

初心者のためのDNS運用入門 - トラブル事例とその解決のポイント -

2014年6月26日

DNS Summer Days 2014

株式会社日本レジストリサービス(JPRS)

水野 貴史

講師自己紹介

- 氏名: 水野 貴史 (みずの たかふみ)
- 生年月日: 1988年3月3日 (26歳)
- 所属: 株式会社日本レジストリサービス (JPRS) システム部
- Unix歴: 9年目 (FreeBSD、OS Xを中心に)
- 職歴:
 - 2013年4月 JPRS入社
 - 2013年4月～6月 新人研修
 - 2013年7月 DNS Summer Days 2013講師
 - 2013年8月～ レジストリ基盤開発
 - 2013年11月 Internet Week 2013 DNS DAY「JP DNS UPDATE」
 - **2014年6月 DNS Summer Days 2014講師**

本日の内容

1. DNSの基礎知識とトラブルシューティングの基本
 - DNSの全体構成
 - 区別すべき2種類のDNSサーバー/問い合わせ
 - トラブルシューティングの基本
2. 道具の使い方
 - コマンドラインツールの使い方
 - 便利なWebサービスの紹介
3. よくあるトラブル事例とトラブルシューティング
 - 設定がうまくいかない
 - 名前が引けない
 - 名前を引くのにかかる

ポイントと想定する対象者

- ツールの紹介と使い方
 - コマンドラインツールとWebサービス
 - トラブルシューティングについて、具体例を挙げながら解説

- 対象
 - DNSサーバーをこれから運用される方
 - DNSサーバーの運用を始めて間もない初学技術者の方

そして、

- 初学技術者ではない方々の知識のおさらい、再確認
- 社内セミナーの資料

としても活用可能なものとすることをめざします

まずは、おさらいとして……

1. DNSの基礎知識と トラブルシューティングの基本

区別すべき2種類のDNSサーバ

- DNSには

- 階層構造を**構成する** (分散管理)
- 階層構造を**たどる** (名前解決)

という**2つの役割**がある

- 「DNSサーバ」にはそれぞれの役割を担当する

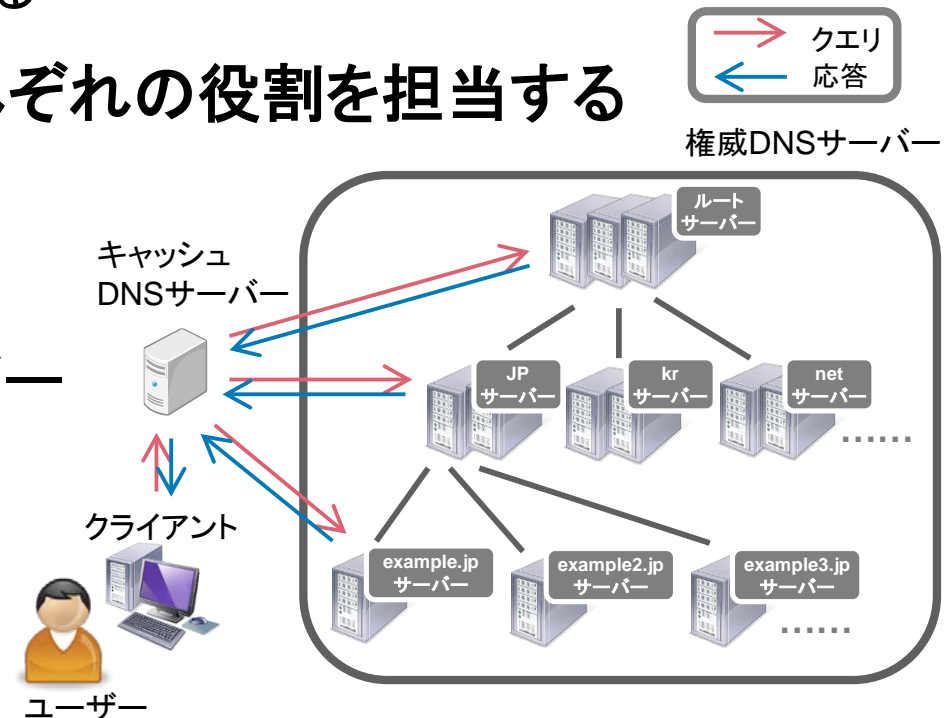
1. 権威DNSサーバ

→ 階層構造を構成

2. キャッシュDNSサーバ

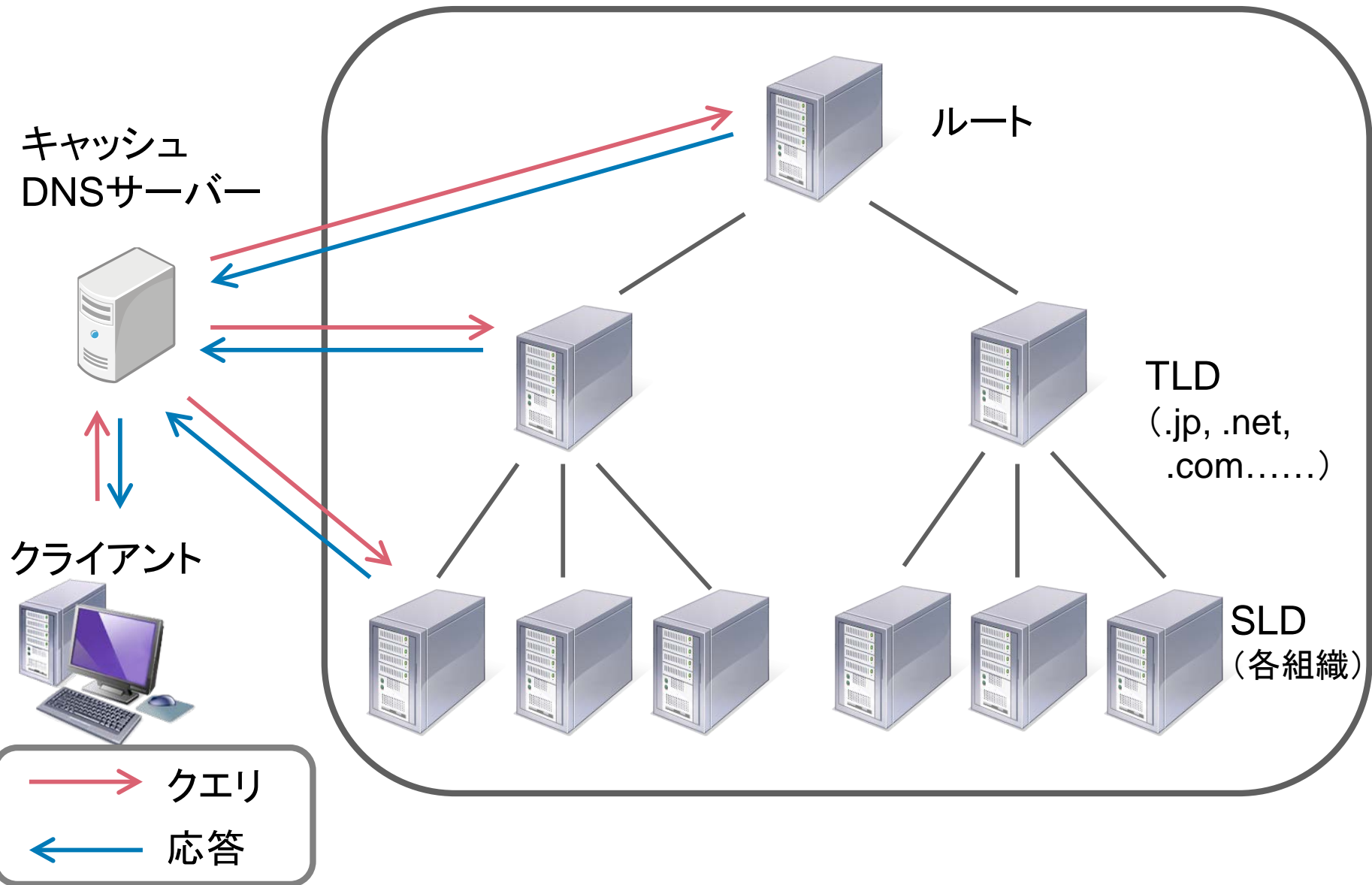
→ 階層構造をたどる

の**2種類**が存在する



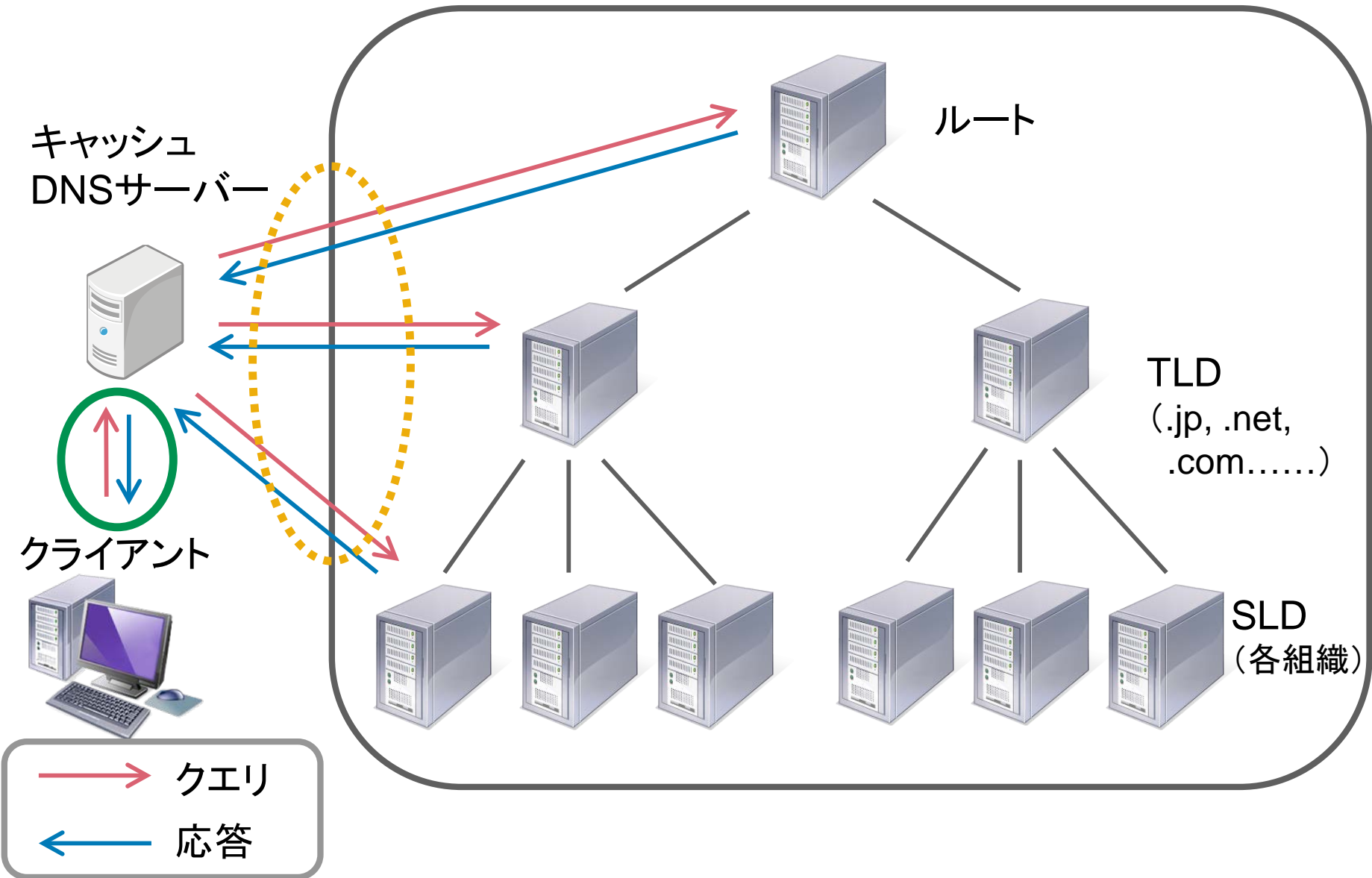
DNSの全体構成

権威DNSサーバー



区別すべき2種類のクエリ

権威DNSサーバー

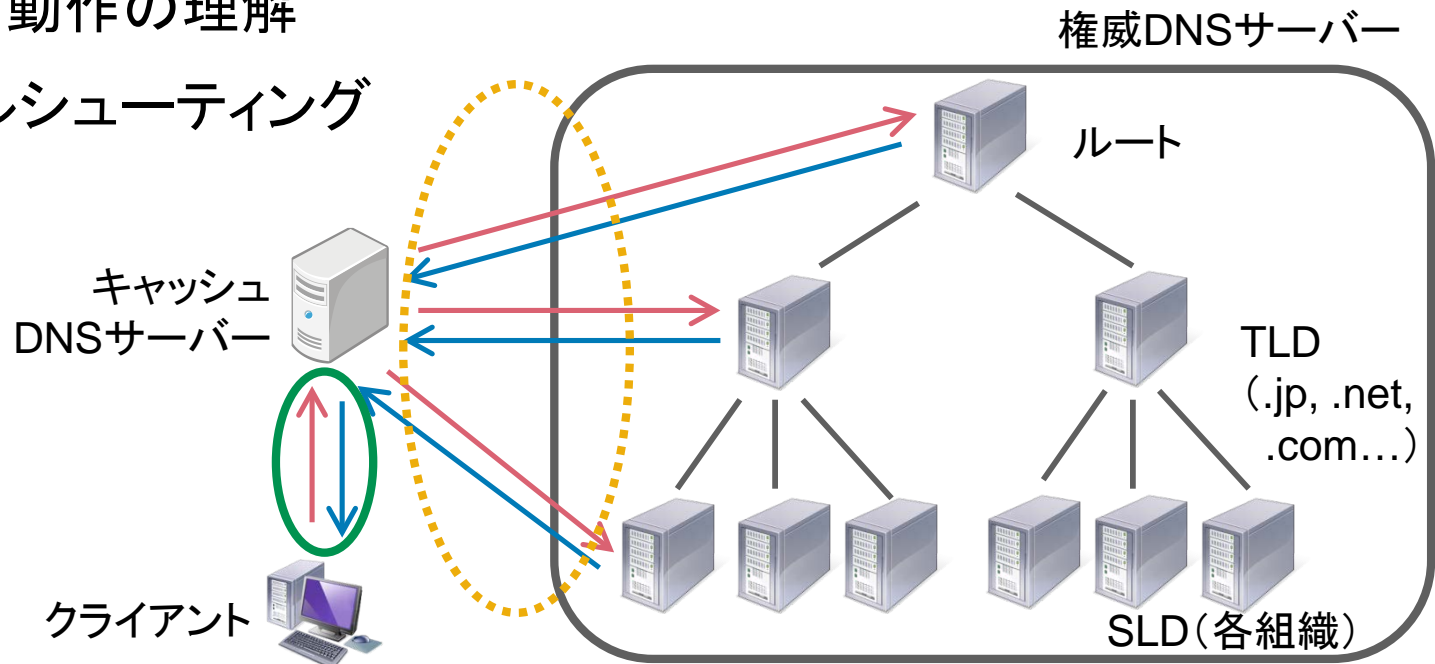


区別すべき2種類のクエリ

- クライアントからキャッシュDNSサーバーへのクエリ
- キャッシュDNSサーバーから権威DNSサーバーへのクエリ

→ この2種類のクエリを明確に区別することがすべての基本

- DNSの動作の理解
- トラブルシューティング



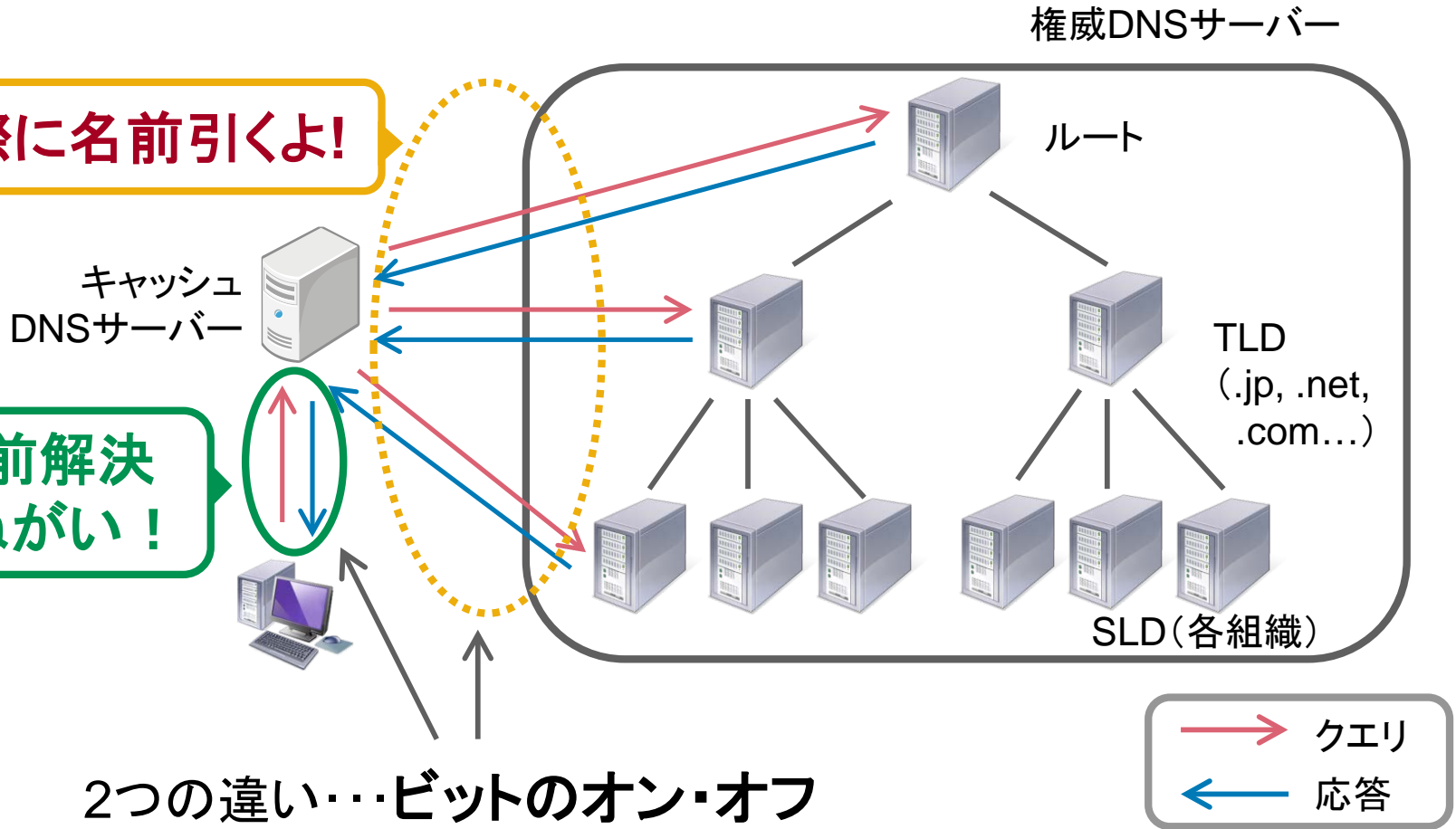
区別すべき2種類のクエリ

実行

実際に名前引くよ!

