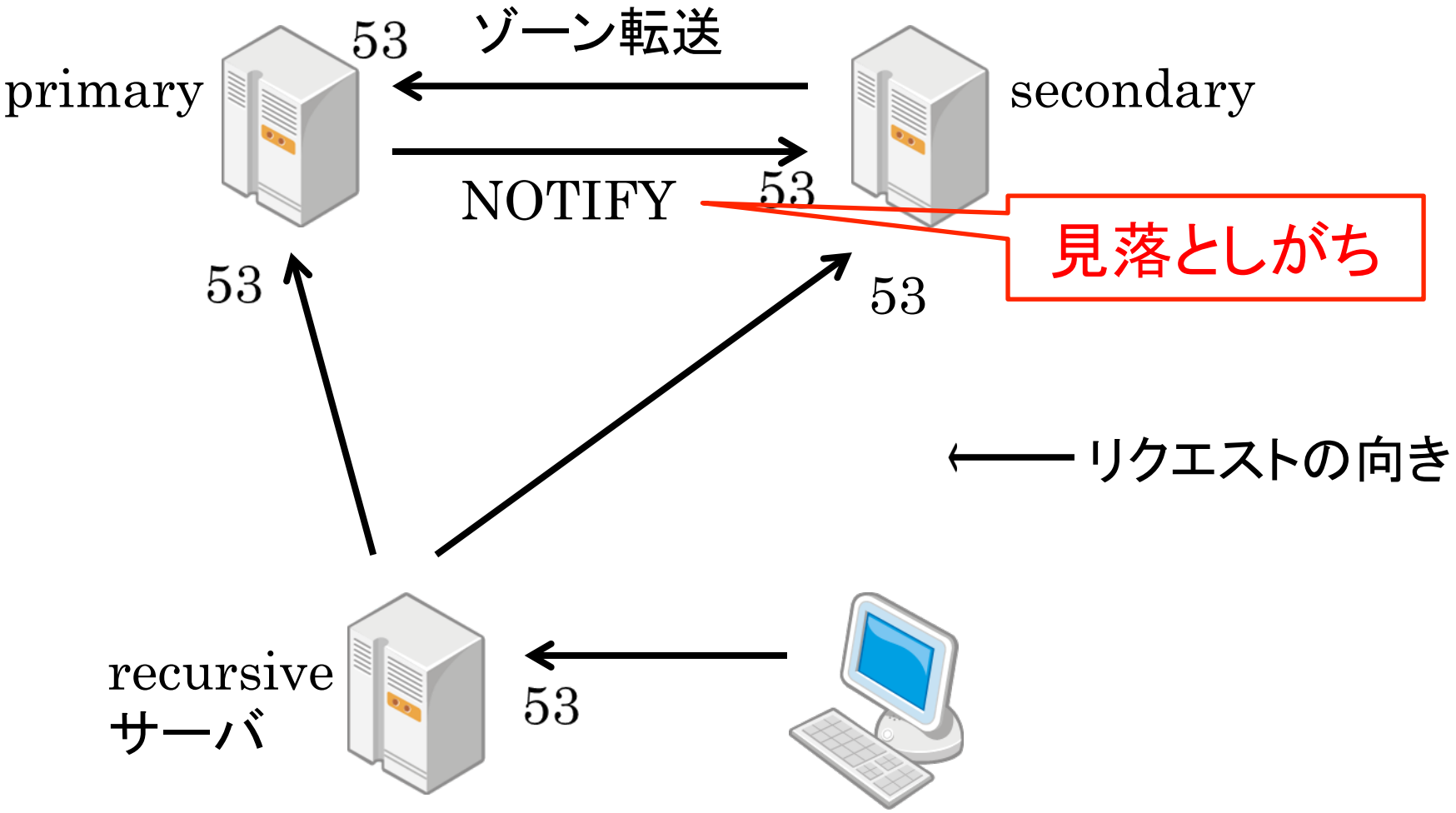
トランスポート層

- primary
  - 通常のquery
    - passive open側なので53
  - ゾーン転送
    - リクエストを受ける側=passive open側なので53
  - NOTIFY
    - active open側なので53以外のポート
- secondary
  - ゾーン転送
    - リクエストする側=active open側なので53以外のポート
  - NOTIFY
    - 受けるだけでなく、Notify Set同士もNOTIFYを送り合う。

