SIer、Developerから見た BINDからの移行

DNS Summer Day 2018

2018年6月27日

MITSUBISHI ELECTRIC
Changes for the Better
三菱電機インフォメーションシステムズ株式会社

XACK

発表者

名前

佐藤　匠（さとう　たくみ）

所属

三菱電機インフォメーションシステムズ株式会社（MDIS）

仕事内容

SIer

通信キャリア向けネットワークインフラシステム構築

(RADIUS、DHCP、DNS)

MITSUBISHI ELECTRIC
Changes for the Better
三菱電機インフォメーションシステムズ株式会社

XACK

発表者

名前

矢島　崇史（やじま　たかし）

所属

株式会社XACK

仕事内容

通信キャリア向けのソフトウェア開発

(RADIUS、DHCP、DNS)

XACK DNS

・100%自社開発のDNSサーバー

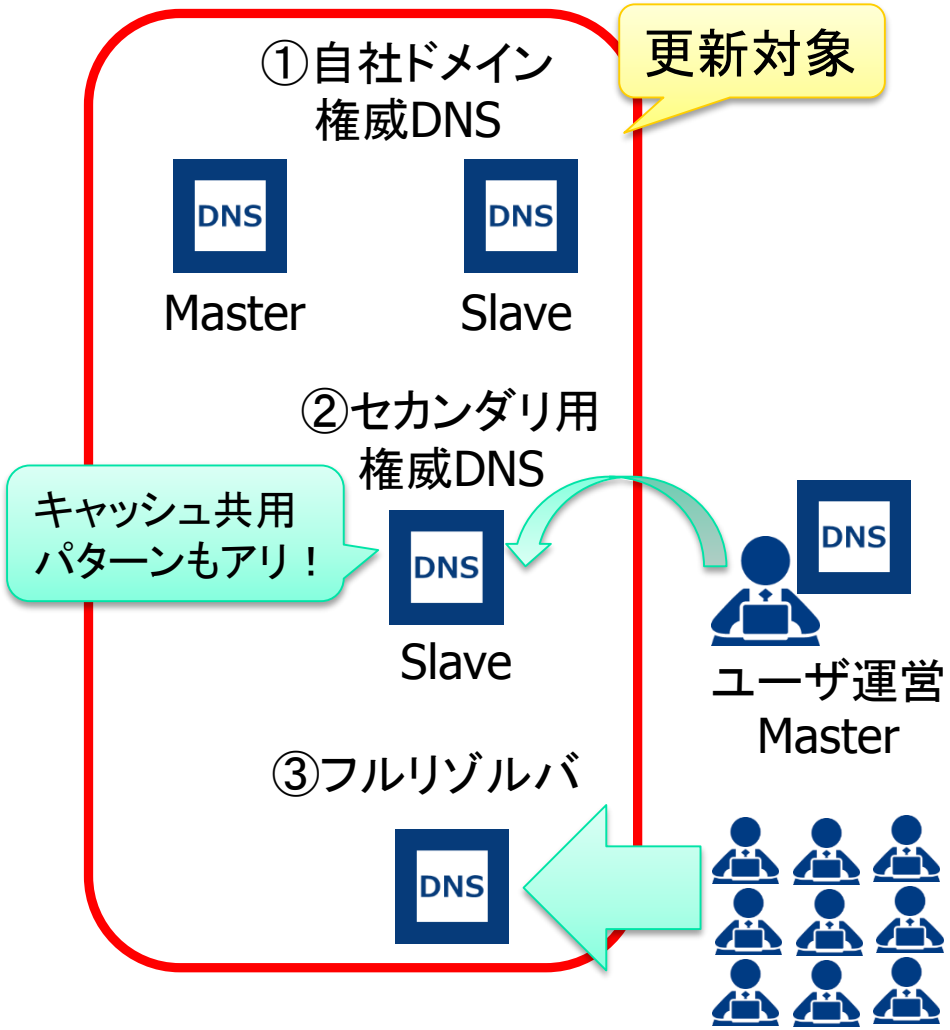
・UNIX系OSで動作するソフトウェア、アプライアンス

・権威サーバー、フルリゾルバー、フォワーダー、etc...