依頼

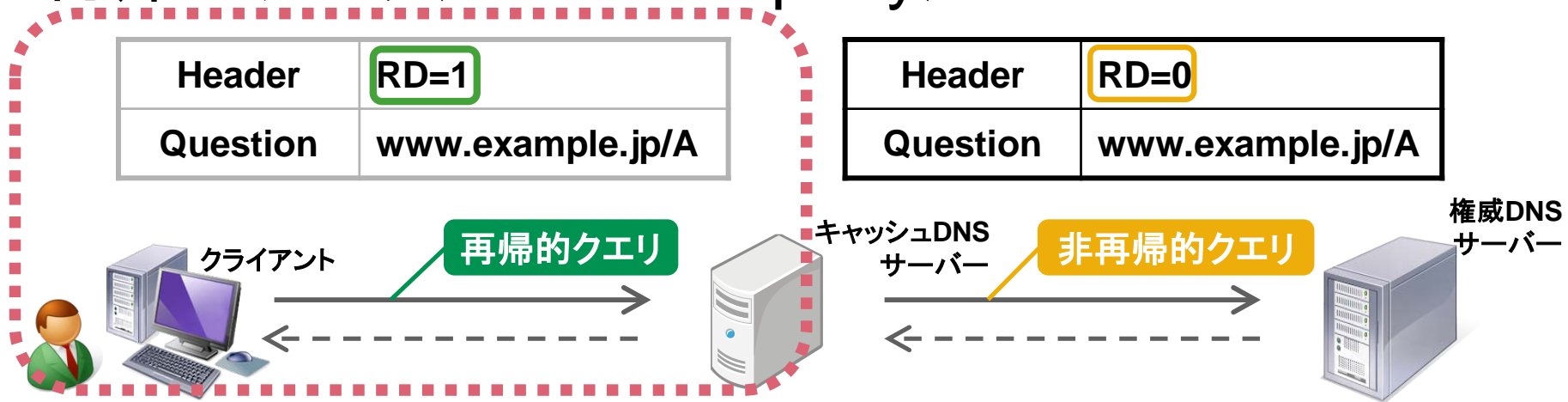
名前解決
おねがい!



2つの違い...ビットのオン・オフ

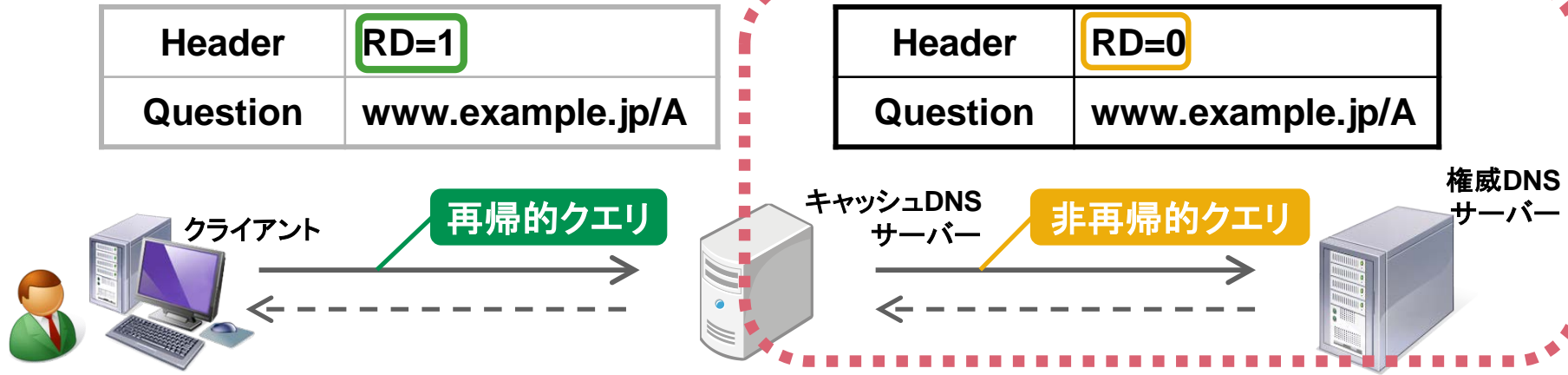
- ✓ 区別しないと、調査の際、問題の切り分けができない
- ✓ どの部分が問題か? どの部分を調べているのか?

再帰的クエリ (recursive query)



- クライアントからキャッシュDNSサーバーへのクエリ
- クエリ中のRDビットがセット**されている**
- クライアントはRDビットをセットしたクエリを送信することにより、キャッシュDNSサーバーに階層構造をたどらせる
 - これを名前解決要求という

非再帰的クエリ (non-recursive query)



- キャッシュDNSサーバーから権威DNSサーバーへのクエリ
 - クエリ中のRDビットがセット**されていない**
- クライアントからの名前解決要求によって発生する
- 再帰的クエリと**同じ内容**がRDビットをクリアした上で送信される

区別すべき2種類のクエリ

権威DNSサーバー

非再帰的クエリ

キャッシュ
DNSサーバー

ルート

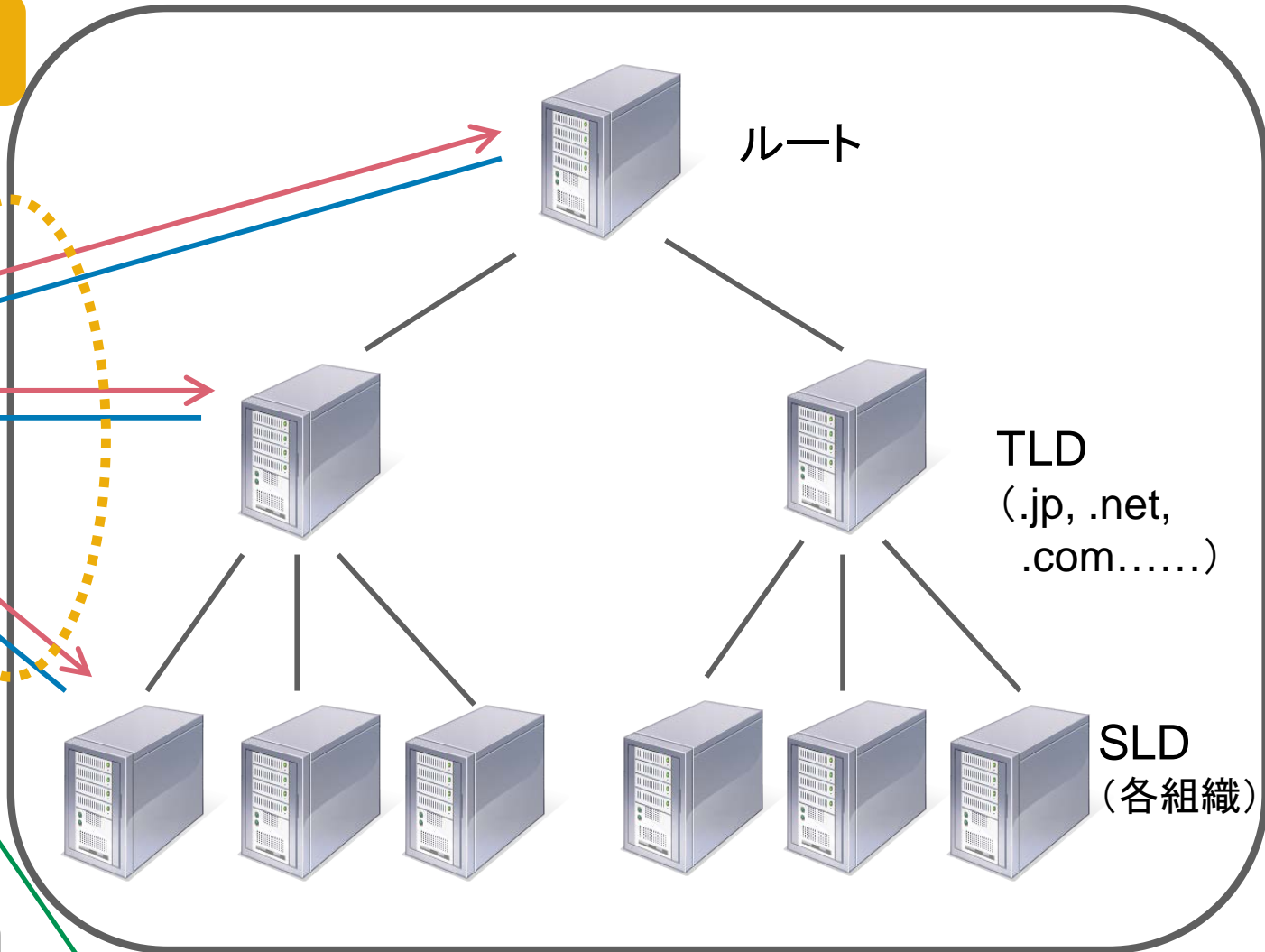
TLD
(.jp, .net,
.com.....)

クライアント

SLD
(各組織)

→ クエリ
← 応答

再帰的クエリ



DNSの基礎知識まとめ

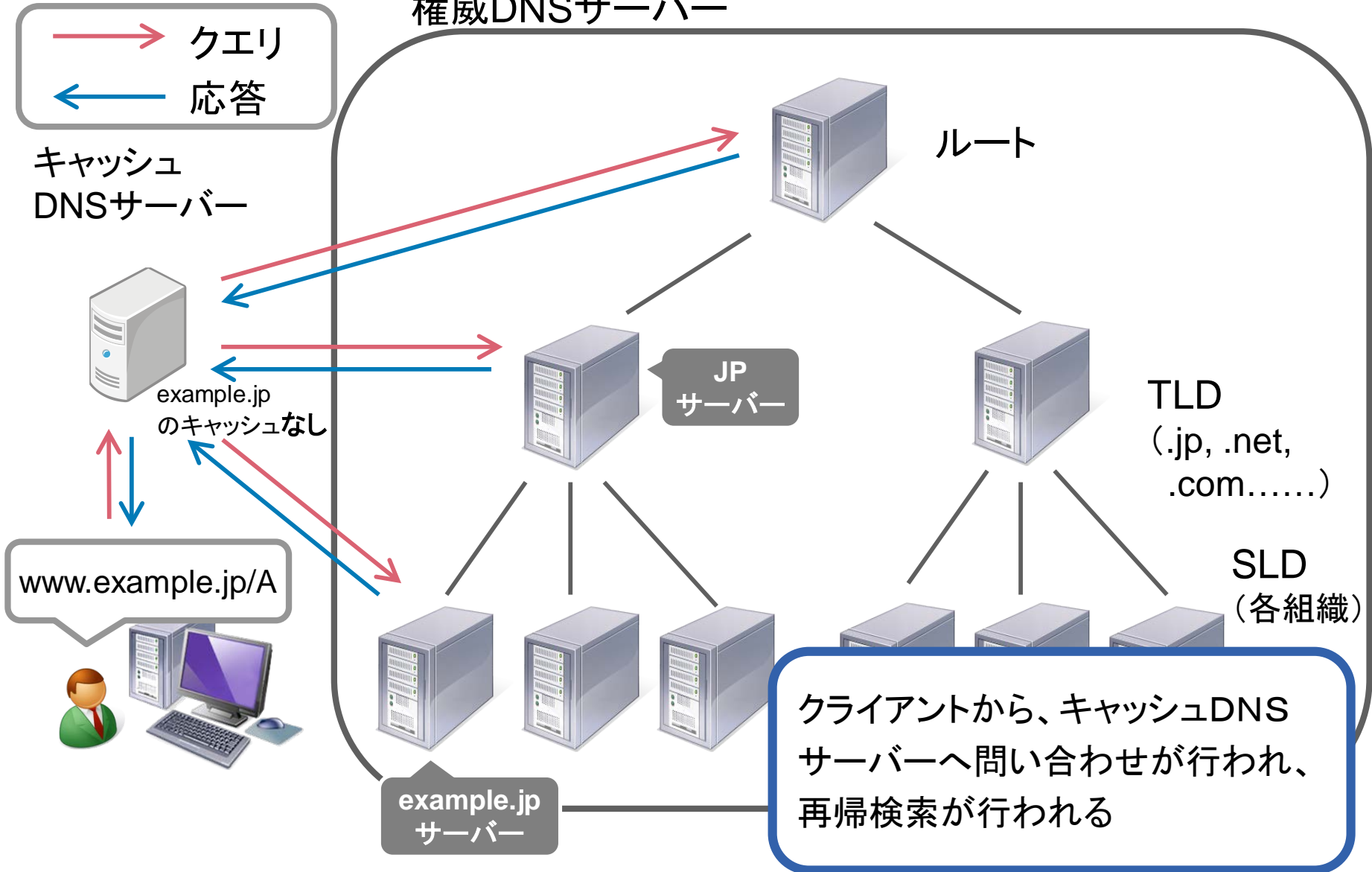
- 権威DNSサーバー

- DNSサーバーの階層構造を構成
- 通常、クライアントは直接利用しない

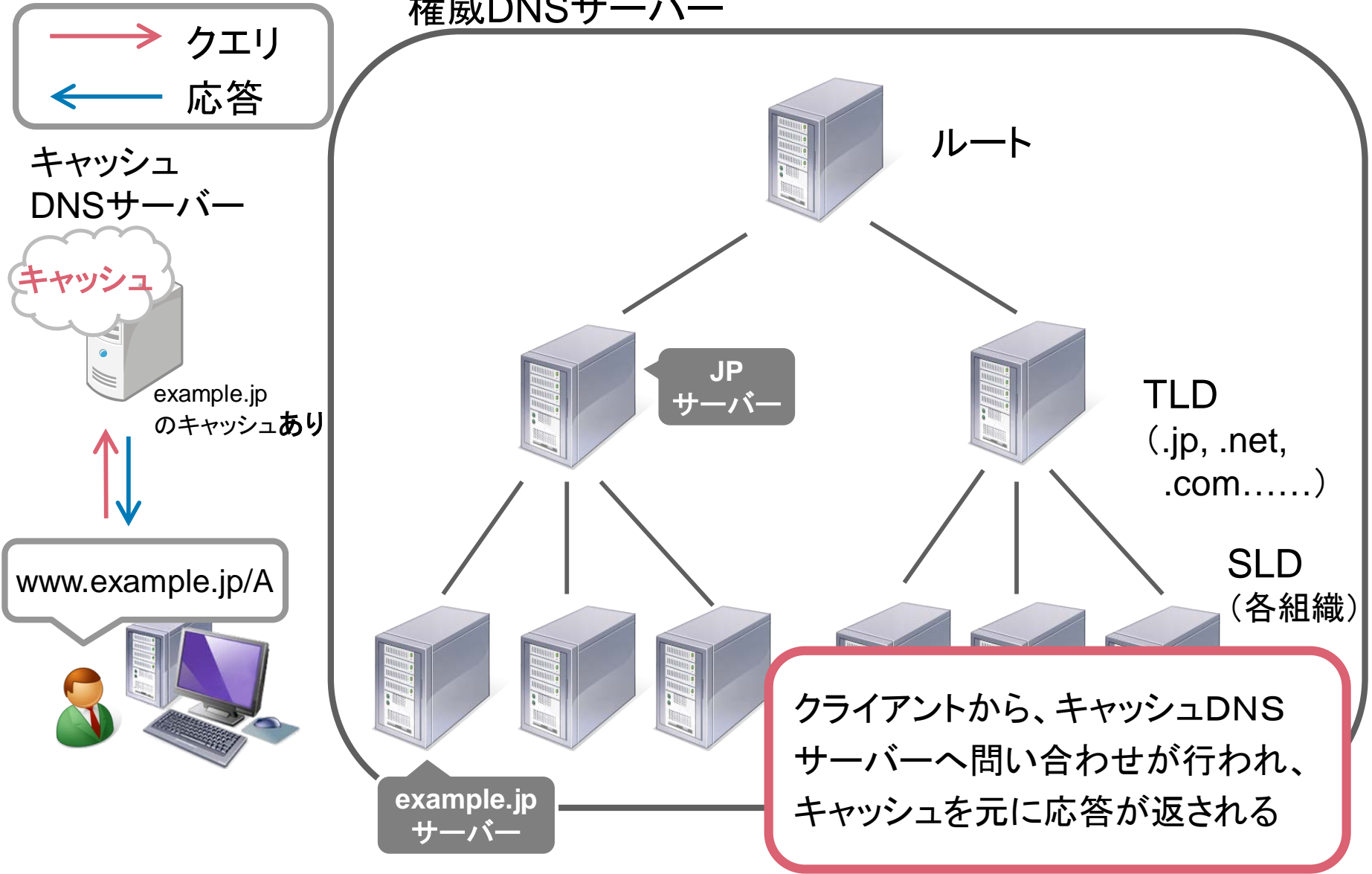
- キャッシュDNSサーバー

- クライアントが直接利用する
- 名前解決の結果をしばらく保持する
 - キャッシュがあると早い(良く使う名前ほどキャッシュされる)
- クライアントからのクエリを受ける
 - 再帰的クエリ / RDビット = 1
- キャッシュにあれば、そこから応答を返す
- キャッシュになければ、非再帰クエリを発行して、その結果を返す
 - 非再帰的クエリ / RDビット = 0
 - 再帰的クエリのRDビットをクリアし、同じ内容にて非再帰クエリを発行する

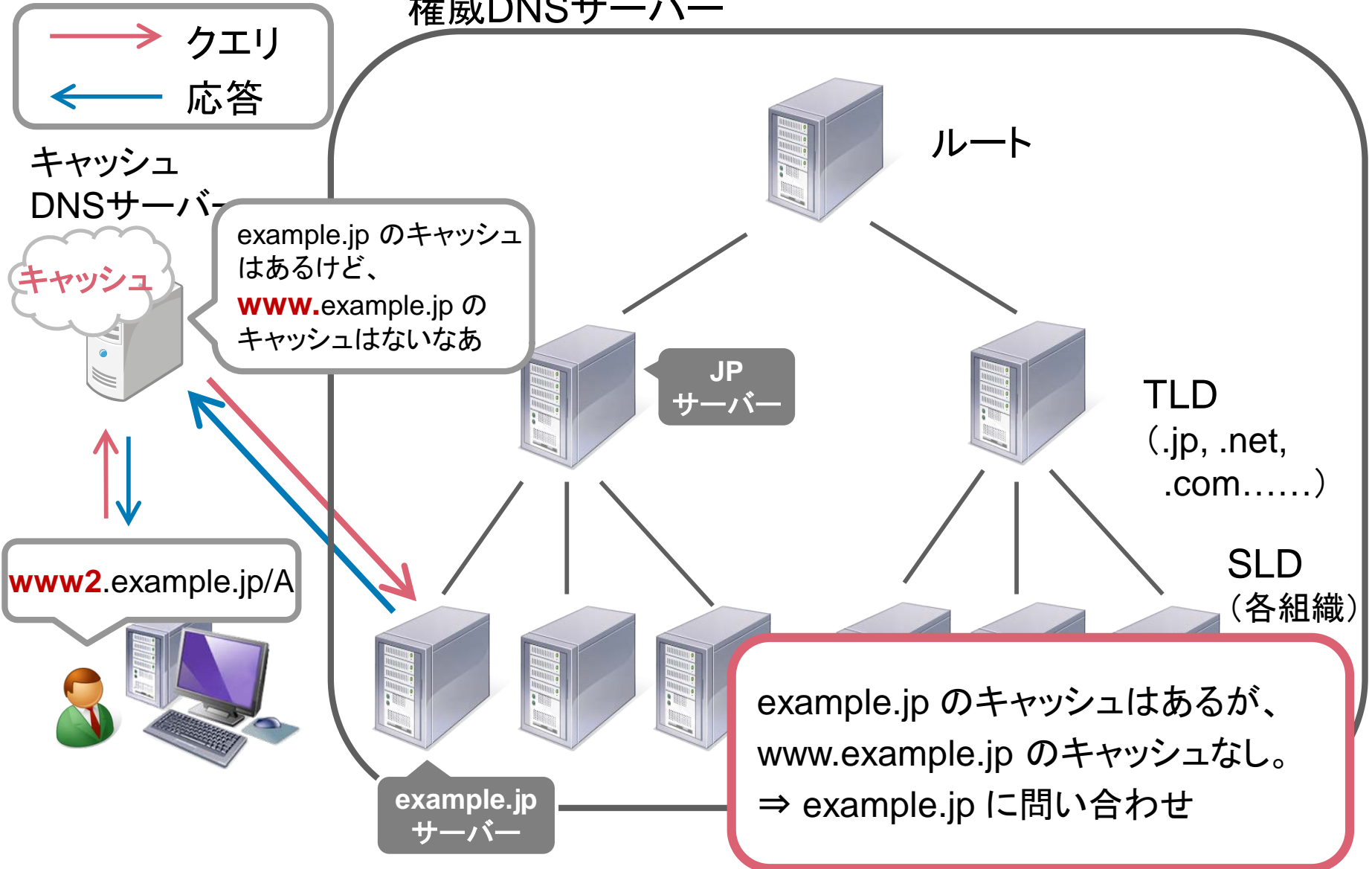
キャッシュDNSサーバーにキャッシュがない場合 権威DNSサーバー



キャッシュDNSサーバーにキャッシュがある場合(1) 権威DNSサーバー



キャッシュDNSサーバーにキャッシュがある場合(2) 権威DNSサーバー



トラブルシューティングの基本

- **Where?** - 原因はどこか？
 - 手元のキャッシュDNSサーバーか？
 - 権威DNSサーバーのいずれかか？
 - 各DNSサーバーまでのネットワークか？
- **How?** - どこをどう調べればよいか？
 - どんなツールやWebサービスを使えばよいか？
- 調査の際には「**再帰的クエリ**」と「**非再帰的クエリ**」を明確に
区別すべき
 - 調査対象がキャッシュDNSサーバーか？権威DNSサーバーか？
- それぞれのサーバーに合った形での調査が必要
 - dig/drillコマンドのオプションなど → 以降で詳しく説明します