トランスポート層

- Notify Set
  - NS RRsetに書いてあるサーバ全部
  - だけどmnameに書いてあるの(RFC 1035的にはprimary)は除く
  - ので、整理すると、一般的には全secondary。

トランスポート層(続き)


primary
53
ゾーン転送
secondary
NOTIFY
53
53
53
見落としがち
リクエストの向き
recursive
サーバ
53

DNSが動かない:事例

- お客様からサポート要請
  - 「自分のドメインはアクセスできるが、外がアクセスできない。」
- 回線/ルーティング障害?? すわー大事!!
- 詳しくヒアリングしてみると
  - IPアドレスでアクセスすればコネクティビティはO.K.
  - 中からはauthorityを持っている名前は索けるが、外部の名前が索けない。
  - 外からはそのサーバが索けない。
- 実際にアクセスしてみると
  - '.'のNSが索けない。

DNSが動かない:事例(続き)

BroadBand Tower, Inc.

- 推測
  - named.rootが悪い?
    - 正しく入手したものだった。
    - 外部から索けないことの説明がつかない。
- 結論
  - BIND4時代の知識でファイアウォールで53番同士のパケットしか通してなかった。
    - 使っていたのはBIND8。
    - query-source port 53;を仮に設定してもらい切り分け。

トランスポート層(続き)

- ゾーン転送は最初からTCPを使う。
- それ以外のqueryは、まずUDPを使う。
- でも、パケットサイズがある大きさを越えるとTCPに切り替える。
  - EDNS0(RFC 2671)を使わなければ512 octet
  - EDNS0では、受信できるサイズを相手に通知
- query送出側は不特定のポート番号を使う。
  - BIND 4.xは53<->53だった。
    - 大昔の参考書を読むときは注意。

EDNS0

- EDNS0の拡張を使うと、UDPで送れるデータサイズを拡大できる。
  - requesterはadditional sectionに、自分が受け入れられるUDPのサイズなどを設定したpseudo-RRを入れてqueryを送出する。
  - 対応していないresponderは、NOTIMPL、FORMERR、SERVFAILを返すだろう。
    - そうしたら、EDNS0なしでretry。
    - 自分がEDNS0に対応していても、相手が対応していなければ、EDNS0は使われない。
  - EDNS0に対応しているresponderなら、平然と応答する。
    - パケットサイズの上限はrequesterの要求に沿う。

EDNS0なしでretry

BroadBand Tower, Inc.

