90分で学び直す
DNSとDNSSECの基本

2025年11月18日

Internet Week 2025 オンラインWeek

株式会社日本レジストリサービス（JPRS）

鍛冶 典彦・倉持 玲介・森下 泰宏

講師自己紹介

- **鍛冶 典彦（かじ ふみひこ）**
  - 所属：JPRS システム部
  - 主な業務内容：権威DNSサーバー・ネットワーク機器の更新・メンテナンス、ビジネスパートナー向けセミナー、外部イベントにおける講演など

- **倉持 玲介（くらもち りょうすけ）**
  - 所属：JPRS システム部
  - 主な業務内容：権威DNSサーバー・ネットワーク機器の更新・メンテナンス、DNSに関する社内研修の講師、外部イベントにおける講演など

- **森下 泰宏（もりした やすひろ）**
  - 所属：JPRS 技術広報担当・技術研修センター
  - 主な業務内容：技術広報活動全般・社内外の人材育成など

勤務先紹介

株式会社 日本レジストリサービス

JaPan Registry Services

■ 主な業務内容：**ドメイン名サービス**

- **JPドメイン名（.jp）**の登録管理
- **gTLD（.com、.netなど）**の登録取り次ぎ
- **JP DNSサーバー**の運用・**MルートDNSサーバー**の共同運用
- **サーバー証明書**の発行（**認証局**）
- **ICANN・IETF**などの**国際活動・研究開発活動**への貢献

想定する対象

- DNSは知っているけど、**仕組みを理解しているか自信がない**方
- できるだけコンパクトに、**DNSとDNSSECの基本を学び直したい**方
- 担当者になったので、**DNSの仕組みを改めて勉強したい**方
- 新入社員や担当者に、**DNSとDNSSECの基本を教えたい**方

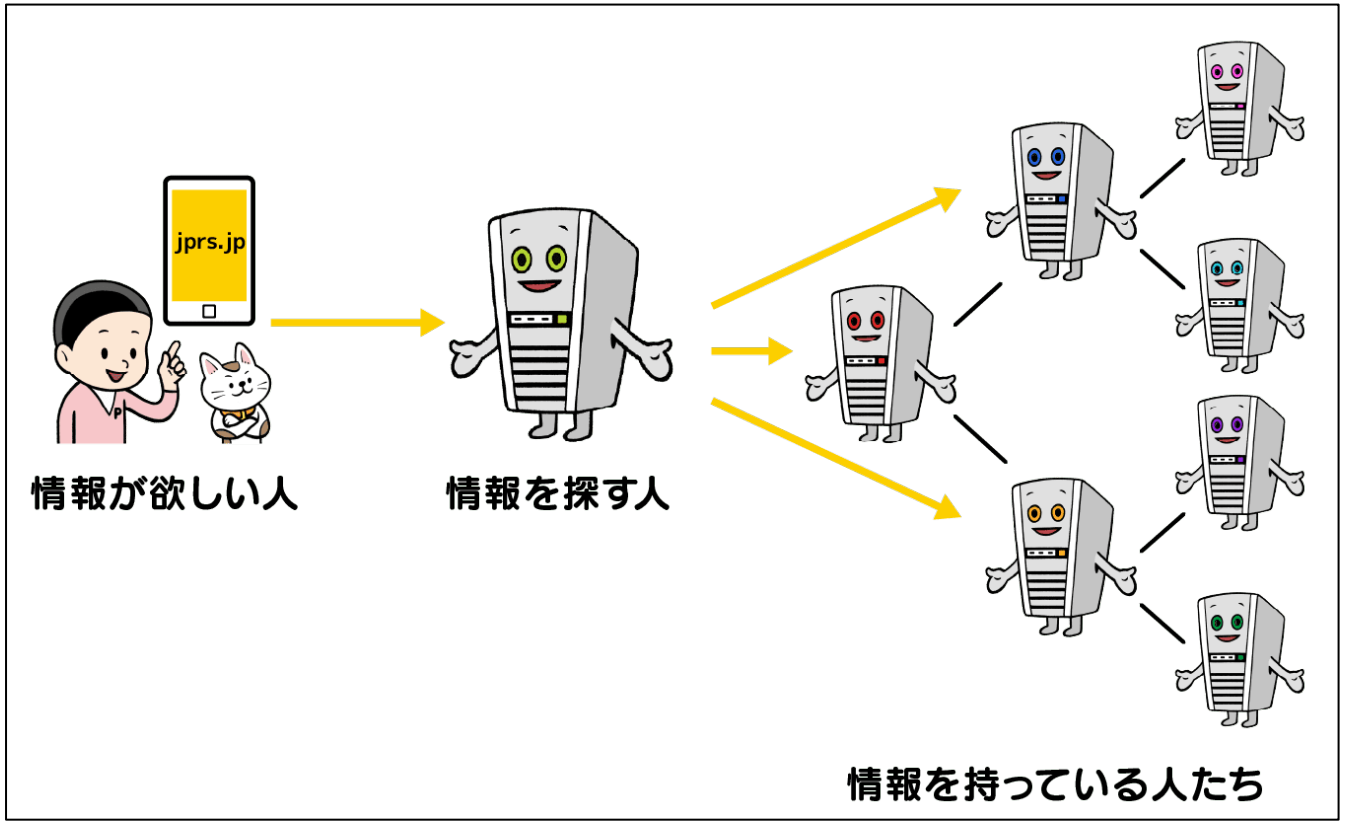
このチュートリアルの構成

- 以下の4パートで構成
  1. **DNSの構成要素と分散管理の仕組み（担当：鍛冶）**
  2. **名前解決の概要と具体的な動作（担当：倉持･鍛冶）**
  3. **プライバシー上の懸念点と名前解決の動作の変化（担当：倉持）**
  4. **DNSSECの概要と仕組み（担当：森下）**

1. DNSの構成要素と分散管理の仕組み

3種類の構成要素

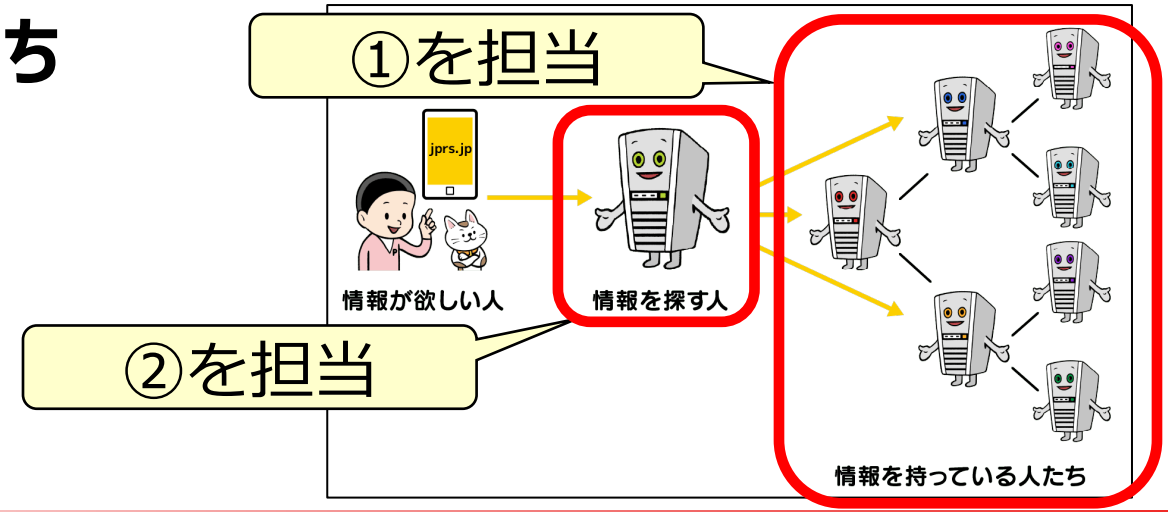
- DNSには、**3種類の登場人物（構成要素）**が存在する
  - **① 情報が欲しい人**
  - **② 情報を探す人**
  - **③ 情報を持っている人たち**
- DNSは、これらの構成要素が互いに連携し合って動作している



画像引用元：<https://withponta.jp/>

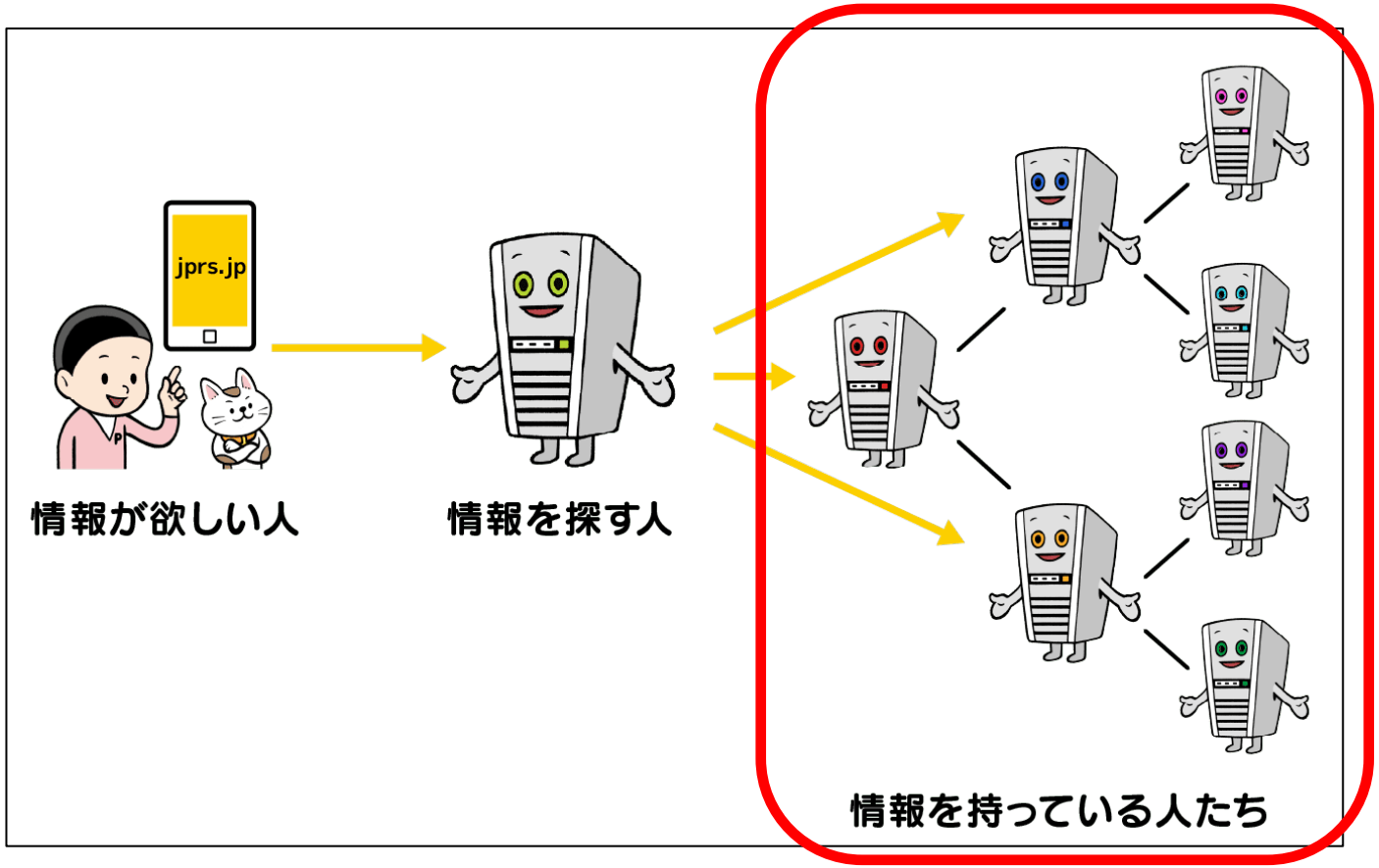
DNSの役割と構成要素の関係

- DNSには、**2種類の役割**が存在する
  - **① 名前と対象（IPアドレスなど）を、あらかじめ対応付けておく**
  - **② 名前と対象の対応付けを調べて、問い合わせ先に返す**
- ①と②の役割は、**別の構成要素が担当**している
  - ①を担当：**情報を持っている人たち**
  - ②を担当：**情報を探す人**



情報を持っている「人たち」な理由は？

- なぜ、情報を持っている「**人たち**」なのか？



DNS以前は「情報を持っている人」だった

- DNSが導入されるまで、対応付けは**集中管理**されていた
  - 米国のSRI-NICという組織が「**持っている人**」を担当していた
  - SRI-NICは**すべての対応付けが書かれたファイル（HOSTS.TXT）**を、**オンラインで公開**していた

参考：1983年11月4日のHOSTS.TXT（抜粋）

名前とIPアドレス以外に、
ホストの機種・OS・サポートする
プロトコルなども記述されていた

引用元：<https://emaillab.jp/dns/hosts/>

HOSTS.TXTによる集中管理の限界

- HOSTS.TXTは、以下の手順で更新されていた
  ① 各組織がホストを接続する際に、**SRI-NICに申請**する
  ② SRI-NICが各組織からの申請内容を、**HOSTS.TXTに反映・公開**する
  ③ 各組織がHOSTS.TXTを、**オンラインで入手・利用**する
- インターネット自身の成長により、この管理方法ではHOSTS.TXTの**サイズと更新頻度**が、**管理の限界を迎えてしまう**と考えられた

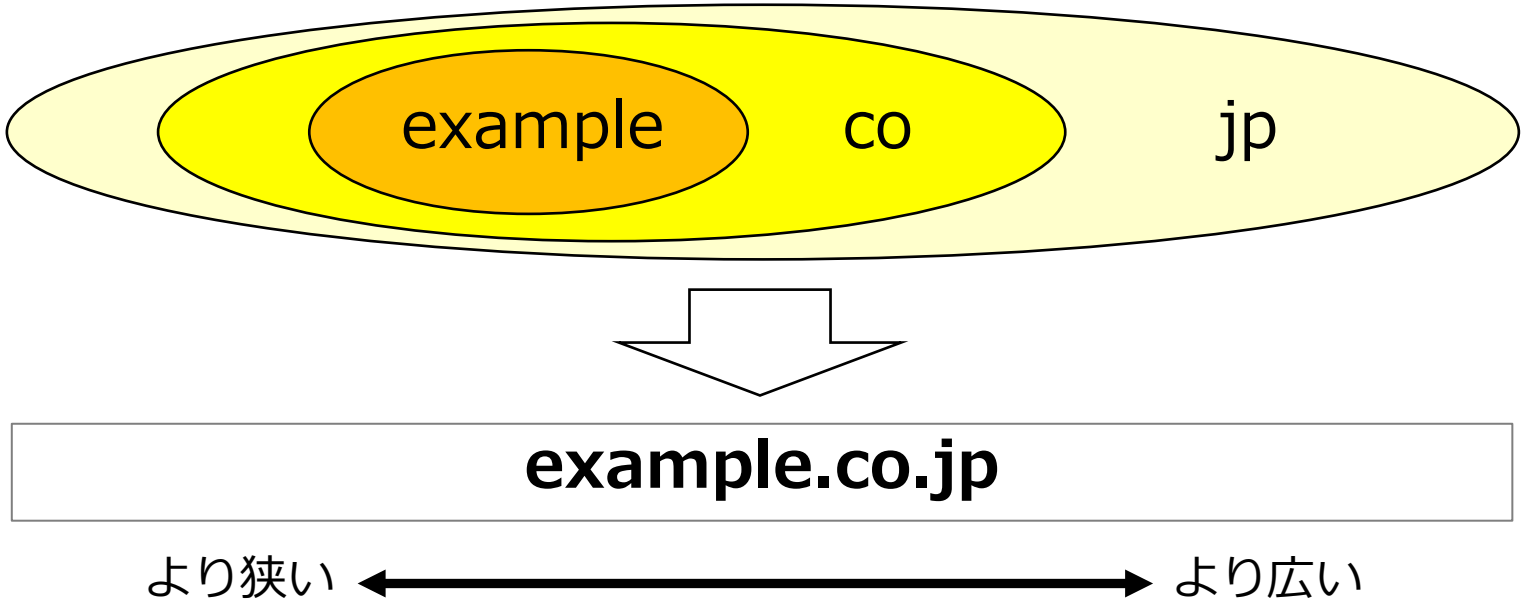
限界を迎えないようにするための仕組み

- 管理の限界を迎えないようにするため、**情報を手分けして持ち**、かつ**HOSTS.TXTと同様の一元管理を実現する**ための**仕組み**として、**Domain Name System（DNS）**が開発された
- 以降で、DNSに導入された**分散管理の仕組み**について解説する