トラブルシューティングの心構え

- **キャッシュDNSサーバーの気持ちになって考える**

- 権威DNSサーバーからの応答の意味を考える
- 権威DNSサーバの応答を読み解く必要がある
 - その道具がdig/drill
 - 全体を俯瞰するのには向かない

- **目的に合った道具を選ぶ**

- キャッシュDNSサーバーの気持ちになる
 - dig/drill
- 全体を俯瞰する
 - Squish.net DNS traversal checker
 - dnscheck.jp



トラブル解決に役立つ
2. 道具の使い方

調査の基本—どのコマンドを使うべきか？

- DNSサーバーにクエリを送り、調査する
 - リクエストに関するパラメーターを細かく調整して、応答を調査する
 - 基本はコマンドラインツール
- nslookup コマンド……は使うべきでない
 - クエリの細かいパラメーターが指定不可
 - 応答のフラグやセクションの情報を得ることができない
- では、何を使うか？
 - digコマンド、drillコマンド

nslookupとdigの違い

- nslookup

```
$ nslookup jprs.co.jp
Server:          192.0.2.12
Address:         192.0.2.12 #53

Non-authoritative answer:
Name:   jprs.co.jp
Address: 202.11.16.167
```

- dig

```
$ dig jprs.co.jp

; <<>> DiG 9.9.2-P2 <<>> jprs.co.jp
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 41096
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;jprs.co.jp.                IN      A

;; ANSWER SECTION:
jprs.co.jp.                 13883   IN      A      202.11.16.167

;; AUTHORITY SECTION:
jprs.co.jp.                 61085   IN      NS     ns2.jprs.co.jp.
jprs.co.jp.                 61085   IN      NS     ns1.jprs.co.jp.
jprs.co.jp.                 61085   IN      NS     ns3.jprs.co.jp.

;; ADDITIONAL SECTION:
ns1.jprs.co.jp.             26393   IN      A      202.11.16.49
ns1.jprs.co.jp.             74734   IN      AAAA   2001:df0:8::a153
ns2.jprs.co.jp.             71604   IN      A      202.11.16.59
ns2.jprs.co.jp.             53612   IN      AAAA   2001:df0:8::a253
ns3.jprs.co.jp.             73366   IN      A      61.200.83.204

;; Query time: 1 msec
;; SERVER: 192.0.2.12#53(203.0.113.12)
;; WHEN: Wed Jul 17 21:08:42 2013
;; MSG SIZE rcvd: 213
```

情報量の差

digコマンドとdrillコマンド



こちら

- dig コマンド
 - BIND 9 に付属するコマンド
 - 実行例：
 - `$ dig_+dnssec_@192.0.2.53_example.jp._SOA`
- drill コマンド
 - Unboundで用いられているライブラリ「ldns」に付属するコマンド
 - 実行例：
 - `$ drill_-D_example.jp._@192.0.2.53_SOA`

今日はdigコマンドを用いた解説をします

drillとdigの違い

- drill

```
$ drill @localhost jprs.co.jp
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 54357
;; flags: qr rd ra ; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5
;; QUESTION SECTION:
;; jprs.co.jp. IN      A

;; ANSWER SECTION:
jprs.co.jp.      86205  IN      A        202.11.16.167

;; AUTHORITY SECTION:
jprs.co.jp.      86205  IN      NS       ns3.jprs.co.jp.
jprs.co.jp.      86205  IN      NS       ns1.jprs.co.jp.
jprs.co.jp.      86205  IN      NS       ns2.jprs.co.jp.

;; ADDITIONAL SECTION:
ns1.jprs.co.jp. 86205  IN      A        202.11.16.49
ns1.jprs.co.jp. 86205  IN      AAAA    2001:df0:8::a153
ns2.jprs.co.jp. 86205  IN      A        202.11.16.59
ns2.jprs.co.jp. 86205  IN      AAAA    2001:df0:8::a253
ns3.jprs.co.jp. 86205  IN      A        61.200.83.204

;; Query time: 0 msec
;; SERVER: 127.0.0.1
;; WHEN: Fri Jun 20 01:36:22 2014
;; MSG SIZE rcvd: 202
```

- dig

```
$ dig @localhost jprs.co.jp
; <<>> DiG 9.10.0-P1 <<>> +noedns @localhost jprs.co.jp
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63069
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;jprs.co.jp.                IN      A

;; ANSWER SECTION:
jprs.co.jp.                86374  IN      A        202.11.16.167

;; AUTHORITY SECTION:
jprs.co.jp.                86373  IN      NS       ns2.jprs.co.jp.
jprs.co.jp.                86373  IN      NS       ns1.jprs.co.jp.
jprs.co.jp.                86373  IN      NS       ns3.jprs.co.jp.

;; ADDITIONAL SECTION:
ns1.jprs.co.jp.            86373  IN      A        202.11.16.49
ns1.jprs.co.jp.            86373  IN      AAAA    2001:df0:8::a153
ns2.jprs.co.jp.            86373  IN      A        202.11.16.59
ns2.jprs.co.jp.            86373  IN      AAAA    2001:df0:8::a253
ns3.jprs.co.jp.            86373  IN      A        61.200.83.204

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: 火 6月 24 06:12:09 JST 2014
;; MSG SIZE rcvd: 202
```


dig コマンドが使える環境

- Unix系OS
 - ほとんどの環境で標準添付
 - FreeBSD 10以降では、ベースシステムから削除
 - drill(1)コマンドを用いるか、ports/pkg からインストール(dns/bind-tools)
 - OS Xにも標準添付
- Windows
 - Windows版BIND 9のバイナリキットに含まれている
 - 開発元のISCが無償で公開

dig コマンド – 使い方

```
$ dig +rec @192.0.2.53 example.jp. SOA
```

オプション DNSサーバー 対象ドメイン名 クエリタイプ

- 重要なオプション

- RD bit

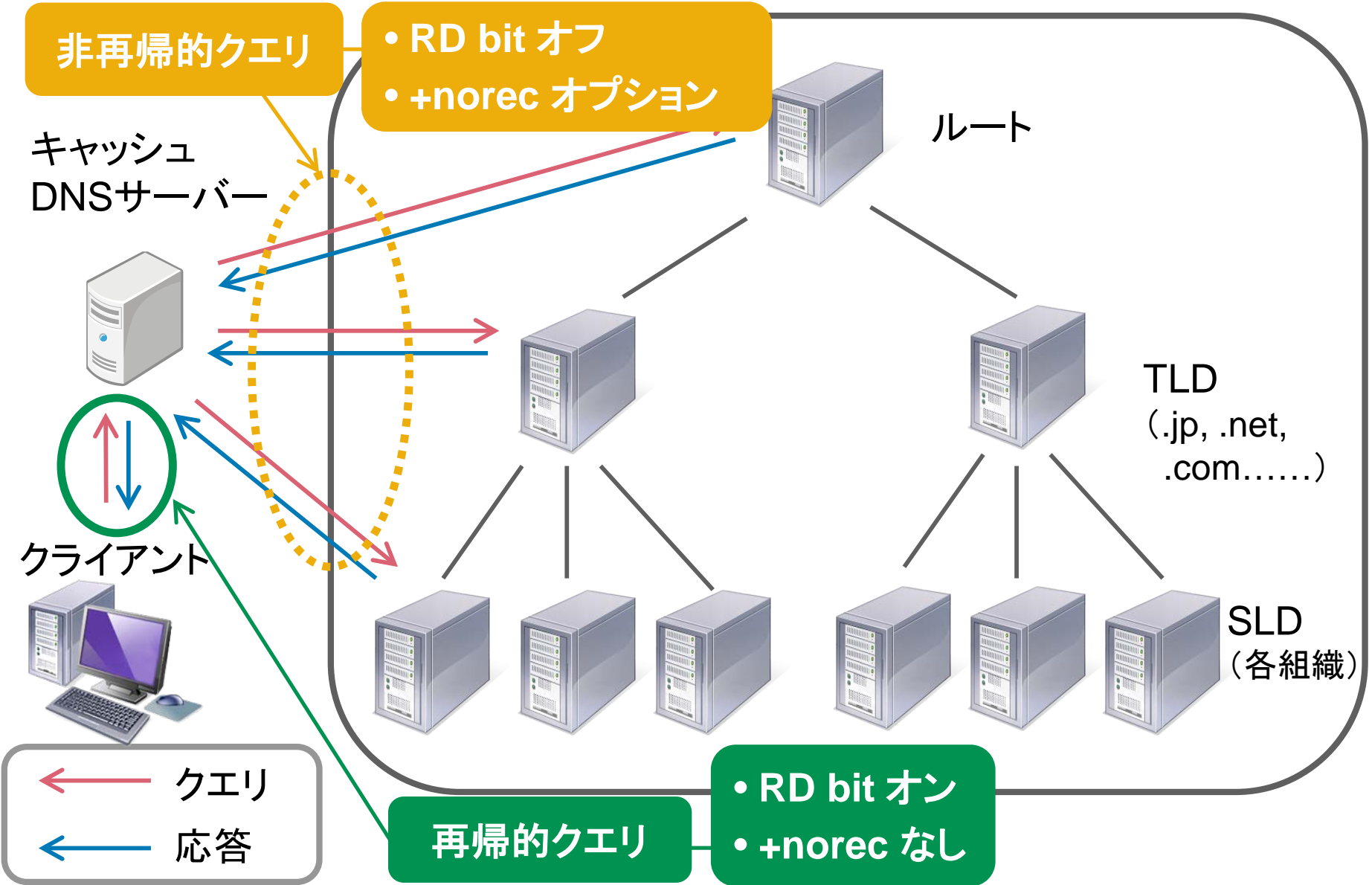
- オン = 階層構造をたどって = +recurse または +rec
- オフ = 持ってる情報を教えて = +norecurse または +norec

- RD bit = **R**ecursion **D**esired bit

- サーバーに対して「DNSの階層構造をたどって！」と伝えるために、クライアント側でセット
- digコマンドやdrillコマンドでは**デフォルトでオン**
- 権威DNSサーバーに対してリクエストを送信する(権威DNSサーバーの動作を調べる)場合には、オフにしておくこと

RD bit と +norec の関係

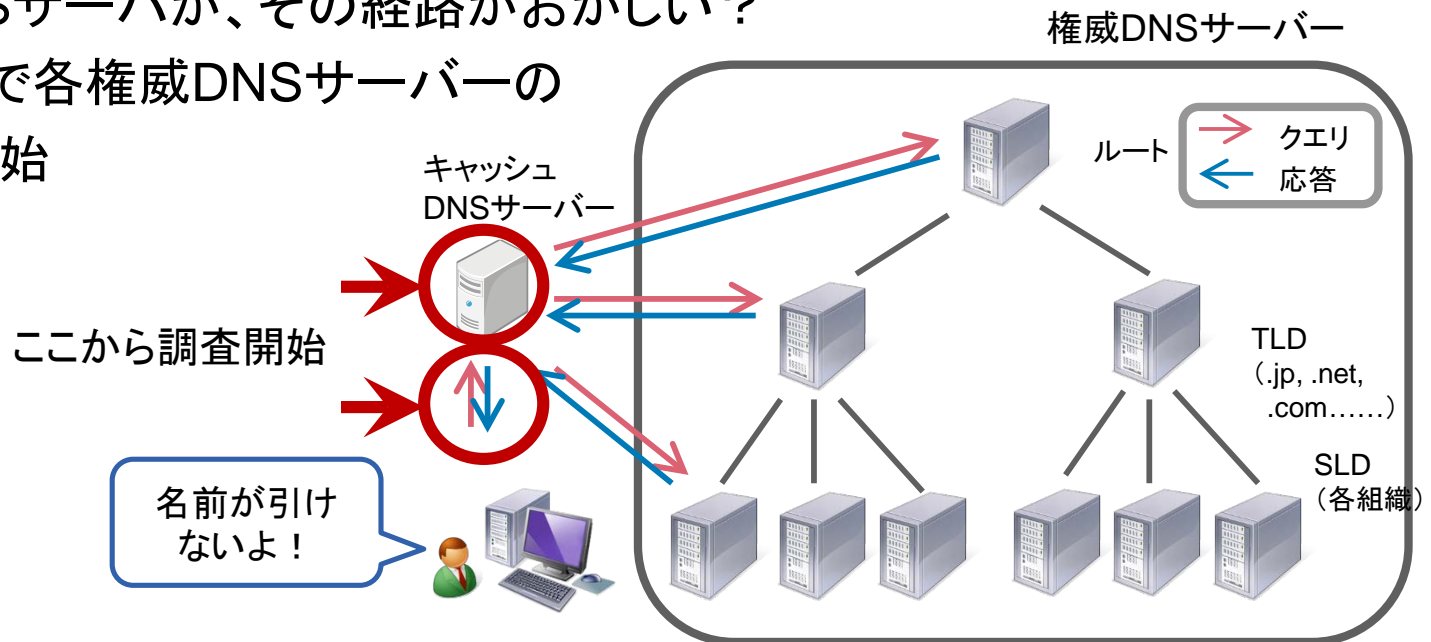
権威DNSサーバー



dig コマンド – +rec / +norec の使いどころ(1)

- 顧客や組織内の利用者から「引けない！」と連絡が来たとき
 - キャッシュDNSサーバーの状況を調査する際に使用
 - +rec をつけての調査から開始
 - クライアントとキャッシュDNSサーバーとの通信は問題なさそうなら……
- 権威DNSサーバか、その経路がおかしい？

+norec で各権威DNSサーバーの調査を開始



dig コマンド – +rec / +norec の使いどころ(2)

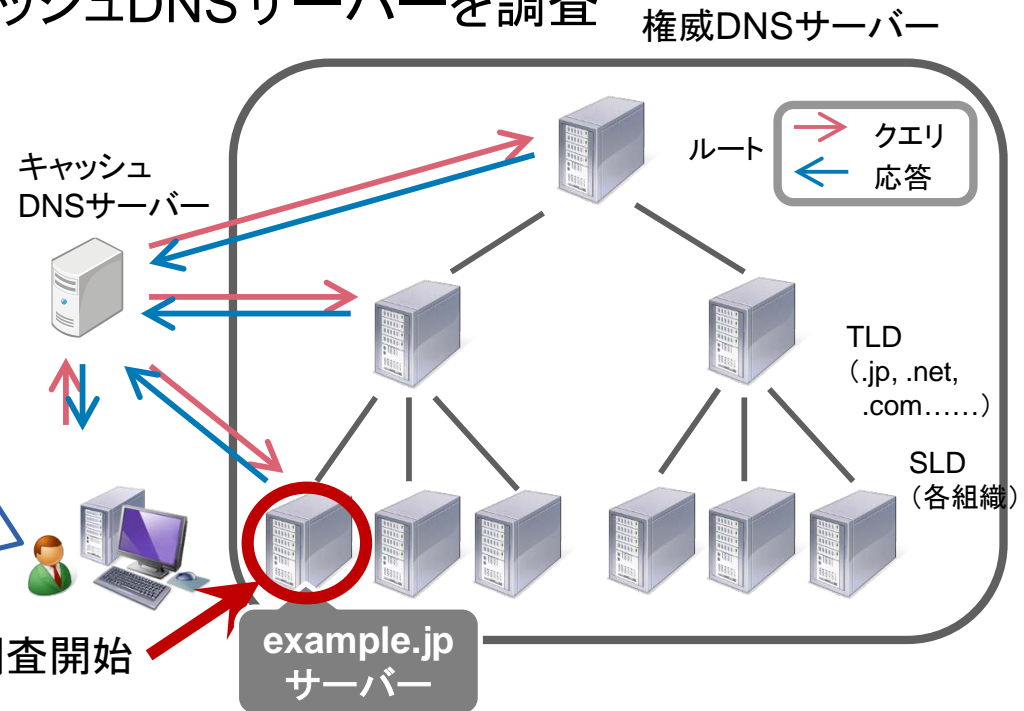
- 顧客から「登録したドメイン名が利用できない」と連絡がきたとき
 - 権威DNSサーバー の設定不具合？
 - +norec をつけて、権威DNSサーバーから調査開始
 - 特定のキャッシュDNSサーバーだけがおかしい？
- +rec をつけて、該当のキャッシュDNSサーバーを調査

• どちらを(どこを)
• どう調べているのか
を意識！

example.jp
登録したのに使えない！

ここから調査開始

example.jp
サーバー



[補足] ネガティブキャッシュとは

- 顧客から「登録したドメイン名が利用できない」と連絡がきたとき
- 顧客が利用するキャッシュDNSサーバー以外は名前が引ける



ネガティブキャッシュが原因かも？

- ネガティブキャッシュとは？
 - 「そのドメイン名は存在しない」という情報のキャッシュ
 - ドメイン名の登録設定(上位ゾーンからの委譲の設定)が行われる前に名前を引こうとすると……

※ RFC2308 - Negative Caching of DNS Queries (DNS NCACHE) (DNSクエリのネガティブキャッシュ)

dig コマンド – 嬉しいオプション +multi

- +multi (+multiline)

```
% dig +dnssec jprs.co.jp SOA
;; ANSWER SECTION:
jprs.co.jp.          85892  IN      SOA     ns1.jprs.co.jp. postmaster.jprs.c
o.jp. 1403054817 3600 900 1814400 900
jprs.co.jp.          85892  IN      RRSIG   SOA 8 3 86400 20140718002657 2014
0618002657 18384 jprs.co.jp. N+shK12/CcvmzZEdTJsZF3jjILl1jxyQgX0Ztf9STW0mNf5KR4/9E
qW/r KmDjeAjJ4nDw10AJaYaS1Y0GYsQt0WxsH5KXdVs2sVkiGFyeTECoSUu9BT40EPLdsQY5xJn3Tr0
5Ftrh4PRnHjnLAA3YsBjZP0x90LWHiMafQqsu d80=
```