ある実装

- まずEDNS0なし
  ↓truncateしたら
- EDNS0を使ってみる
  ↓エラーだったら
- TCP

別の実装

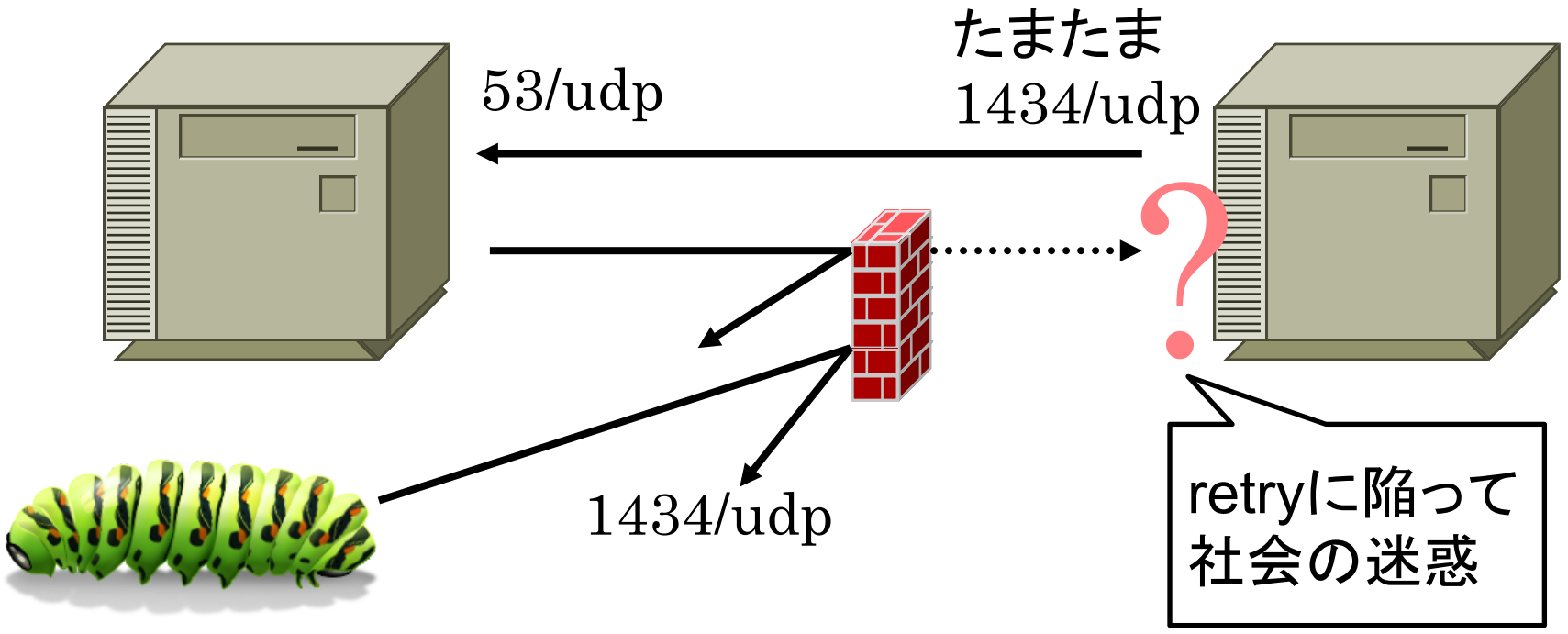
- とにかくEDNS0を使う
  ↓エラーだったら
- EDNS0なし
  ↓truncateしたら
- TCP

トランスポート層(続き)

- ゾーン転送以外にもTCPが使われることはある。
  - 誤: パケットフィルタでTCPはsecondaryだけに許可。
  - 正: ゾーン転送のアクセス制限はアプリケーション層で。
- フラグメントは大丈夫ですか?
  - ブロードバンドルータなど、宅内小箱たち。

トランスポート層(続き)

- アタック、warmに対する防御のために特定ポート宛のパケットをフィルタリングするときに、DNSパケットを巻き添えにしないように。
  - ソースポートが53番だったら、DNSパケットの可能性あり。



関連RFC

- RFC 1035: Domain names - implementation and specification
- RFC 2671: Extension Mechanisms for DNS (EDNS0)
- RFC 5966: DNS Transport over TCP - Implementation Requirements

lame delegation

BroadBand Tower, Inc.

- 正しい委任とは
  - 親がNS RR(とglue)で委任を提示している先のネームサーバが、そのゾーンのauthoritative answerを返す。
  - 子がNS RRでauthorityの所在を主張しているネームサーバが、そのゾーンのauthoritative answerを返す。
  - それぞれのネームサーバが同じ応答を返す。
  - キャッシュ上のデータを考慮しても、これらが成立する。
- lame delegationとは
  - 正しい委任が成立していないこと。

lame delegation(続き)

- 例えばauthoritativeサーバを変更するとき
  - 旧サーバが提供したデータは、最長でTTLの間は世の中のキャッシュに載り続ける。
  - これは親ゾーンの委任情報に由来するデータより強い。
  - だから、親ゾーンの委任情報を更新したから、といって、旧サーバが旧データを提供し続けたり、逆にすぐ停止してしまったら、TTLが満了するまでlame delegationが発生し得る。

lame delegation(続き)