・モジュール化による機能の足し引きが可能

BINDをXACK DNSにしてみよう！

お客様からDNSサーバ更新の提案機会をいただきました。



- 現在はBINDだが、次回はOSS以外がよい
- 権威キャッシュ共用を現IPで継続したい
- ①②③パターンができるDNSソフトウェア

・
・
・
・

そうだ！

XACK DNSなら全部できる！

結果

BINDは懐が深くて

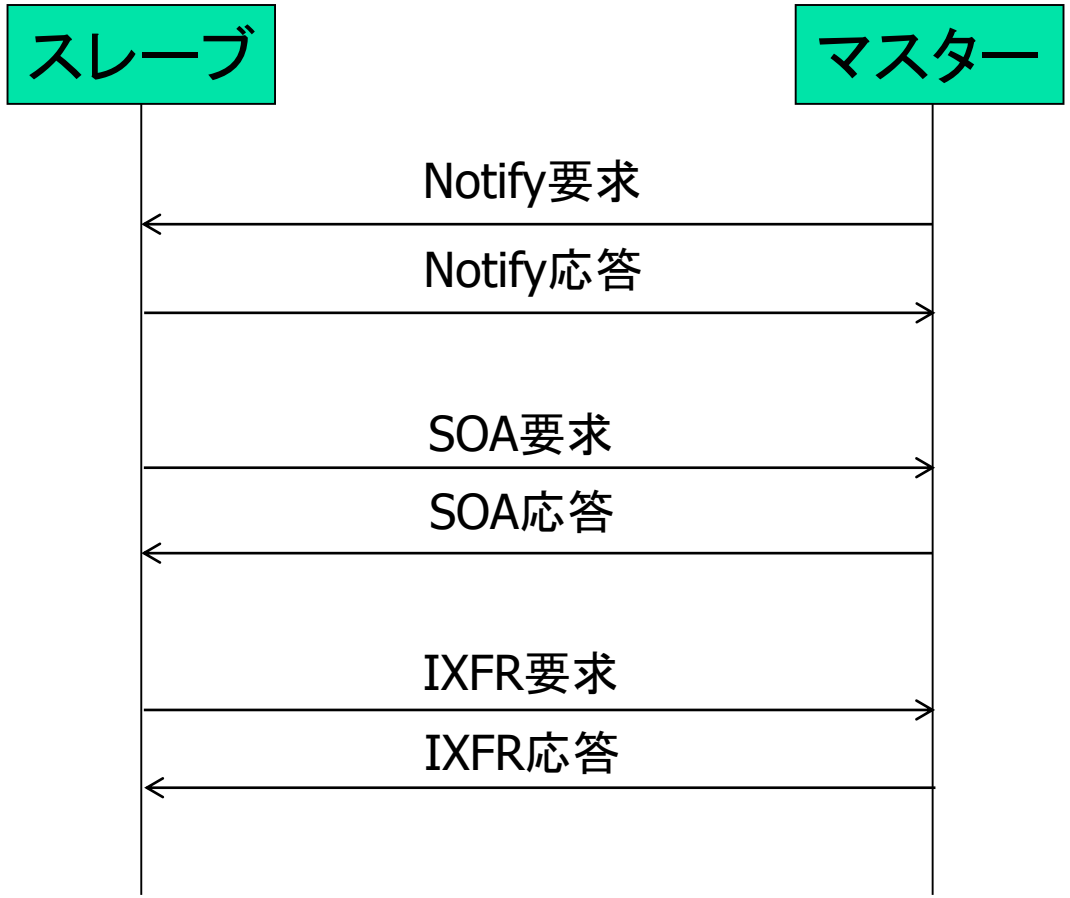
やさしくて

適当で

学ぶことが多くありました！

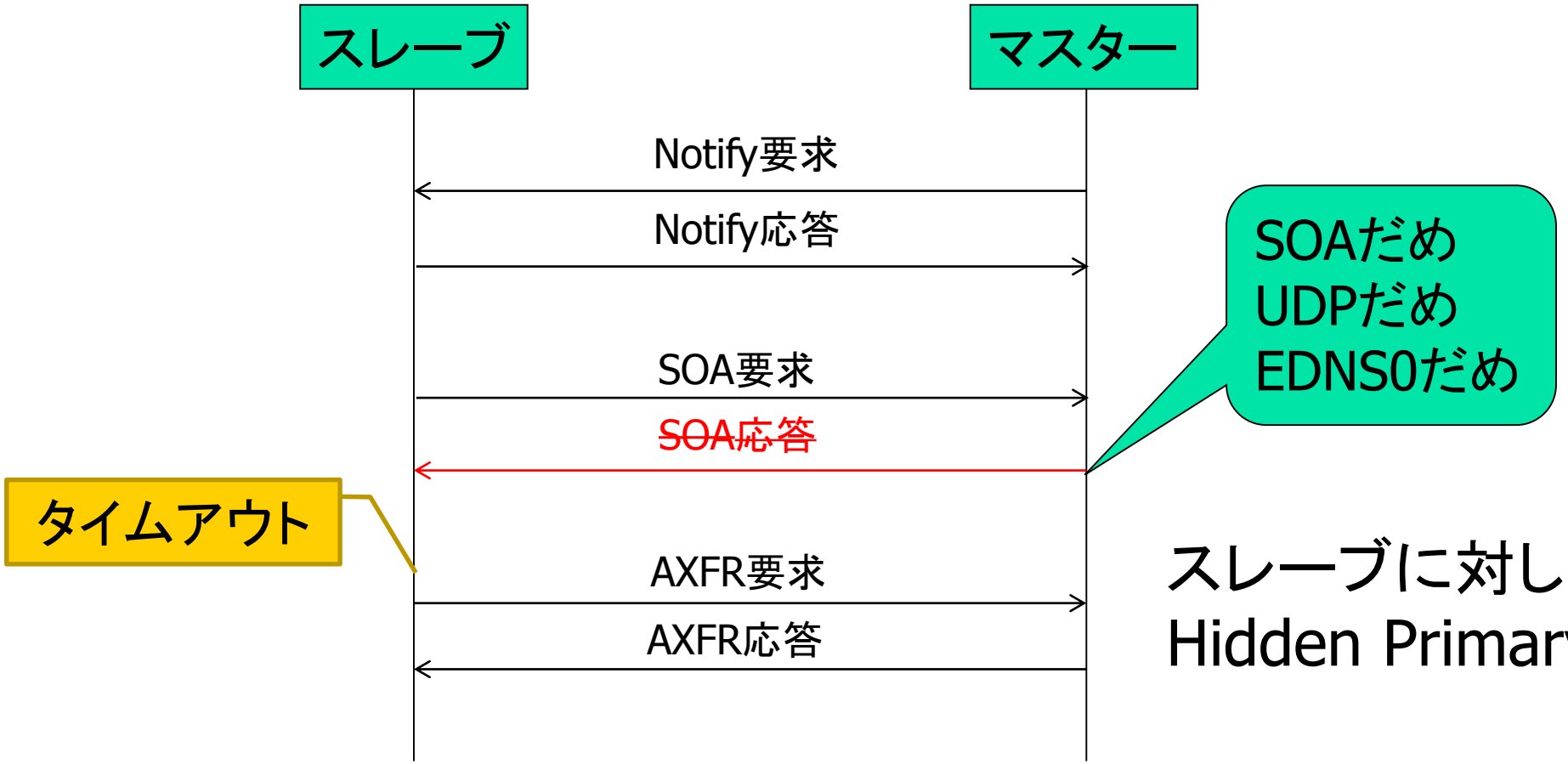
事例1:ゾーン転送

よくあるシーケンス



事例1:ゾーン転送

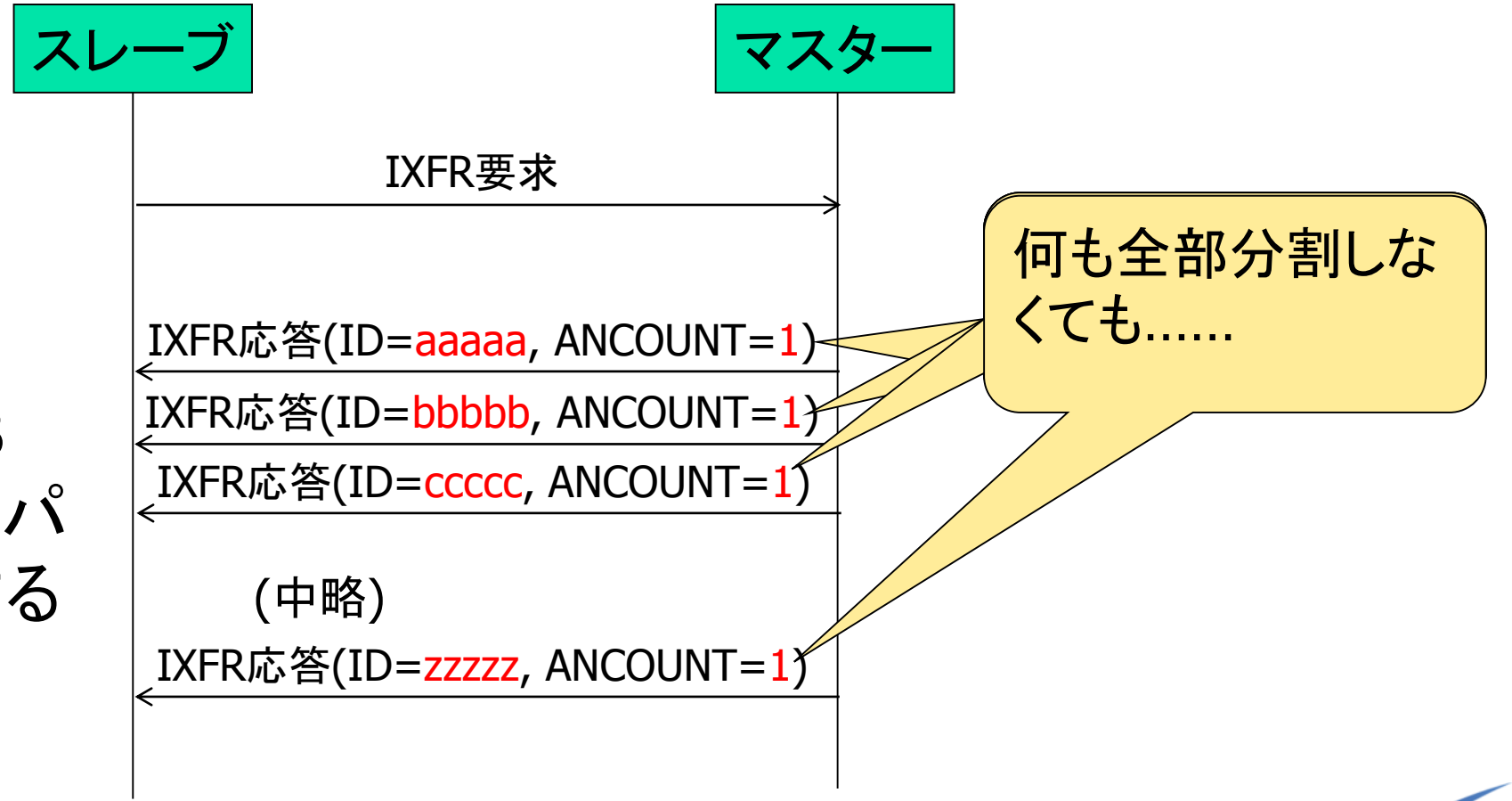
ケース1: 色々応答してくれない



事例1:ゾーン転送

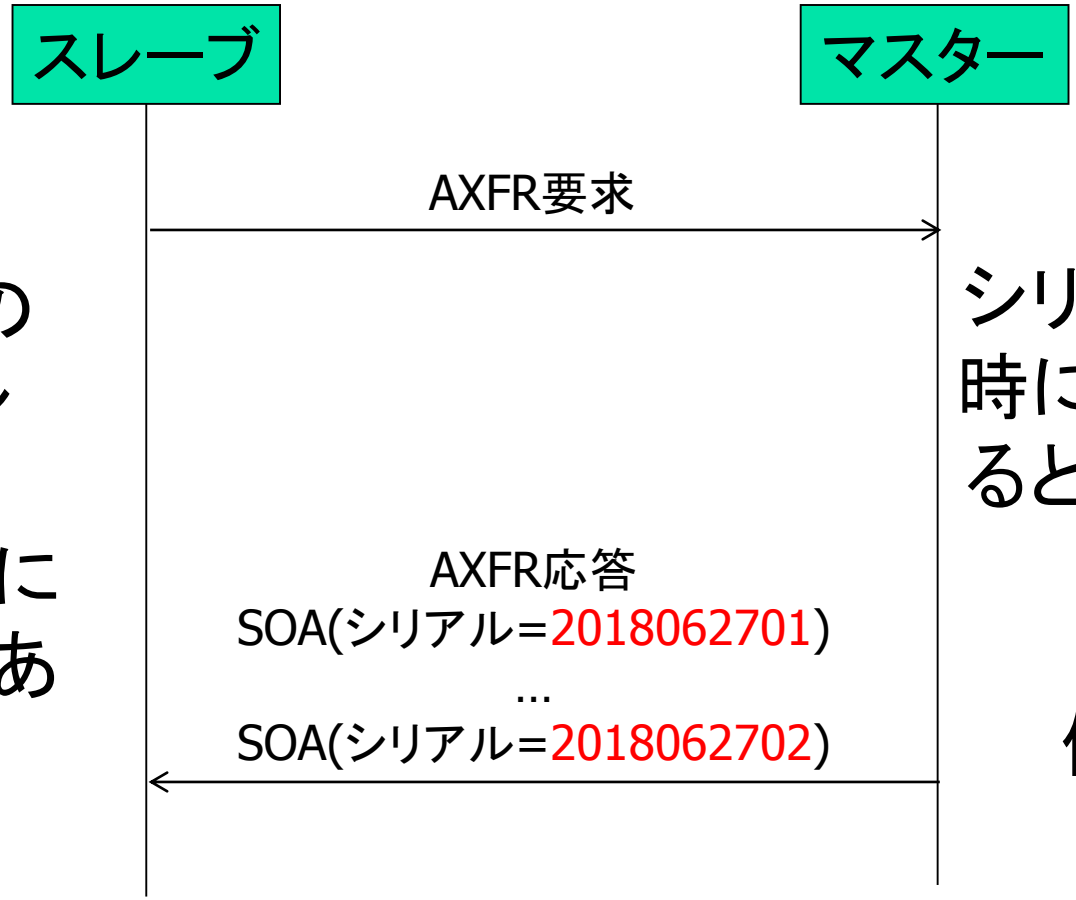
ケース2: RR毎にDNSパケットを分割する。ID違いで

DNSパケットの
最大サイズは64KB
⇒64KB超のゾーンはパケット分割して送信する



事例1:ゾーン転送

ケース3: AXFR応答の最初と最後のSOAが違う

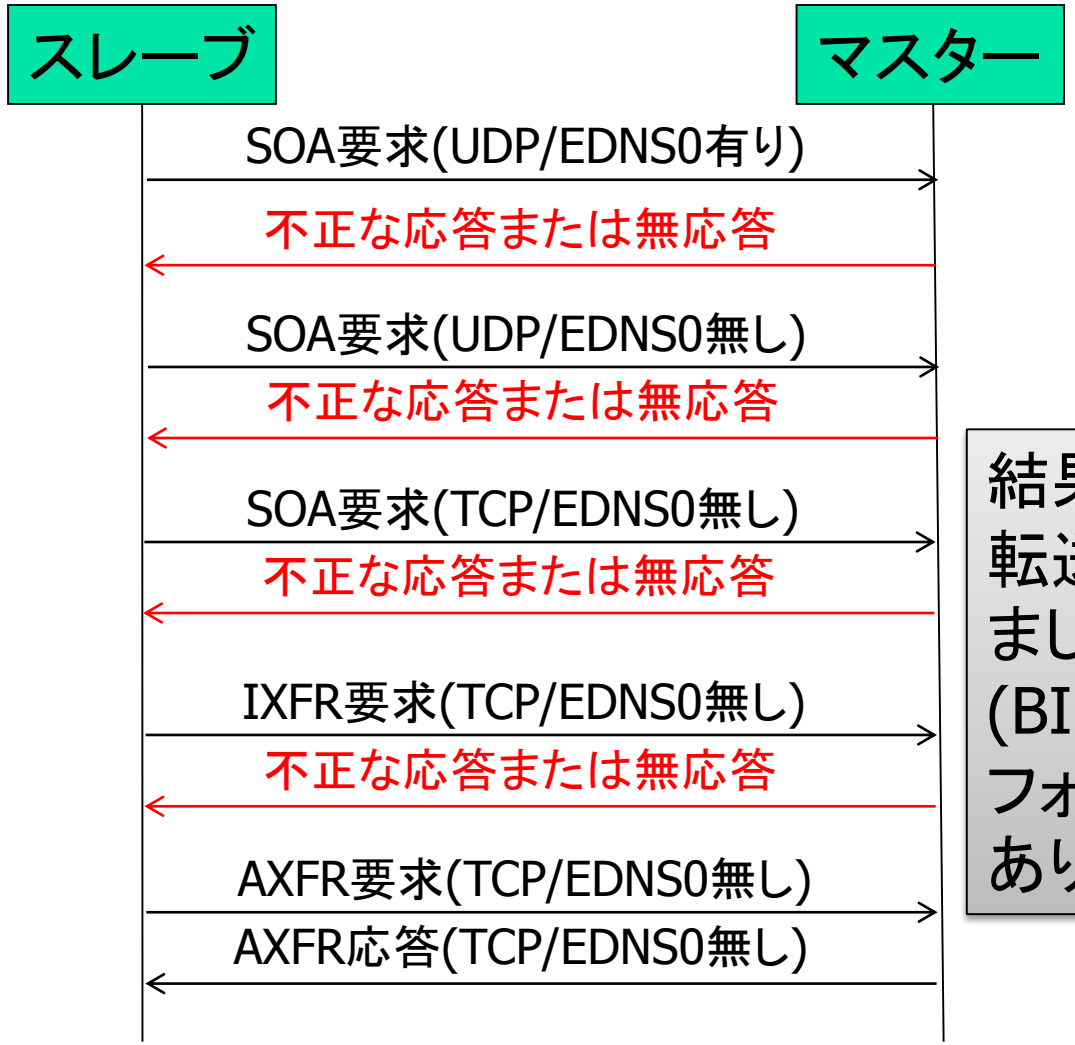


分割されたパケットの先頭、終端はSOAレコードで判別する
⇒一度のゾーン転送に2つのSOAレコードがある

シリアルは差分ゾーン転送時に重要な情報、値が異なると出来上がるゾーンが異なってしまうかも