SOAを読みやすくする

```
% dig +dnssec +multi jprs.co.jp SOA
;; ANSWER SECTION:
jprs.co.jp.          85620 IN SOA ns1.jprs.co.jp. postmaster.jprs.co.jp. (
                        1403054817 ; serial
                        3600      ; refresh (1 hour)
                        900       ; retry (15 minutes)
                        1814400   ; expire (3 weeks)
                        900       ; minimum (15 minutes)
                        )
jprs.co.jp.          85620 IN RRSIG SOA 8 3 86400 (
                        20140718002657 20140618002657 18384 jprs.co.jp.
                        N+shK12/CcvmzZEdTJsZF3jjILl1jxyQgX0Ztf9STW0mN
                        f5KR4/9EqW/rKmDjeAjJ4nDw10AJaYaS1Y0GYsQt0Wxs
                        H5KXdVs2sVkiGFyeTECoSUu9BT40EPLdsQY5xJn3Tr05
                        Ftrh4PRnHjnLAA3YsBjZP0x90LWHiMafQqsud80= )
```

dig コマンド – その他のオプション

- +VC
 - 初めからTCPで問い合わせる
 - Tcp fallback のテスト用に利用
- +edns
 - edns0(後述) を有効にする
 - BIND 9.9からデフォルトでon
 - +noedns で9.8以前と同じ動作に

その他多数オプションあり。

\$ man 1 dig

で確認！

+multi のように、常に設定しておきたいオプションは、ホームディレクトリに”.digrc”を

dig コマンド – 出力の読み方

特に注目

```
$ dig +norec @ns1.jprs.jp jprs.jp
; <<>> DiG 9.9.2-P2 <<>> +norec @ns1.jprs.jp jprs.jp
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34174
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5
```

ヘッダー

```
;; QUESTION SECTION:
jprs.jp.                IN      A
```

Question

```
;; ANSWER SECTION:
jprs.jp.                86400   IN      A      202.11.16.167
```

Answer

```
;; AUTHORITY SECTION:
jprs.jp.                86400   IN      NS     ns2.jprs.jp.
jprs.jp.                86400   IN      NS     ns3.jprs.jp.
jprs.jp.                86400   IN      NS     ns1.jprs.jp.
```

Authority

```
;; ADDITIONAL SECTION:
ns1.jprs.jp.           86400   IN      A      202.11.16.49
ns1.jprs.jp.           86400   IN      AAAA   2001:df0:8::a153
ns2.jprs.jp.           86400   IN      A      202.11.16.59
ns2.jprs.jp.           86400   IN      AAAA   2001:df0:8::a253
ns3.jprs.jp.           86400   IN      A      61.200.83.204
```

Additional

```
;; Query time: 1 msec
;; SERVER: 203.0.113.12#53(203.0.113.12)
;; WHEN: Thu May 02 15:20:20 2013
;; MSG SIZE rcvd: 199
```

応答時間、サーバーの
IPアドレス、サイズなど

dig コマンド – 出力の読み方 (ヘッダー)(1/2)

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34174  
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5
```

- ヘッダの内容
 - 各セクションに関する情報やステータス、フラグなどを格納
- 主な status (RCODE: 応答コード)
 - NOERROR 正常な応答(該当するタイプがない場合も含む)
 - FORMERR DNSメッセージのフォーマットが不正
 - SERVFAIL DNSサーバー側の異常^{†1}
 - NXDOMAIN リクエストされた名前が存在しない
 - REFUSED リクエストが拒否された
 - NXRRSET 存在すべきレコードが存在しない^{†2}

^{†1} DNSSEC 検証エラーや、全ての権威DNSサーバーが応答しない場合にも出力される

^{†2} ダイナミックアップデートの際、返りうるエラー

dig コマンド – 出力の読み方 (ヘッダー) (2/2)

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34174  
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 5
```

- 注目すべき主な flags (ヘッダ等に含まれるビット)
 - qr: 応答であることを示す (Query / Response)
 - dig/drill コマンドの出力(応答)では、通常オン
 - aa: 権威ある応答であることを示す (Authoritative Answer)
 - 通常、問い合わせたゾーンの権威DNSサーバーからの応答はオン
 - キャッシュDNSサーバーからの応答や、他のDNSサーバーに委任していることを示す応答ではオフ
 - ra: 再帰検索要求が処理可能なことを示す (Recursion Available)
 - 通常、キャッシュDNSサーバーからの応答ではオン
 - キャッシュDNSサーバーと、権威DNSサーバーの分離ができていない (オープンリゾルバーである可能性がある)
 - tc: 応答の一部が切り捨てられたことを示す (TrunCation)
 - TCPに切り替えて(TCPフォールバック)再度問い合わせる
 - digコマンドは自動的にTCPフォールバックするため、通常は表示されない (「+ignore オプション」で抑制できる)

dig コマンド – 出力の読み方 (Question)

```
;; QUESTION SECTION:  
;jprs.jp.                IN      A
```

- Question セクションの内容
 - 問い合わせた内容(名前、型)がそのままコピーされている



```
$ dig +norec @ns1.jprs.jp jprs.jp
```

dig コマンド – 出力の読み方 (Answer)

```
;; ANSWER SECTION:  
jprs.jp.                86400    IN       A        202.11.16.167
```

- Answerセクション

- 問い合わせた内容に対応するリソースレコードセット(RRSet)が格納される
 - RRSET : その名前に存在する当該リソースレコードのセット
- 問い合わせた名前 / タイプが存在しない場合や、他のDNSサーバーにゾーンが委任されている場合は空

dig コマンド – 出力の読み方 (Authority)

```
;; AUTHORITY SECTION:  
jprs.jp.           86400   IN      NS      ns2.jprs.jp.  
jprs.jp.           86400   IN      NS      ns3.jprs.jp.  
jprs.jp.           86400   IN      NS      ns1.jprs.jp.
```

- Authorityセクション

- 権威を持っているDNSサーバーの情報が格納される
- 問い合わせた名前 / タイプが存在しないことを示す場合、SOA RRが格納される
- 委任応答の場合、委任先の権威DNSサーバーのホスト名 (NS) が格納される

dig コマンド – 出力の読み方 (Additional)

```
;; ADDITIONAL SECTION:
ns1.jprs.jp.      86400   IN      A       202.11.16.49
ns1.jprs.jp.      86400   IN      AAAA    2001:df0:8::a153
ns2.jprs.jp.      86400   IN      A       202.11.16.59
ns2.jprs.jp.      86400   IN      AAAA    2001:df0:8::a253
ns3.jprs.jp.      86400   IN      A       61.200.83.204
```

- Additionalセクション

- 付加的な情報が格納される

- Authorityセクションに含まれるDNSサーバーのA、AAAA RRなど

- 委任応答で委任先が内部名の場合、グルーレコードと呼ばれる

dig コマンド – 出力例(1)

```
$ dig +norec @ns1.jprs.jp jprs.jp PTR
```

```
; <<>> DiG 9.9.2-P2 <<>> +norec @ns1.jprs.jp jprs.jp PTR
; (2 servers found)
```

```
;; global options: +cmd
;; Got answer:
```

(1) ステータスは NOERROR

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15556
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
```

ヘッダー

```
;; QUESTION SECTION:
;jprs.jp.
```

IN PTR

Question

(2) Answerセクションなし

```
;; AUTHORITY SECTION:
;jprs.jp.
```

86400

IN

SOA

```
ns1.jprs.co.jp. ¥
postmaster.jprs.co.jp. ¥
1402803013 3600 900 ¥
1814400 86400
```

Authority

(3) AuthorityセクションにSOALレコード

```
;; Query time: 1 msec
;; SERVER: 203.0.113.12#53(203.0.113.12)
;; WHEN: Thu May 02 15:20:20 2013
;; MSG SIZE rcvd: 199
```

応答時間、サーバーの
IPアドレス、サイズなど

→ 問い合わせた名前は存在するが、タイプが存在しないケース

※ (3)のSOALレコードのminimumはネガティブキャッシュのTTLとして扱われる

dig コマンド – 出力例(2)

```
$ dig +norec @ns1.jprs.jp nameerror.jprs.jp
; <<>> DiG 9.9.2-P2 <<>> +norec @ns1.jprs.jp nameerror.jprs.jp
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 32704
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
```

(1)ステータスは NXDOMAIN

ヘッダー

```
;; QUESTION SECTION:
;nameerror.jprs.jp.          IN      A
```

Question

(2) Answerセクションなし

```
;; AUTHORITY SECTION:
jprs.jp.          86400   IN      SOA     ns1.jprs.co.jp.  ¥
                  postmaster.jprs.co.jp.  ¥
                  1402803013 3600 900 ¥
                  1814400 86400
```

Authority

(3) AuthorityセクションにSOALレコード

```
;; Query time: 1 msec
;; SERVER: 203.0.113.12#53(203.0.113.12)
;; WHEN: Thu May 02 15:20:20 2013
;; MSG SIZE rcvd: 199
```

応答時間、サーバーの
IPアドレス、サイズなど

→ 問い合わせた名前自体が存在しないケース

※ (3)のSOALレコードのminimumはネガティブキャッシュのTTLとして扱われる

dig コマンド – 結果が違う？

```
% dig @localhost jprs.co.jp

; <<>> DiG 9.10.0-P1 <<>> @localhost jprs.co.jp
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19999
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;jprs.co.jp.                IN      A

;; ANSWER SECTION:
jprs.co.jp.                86400  IN      A      202.11.16.167

;; AUTHORITY SECTION:
jprs.co.jp.                86400  IN      NS     ns1.jprs.co.jp.
jprs.co.jp.                86400  IN      NS     ns2.jprs.co.jp.
jprs.co.jp.                86400  IN      NS     ns3.jprs.co.jp.

;; ADDITIONAL SECTION:
ns1.jprs.co.jp.           86400  IN      A      202.11.16.49
ns1.jprs.co.jp.           86400  IN      AAAA   2001:df0:8::a153
ns2.jprs.co.jp.           86400  IN      A      202.11.16.59
ns2.jprs.co.jp.           86400  IN      AAAA   2001:df0:8::a253
ns3.jprs.co.jp.           86400  IN      A      61.200.83.204

;; Query time: 934 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 20 00:40:21 JST 2014
;; MSG SIZE rcvd: 213
```

```
% dig @localhost jprs.co.jp

; <<>> DiG 9.10.0-P1 <<>> @localhost jprs.co.jp
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27868
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;jprs.co.jp.                IN      A

;; ANSWER SECTION:
jprs.co.jp.                86400  IN      A      202.11.16.167

;; Query time: 238 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 20 00:49:54 JST 2014
;; MSG SIZE rcvd: 55
```

**AUTHORITY SECTION
ADDITIONAL SECTION
がない**

→ キャッシュDNSのサーバーの実装、設定による

※ 右の例はキャッシュDNSサーバーにBIND 9を用い、
minimal-responses オプションを有効にした場合

調査に使えるWebサービス

- DNSの設定などを、GUIで可視化・チェック可能
ここでは2種類のツールを紹介します(この他にもあります)
- Squish.net DNS traversal checker(個人提供:James氏)
 - DNS可視化ツール
 - 応答のおかしいDNSサーバーなどを調べることが可能
- dnscheck.jp(JPRS提供)
 - DNSの設定チェックツール
 - 今現在の設定の確認
 - **これからしようと思っている設定**を調べることが可能

調査に使えるWebサービス – Squish

- Squish.net DNS traversal checker
 - <http://dns.squish.net/>
 - ルートサーバーから再帰的に名前解決した結果を、視覚的に表示
 - 設定に問題があるサーバを調査可能
 - サーバーによって問い合わせ結果が違う
 - 権威サーバーなのに再帰検索可能

Squish DNS traversal checker

Login Register Home » Traversals » jprs.co.jp » detail

Name: jprs.co.jp
 Type: A
 Date: 2014-06-22 05:27:01 UTC
 Initial Root: c.root-servers.net, 192.33.4.12 (1 roots returned)
 Query Key: A, jprs.co.jp

Traversing for jprs.co.jp type A starting at the root(s)