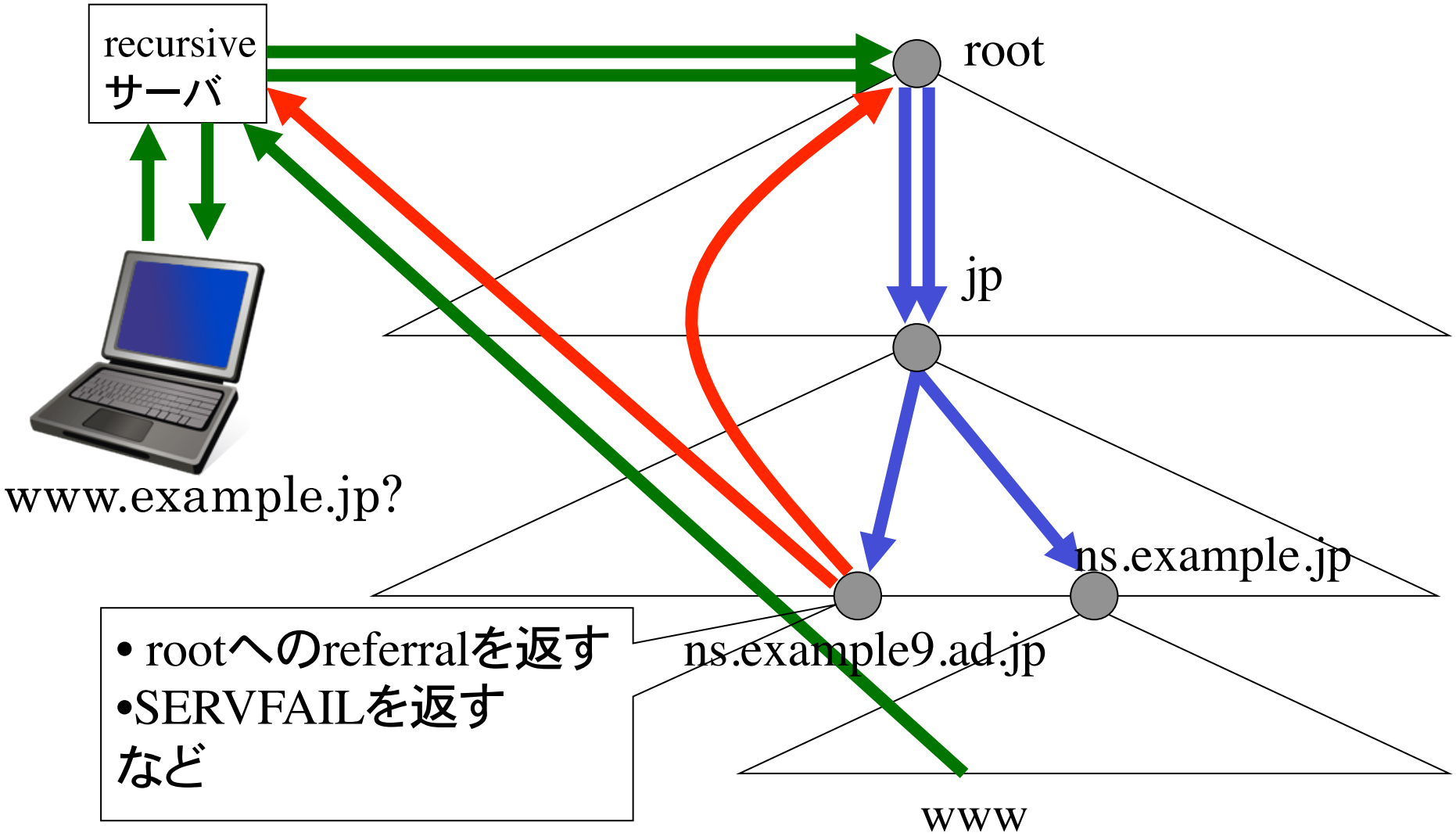
- 複数のauthoritativeサーバがあるとき
  - 一部が応答しないこと自体は異常ではない。
    - それを認めなきゃ、何のためのsecondaryなんだか。
  - 応答の不一致も、即、異常ではない。
    - primary→secondaryの伝播遅延。
  - 過渡的ではなく継続的だと異常。
  - ウソを答える奴が混じっていてはいけない。

lame delegation(続き)

```
example.jp.   IN   NS   ns.example.jp.
                   NS   ns.example9.ad.jp.
```

- ns.example.jp
  - example.jpのauthorityを持っている。
- ns.example9.ad.jp
  - example.jpのauthorityを持っていない。

何が起こる? :非効率編


recursive
サーバ
root
jp
www.example.jp?
ns.example.jp
ns.example9.ad.jp
www
• rootへのreferralを返す
•SERVFAILを返す
など

なぜ起こる?