キーワード：ドメイン名・階層化・委任・ゾーン

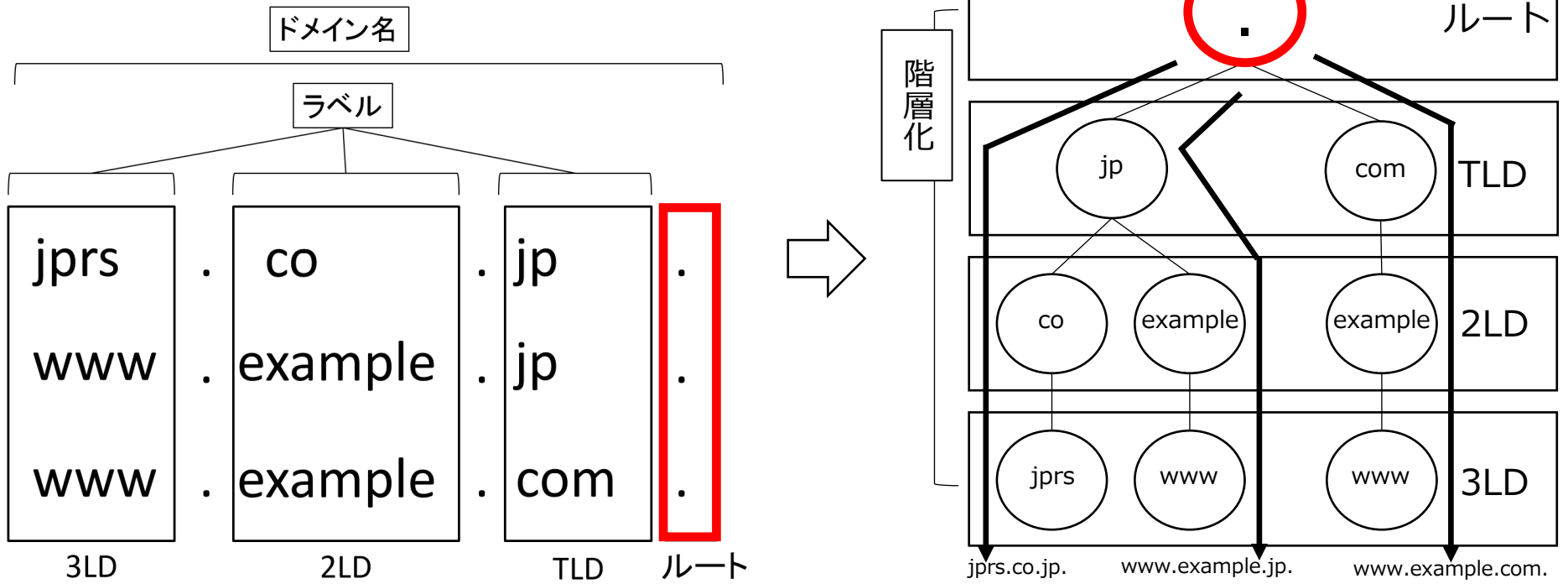
ドメイン名

- 管理する**範囲（ドメイン）**を、**名前で表せる**ようにした
  - 部分的な名前（**ラベル**）を、**ドット（.）でつないだ形**で表す
  - **左側のラベルほど、範囲が狭くなる**ように設計された



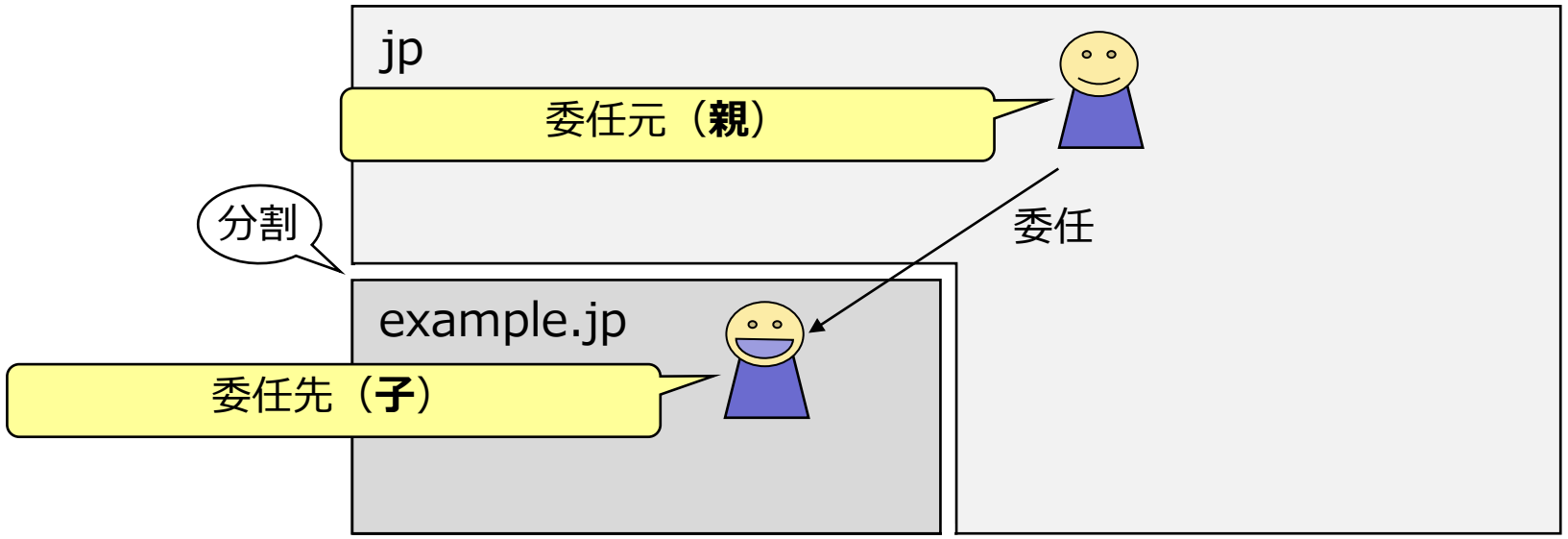
階層化

- **一つのルートを起点とする形**で、ドメイン名を**階層化**した
  - HOSTS.TXTと同様の、一つの**名前空間**を使う**一元管理**が可能になった



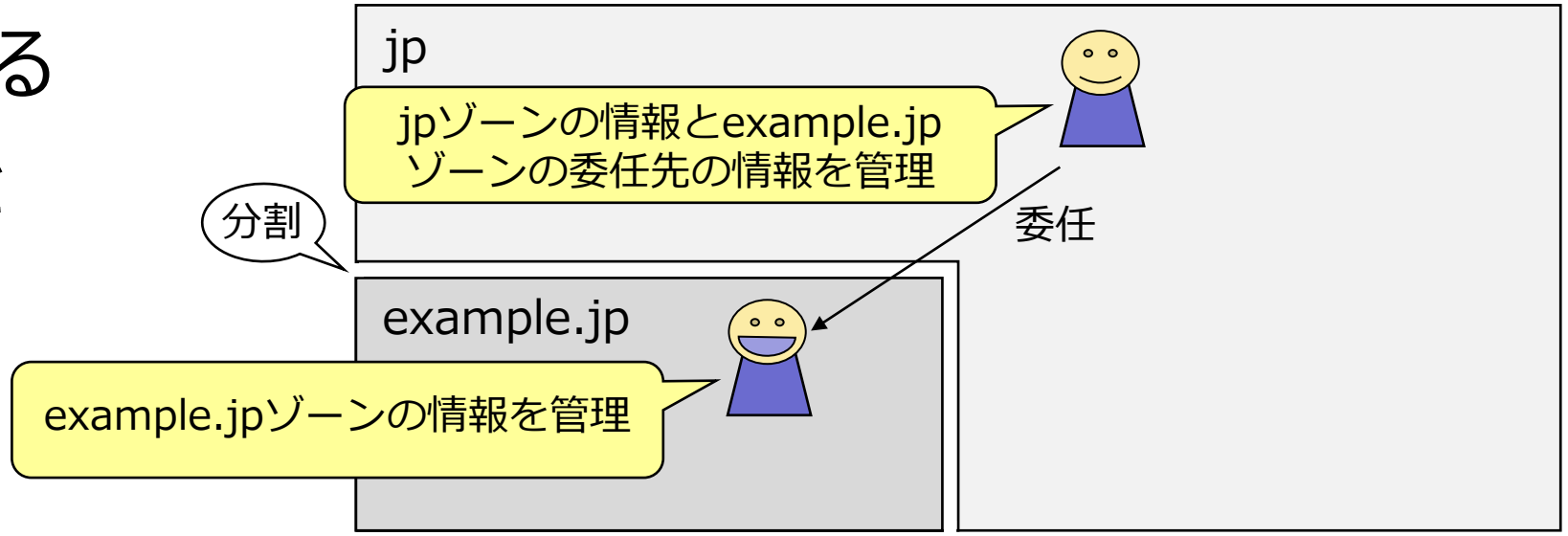
委任

- **階層を分割**して、**管理を任せられる**ようにした
  - 下記の例では、委任元（jp）は委任先（example.jp）の情報のみを管理し、example.jpの管理はexample.jpの管理者が担当する
- 委任元を**親**、
  委任先を**子**と呼ぶ



ゾーン

- 「情報を持っている人たち」は、以下の情報**のみ**を管理する
  - **自分が管理するドメイン名**の情報
  - **自分が委任したドメイン名の委任先**の情報
- 委任によって作られる管理の単位のことを「**ゾーン**」と呼ぶ



DNSの分散管理と一元管理

- **ドメイン名**を導入して名前を**階層化**し、**委任**によって管理の単位を**ゾーン**に**分割**することで、**分散管理**を実現している
- **一つのルート**を**階層構造の起点**とすることで、**HOSTS.TXTと同様の一元管理**が可能になる
  - すべての名前がインターネット全体で**同じ意味**を持つ（**名前空間の一意性**）

分散管理と**一元管理**の**双方を実現**している

まとめ：DNSの構成要素と分散管理の仕組み

- DNSには**3種類の構成要素**があり、連携し合って動いている

情報が欲しい人・情報を探す人・情報を持っている人たち

- 情報を手分けして持つ**分散管理**と、すべての名前がインターネット上で同じ意味を持つ**一元管理**の**双方を実現**している

ドメイン名・階層化・委任・ゾーン

- 次のパートでは構成要素が連携する形で実行される、**DNSの名前解決の概要と具体的な動作**について解説する

2. 名前解決の概要と具体的な動作

2.1 名前解決の概要

2.2 名前解決の具体的な動作

2.1 名前解決の概要

名前解決とは？

- 利用者の要求に応じ、**名前に対応する情報を取り出す**こと
  - 情報の例：名前に対応するIPアドレス、メールサーバーホスト名など

example.jpのIPv4アドレス → 192.0.2.1

example.jpのメールサーバー → mail.example.jp

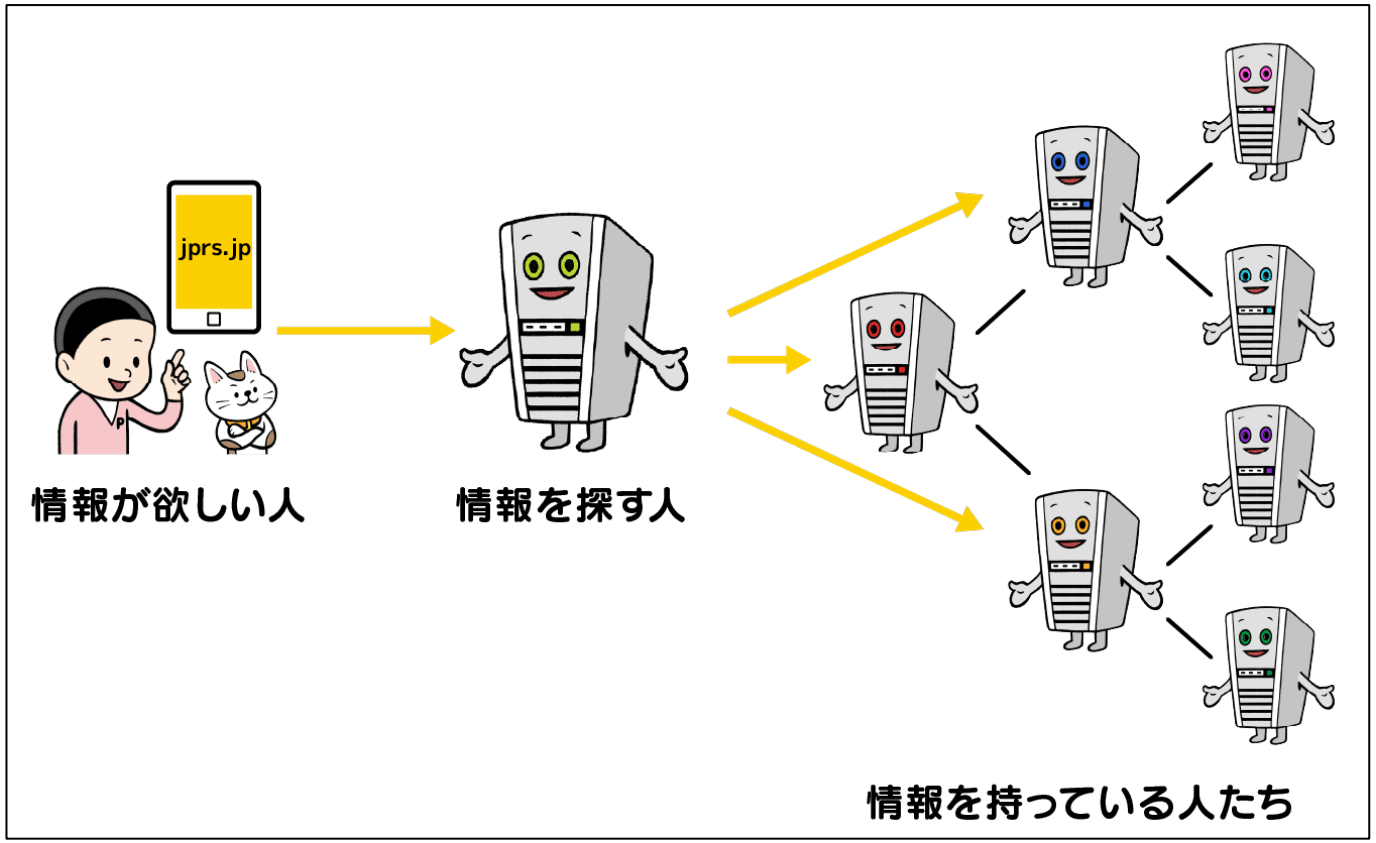
- DNSの名前解決では、**ドメイン名と情報の種類（タイプ）の双方を指定**して問い合わせる