```

  | c.root-servers.net (192.33.4.12) =>
  | | f.dns.jp (150.100.8.8) =>
  | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | g.dns.jp (210.138.175.244) =>
  | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | a.dns.jp (203.119.1.1) =>
  | | | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | | | g.dns.jp (203.119.40.1) =>
  | | | | | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | | | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | | | | | | e.dns.jp (192.50.43.63) =>
  | | | | | | | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | | | | | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | | | | | | | | c.dns.jp (156.154.100.5) =>
  | | | | | | | | | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  | | | | | | | | | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | | | | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | | | | | | | | | | b.dns.jp (202.12.30.131) =>
  | | | | | | | | | | | | | | | ns3.jprs.co.jp (61.200.83.204) <jprs.co.jp>
  | | | | | | | | | | | | | | | ns1.jprs.co.jp (202.11.16.49) <jprs.co.jp>
  | | | | | | | | | | | | | | | ns2.jprs.co.jp (202.11.16.59) <jprs.co.jp>
  
```

Results

33.3% Answered from ns1.jprs.co.jp (202.11.16.49)

jprs.co.jp.	66400	IN	A	202.11.16.167
-------------	-------	----	---	---------------

33.3% Answered from ns2.jprs.co.jp (202.11.16.59)

jprs.co.jp.	66400	IN	A	202.11.16.167
-------------	-------	----	---	---------------

33.3% Answered from ns3.jprs.co.jp (61.200.83.204)

jprs.co.jp.	66400	IN	A	202.11.16.167
-------------	-------	----	---	---------------

dnscheck.jpの使用例

「jprs.jp」の出力結果

●チェック結果詳細

1.ドメイン名に対するチェック結果

値	重要度	チェック結果
JPRS.JP	OK	

2.各ホスト名に対するチェック結果

値	重要度	チェック結果
ns1.jprs.jp	OK	
202.11.16.49		
ns1.jprs.co.jp.		

●緑子 正常に追加して取得した値
●赤字 設定されているべきだが、設定されていない値

1.ドメイン名に対するチェック結果

値	重要度	チェック結果
JPRS.JP	OK	

2.各ホスト名に対するチェック結果

値	重要度	チェック結果
ns1.jprs.jp	OK	
202.11.16.49		
ns1.jprs.co.jp.		

SOA 6 2 86400 20130814023006
20130715023006 64404 jprs.jp.
op+98vIzW6wAPyJEIsJDY2OellF03T
CQy5LqoHqWuV+2Ehd9M8IErJlodzn
DAIq5OKo2ywb8MKo6Be+8T7Qaim9al
wZ5qwsjOOKHs12pQoA1xu4+E7J5YTAY
BR16LUWFMkRBPxME6J3nWZxPk+X9vV
D0xeQJAJe4=

256 3 8
AwEAacpH81ty4DTycAnZzcilR577+G1
mZ70sZ7OcalM3pb+dSHk1Xxa115RDn
2Ep0Y/NGFdz4CKAyTTZ/N6/F5sd3xax
36cNSGUMw6HHQ7oia88qHB1bXvY5w
o40TZP8PPTKURzSodh2zFIrF5puCD
uGekcsgtPaxjM3

257 3 8
AwEAAo1a0eeyID+lcc+NopB0v0TFvml
Dtlw4PYkaPbG50ZLwNNDQYU3S50SF
eC6BhVRY8vTrPTLULUMvM8iRrsmh
utJpIlgkwS/C0vWvDEhbK1UpE3Ew
D5e05yK00xIzxTPk/L2R89Njk+Qrud
QePT5GedzZ9geT+FN+290YUqXWHJ89
+STUL7K0WEiodmhwSwKSIW5meCMHk
WkK...

トラブルシューティングの前に……

3. トラブルを事前回避！

問題を起こさないための設定

トラブル事前回避


- トラブル発生！
 - 対処！解決！問題なし！
- ちょっと待って
 - そのトラブル、事前に防げたのでは？
 - 防げなくても、最小限に留められていたのでは？



→ 問題が発生しにくい設定・お作法

→ 問題が起きても、被害を最小限に食い止める設定、お作法

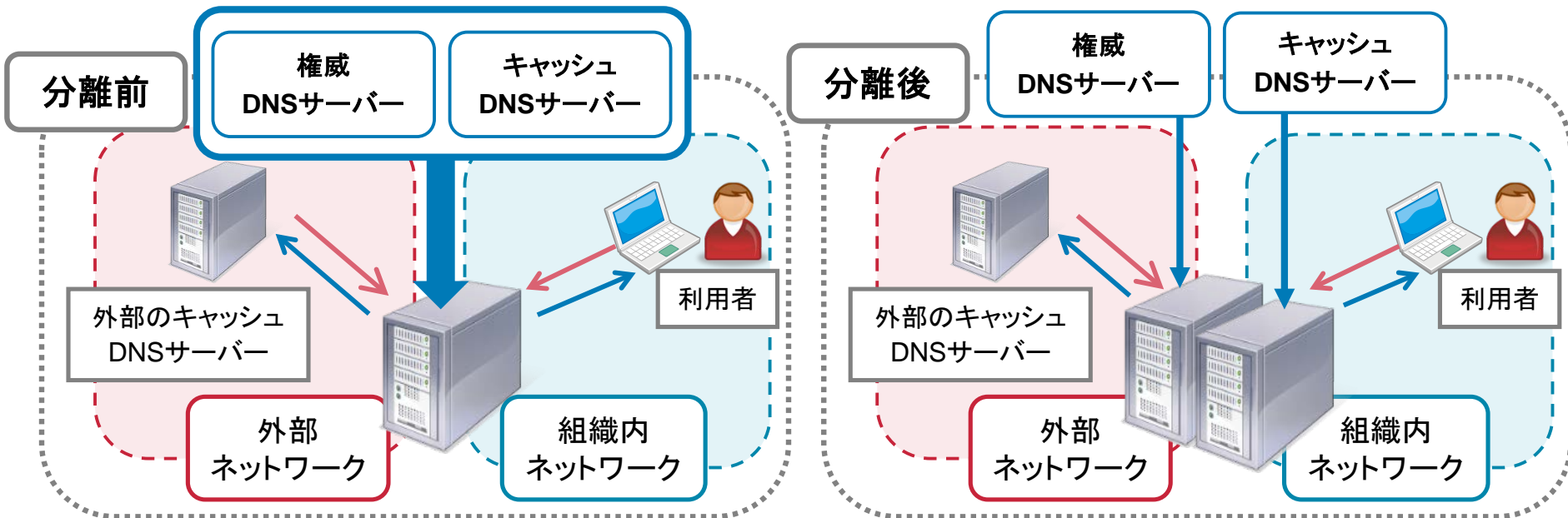
トラブル事前回避 – 設定を事前にチェック

- named-checkconf
 - named.conf の構文チェック
 - \$ named-checkconf named設定ファイル名
 - named-checkzone
 - \$ named-checkzone ゾーン名 ゾーンファイル名
- 
- “)”や”;"の抜けなどの、文法チェック用
 - シリアル番号の上げ忘れ(後述)や、ホスト名末尾の”.”付け忘れ(後述)などはチェックしてくれない

トラブル事前回避 – サーバーの分離とアクセス制限

- キャッシュDNSサーバーと権威DNSサーバーの分離
 - 「ドメイン名の浸透問題」の原因になるかも？
 - 「DNSリフレクター攻撃」の加害者になるかも？
 - 「DNSキャッシュポイズニング」されるかも？

✓ 分離しましょう



DNSキャッシュポイズニング？DNSリフレクター攻撃？

- DNSキャッシュポイズニング

- 偽のDNS情報をキャッシュとして蓄積させる
 - ⇒ フィッシングサイトなどへ誘導
- ■ 権威／キャッシュDNSサーバーの兼用によるDNSポイズニングの危険性について
 - <http://jprs.jp/tech/security/2012-07-04-risk-of-auth-and-recurse.html>

- DNSリフレクター攻撃

- 問い合わせパケットサイズよりも、応答パケットサイズが大きくなるDNSサーバーの特性を利用した攻撃手
 - ⇒ 被害者ではなく、**加害者**になる可能性がある
- ■ 技術解説:「DNS Reflector Attacks (DNSリフレクター攻撃)」について
 - <http://jprs.jp/tech/notice/2013-04-18-reflector-attacks.html>

困った！ どうしてこうなる？

3.よくあるトラブル事例と トラブルシューティング

今日紹介するトラブル事例

A) 設定がうまくいかない・間違えた

1. ゾーン転送がうまくいかない
 1. マスタサーバーにDNSが稼動していない
 2. マスタサーバー側のファイヤーウォールでブロックされている場合
 3. マスタサーバー側でゾーン転送が許可されていない場合
2. ピリオドを付け忘れた
3. SOAシリアルを上げ忘れた
4. SOAシリアルを上げ損ねた
(シリアルを巻き戻したい)

B) 名前が引けない

1. DNSサーバーがダウンしている
2. CNAMEの循環
3. ふぞろいのzone情報たち

C) 名前を引くのに時間が掛かる

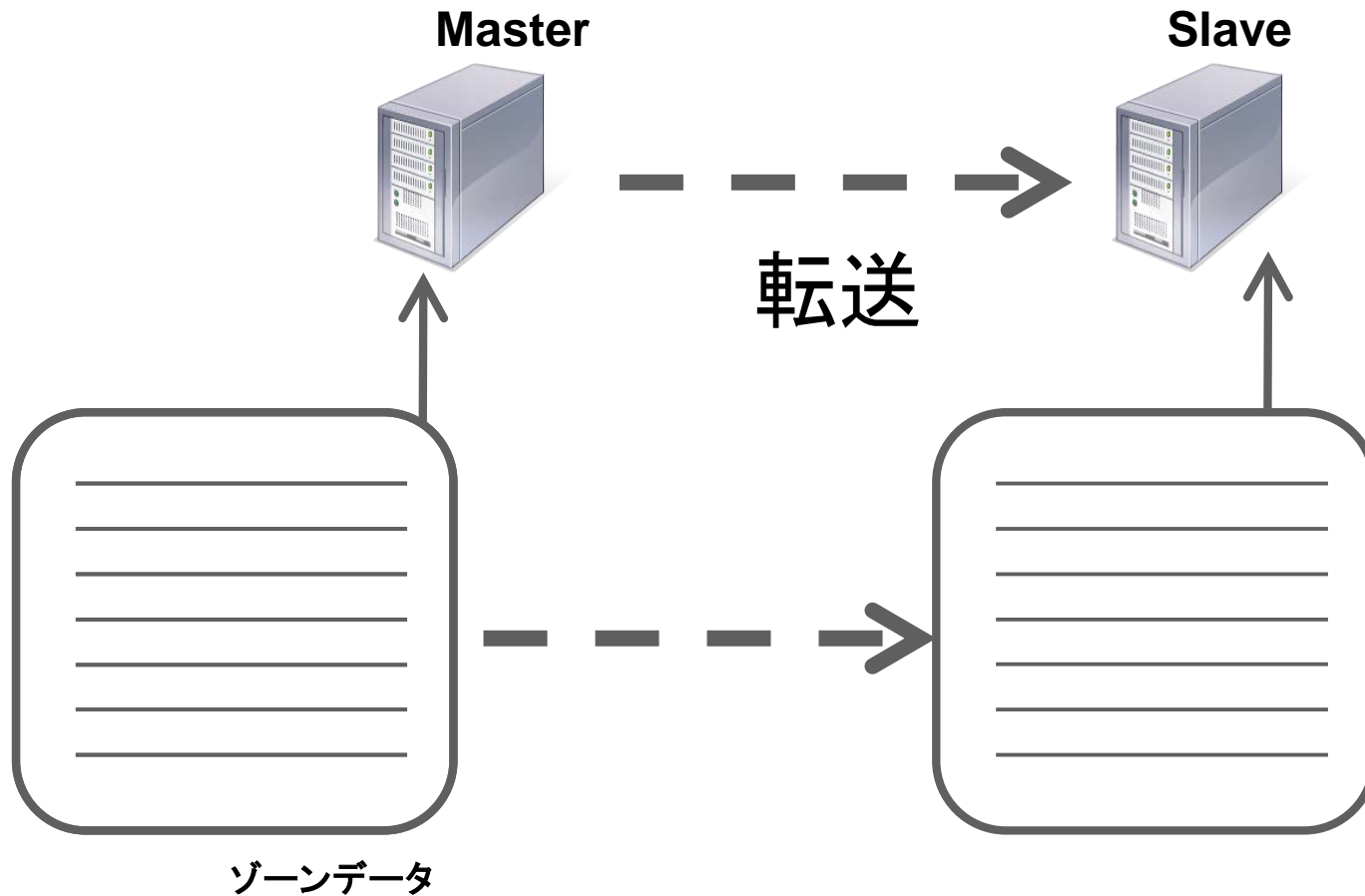
1. TCPフォールバック
2. 権威DNSサーバーの一部がダウンしている

DNSトラブル事例

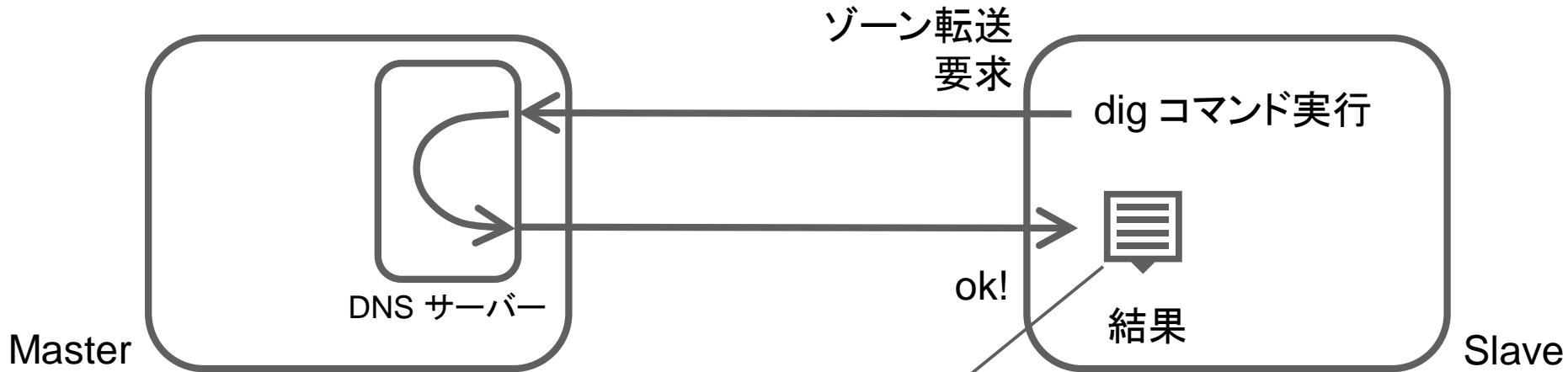
▶ A. 設定を間違えた

1. ゾーン転送がうまくいかない

- ゾーン転送とは……



1. ゾーン転送がうまくいかない – 正常な例



```
$ dig +norec @(Master) example.jp. AXFR

; <<>> DiG 9.8.1-P1 <<>> +norec @(Master) example.jp. AXFR
; (1 server found)
;; global options: +cmd
example.jp.          10800  IN      SOA     (中略)
example.jp.          10800  IN      NS      ns1.example.jp.
(中略)
example.jp.          10800  IN      SOA     ns1.example.jp. root.example.jp. (中略)
;; Query time: 1 msec
;; SERVER: (Master)#53((Master))
;; WHEN: Fri Jul 12 17:56:17 2013
;; XFR size: 31 records (messages 1, bytes 3380)
```

1. ゾーン転送がうまくいかない – よくある原因

- 原因

- TCP 53番ポートがフィルタされている？
- ゾーン転送の設定を間違っている？
- あるいは他の何か？

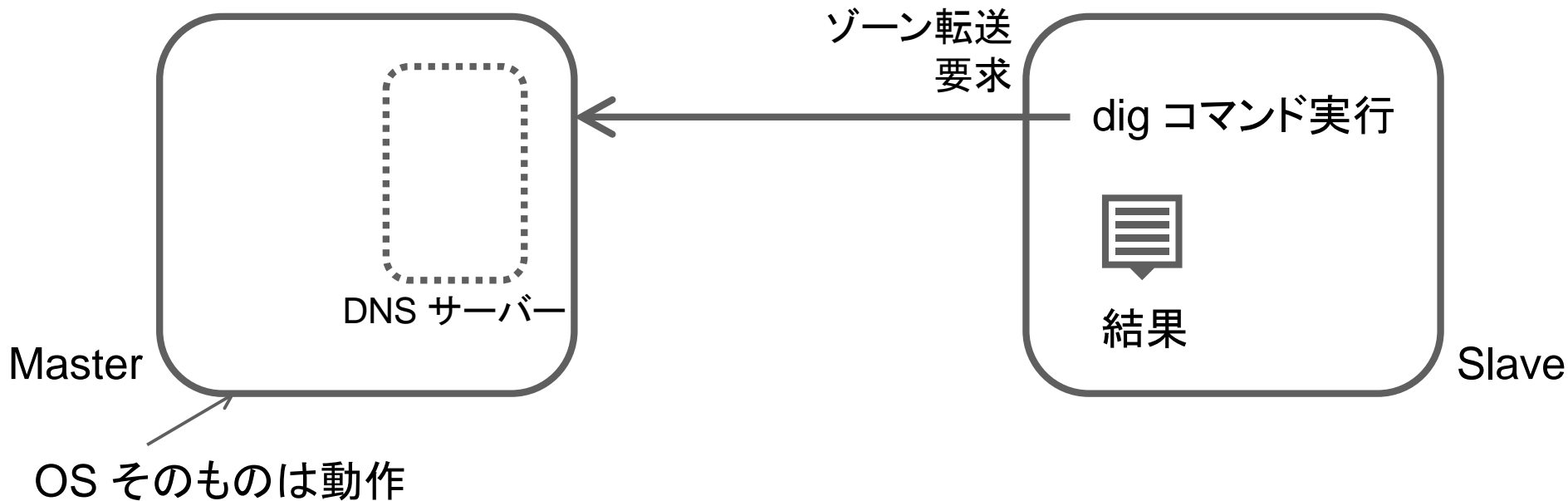
どう切り分ける……？

- 調査法

- dig コマンドを使う
- コマンド例
 - `$ dig_+norec_@(マスター)_example.jp_axfr`
 - スレーブサーバー側で実行！

1. ゾーン転送がうまくいかない – 調査と具体例

1. マスターサーバーでDNSサーバープロセスが稼動していない場合

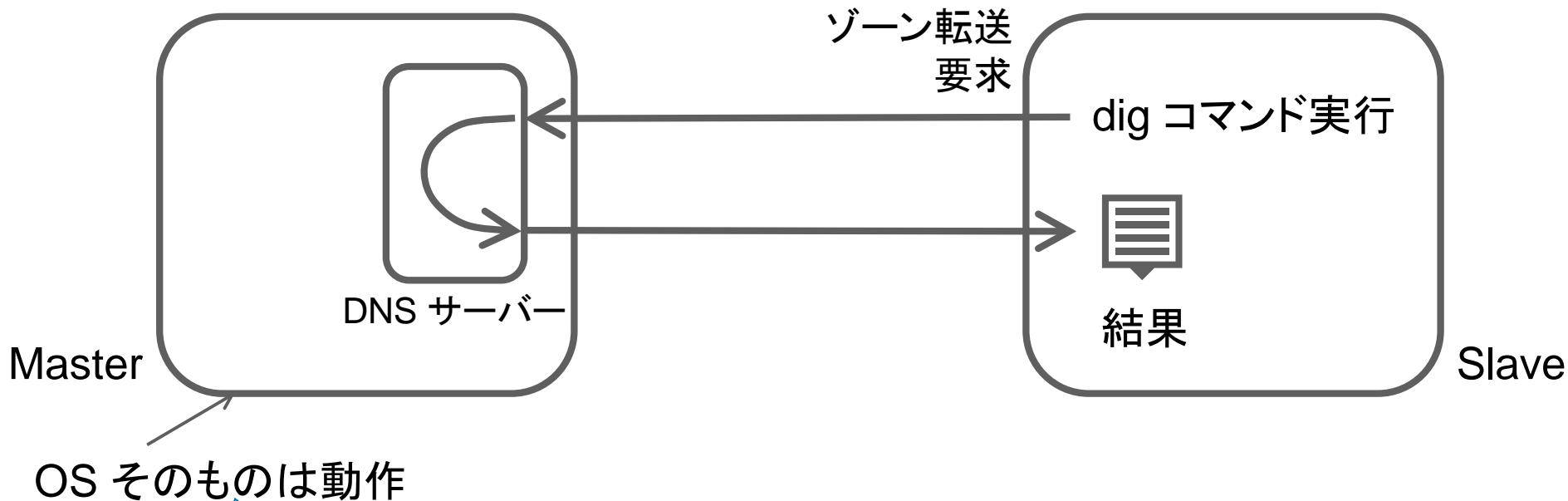


実行結果例

```
$ dig +noredc @(マスター) example.jp axfr
;; Connection to 203.0.113.8 #53(203.0.113.8)
   for example.jp failed: connection refused.
```

1. ゾーン転送がうまくいかない – 調査と具体例

1. マスターサーバーでDNSサーバープロセスが稼動していない場合



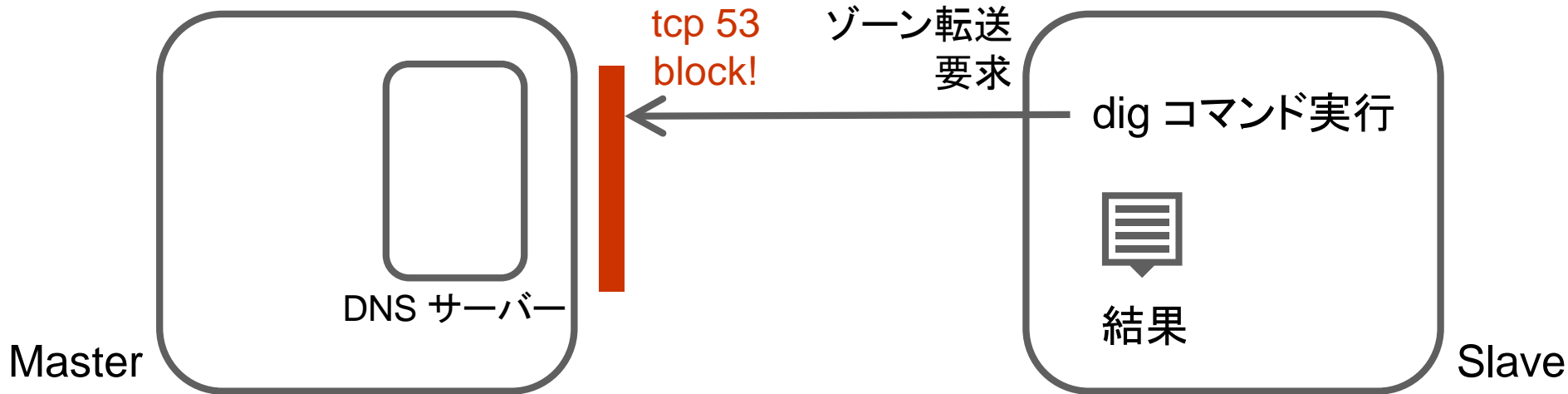
対応

✓ DNS サーバープロセスを立ち上げ直す

- 実は気づかないうちに落ちていたのかも……
- **必ず**原因究明を並行してすすめること
- サーバーのログのチェックなど……

1. ゾーン転送がうまくいかない – 調査と具体例

2. マスターサーバー側のファイヤーウォールでブロックされている場合



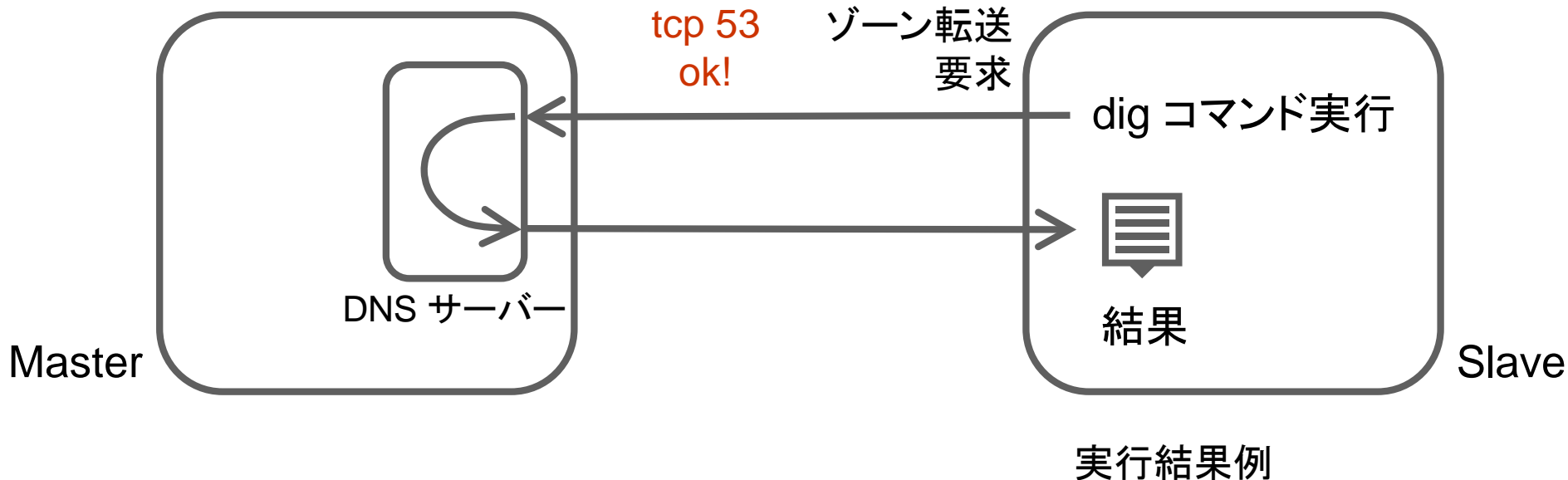
実行結果例