⇒どっちのシリアル使えばいいんです？

事例1:ゾーン転送

対処: 色々失敗してもフォールバック、応答のチェックを許容的に



結果としてBINDよりもゾーン転送に対応できるようになりました。
(BINDはIXFRからAXFRにはフォールバックしないパターンあり)

事例2:ゾーンファイル読み込み

ケース4: 委任先のグルーがない

```
$ORIGIN example.com.
$TTL 300
@       IN      SOA     (SOA RDATA略)
                NS      ns1
ns1             A       192.0.2.1
sub             NS      no-glue
;no-glueのAレコードはなし
```

・委任先内部名のグルーがないと委任先の権威サーバーのIPアドレスが分からずその先の問い合わせができない。

・設定でそのようなゾーンファイルを許容できる機能を追加(推奨しません)

事例2:ゾーンファイル読み込み

ケース5: TTLが10進数でない

RFC1035 5.1項では以下のように書かれています。

> The RR begins with optional TTL and class fields, followed by a type and RDATA field appropriate to the type and class. Class and type use the standard mnemonics, TTL is a decimal integer.

→TTLは10進数の整数である。

しかし色々なゾーンファイルを見てみると...

```
$TTL 1w3d12h59m1s
```

```
$TTL 1m1m1m1m1m
```

```
$TTL 39m421S899s981M1h4M9H187232131s
```

SOAのRDATAでこんなの見たことある！→TTLでも許容するようにしました。

事例3:この応答の違い、分かりますか？

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21963
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            12715   IN      A
93.184.216.34

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 15 11:35:35 2018
;; MSG SIZE  rcvd: 45
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21963
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            12715   IN      A
93.184.216.34

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 15 11:35:35 2018
;; MSG SIZE  rcvd: 56
```

事例3:メッセージサイズが違う！

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
21963
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY:
0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            12715   IN      A
93.184.216.34

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 15 11:35:35 2018
;; MSG SIZE  rcvd: 45
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
21963
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY:
0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            12715   IN      A
93.184.216.34

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 15 11:35:35 2018
;; MSG SIZE  rcvd: 56
```