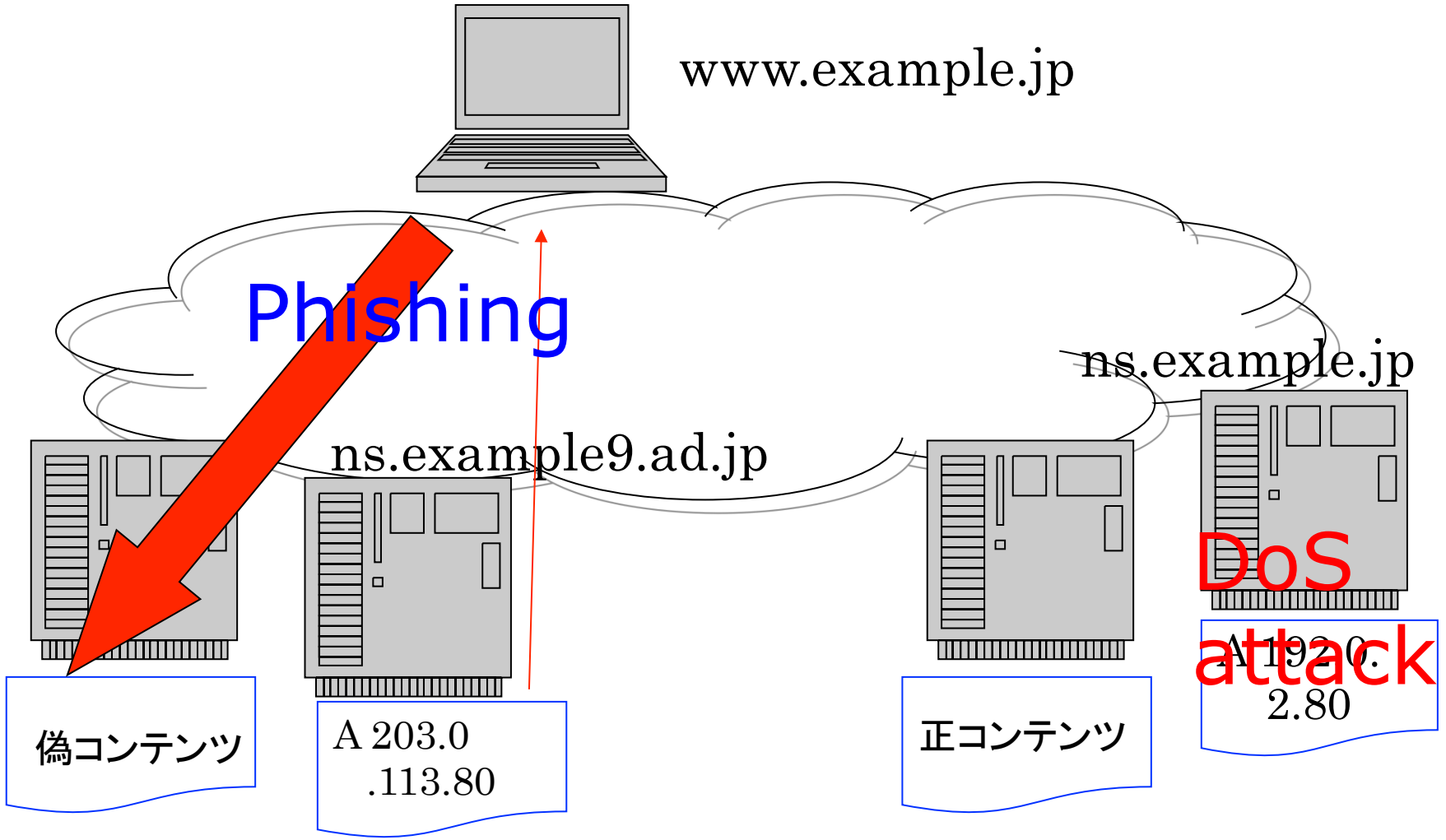
- authorityを持っているはずのネームサーバがauthorityを失くした。
  - 設定ミス
  - secondaryがexpireした。
- authorityを持っていないネームサーバにauthorityを委任する設定/手続きをしてしまった。
  - xSPにsecondaryを依頼するときの手続きミス。
- ISPを替えたのに、レジストリの登録を変更し忘れた。
  - 旧ISPは解約されたので設定を消した。
  - primaryもrenumber…→glueとの食い違い。
- etc,etc…