```
$ dig +noredc @(マスター) example.jp axfr

; <<>> DiG 9.9.2-P2 <<>> +noredc @(マスター) example.jp axfr
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached
```

1. ゾーン転送がうまくいかない – 調査と具体例

2. マスターサーバー側のファイヤーウォールでブロックされている場合



対応

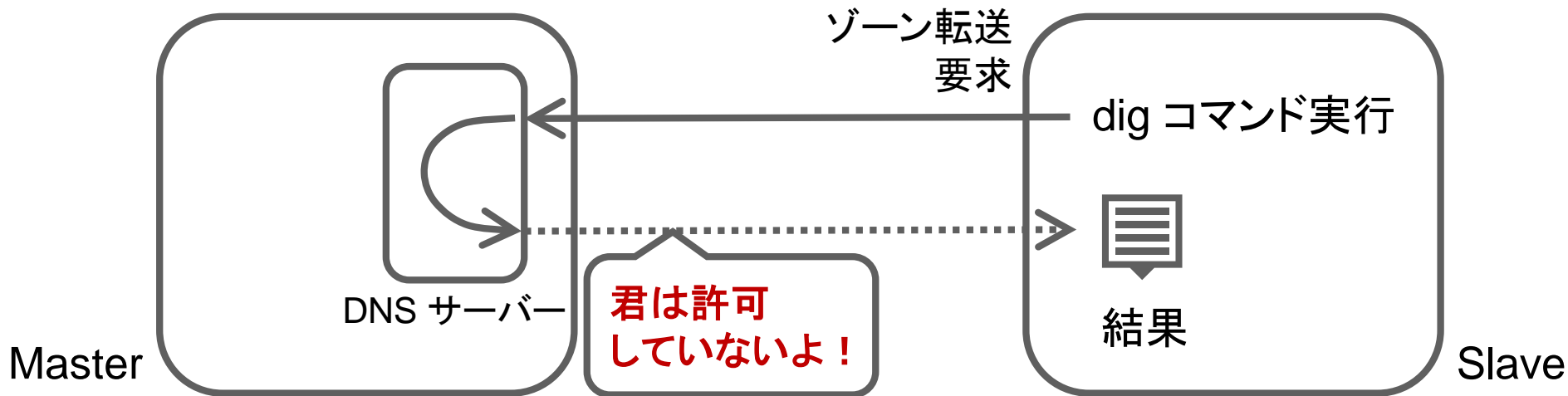
✓ TCP 53番ポートへのアクセスを許可する

→ UDP だけの許可かも……？

→ そもそもDNSサーバーではTCP 53番のオープンが必須！

1. ゾーン転送がうまくいかない – 調査と具体例

3. マスターサーバー側でゾーン転送が許可されていない場合

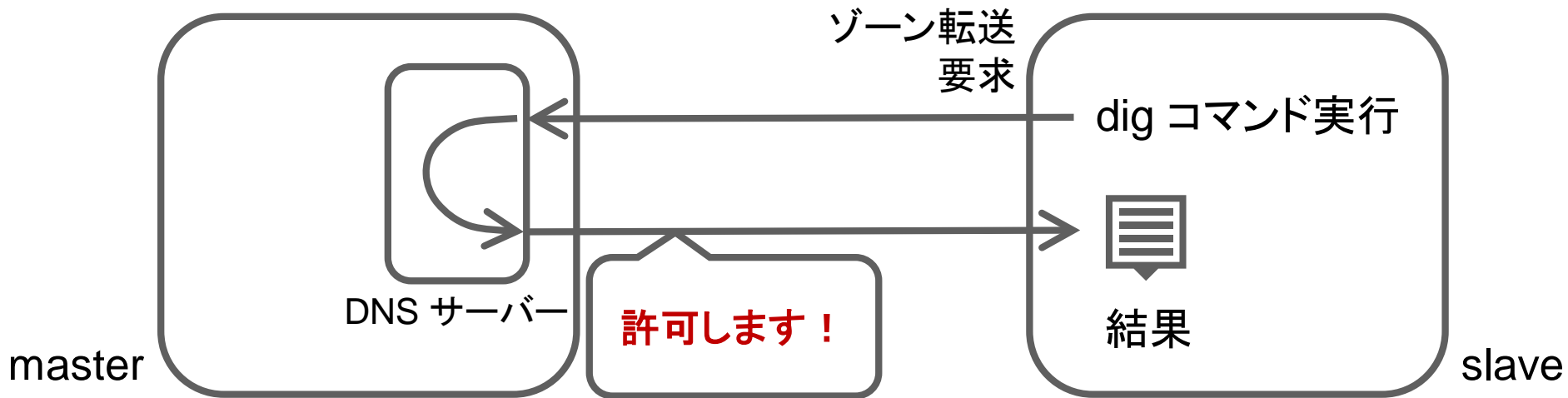


実行結果例

```
$ dig +noredc @(マスター) example.jp axfr
; <<>> DiG 9.9.2-P2 <<>> +noredc @(マスター) example.jp axfr
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

1. ゾーン転送がうまくいかない – 調査と具体例

3. マスタサーバー側でゾーン転送が許可されていない場合



対応

✓ ゾーン転送の設定を見直す

→ 許可ホストの設定を間違えているかも……

2. ピリオドを忘れた - [出題編]

```

$ORIGIN      a.example.
$TTL        86400
@           IN          SOA      ns1.a.example.  root.localhost. (
                                           1047
                                           604800
                                           86400
                                           2419200
                                           3600
                                           )

           IN          NS       ns1.a.example.
           IN          MX       10 mail.a.example

ns1.a.example. IN      A        192.0.2.54
ns1.a.example. IN      A        2001:db8:53::53
mail.a.example. IN     A        192.0.2.57
mail.a.example. IN     AAAA     2001:db8:53::25
www.a.example.  IN     A        192.0.2.58
mail.a.example. IN     AAAA     2001:db8:53::80

```

2. ピリオドを忘れた - [回答編]

```

$ORIGIN      a.example.
$TTL         86400
@            IN          SOA      ns1.a.example.  root.localhost. (
                                                1047
                                                604800
                                                86400
                                                2419200
                                                3600
                                                )

            IN          NS       ns1.a.example.
            IN          MX       10 mail.a.example.
ns1.a.example. IN      A         192.0.2.54
ns1.a.example. IN      A         2001:db8:53::53
mail.a.example. IN     A         192.0.2.57
mail.a.example. IN     AAAA      2001:db8:53::25
www.a.example. IN     A         192.0.2.58
mail.a.example. IN     AAAA      2001:db8:53::80

```


2. ピリオドを忘れた - [回答編] ~dig の場合~

```
$ dig a.example. MX @127.0.0.1

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> a.example. MX @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8642
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;a.example.                IN

;; ANSWER SECTION:
a.example.                15      IN      MX      10      mail.a.example.a.example.

;; AUTHORITY SECTION:
a.example.                8       IN      NS      ns1.a.example.

;; ADDITIONAL SECTION:
ns1.a.example.           8       IN      A       192.0.2.54

;; Query time: 4 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Jul 18 20:47:26 2013
;; MSG SIZE  rcvd: 92
```

mail.a.example.a.example.

mail.a.example.a.example.

3. SOAのシリアルを上げ忘れた

```

$ORIGIN      a.example.
$TTL        86400
@           IN           SOA      ns1.example.  root.example.jp. (
                                2014062001
                                3600
                                900
                                64800
                                3600
                                )

```

- マスター / スレーブ を構築している場合、**スレーブが情報更新されない**
- 権威DNSサーバーによって返す応答が違う
 - ゾーンデータの新鮮さは、**シリアルの値の大小のみ**によって判断
 - ゾーン情報を書き換えた後は、
\$ dig @(スレーブ) domain SOA +norec
を実行
⇒ マスターと一致していることを確認！

更新したよ！

シリアルあがってない。
更新なくてよいか。



情報
新

master
(ns1.example.jp)



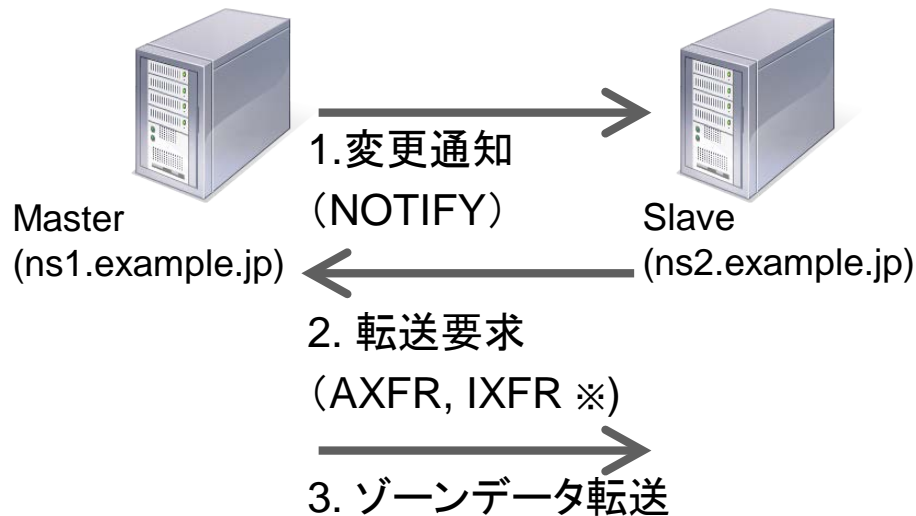
情報
旧

slave
(ns2.example.jp)

[補足] NOTIFY (変更通知) によるゾーン情報の更新

- 権威DNSサーバーを複数設置

- ゾーン情報を全て手作業で更新するのは大変！
 - 1台をマスターにして、残りのマシンもこれに追従させる
- NOTIFY (変更通知) による zone 情報の更新



外部からは、権威DNSサーバーのプライマリとセカンダリは**区別されない**。本スライドでのマスター/スレーブは、ゾーン転送のときにのみに用いる概念

もし、「シリアルの上げ忘れ」で、スレーブへのゾーン転送が失敗していると……

※ AXFRはゾーンデータを全て転送、IXFR は差分転送

4. SOAのシリアルを上げ損ねた

- シリアルを上げよう
 - “YYYYMMDDnn”(nn：連番)だから、“2014062601”.....
 - “2114062601”にしちゃった
- シリアル戻そう！
 - ん？**シリアルを変更するには加算**するしかないのでは……？



シリアル巻き戻し、2回上げテクニックを使う

4. SOAのシリアルを上げ損ねた – シリアルの巻き戻し

- シリアルを2度上げる

1. マスターで「現在の値 + $2^{31}-1 (=2147483647)$ 」をセット、反映

- 例) $2114062601 + 2147483647 = 「4261546248」$ をセット

2. スレーブへの反映/確認

- `$ dig @(スレーブ) domain SOA +norec`

3. マスターで「目的の値」をセット

- 例) 「2014062601」をセット

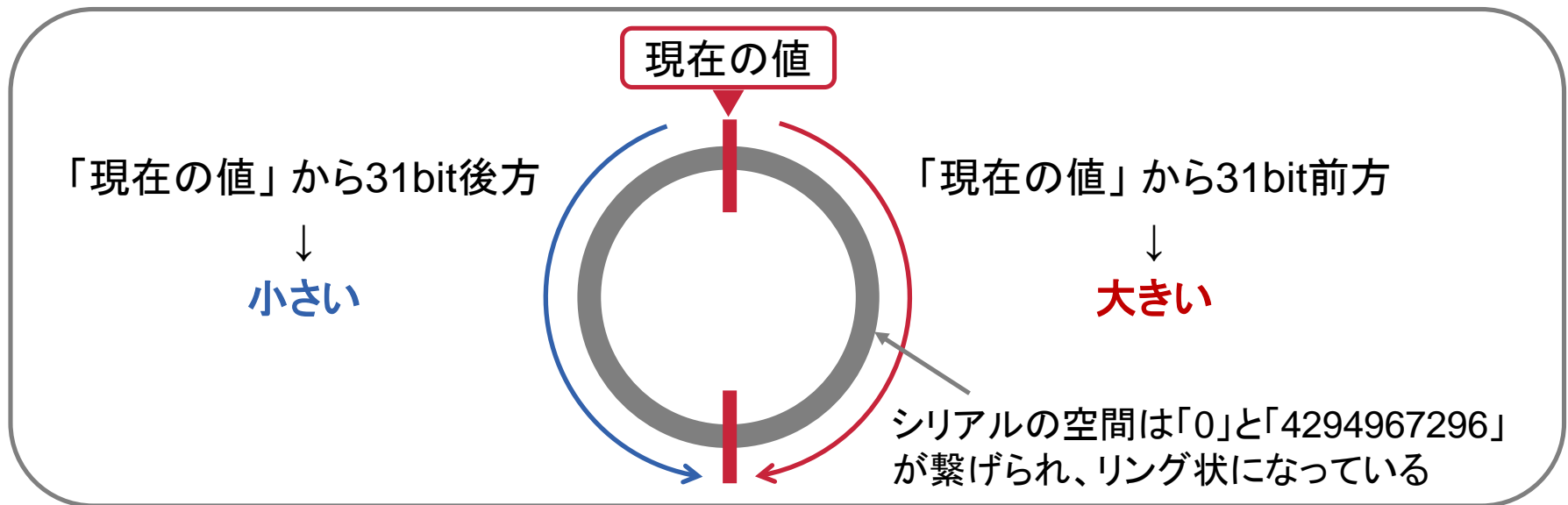
4. スレーブへの反映/確認

- もう一度 dig して目的のシリアル番号になっていることを確認

[補足] なぜシリアルを巻き戻せる？

- 「2114062601」を「2014062601」に戻す(1)

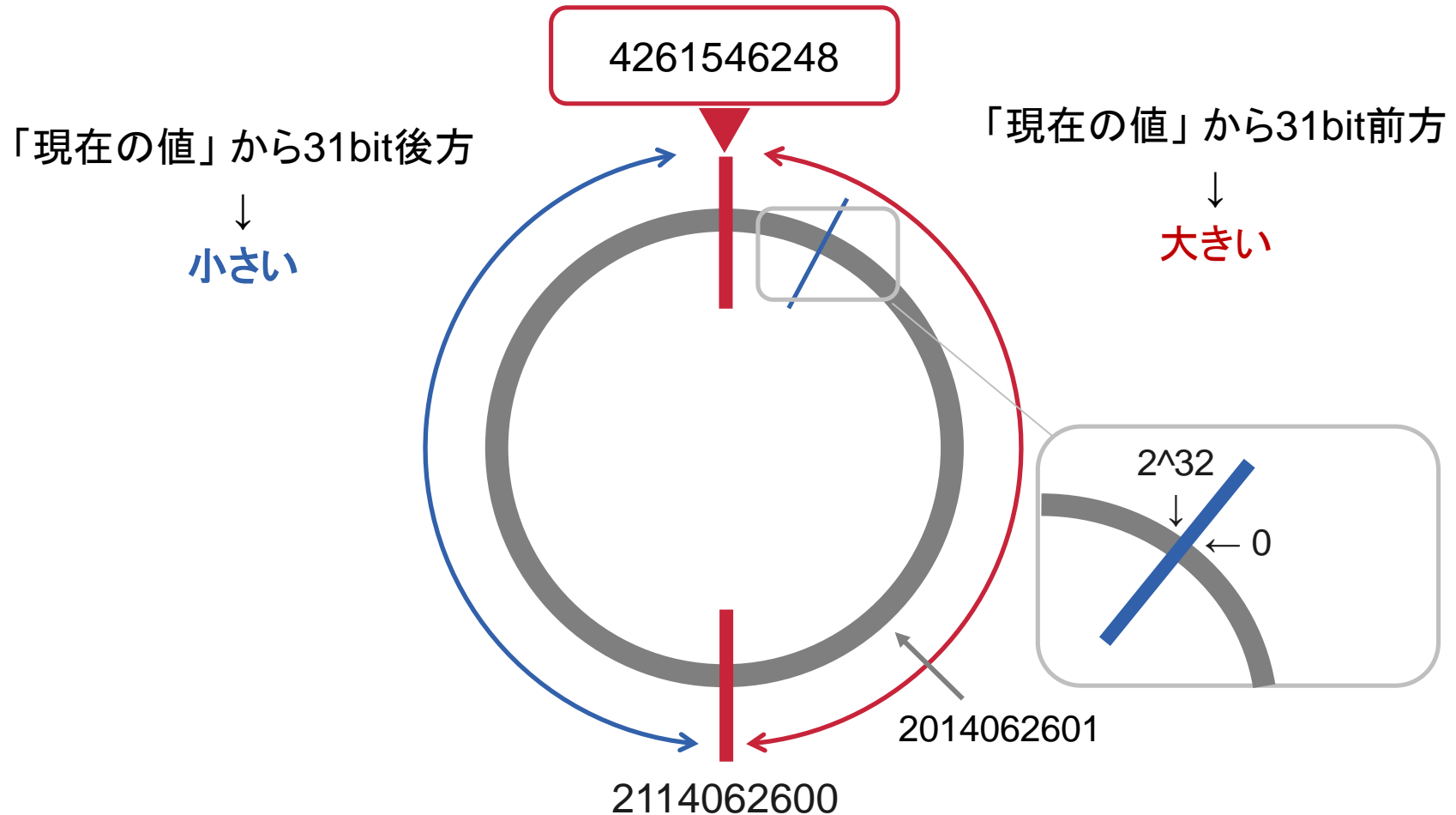
- シリアルは「常に加算」だから巻き戻せないのでは？
 - SOAシリアルの数値空間は、0と 2^{32} が接続されたリング状
 - SOAシリアルは、現在値から相対的に大小判断が行われる



※ RFC 1982 - Serial Number Arithmetic(シリアル番号の計算)

[補足] なぜシリアルを巻き戻せる？

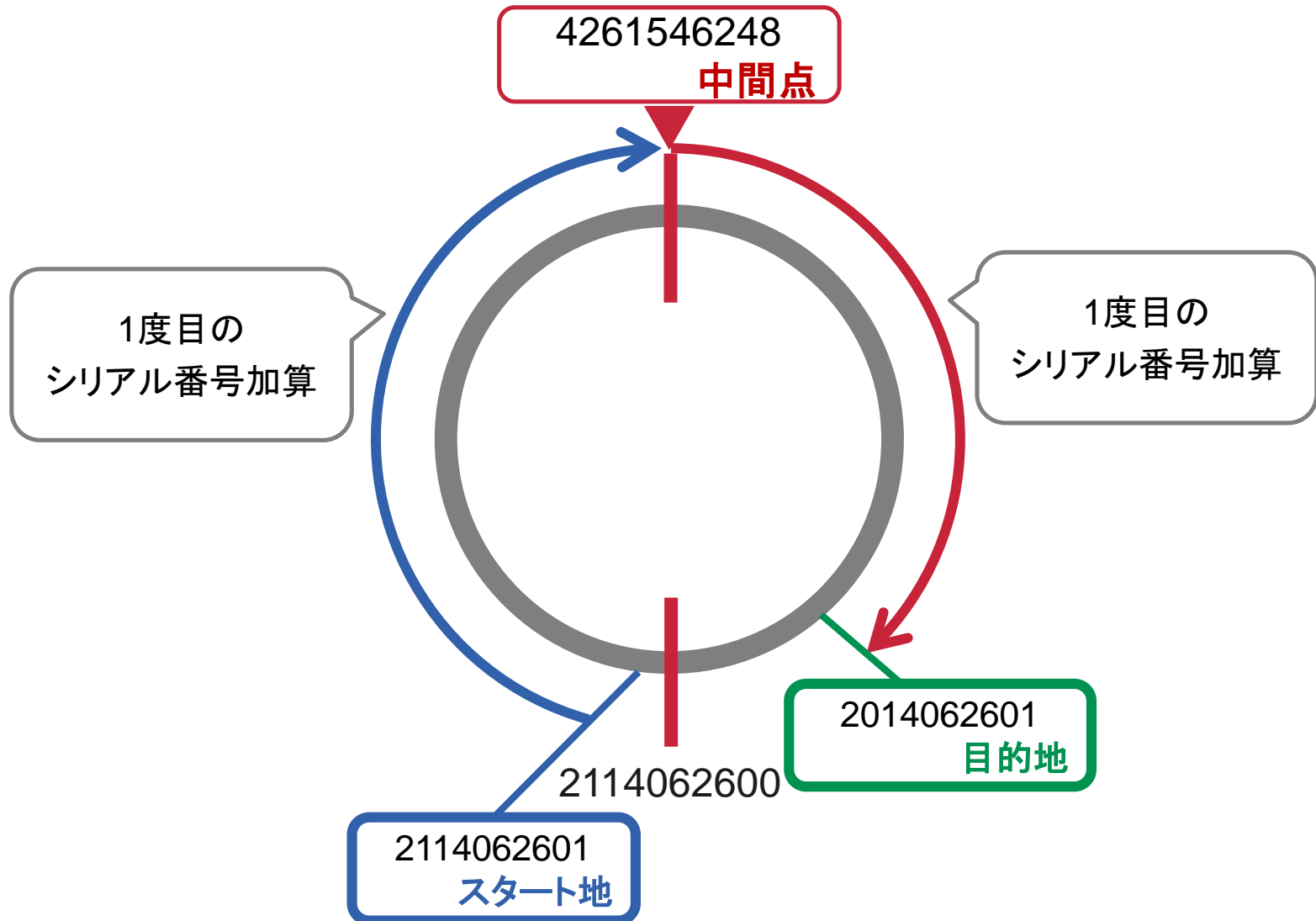
- 「2114062601」を「2014062601」に戻す(2)