example.jpのIPv4アドレスは？

example.jpのメールサーバーホスト名は？

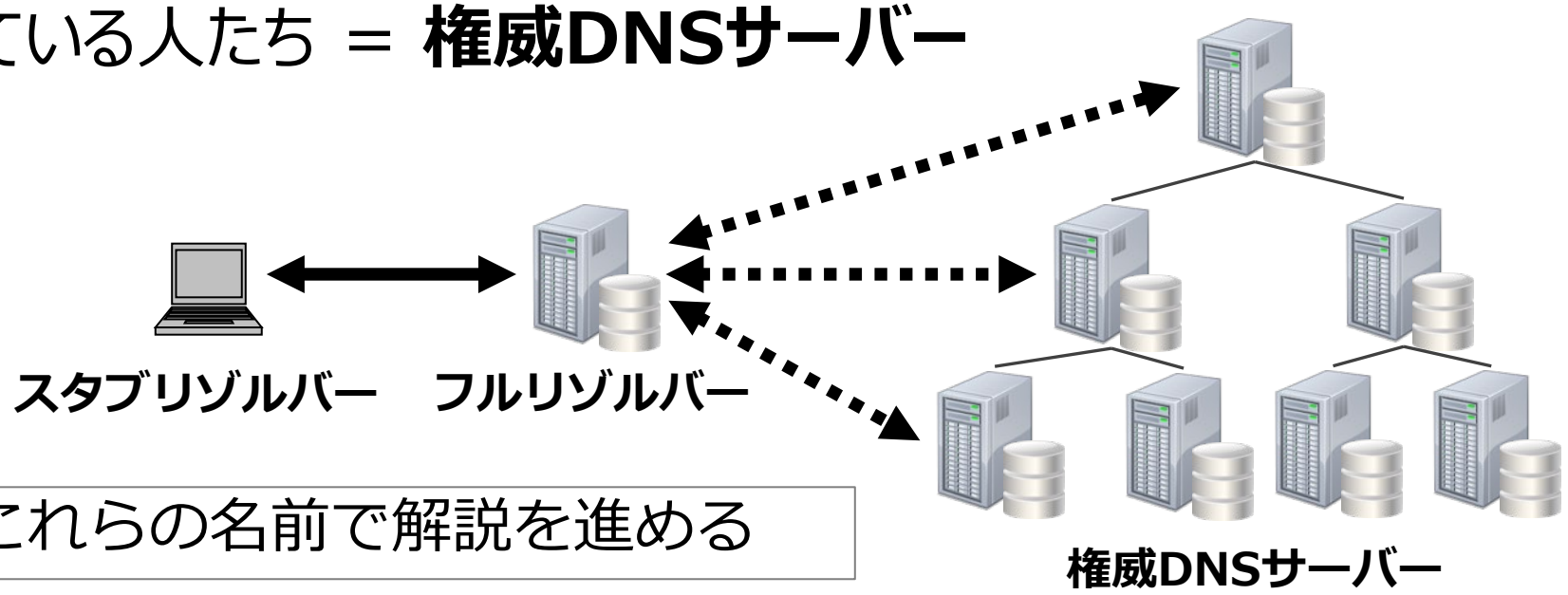
おさらい：DNSの構成要素

- 情報が欲しい人・情報を探す人・情報を持っている人たち



構成要素の名前

- これらの構成要素には、名前が付けられている
  - 情報が欲しい人 ＝ **スタブリゾルバー**
  - 情報を探す人 ＝ **フルサービスリゾルバー（フルリゾルバー）**
  - 情報を持っている人たち ＝ **権威DNSサーバー**



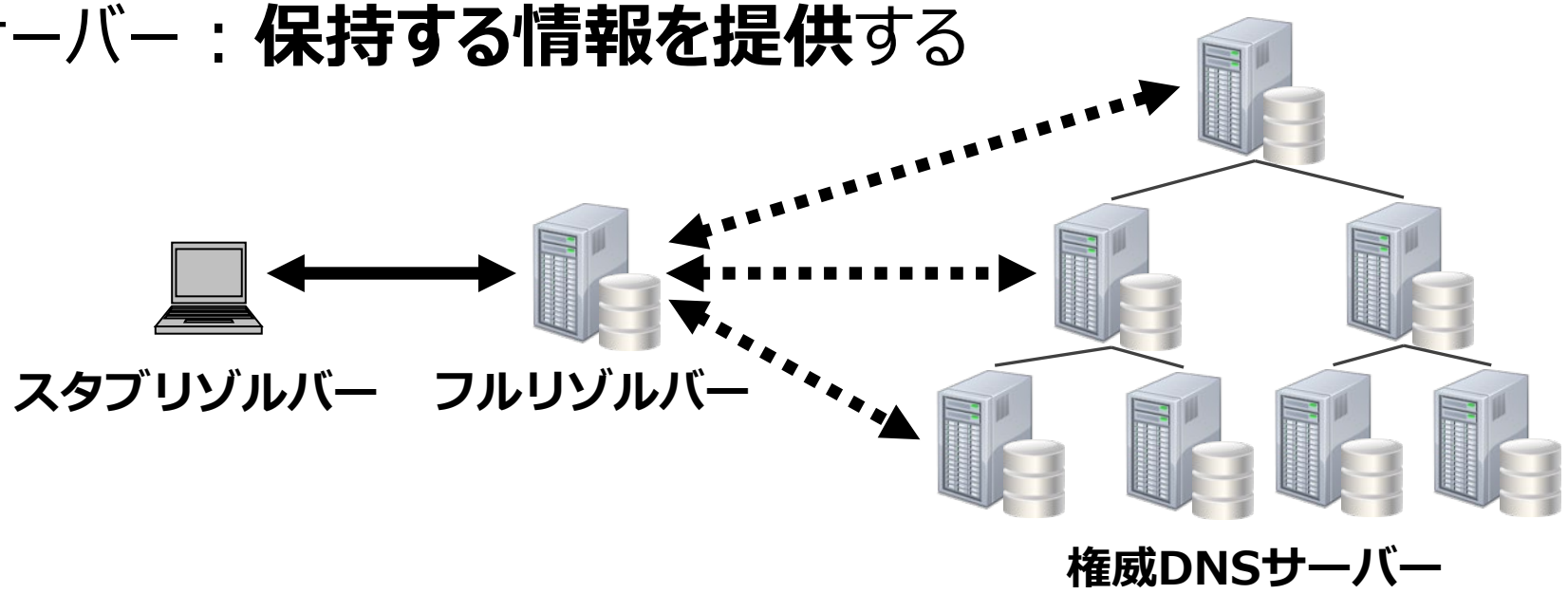
以降では、これらの名前で解説を進める

「DNSサーバー」には要注意

- 構成要素の「フルリゾルバー」と「権威DNSサーバー」はいずれも、「**DNSサーバー**」と呼ばれている
  - かつ、「フルリゾルバー」「権威DNSサーバー」「フルリゾルバーと権威DNSサーバーの両方」の、**いずれの意味でも使われる**
- 書籍や資料、会話などでDNSサーバーという用語が出て来た場合、**どの構成要素を指しているかを把握する**必要がある
  - 誤解を避けるため、**自分からはできるだけ使わない**ことを推奨する

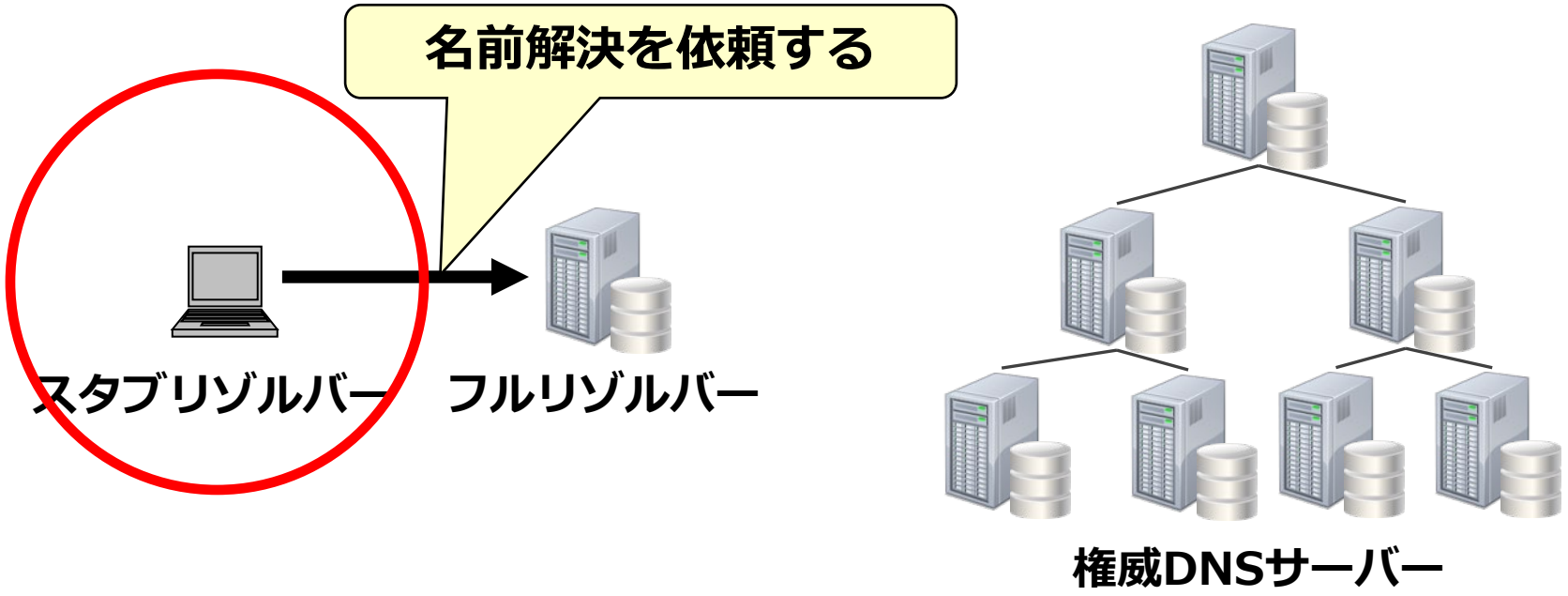
構成要素の役割

- 以降で、それぞれの構成要素の役割について解説する
  - スタブリゾルバー：**名前解決を依頼する**
  - フルリゾルバー：**名前解決を実行**し、得られた**応答を蓄える**
  - 権威DNSサーバー：**保持する情報を提供**する



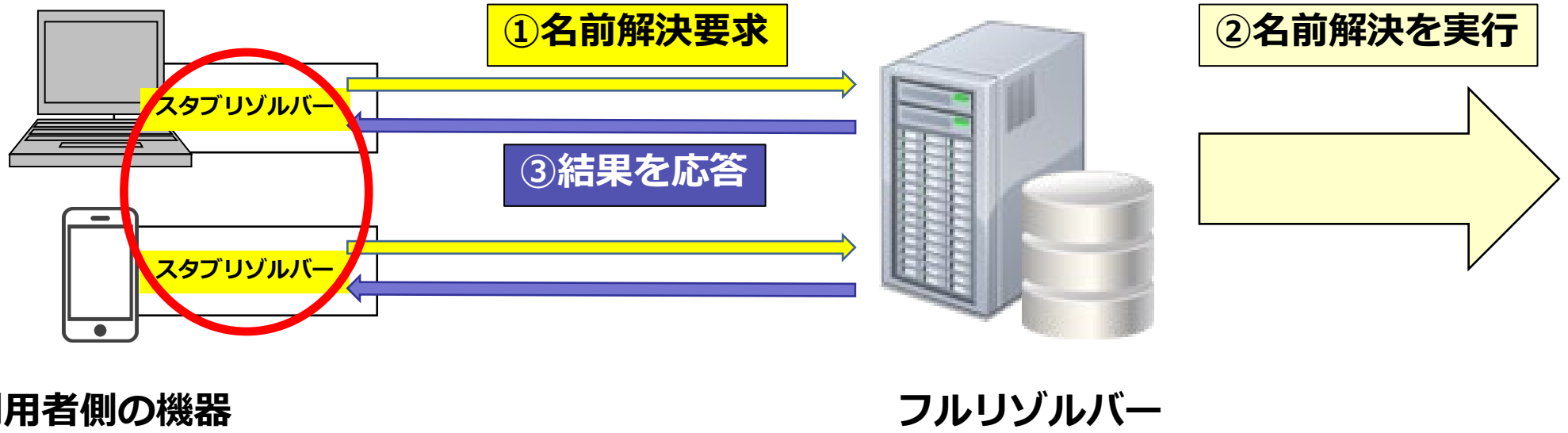
スタブリゾルバー

- Webブラウザーなどのアプリケーションから呼び出され、フルリゾルバーに**名前解決を依頼する**
  - この動作を「**名前解決要求**」と呼ぶ



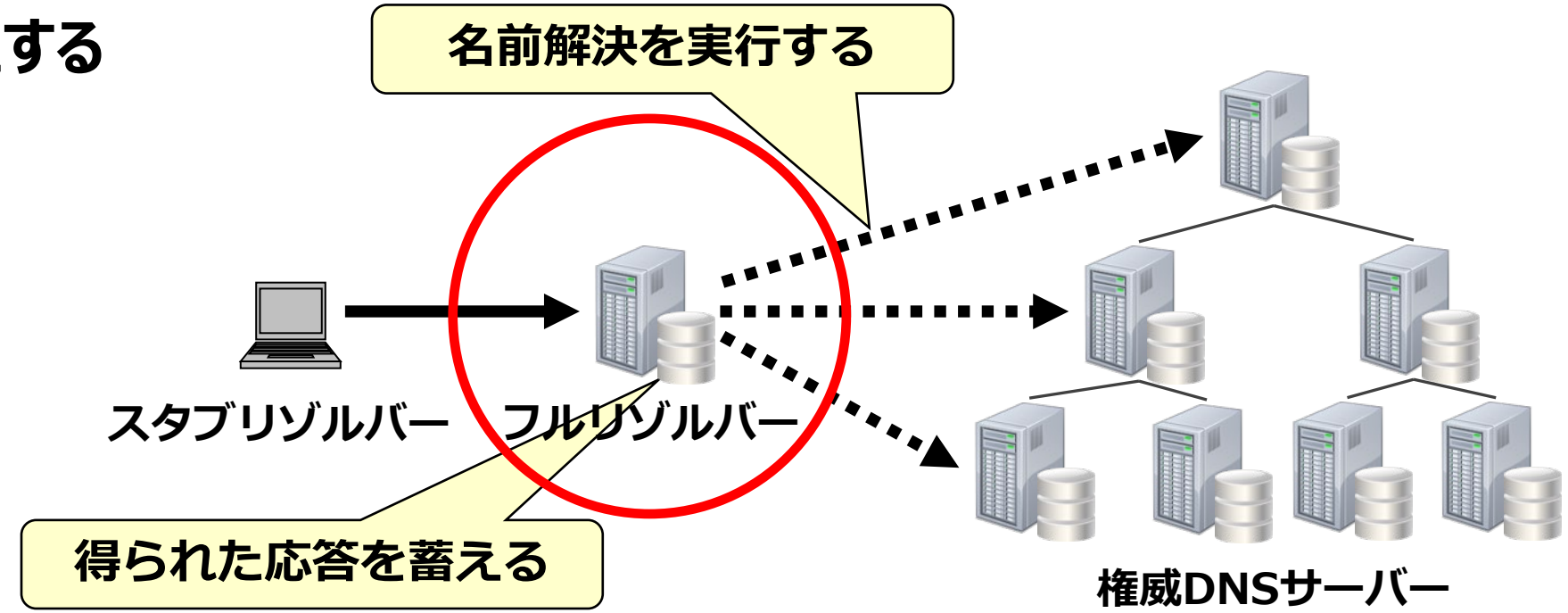
名前解決要求

- 「**私の代わりに名前解決を実行して、結果を教えてください**」という問い合わせを示す
  - 名前解決要求を受け取ったフルリゾルバーが名前解決を実行し、結果をスタブリゾルバーに応答する