事例3:メッセージ圧縮

同一の名前が存在する場合、他方を参照するポインターを設定することで圧縮することが出来ます。

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21963
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            12715   IN      A       93.184.216.34

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 15 11:35:35 2018
;; MSG SIZE  rcvd: 45
```

事例3:メッセージ圧縮

ケース6:後方参照圧縮

```
$ORIGIN example.jp.
$TTL 3600
@               IN      SOA     ns1.test.jp.  a.test.jp.  (
                        3147483648      ; serial
                        3000            ; refresh
                        3000            ; retry
                        100             ; expire
                        3000            ; minimum
                        )
                        PTR     foo.bar.
                        NS      a.foo.bar.
                        NS      b.foo.bar.
```

→ “example.jp.”のPTRレコードを問い合わせると…

事例3:メッセージ圧縮

ケース6:後方参照圧縮

drillの場合

```
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 44216
;; flags: qr aa ; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;; example.jp.   IN      PTR

;; ANSWER SECTION:
example.jp.      3600    IN      PTR     foo.bar.

;; AUTHORITY SECTION:
example.jp.      3600    IN      NS      a.foo.bar.
example.jp.      3600    IN      NS      b.foo.bar.

;; ADDITIONAL SECTION:

;; Query time: 0 msec
;; SERVER: 127.0.0.1
;; WHEN: Wed Dec 20 17:17:16 2017
;; MSG SIZE  rcvd: 81
```

事例3:メッセージ圧縮

ケース6:後方参照圧縮

digの場合

```
;; Got bad packet: bad compression pointer
81 bytes
2a d4 84 80 00 0c 00 01 00 02 00 00 07 65 78 61          *............exa
6d 70 6c 65 02 6a 70 00 00 01 00 01 c0 0c 00 0c          mple.jp.........
00 01 00 00 0e 10 00 04 01 62 c0 3a c0 0c 00 02          .........b.:....
00 01 00 00 0e 10 00 0b 01 61 03 66 6f 6f 03 62          .........a.foo.b
61 72 00 c0 0c 00 02 00 01 00 00 0e 10 00 02 c0          ar..............
28                                                       (
```

事例3:メッセージ圧縮

ケース6:後方参照圧縮

```
;; ->>HEADER<<- opcode: QUERY, rcode: NOERR
;; flags: qr aa ; QUERY: 1, ANSWER: 1, AUTH
;; QUESTION SECTION:
;; example.jp.   IN      PTR

;; ANSWER SECTION:
example.jp.      3600    IN      PTR     foo.bar.

;; AUTHORITY SECTION:
example.jp.      3600    IN      NS      a.foo.bar.
example.jp.      3600    IN      NS      b.foo.bar.

;; ADDITIONAL SECTION:
```

(2)PTRは後方にある1番目のNSを参照
⇒後方参照

(1)2番目のNSは前方にあるPTRを参照

→BINDは(2)の解釈に失敗しますが、Unboundは対応しているようです。

事例4:誰も守っていないRFC

ケース7: 応答のEDNS0のサイズ

RFC6891 6.1.2項では以下のように記載されています。

The fixed part of an OPT RR is structured as follows:

Field Name	Field Type	Description
NAME	domain name	MUST be 0 (root domain)
TYPE	u_int16_t	OPT (41)
CLASS	u_int16_t	requestor's UDP payload size
TTL	u_int32_t	extended RCODE and flags
RDLEN	u_int16_t	length of all RDATA
RDATA	octet stream	{attribute,value} pairs

OPT RRのCLASSフィールドには「要求者」、つまるところクエリを送信した側のUDPペイロードサイズを設定するように定められているようです

(昔はsender's UDP payload sizeでした)。

事例4:誰も守っていないRFC

ケース7: 応答のEDNS0のサイズ

尚、実際の所誰も守っていません。皆さん自身のサーバーの設定を返すようです。

```
$ dig +norec @198.41.0.4 +bufsize=1024 a.root-servers.net. A; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5940
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 26