以上より、シリアル上は $4261546248 < 2014062601$

[補足] なぜシリアルを巻き戻せる？

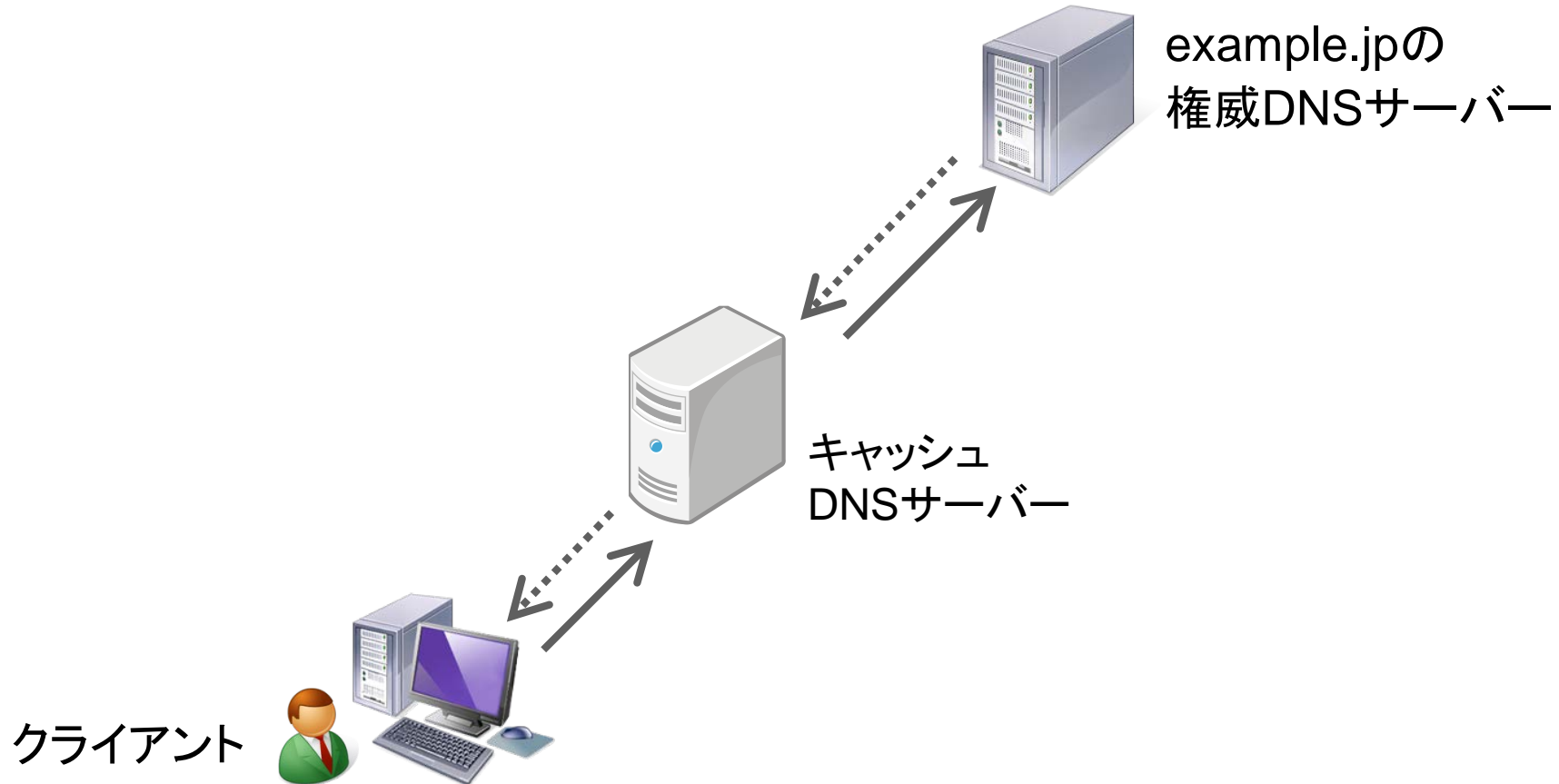
- 「2114062601」を「2014062601」に戻す (3)



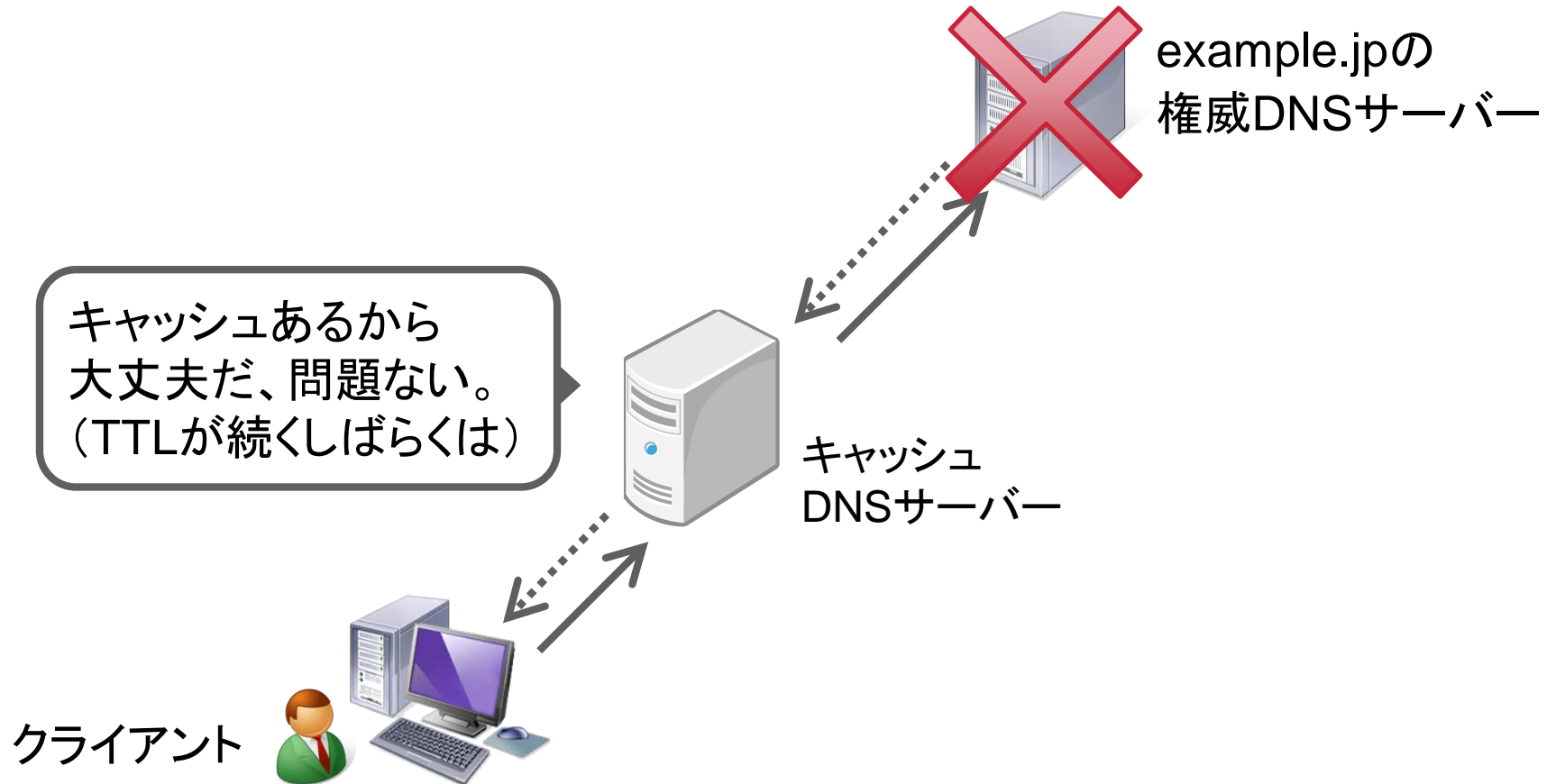
DNSTラブル事例

▶ B. 名前が引けない

1. 権威DNSサーバーがダウンしている

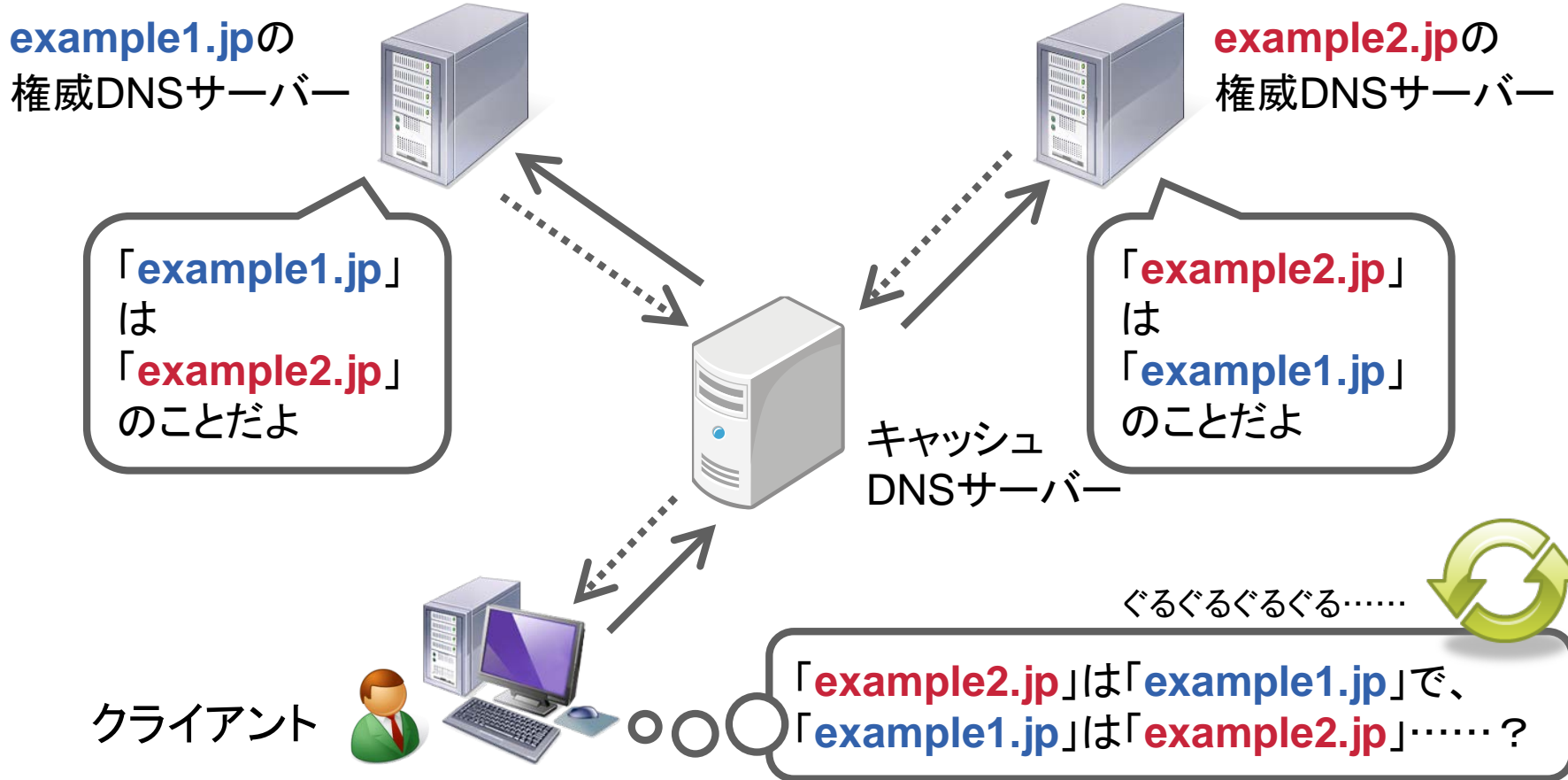


1. 権威DNSサーバーがダウンしている



キャッシュDNSサーバーのキャッシュで、気づくのが遅れることも……

2. CNAME の循環



アプリケーションによってはエラーが出たり、そのまま固まったり……

2. CNAME の循環 - dig の実行結果

```
$ dig cname.a.example. @127.0.0.1

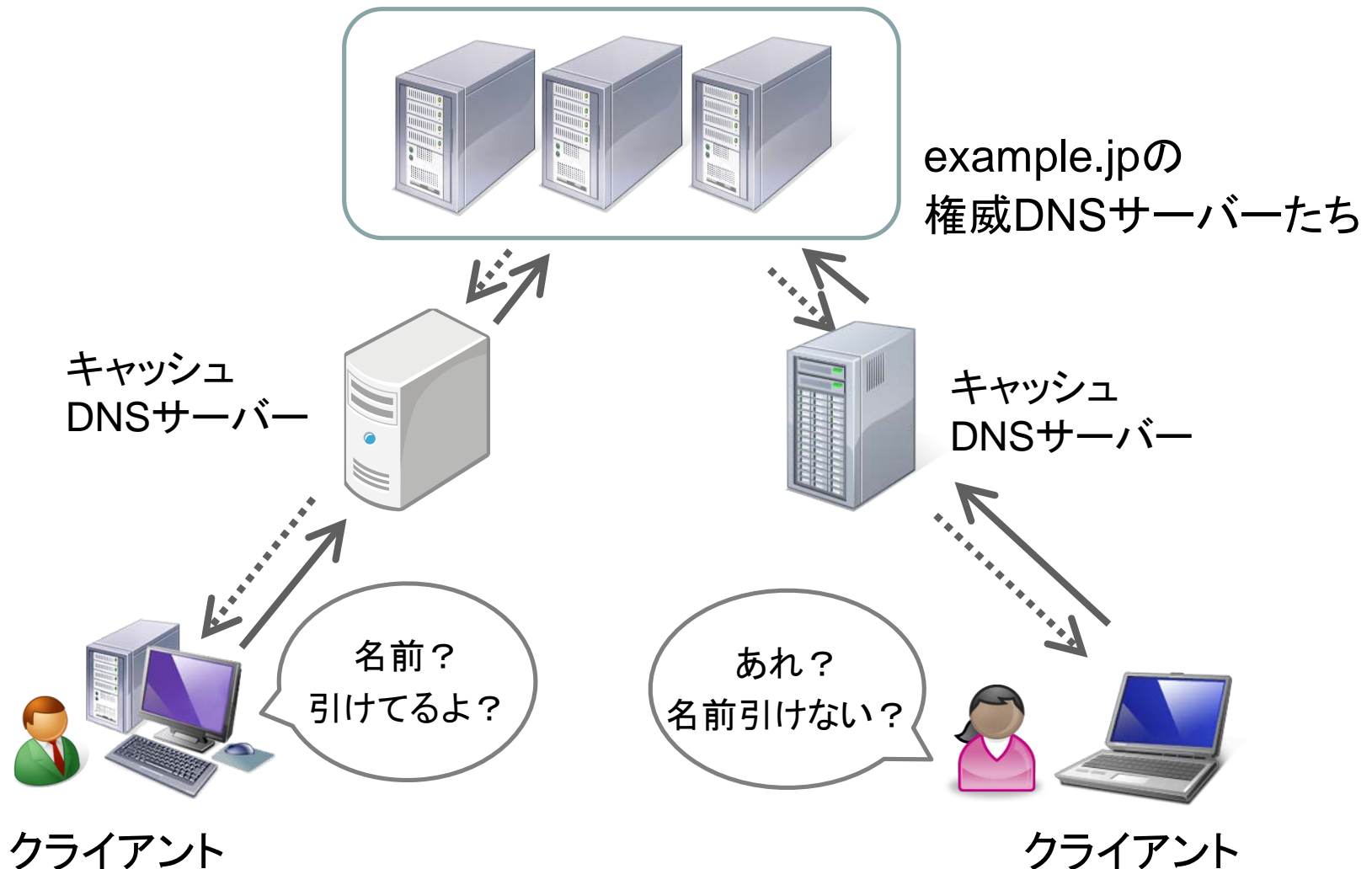
; <<>> DiG 9.8.4-rpz2+r1005.12-P1 <<>> cname.a.example. @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20338
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;cname.a.example.          IN      A

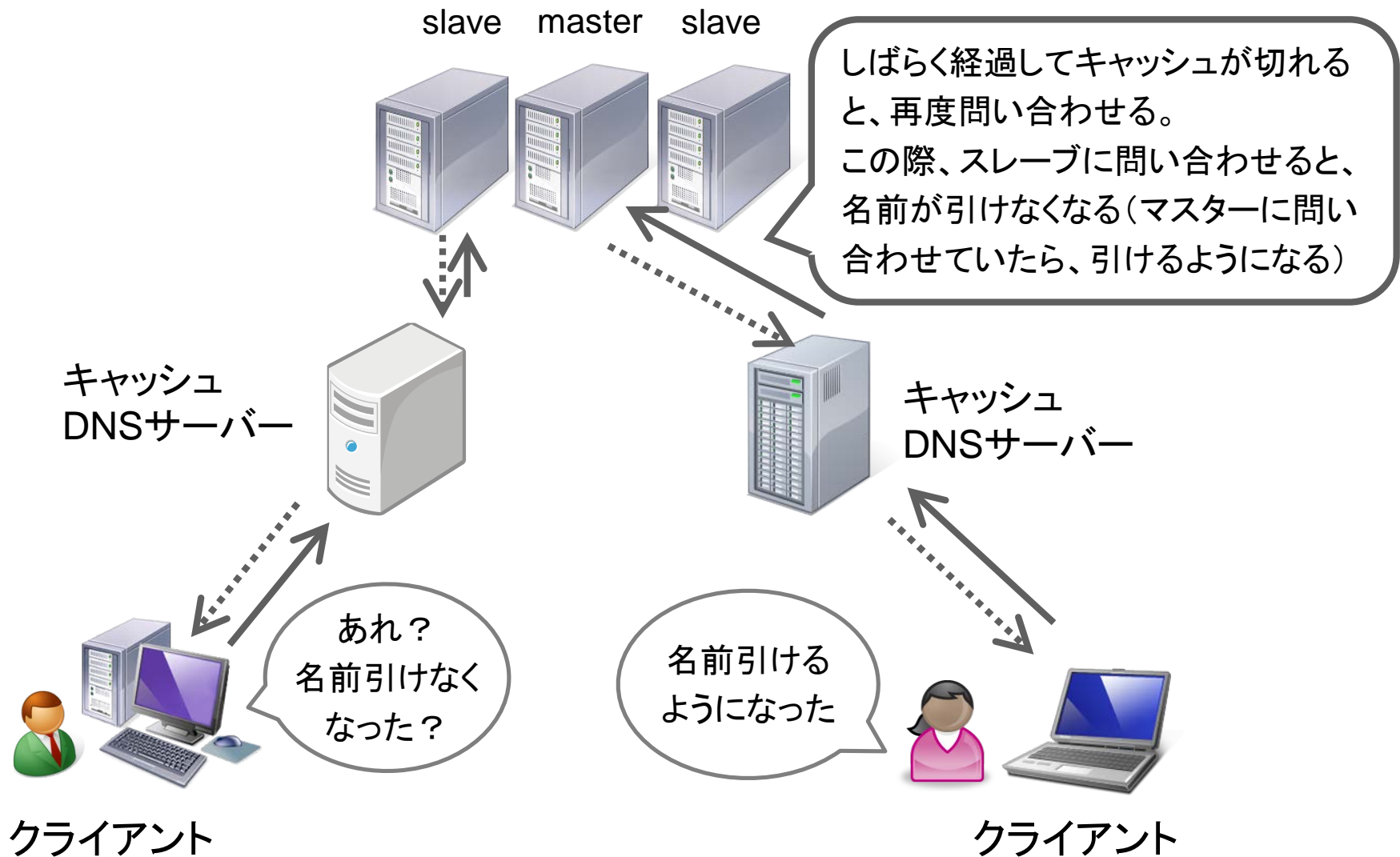
;; ANSWER SECTION:
cname.a.example.          15      IN      CNAME   cname.b.example.
cname.b.example.          15      IN      CNAME   cname.a.example.

;; Query time: 15 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Jul 18 20:40:32 2013
;; MSG SIZE  rcvd: 69
```