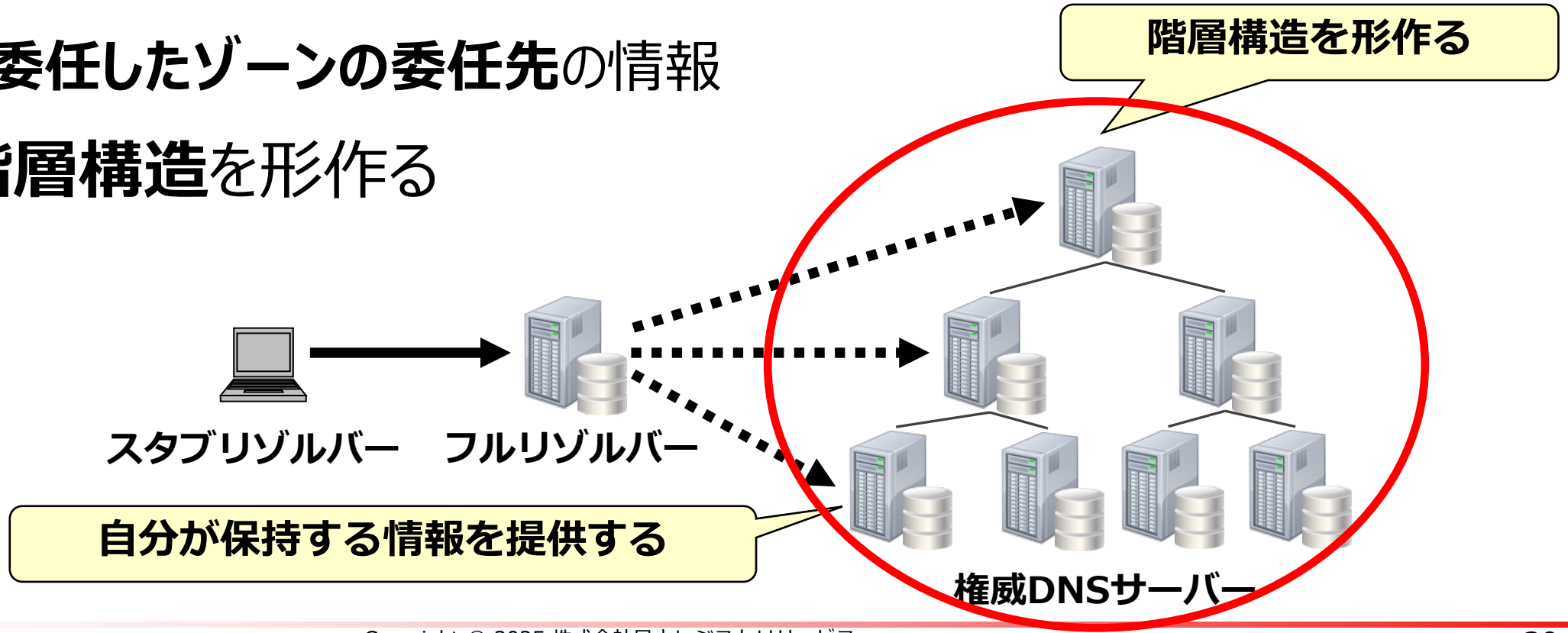
フルリゾルバー

- 名前解決要求を受け取って**名前解決を実行**し、結果を返す
  - ルートから始めて、権威DNSサーバーへの問い合わせを繰り返す
- 次回以降の名前解決に使うため、**得られた応答を蓄える**
  - **キャッシュする**



権威DNSサーバー

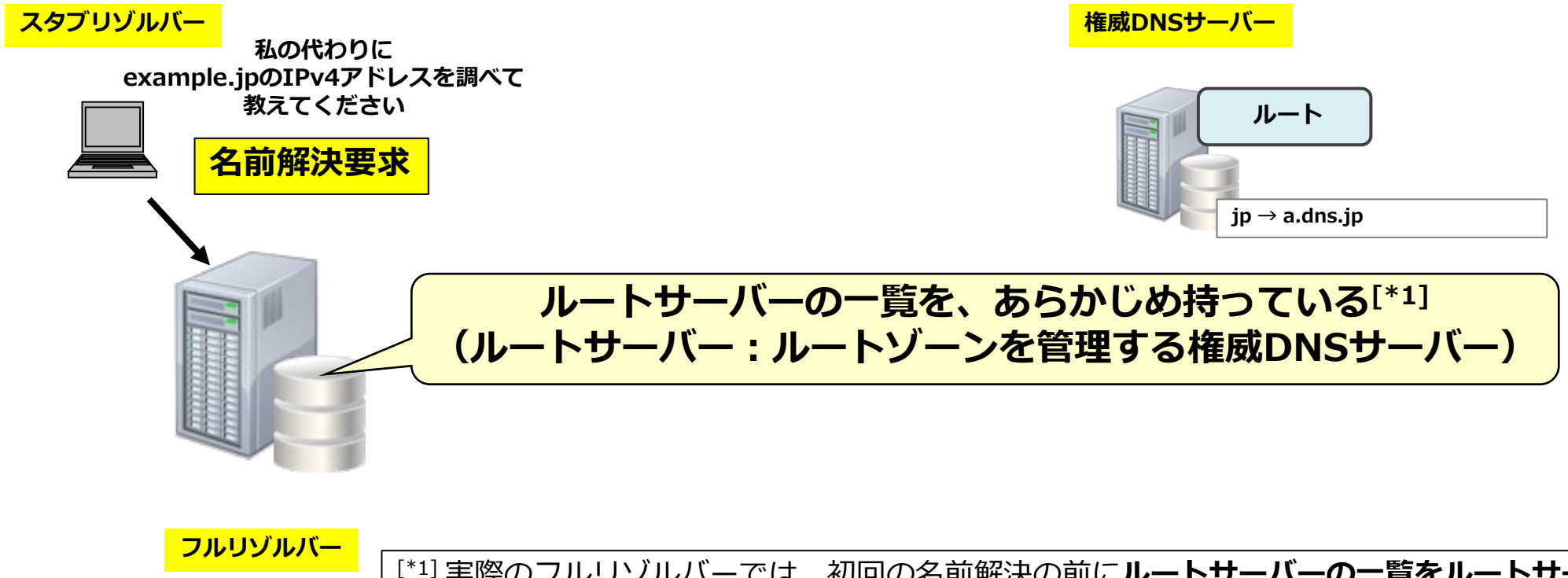
- 問い合わせに応じ、**自分が保持する情報を提供**する
  - **自分が管理するゾーン**の情報
  - **自分が委任したゾーンの委任先**の情報
- DNSの**階層構造**を形作る



2.2 名前解決の具体的な動作

名前解決の流れ

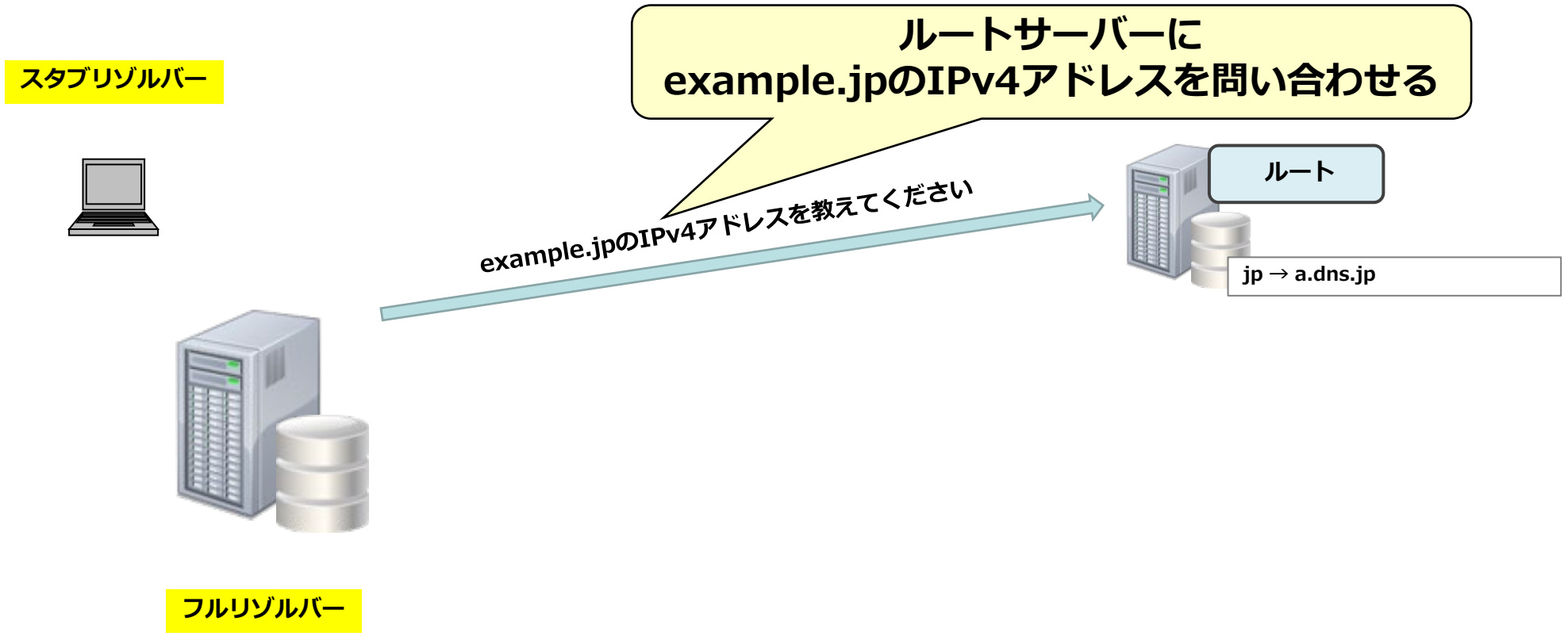
example.jpのIPv4アドレスを名前解決する例で説明する



[*1] 実際のフルリゾルバーでは、初回の名前解決の前に**ルートサーバーの一覧をルートサーバー自身に問い合わせて応答をキャッシュし、以降はそれを使う（プライミング**：このチュートリアルでは説明を省略）

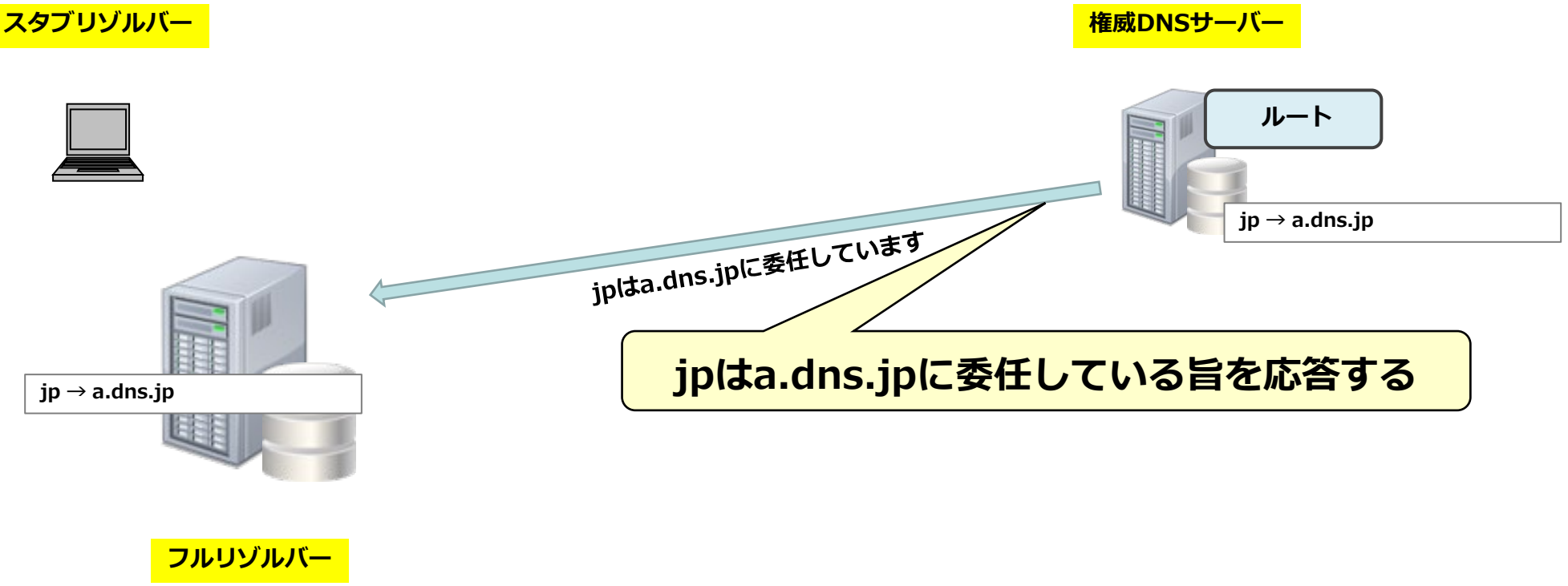
名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する