何が起こる? :ヤバい編

```
example.jp.   IN   NS   ns.example.jp.
                   NS   ns.example9.ad.jp.
```

- example9.ad.jpドメインが☆◎※な理由で消滅。
  - ☆◎※には「買収」、「事業撤退」などお好きな言葉を当てはめて下さい。
- lame delegationが発生。
- 別の組織がexample9.ad.jpというドメイン名を登録して、ns.example9.ad.jpという名前でネームサービスを提供。
- 悪意の有無は別としてトラブルの元。

何が起こる? :ヤバい編 (続き)



ゾーンデータ寄りの話

コメント記号

- コメント記号は';'
  - '#'はコメント記号ではない。
    - UNIXの常識に反している。
      - DNSサーバの最初の実装、JeevesはUNIXではなくTOPS-20で開発された。
    - named.confと不統一でまぎらわしい。

相対表記に関する失敗例

- 名前の末尾に‘.’を忘れる。

```
$ORIGIN example.jp.
  @   IN   SOA …(…)
      IN   NS   ns.example.jp
      IN   NS   ns.example9.ad.jp
```

は

```
@   IN   SOA …(…)
    IN   NS   ns.example.jp.example.jp.
    IN   NS   ns.example9.ad.jp.example.jp.
```

に展開されてしまう。

– NSじゃなくってPTRで忘れると、tracerouteの結果が恥ずかしいことになる。

相対表記に関する失敗例 (続き)

BroadBand Tower, Inc.

- tips
  - 相対表記、絶対表記に関しては、自分の流儀を決めておく。
    - 相対表記は使わない。必ず絶対表記。
    - ownerは必ず相対表記、RDATAは必ず絶対表記。
    - 相対表記できるところは必ずする。
      - etc
  - 設定ミスを見つけやすくなる。

ドメイン名の制約

- 文字種
  - RFC 1035に、'host name'のlabelは
    - アルファベットで始まり
    - アルファベット、数字、’-’ (ハイフン)の繰り返し
    - アルファベットまたは数字で終わる

    のが無難だろう、という意味のことが書いてある。
  - RFC 1123で1文字目が数字のドメイン名もよいことになった。
    - 3com.com、0123.co.jp、…
  - ありがちな間違い: 'host name'に’_’を使ってしまう。
    - SRV RRなど'host name'じゃない名前はいいと解釈されている。

関連RFC

- RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION
- RFC 1123: Requirements for Internet Hosts - Application and Support

してはいけないこと

- CNAMEを定義したownerに対して、他のRRを定義してはいけない。

```
www   IN   CNAME   server156
           MX      10   server154
```

はダメ。

してはいけないこと(続き)

- user@example.jpというメールアドレスを使いたい。

```
$ORIGIN example.jp.
@   IN   SOA     ns   hostmaster (
                 2010060801
                 20m
                 15m
                 4w
                 15m)
          NS     ns
          CNAME  server154 ; MXを書かなきゃダメ
```

してはいけないこと(続き)

- 2つ目のCNAMEもダメ

```
www    IN    CNAME    backend1
             CNAME    backend2
```

してはいけないこと(続き)

- NSやMXのRDATAに、CNAMEで定義したaliasを書いてはいけない。
  - RFC 974には、よくない、という趣旨のことが書いてある。
  - RFC 2181にはmust not be an aliasと書いてある。

```
@    IN NS     ns
ns      CNAME cabernet-sauvegnon
```

はダメ

しない方がいいこと

- CNAMEのCNAMEは避ける。
  - 循環参照回避

```
alias1   IN   CNAME   alias2
alias2        CNAME   alias3
alias3        CNAME   alias1
```

しない方がいいこと(続き)

- RFCでは禁止していないが、逆にxx段まで動作すること、というような記述もない。
- unboundでは8段、BIND 9では16段で正規化を打ち切る。
- NSDは実行時ではなくゾーンコンパイラにかける時点でチェックしているので、循環参照さえなければ、いくらでもいける模様。
- 1段でダメな実装はRFC違反だが、2段でダメでもRFC違反ではない。
- 自サイトのauthoritativeサーバだけでなく、相手サイトのrecursiveサーバにも依存。
  - 「うちはBIND 9だから16段まで大丈夫」ではなく「unboundに索かれたら9段でアウト」。