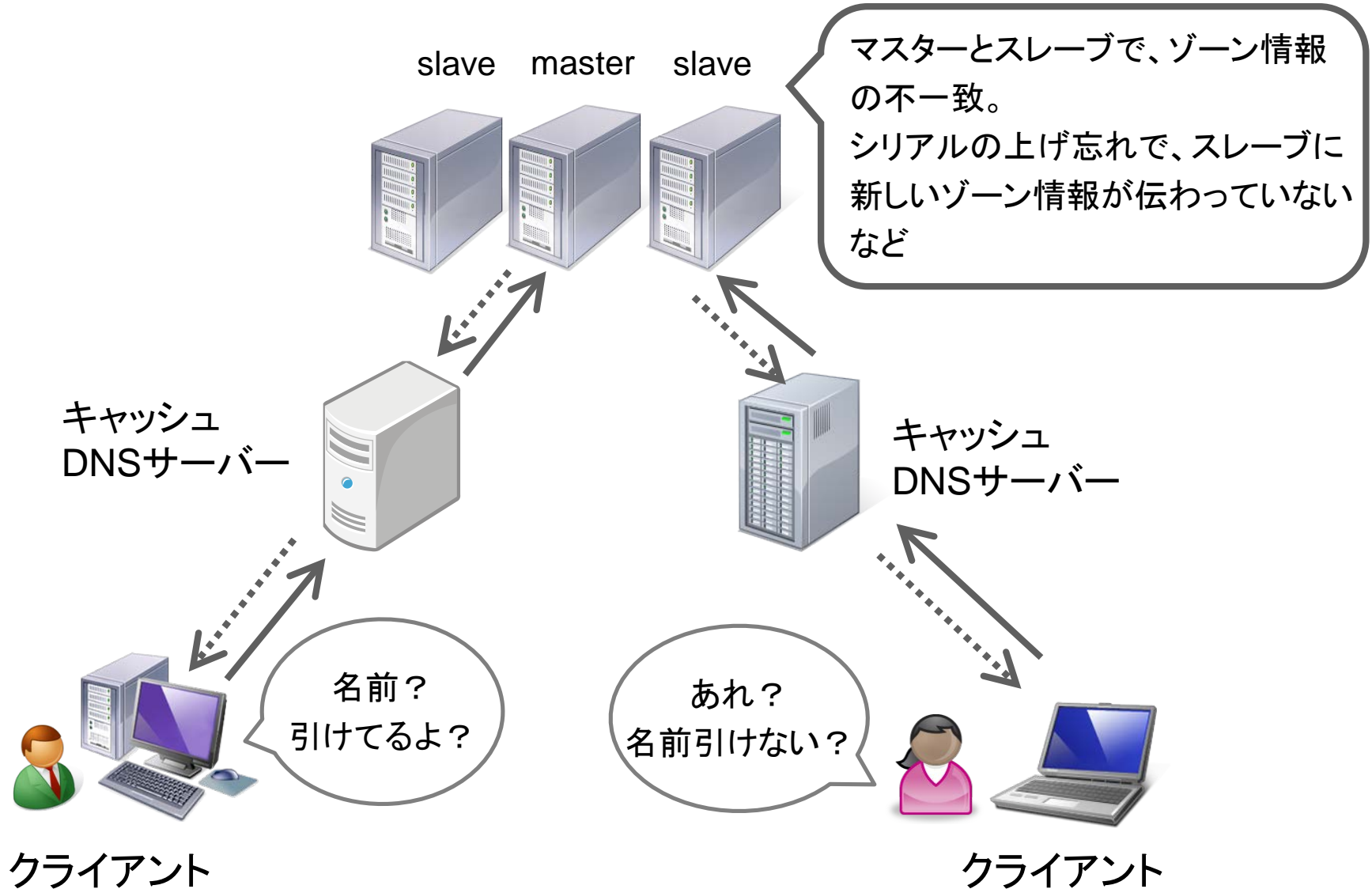
3. ふぞろいのゾーン情報たち(1)



3. ふぞろいのゾーン情報たち(2)



3. ふぞろいのゾーン情報たち(2)



DNSトラブル事例

▶ C.名前を引くのに時間が掛かる

1. TCPフォールバック

- DNS の 512bytes の壁

- 応答はできるだけ 512 bytes 以下に収め、UDP 一発で送信できるのがよい
- 近頃のトレンド: 応答サイズの増大
 - IPv6、DNSSEC、spam対策 (SPF情報: TXTレコード)



- どうなる？

- 最初に UDP で問い合わせ、512 bytes に収まらないことが分かたら TCP で再度問い合わせる
 - udp での問い合わせで tc ビットがオンになっている

→再問い合わせの分遅くなる

- 最近は「EDNS0」という仕組みが使われる

[補足] EDNS0とは？ - 背景

- 従来のDNSプロトコルに存在する **512 bytes** の壁
 - UDPによる問い合わせ
 - 応答の大きさの上限
- IPv6やDNSSECの普及に伴う、DNSの応答に含まれる情報量の増加



512 bytes の壁を超えるための仕組み

EDNS0

[補足] EDNS0とは？ - 「512 bytes の壁」の例

```
% dig +ignore ***.com txt
```

(途中略)

不完全な応答をそのまま表示させる

```
; flags: qr aa tc; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
```

(途中略)

応答が 512 bytes を越えたため、切り詰めが発生

```
;; ANSWER SECTION:
```

```
***.com.          300      IN       TXT      "spf2.0/pra ip4: x x
x .xxx.xxx.0/24 ip4:xxx.xxx.xxx.0/24 ip4:xxx.xxx.xxx.0/24
ip4:xxx.xxx.xxx.0/23 ip4:xxx.xxx.xxx.0/24 ip4:xx.xx.xxx.0/23
ip4:xx.xx.xxx.0/24 ip4:xx.xx.xxx.xx/32 ip4:xx.xx.xxx.xxx/32
ip4:xx.xx.xxx.xxx/32 ptr:mx.***.com ?all"
```

```
;; Query time: 180 msec
```

```
;; SERVER: xx.xx.xx.xxx#53(***.***.***.***)
```

```
;; WHEN: Tue Jun 10 16:32:56 2008
```

```
;; MSG SIZE rcvd: 273
```

得られた応答の大きさ

実際には 668 bytes のデータがサーバに存在

→ 従来のDNSプロトコルでは、UDPで
512bytes を越えるデータを送受信できない

[補足] EDNS0とは？ - TCPフォールバックの場合

- 512 bytes の壁を越えるには？
 - TCPフォールバック
 - EDNS0
- TCPで再度同じサーバーに同じデータを要求
 - データの切り詰めをクエリの送信者に通知、TCPで再接続
 - 応答サイズの制限を緩和 65,535 bytes まで OK!
 - DNSができた当初から存在する方法

- 信頼性のある通信路(コネクションを確保)
- 通信の信頼性は高いが、通信にかかる負荷は大きい
- 再接続するため、時間が掛かる

→ 大規模なDNSサーバーでの使用は不向き

[補足] EDNS0とは？ - EDNS0 の場合

- 512 bytes の壁を越えるには？
 - TCPフォールバック
 - **EDNS0**
- DNSプロトコルを改良
 - UDPで大きな応答を受け取れるように
 - 問い合わせ時にUDPで受信できる応答の大きさをサーバへ通知
 - UDPで受信できるサイズを拡張可能

- コネクションの確保を行わず、情報を送信
- 通信の信頼性は低いが、通信にかかる負荷は小さい
- 再接続の必要がないため**応答が速い** (tcpフォールバックと比較)

→ **運用実績のあるUDPをそのまま利用可能**

[補足] EDNS0とは？ - TCPフォールバックとEDNS0

TCPフォールバック

```
% dig ***.com txt  
;; Truncated, retrying in TCP mode.  
(途中略)  
;; Query time: 179 msec  
;; SERVER: ***.***.***.***#53(***.***.***.***)  
;; WHEN: Mon Jun 2 20:31:20 2008  
;; MSG SIZE rcvd: 668
```

切り詰めを実施、TCPで再接続

応答の大きさ

EDNS0

```
% dig ***.com txt +bufsize=4096  
(途中略)  
;; OPT PSEUDOSECTION:  
;; EDNS: version: 0, flags:; udp: 4096  
(途中略)  
;; Query time: 192 msec  
;; SERVER: ***.***.***.***#53(***.***.***.***)  
;; WHEN: Tue Jun 10 17:49:47 2008  
;; MSG SIZE rcvd: 679
```

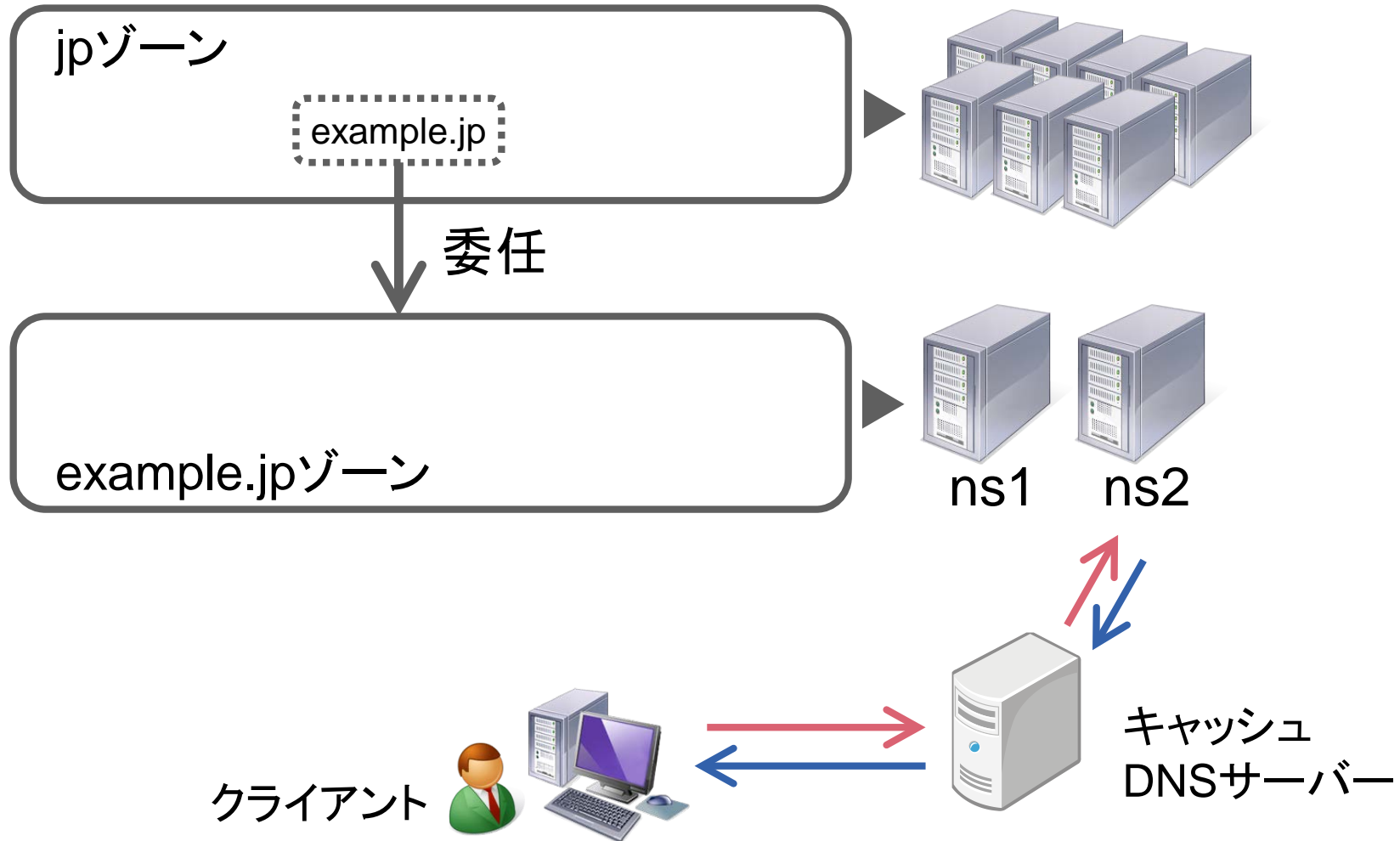
EDNS0 有効

応答の大きさ

→ 上記いずれの場合でも、512 bytes の壁を越えることができています

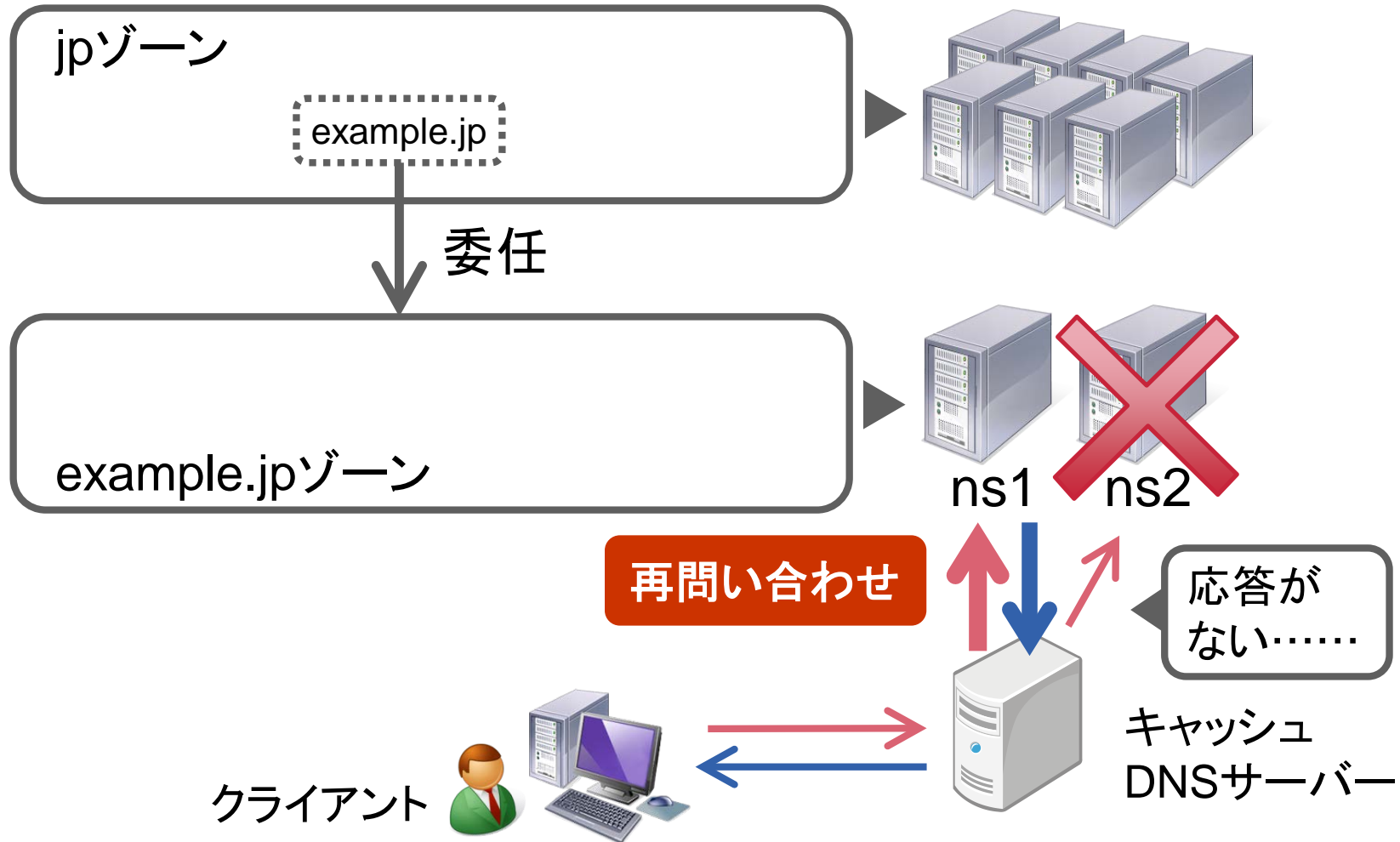
2. 権威DNSサーバーの一部がダウンしている (1/2)

✓ 通常の場合



2. 権威DNSサーバーの一部がダウンしている (2/2)

✓ DNSサーバーの一部がダウンしている場合



2. 権威DNSサーバーの一部がダウンしている (2/2)

✓ DNSサーバーの一部がダウンしている場合

- キャッシュサーバに一度キャッシュされてしまえば、遅延は発生しない
 - 遅延が発生するのは、キャッシュされていないときの問い合わせ
- 今回の例の場合、ns1 にいきなり問い合わせに行ったら、遅延は発生しない
 - 権威 DNS サーバーの選択に、プライマリやセカンダリという概念はない
 - どの権威DNSサーバーに問い合わせに行くかは、各キャッシュDNSサーバーの実装やネットワークの状況に依存
 - ロシアンルーレットのようなもの

気づくのが遅れることも……

まとめ

- どこを調べているのか？を理解しよう
 - 再帰問い合わせ？非再帰問い合わせ？
- 道具の使いかたを知ろう
 - dig は友達
 - nslookupはやめよう
 - Windowsでも動く！
 - @でDNSサーバを指定、+norec オプション
 - 便利なWebサービス
 - DNS可視化の「Squish.net DNS traversal checker」
 - エラーチェックの「dnscheck.jp」
- よくあるトラブル事例
 - まずはログを確認！
 - TCPの53番ポート確認！
 - ファイヤーウォール確認！
 - CNAME 確認！
 - ピリオド確認！
 - シリアル確認！

Q&A