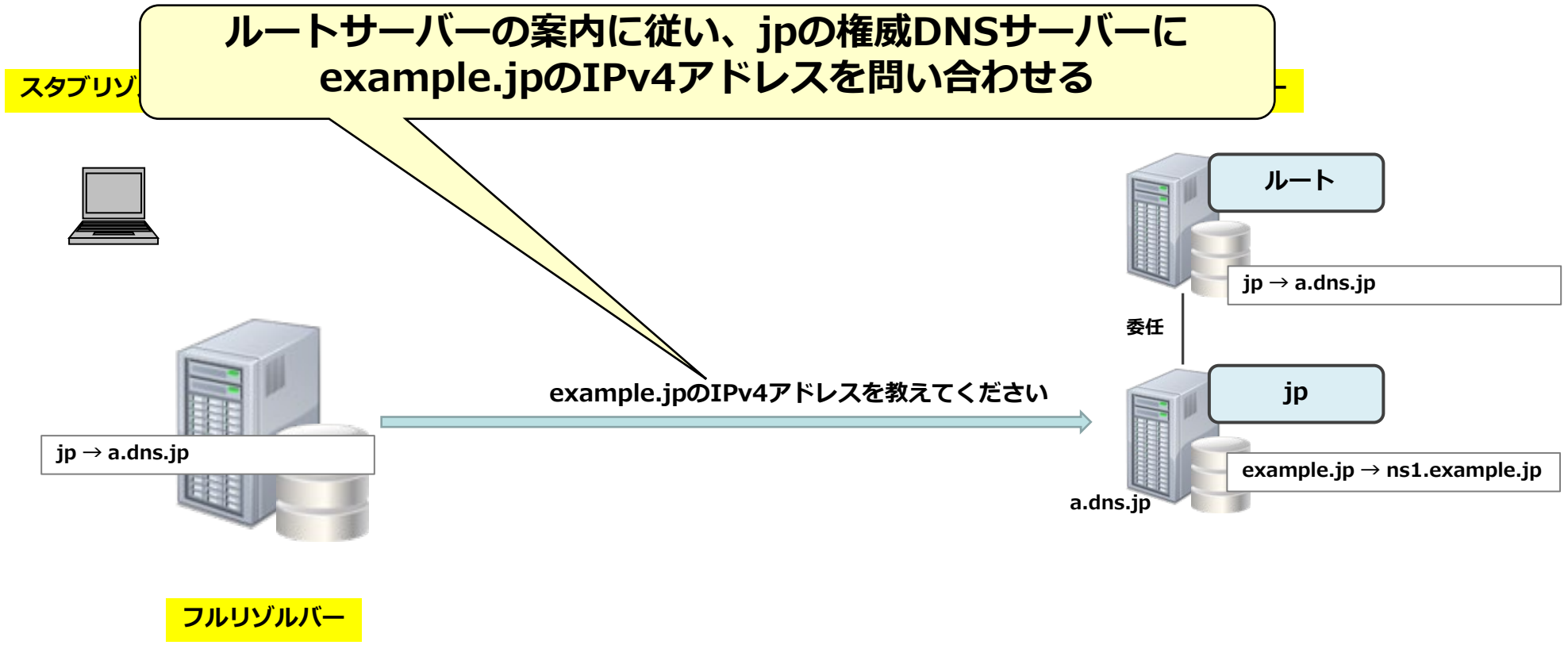
名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する



名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する

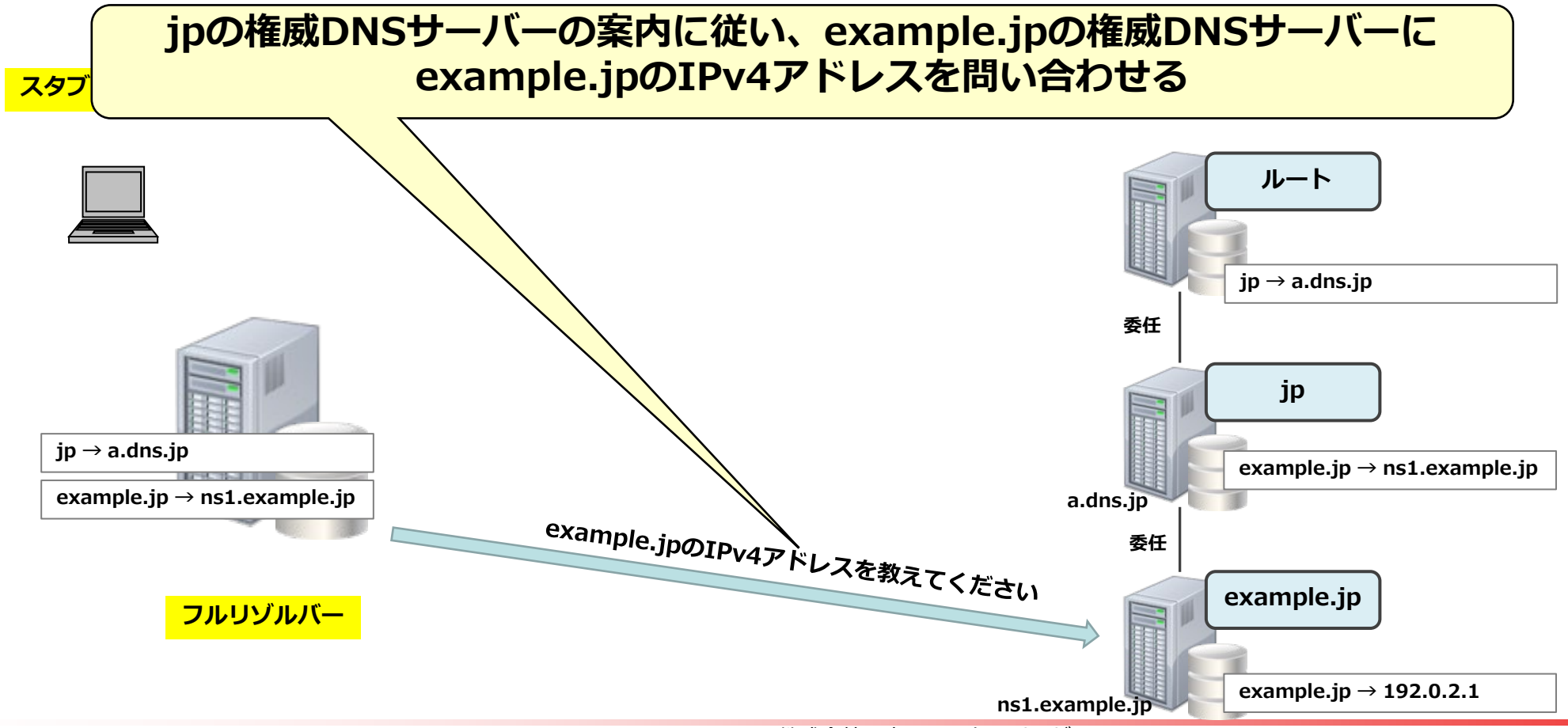


名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する

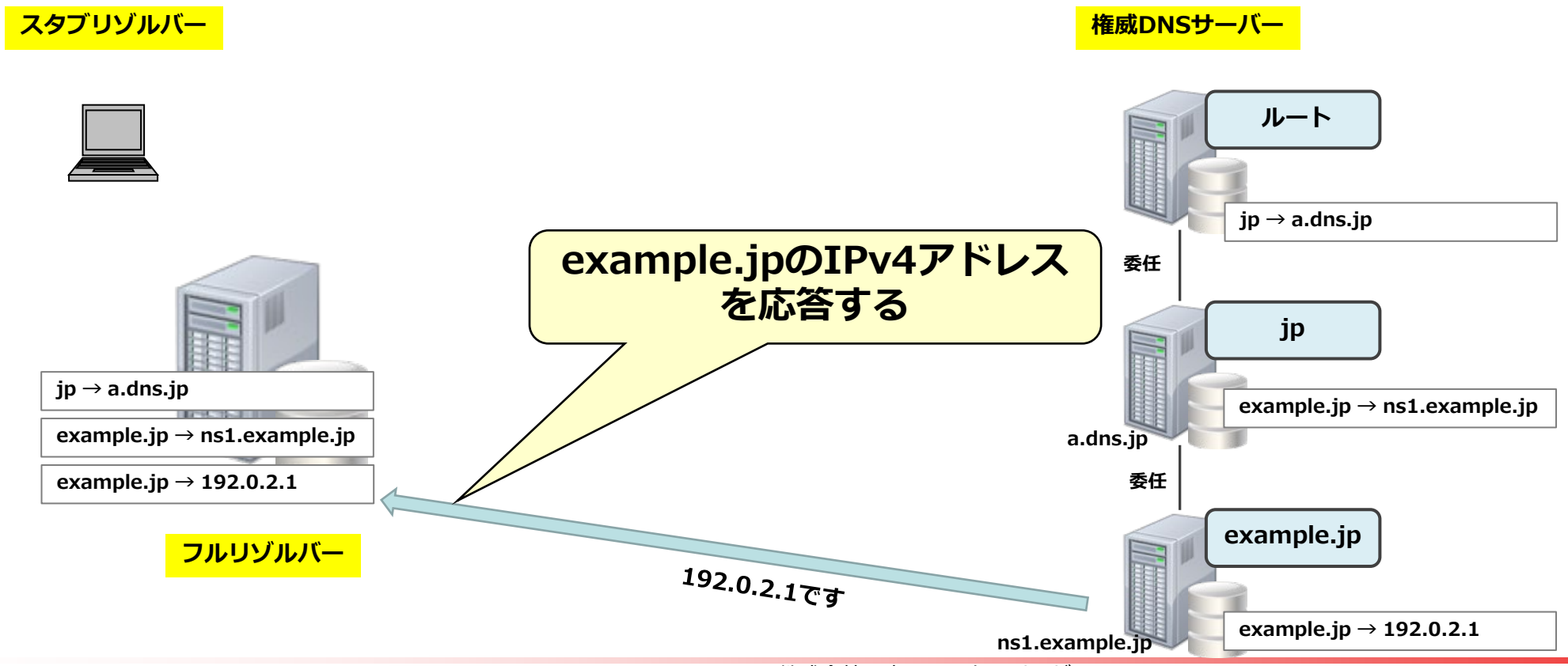
名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する



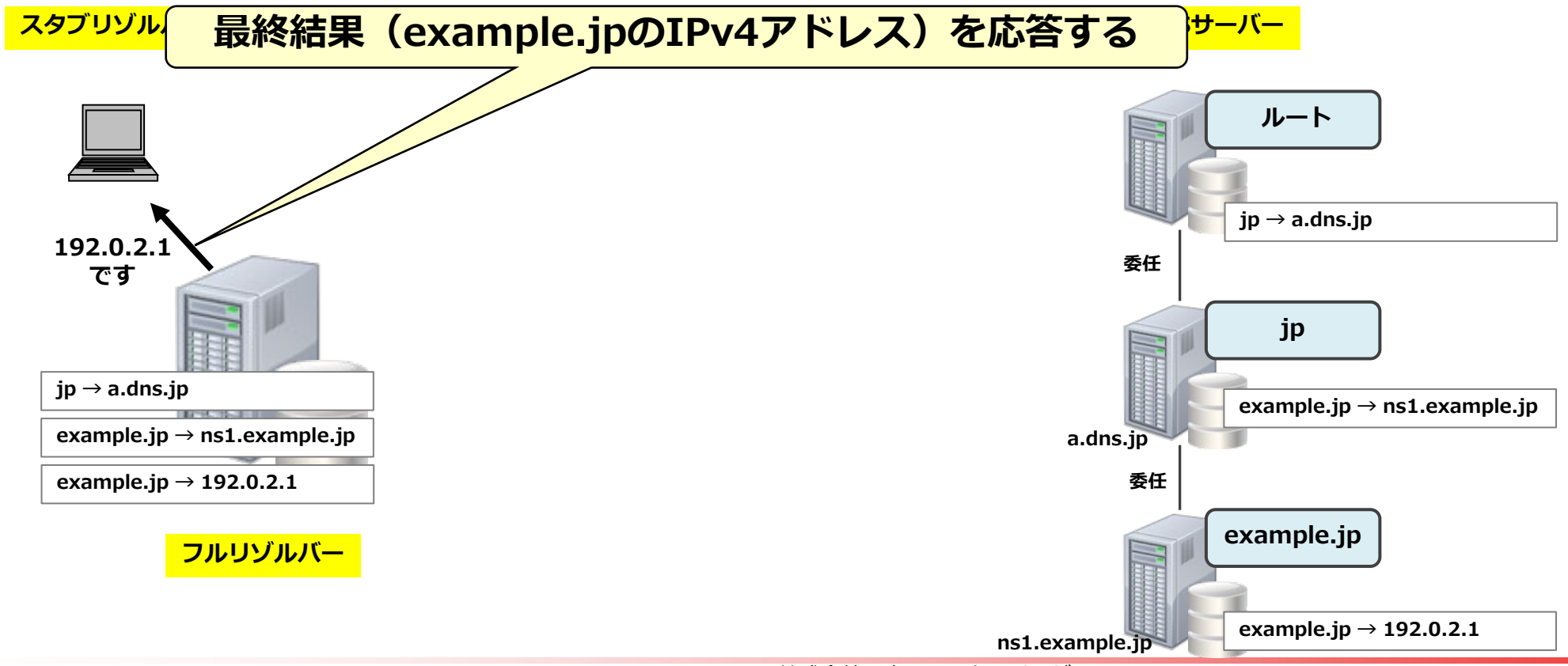
名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する