内部名、外部名、脅威の連鎖

- 内部名
  - そのゾーン(例: example.jp)の中の名前。
    - ns2.example.jp
    - www3.example.jp
  - 下位のゾーンでも、別ゾーンの中の名前は外部名。
    - ~~ns.subdom.example.jp~~
- 外部名
  - 内部名じゃない名前。
  - www.example9.ad.jpはexample.jpから見れば外部名。
  - ~~ns.example9.ad.jpはjpから見れば外部名。~~

削除: 親にglueが含まれるので内部名になる。
(当日、森下さんから指摘あり)

CNAMEをゾーンの外へ向けることの意味

```
$ORIGIN example.jp.
www   IN   CNAME rental800.example9.ad.jp.
```

- www.example.jpの設定内容はexample.jpの管理者には(技術的には)制御できない。
  - example9.ad.jpのauthoritativeサーバが乗っ取られると、WWWコンテンツの差し替えが可能。
- CNAMEについて説明したが、NSについても同じ。

CNAMEをゾーンの外へ向けることの意味(続き)

```
$ORIGIN example.jp.
www   IN   CNAME   rental800.example9.ad.jp.
```

```
$ORIGIN example9.ad.jp.
rental800   IN   A   192.0.2.80
```

↓

```
rental800   IN   A   203.0.113.80
```

192.0.2.80

本物の
コンテンツ

203.0.113.80

偽物の
コンテンツ

RRset中の各RRのTTL

- RRsetの中の各RRのTTLはそろえなければならない。

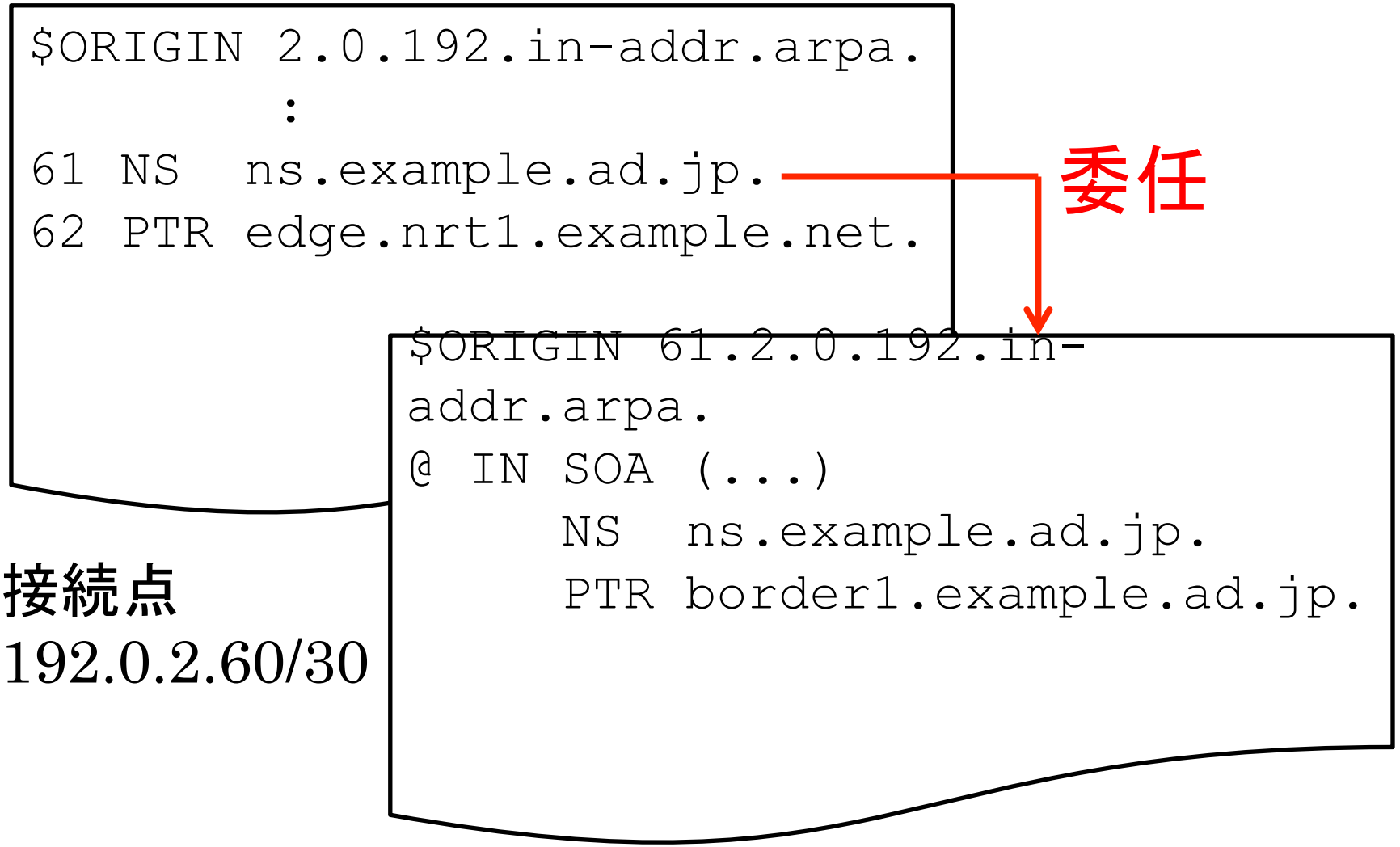
```
www 86400 A 192.0.2.53
    86400 A 192.0.2.153
    60    A 192.0.2.253
```