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
  ...
```

これらの事例を受けて

RFCに書かれていること、書かれていないこと含めて様々な

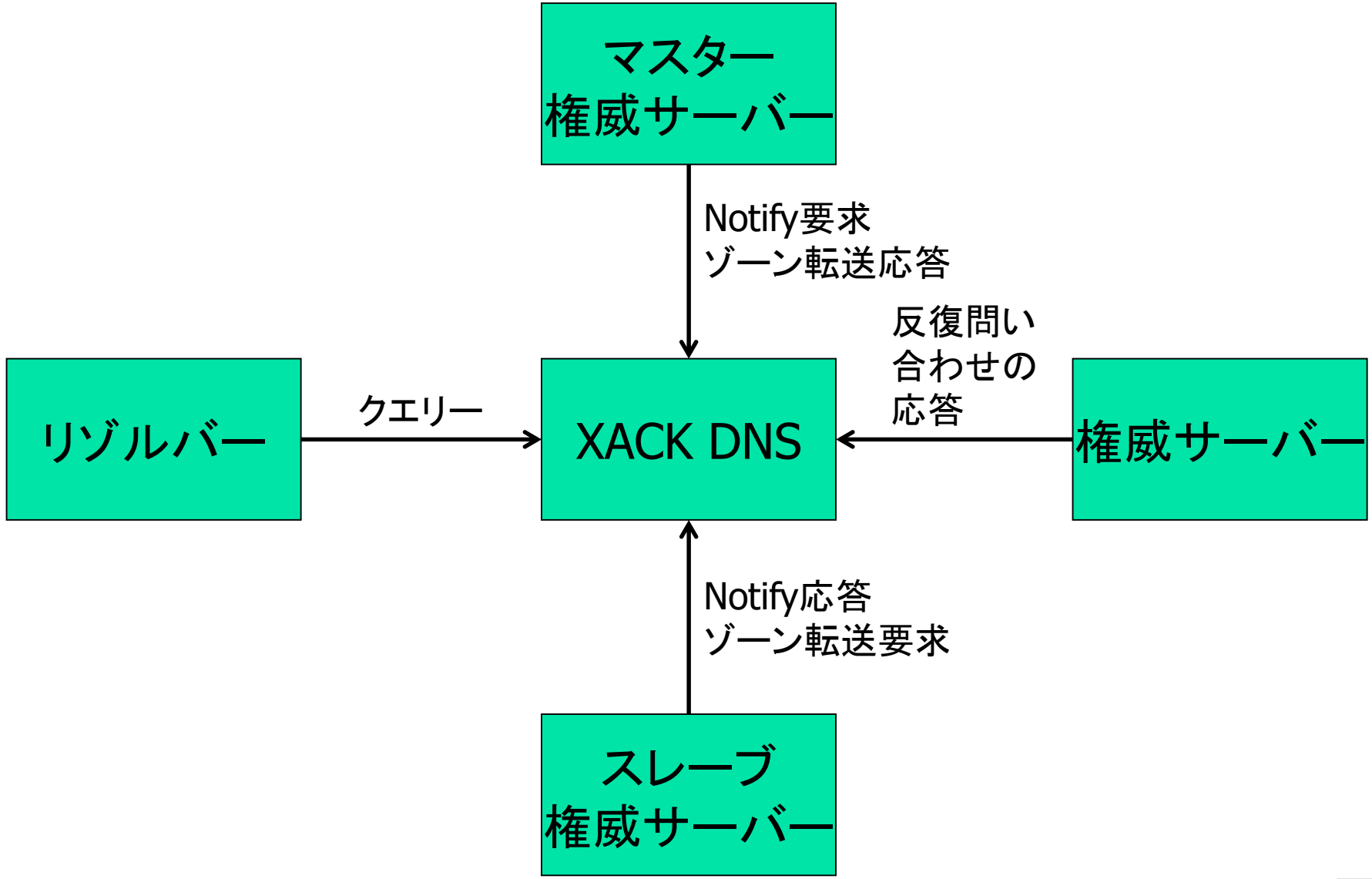
想定外の動作、差分があることが判明

→差分を洗い出したい！

どうやって？

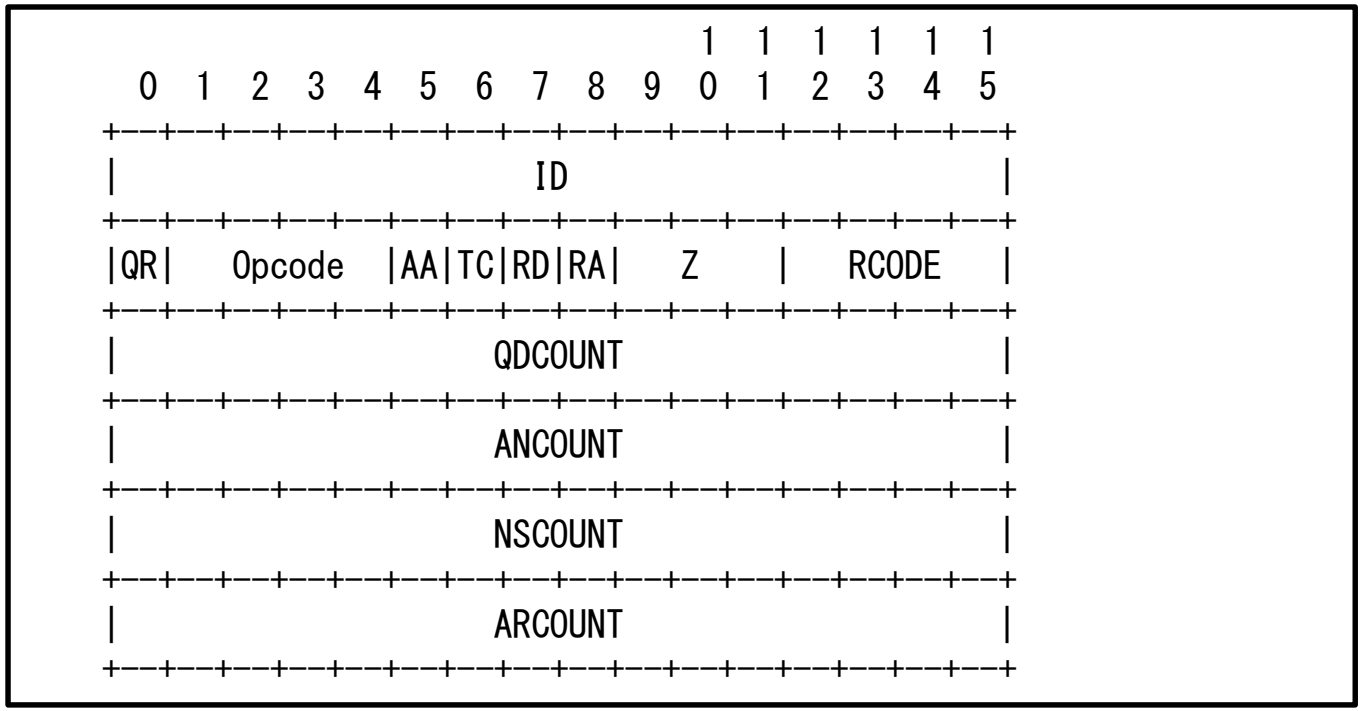
→全通り試せばいい(白目)

対象となる通信


マスター
権威サーバー
Notify要求
ゾーン転送応答
反復問い
合わせの
応答
リゾルバー
クエリー
XACK DNS
権威サーバー
Notify応答
ゾーン転送要求
スレーブ
権威サーバー

やったこと

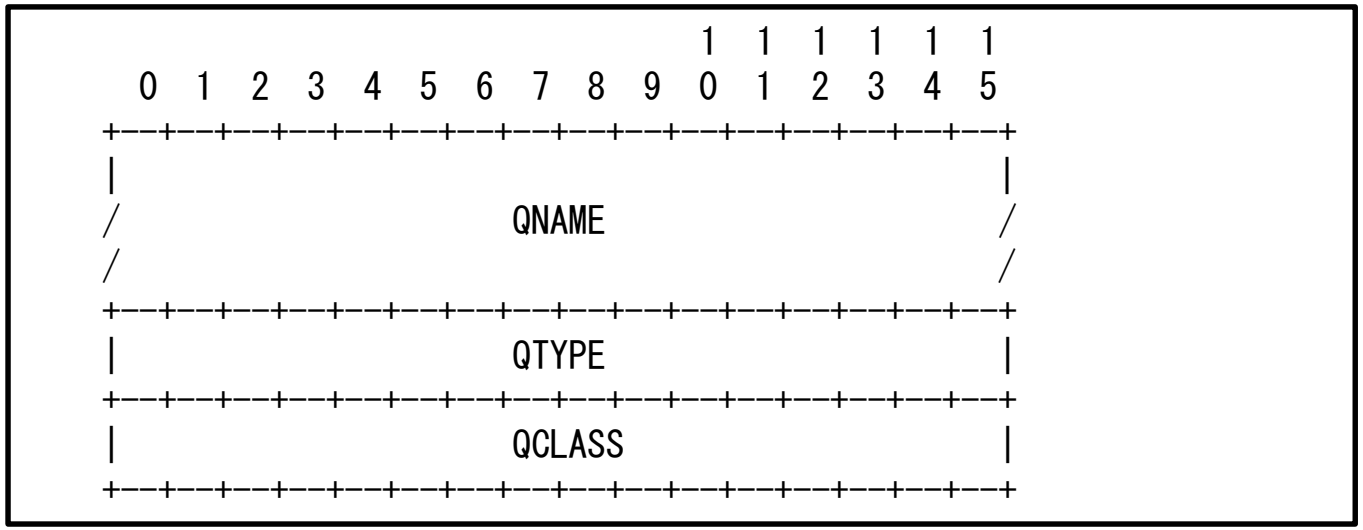
ヘッダーについては概ね全通り試せる。



*COUNTはRRの数に依存するので、気にしなければならないのは先頭32ビットだけ、しかもうち16ビットはID

やったこと

質問部の数については0個、1個、2個で試す(2個以上の質問部をサポートしていないことを想定)。



要求の場合、QTYPE、QCLASSは32ビットなので全通り試せる。

応答の場合、QNAME、QTYPE、QCLASSは要求との一致不一致の2通りに畳み込む。

やったこと

資源レコード(回答部、権威部、追加部)はパターン分けする。

ポジティブ応答、ネガティブ応答、委任、etc…

これらから回答部、権威部、追加部のパターンを作成して組み合わせる。

→ざっくり10億通りくらい

→現実的！！

やりたいこと

DNSパケットのビットレベルでの比較

→思い付き+提供いただけたものについては実施

様々なゾーンファイルの読み込み

→思い付き+提供いただけたものについては実施

資源レコード

→個々のリソースレコードタイプ毎に深堀したい

おわりに

今回の経験を得てXACK DNSもよりやさしく、懐深くなりました。

今後も安心・安定してご使用いただけるソフトウェア、システムを目指して参ります。

(新設、更改のお声掛けお待ちしております！)