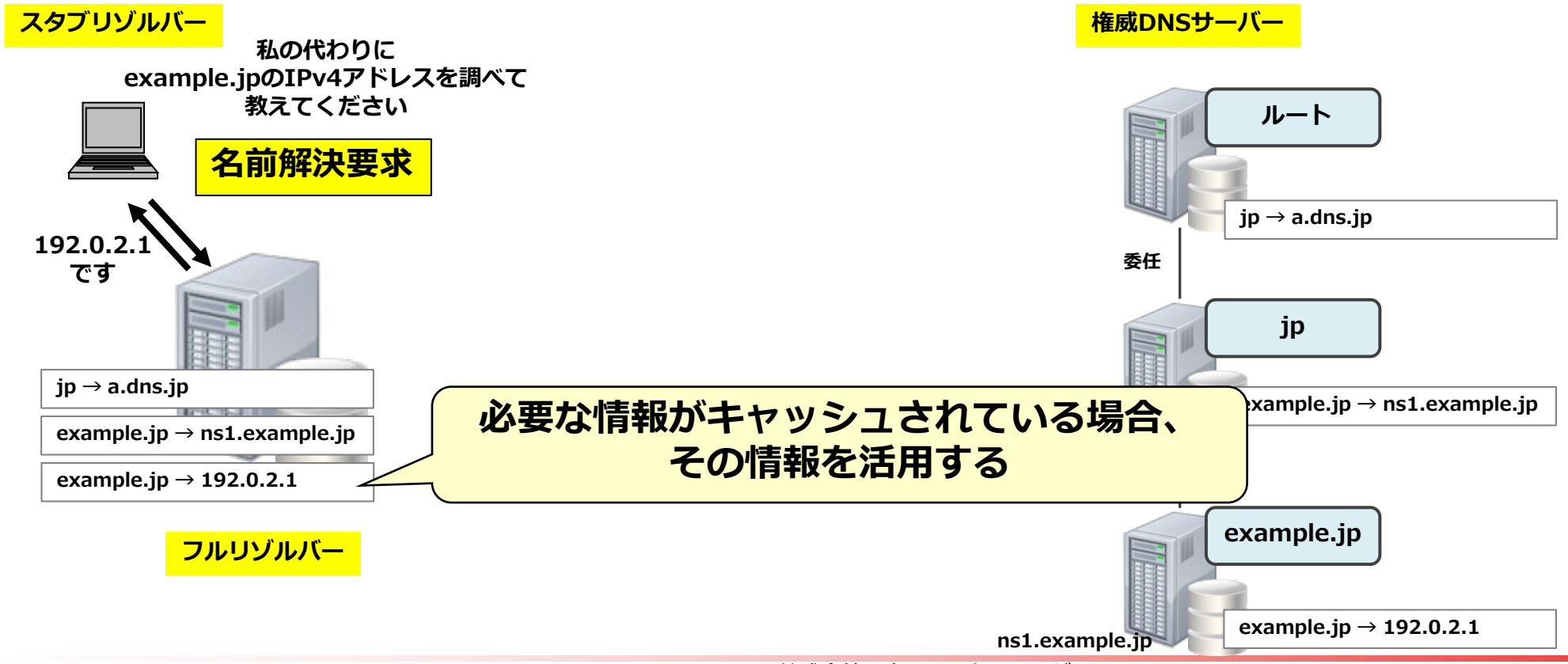
名前解決の流れ

example.jpのIPv4アドレスを名前解決する例で説明する



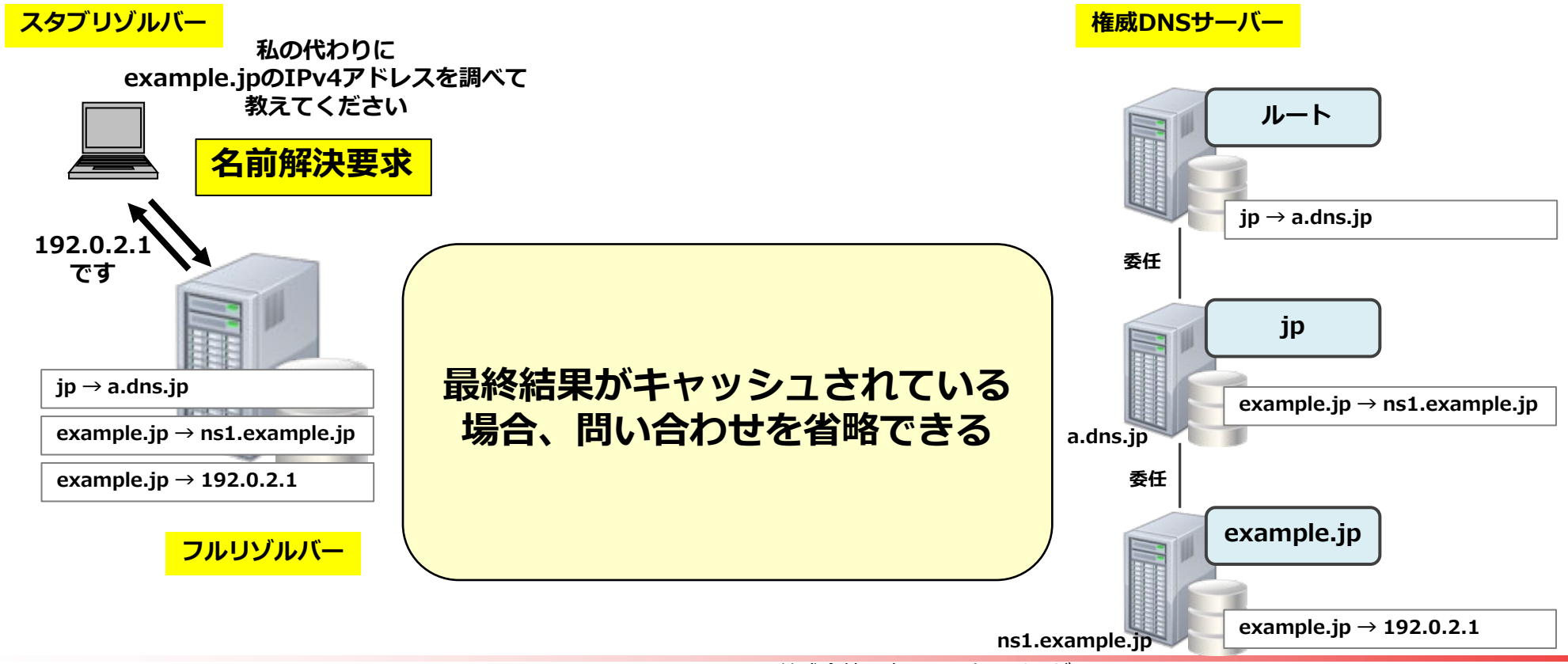
名前解決の流れ（2回目以降）

example.jpのIPv4アドレスを名前解決する例で説明する



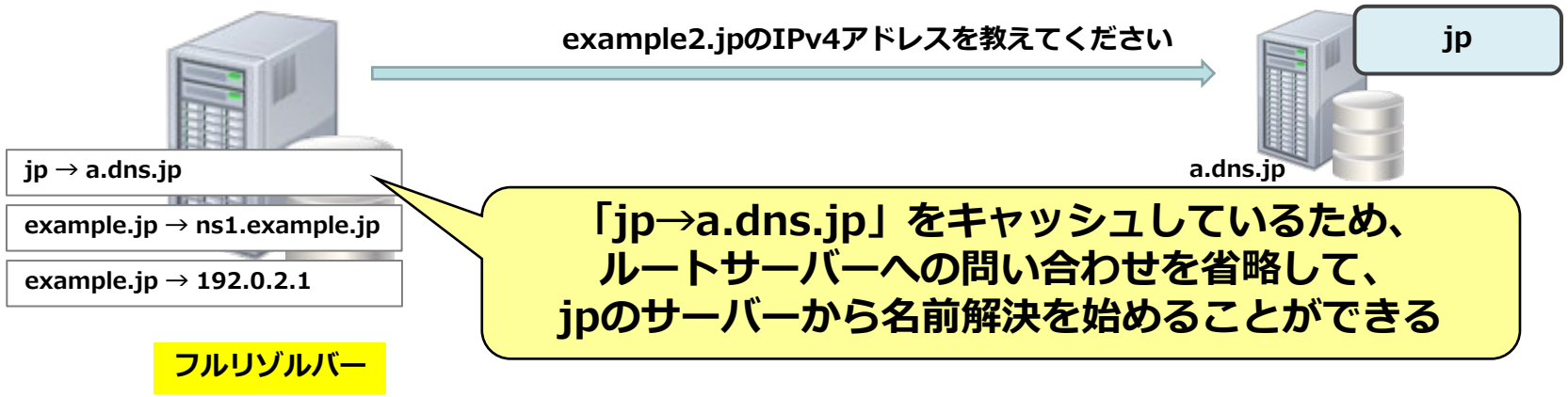
キャッシュの活用

- キャッシュを活用して、名前解決の**負荷と時間を軽減**している



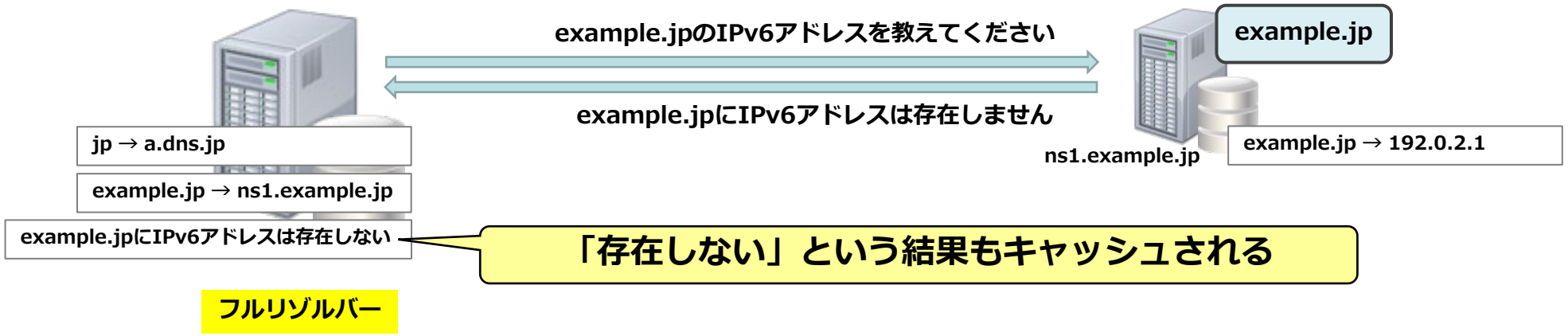
途中の情報もキャッシュ・活用

- 名前解決の**途中で得た情報もキャッシュ・活用**する
  - 例えば、**example.jp**のIPv4アドレスを名前解決した直後に**example2.jp**のIPv4アドレスを名前解決する場合、**ルートサーバーへの問い合わせを省略して、jpのサーバーから名前解決を始める**ことができる



「存在しない」こともキャッシュ・活用

- 名前解決では「**存在しない**」という結果もキャッシュされ、次回以降の名前解決で使われる
- **ネガティブキャッシュ**と呼ばれる



キャッシュの生存時間

- キャッシュは便利だが、情報が無期限にキャッシュされていると**都合が悪い**
  - IPアドレスや委任先などの**DNSデータ**は、**変更される**場合がある
- そのため、DNSでは**自分のゾーンの権威DNSサーバー**で、**そのデータをキャッシュしてよい時間**を設定するようになっている
- この時間を**TTL**（Time To Live）と呼ぶ
  - **キャッシュの生存時間**を**秒単位**で設定する
    - 時間の経過と共にキャッシュの内部で**TTLが減算**され、**0になるとキャッシュが満了**

example.jp → 192.0.2.1

このデータを最大3600秒（1時間）キャッシュさせたい

```
example.jp. 3600 IN A 192.0.2.1
```

ゾーンの管理者がTTLを3600に設定

まとめ：名前解決の概要と具体的な動作（1/2）

- DNSの名前解決では、**ドメイン名**と**情報の種類（タイプ）**を指定して問い合わせる
- DNSの名前解決は、**三つの構成要素**が連携して実行される
  - 情報が欲しい人・情報を探す人・情報を持っている人たち

スタブリゾルバー・フルリゾルバー・権威DNSサーバー

スタブリゾルバーが名前解決を要求し、**フルリゾルバーが権威DNSサーバーの階層構造をルートから順にたどる**ことで、**名前解決が実行**される

まとめ：名前解決の概要と具体的な動作（2/2）

- 名前解決の**負荷と時間を軽減**するために情報を蓄える、**キャッシュ**という仕組みが存在する
- **名前解決の途中で得られた情報もキャッシュ・活用**される
- 「**存在しない**」という結果も**キャッシュ・活用**される
  - **ネガティブキャッシュ**と呼ばれる
- キャッシュの生存時間（**TTL**）は、**各ゾーンの権威DNSサーバーで設定**する

次のパートの内容

- 次のパートではDNSにおける**プライバシー上の懸念点**と、懸念点に対応するために開発された**新しい名前解決方式**の導入に伴う、**名前解決の動作の変化**について解説する

3. プライバシー上の懸念点と名前解決の動作の変化

プライバシー保護の背景

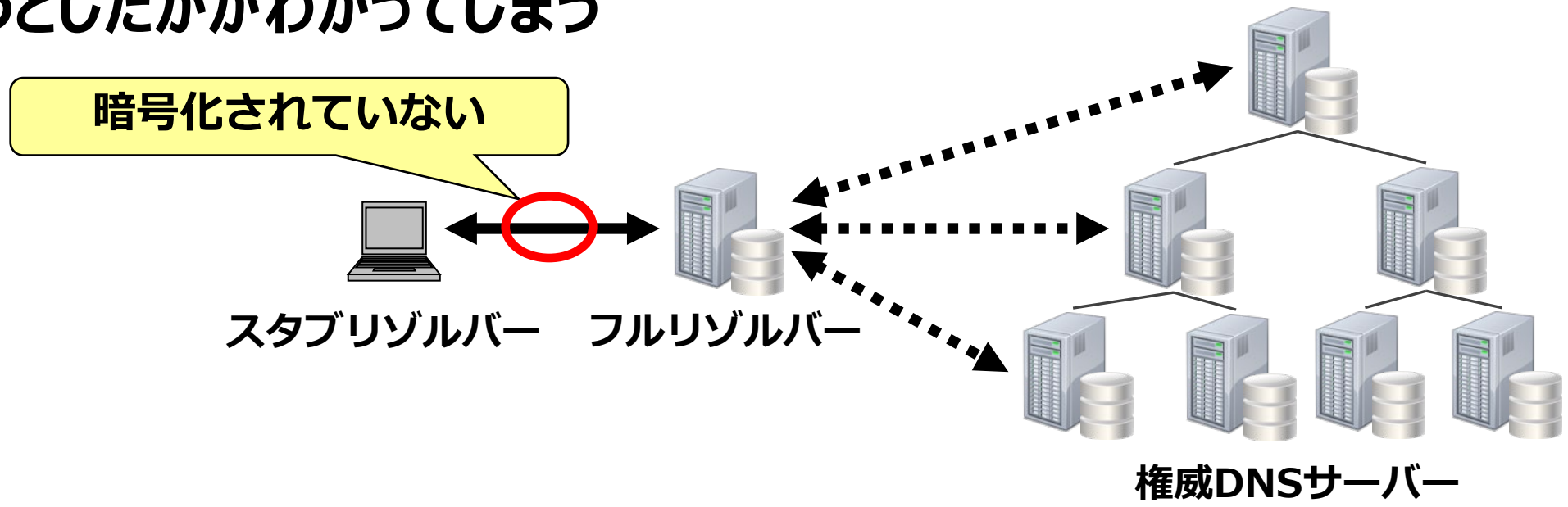
- 利用者の意識の高まりや2013年に発生したスノーデン事件[*1]をきっかけとして、**Pervasive Monitoring[*2]に対抗するための活動**が進められた
- DNSにおいても、**機密性を確保する**ための活動が進められた
  - 機密性：**正当な権限を持つ者のみが情報にアクセスできる**こと

[*1] 米国国家安全保障局（NSA）の業務を請け負っていたエドワード・スノーデン氏が、米国政府の極秘の監視計画「PRISM」を暴露した事件。

[*2] 広域かつ網羅的な通信の傍受による情報収集活動。

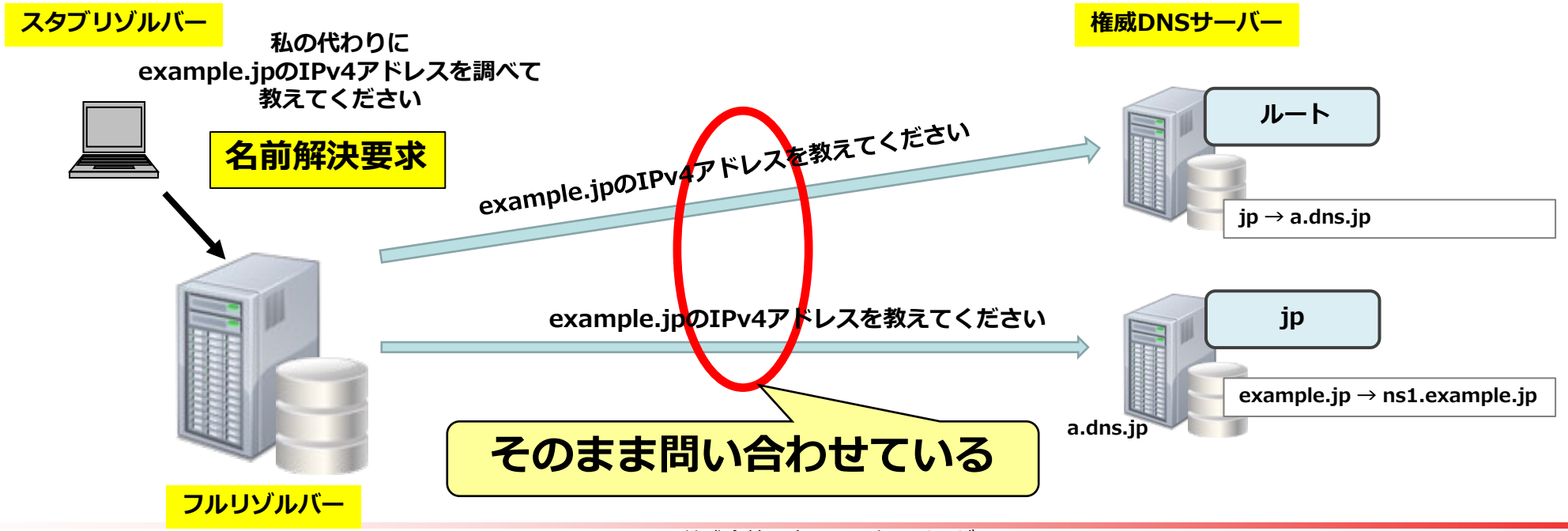
DNSにおけるプライバシー上の懸念点（1/2）

- 懸念点1：スタブリゾルバーとフルリゾルバーの間の通信が**暗号化されていない**
  - 通信を傍受されると、**いつ・どのIPアドレスから・どのドメイン名にアクセスしようとしたかがわかってしまう**