  はダメ。
- TrunCated flagが立たずに断片的なRRが返ることがあり得る。
- 実装は、RRset中のRR全部をRRset中の最小のTTLとして取り扱うべきである。

関連RFC

- RFC 2181: Clarifications to the DNS Specification

4octetに対応するv4の逆索きゾーン



- 接続点
- 192.0.2.60/30

4octetに対応するv4の逆索きゾーン(続き)

BroadBand Tower, Inc.

- v6なら32nibbleに対応する逆索きゾーン。
- 組織間の接続点で、アドレスのownershipとDNSでのauthorityの所在とを一致させられる。
- 魔術的なところは何もない。ただ、あまりよく知られていないだけ。
- ルータ間リンクのPTRには賛否両論ある。
  - tracerouteの結果に出てくる。
  - Layer 3のトラブルシュートに便利。
  - 網構成をナイショにしたい人たち。
  - xe-3-2-101.border1.example.ad.jpとかすると、「あぁ、Jね」とバレて、セキュリティ的に不安という人たち。

'('と')'の意味と応用

```
@   IN   SOA   ns hostmaster (
               2005120601
               1h
               15m
               4w
               15m )
```

- SOAを記述するのに必ず’(’と’)’が登場する。
- 他のところでは見かけない。

'('と')'の意味と応用(続き)

- でも、’(’と’)’はSOAの構文の一部ではない。
  - Parentheses are used to group data that crosses a line boundary. In effect, line terminations are not recognized within parentheses.
    - RFC1035 5. MASTER FILES; 5.1. Formatより

```
0.f.e.d.c.b.a.9.8.7.6.5.4.3.2.1 (
    IN  PTR  www.example.jp.)
```

なんていう使い方もできる。

おわりに

おわりに…まとめにあらず

BroadBand Tower, Inc.

- DNSについてのよくある間違いを、いくつかご紹介しました。
- 間違いとしては、ケアレスミス、検討が不十分なために起こる誤り、誤解がありました。
- ケアレスミスについては、注意を喚起しました。
- 誤解と、設定例の内容を、よく検討せずに引き写しているために起こる間違いとについては、解説を加えました。
- あまり知られていないと思われる事項の紹介から派生して、tipsをいくつかご紹介しました。

おわりたいけど新たなる迷宮の入り口か？

BroadBand Tower, Inc.

- Orangeさんと伊藤との、ゾーンデータでclassは省略できるのか、という議論より…

  当該部分を読むと、TTLとclassは共にオプションだと書いてあり（下記）、最後に明示的に指定されたものがデフォルトになる（上記＆下記）とあるくせに、少なくとも一つは指定しないといけない、という記述は見つかりませんでしたし、省略した時にどうなるかも書いてないようです。

  ＃何て曖昧な。。。

- ってことで、滝澤さん、後はよろしく～