DNSにおけるプライバシー上の懸念点（2/2）

- 懸念点2：フルリゾルバーは名前解決要求の内容を、**ルートサーバーやTLDの権威DNSサーバーにそのまま問い合わせている**
  - ルートサーバーやTLDの権威DNSサーバーに、**問い合わせ内容が伝わる**



懸念点の解決策

- 懸念点1：**通信路を暗号化する**
  - **TLSにより通信路を暗号化**
    - **DNS over TLS･DNS over HTTPS･DNS over QUIC**
    - それぞれ、**DoT･DoH･DoQ**と省略される

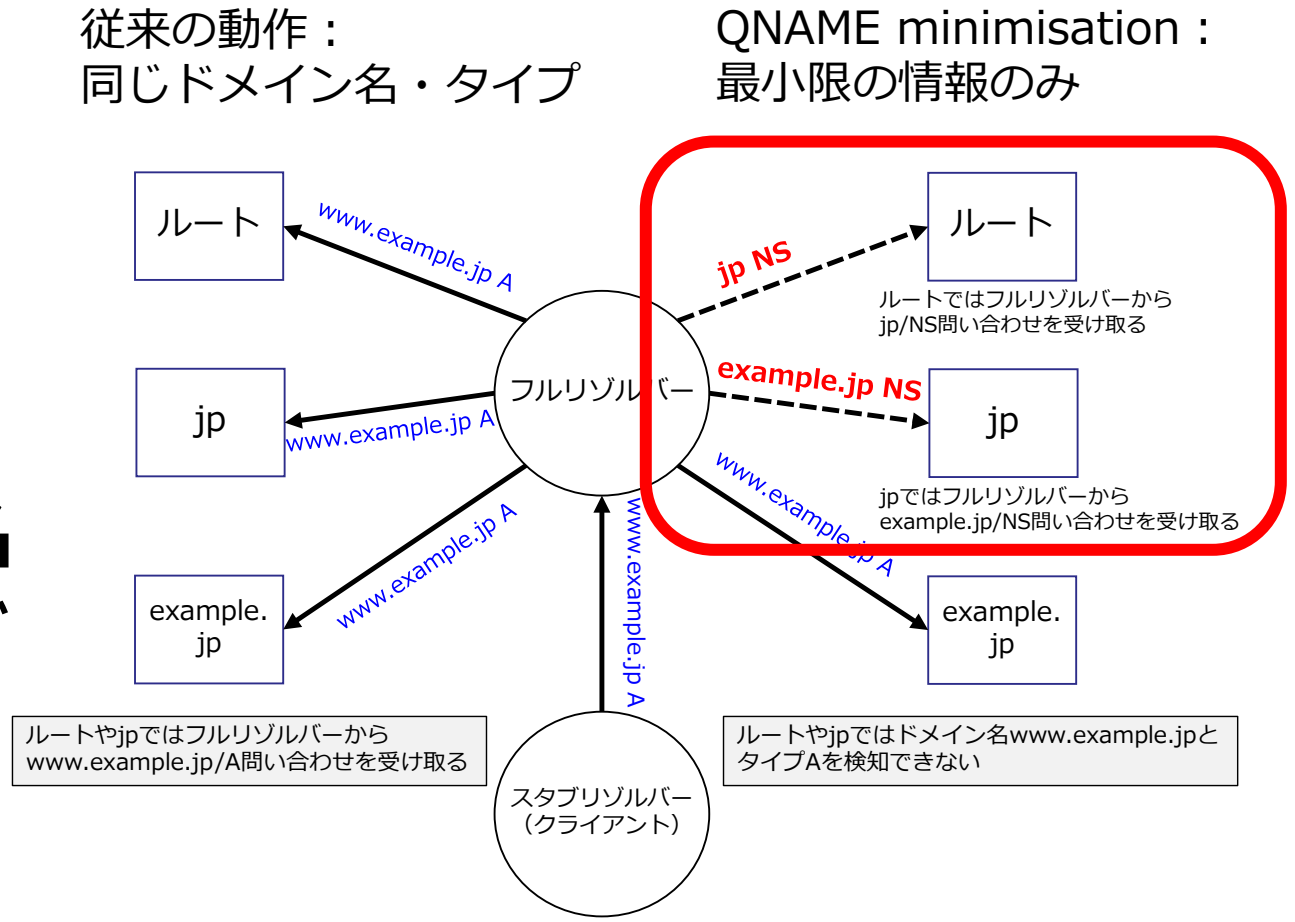
本チュートリアルでは詳細を省略

- 懸念点2：名前解決の**動作を変更する**
  - **QNAME minimisationの導入**

以降で、QNAME minimisationの概要と具体的な動作について解説

QNAME minimisation

- フルリゾルバーの**名前解決アルゴリズムを変更する**
  - 委任元の権威DNSサーバーには、**委任先のみを問い合わせる**
- ルートサーバーやTLDの権威DNSサーバーには、**委任先のドメイン名を利用するつもりであることのみが伝わる**
  - スタブリゾルバーが問い合わせたドメイン名・タイプを**知ることができなくなる**



QNAME minimisationに使われる問い合わせタイプ（1/2）

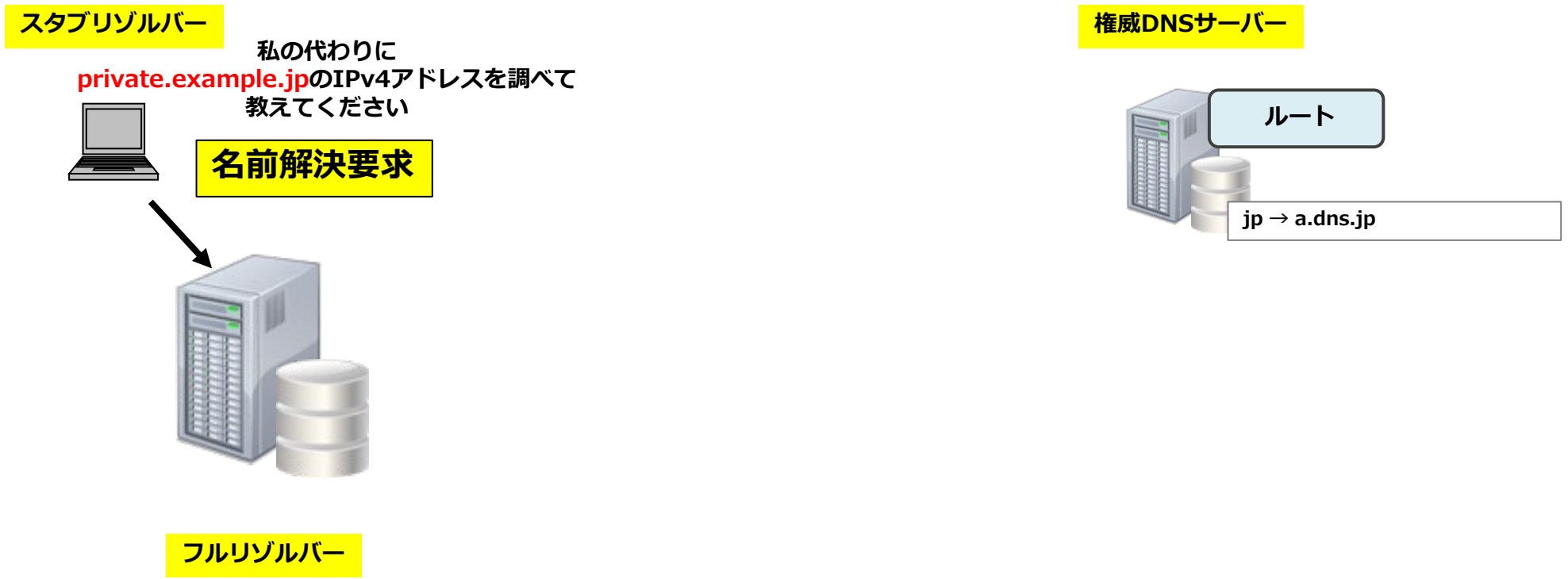
- 2016年に発行されたRFC 7816ではQNAME minimisationの問い合わせのタイプとして、**NSレコード**を使っていた
  - 委任情報は親ゾーンに**NSレコード**で記述される
- QNAME minimisationの運用において、**NSレコードの問い合わせを適切にサポートしないDNSソフトウェアやアプライアンス製品**が存在し、それらに対してQNAME minimisationで問い合わせた場合、**名前解決に失敗してしまう**場合があるという**問題点**が報告された

QNAME minimisationに使われる問い合わせタイプ（2/2）

- この問題を避けるため、2021年発行のRFC 9156ではそれまでの運用で得られた知見を反映する形で、**仕様と動作の改良**が行われた
  - RFC 9156ではQNAME minimisationの問い合わせタイプとして、**AまたはAAAAレコードの使用を推奨**している
  - 問い合わせタイプにA/AAAAを使うことで**問題を回避**し、かつ、QNAME minimisationを**従来のDNS問い合わせに紛れ込ませる**ことができるため、プライバシーをより高めることができる

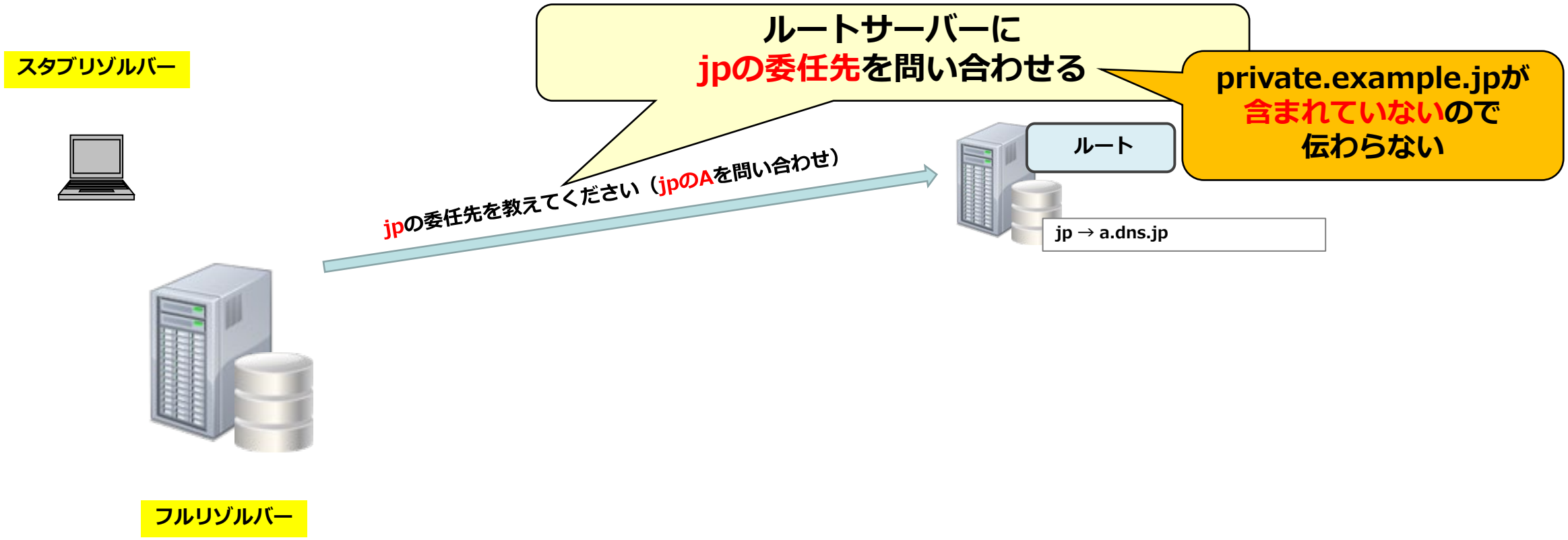
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



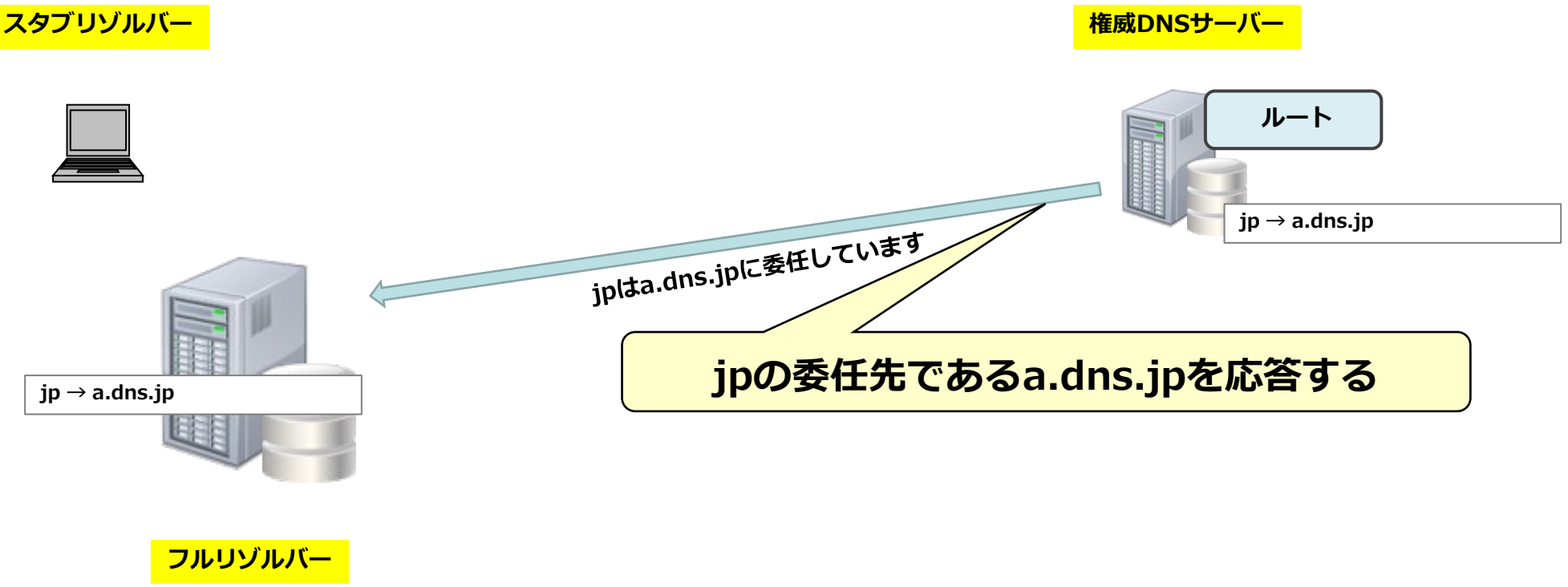
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



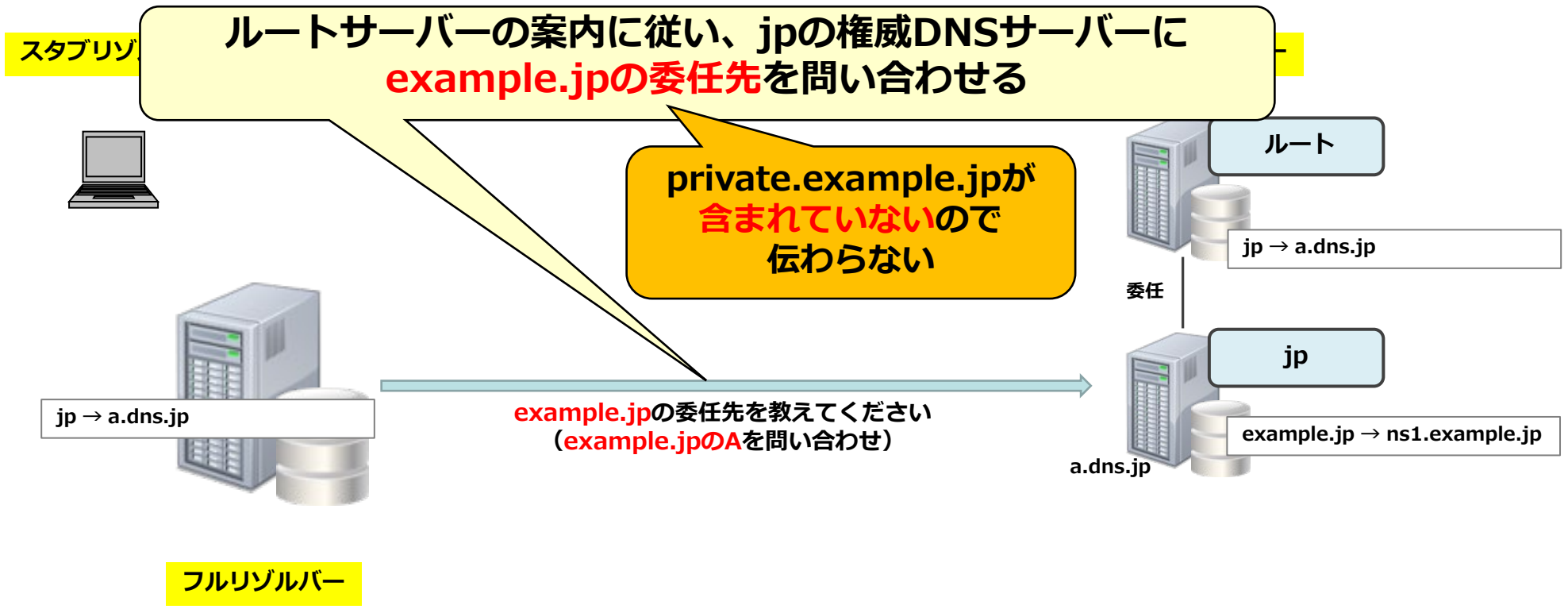
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



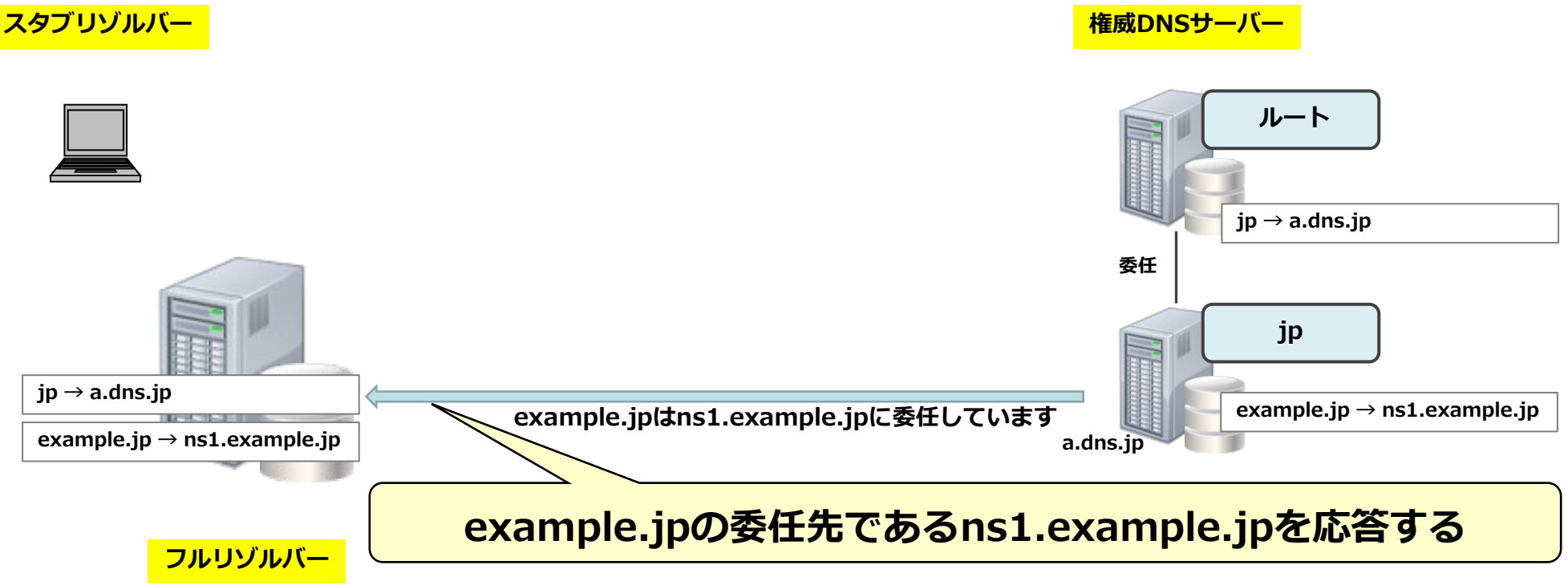
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



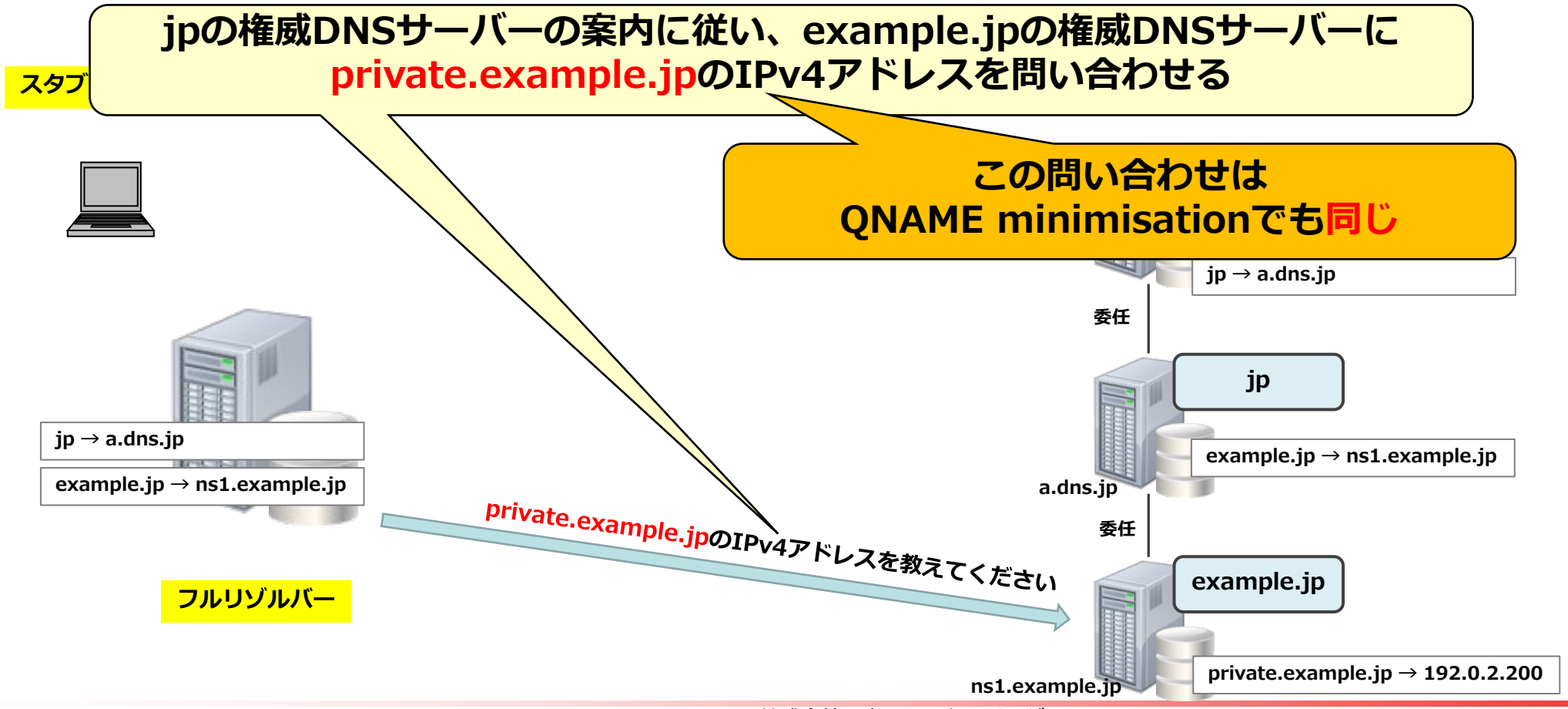
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



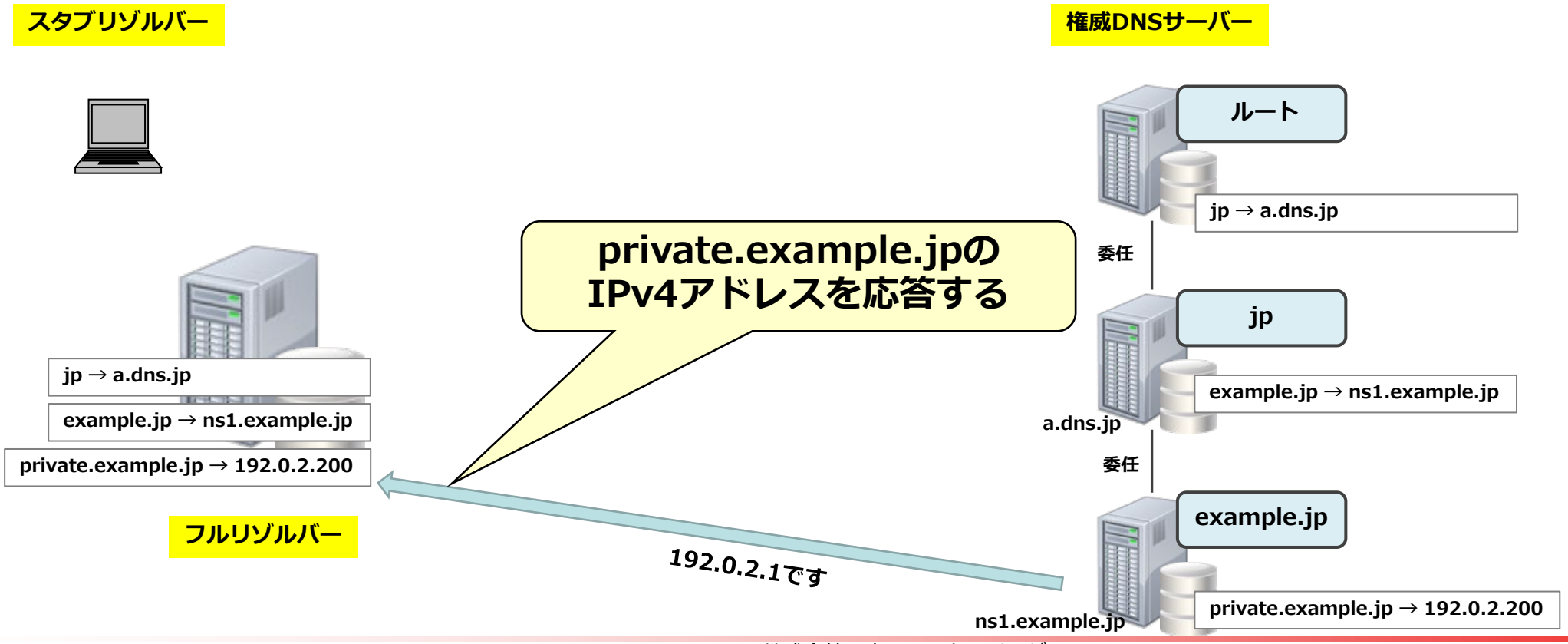
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



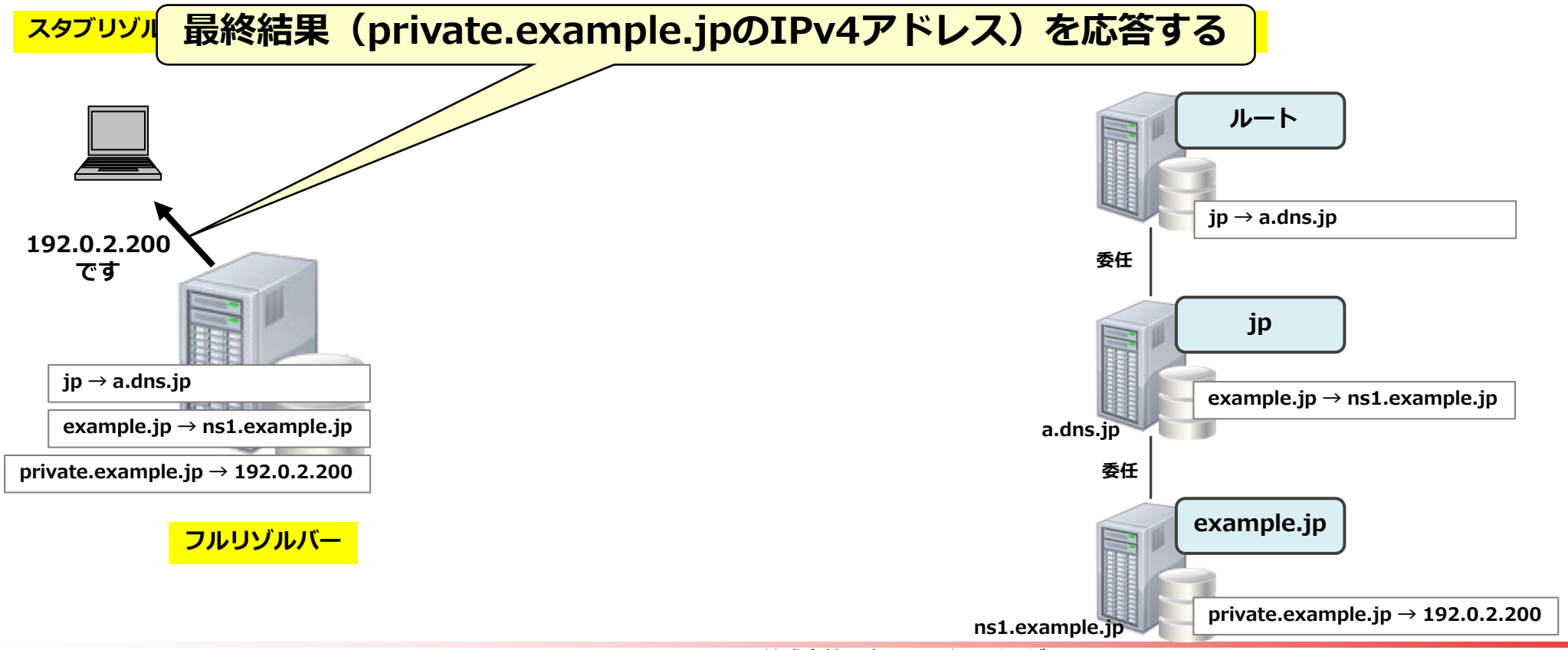
QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



QNAME minimisationによる名前解決の流れ

private.example.jpのIPv4アドレスを名前解決する例で説明する



QNAME minimisationの対応状況

- QNAME minimisationの標準化により、主な**DNSソフトウェアやパブリックDNSサービス**の**名前解決のデフォルトの動作がQNAME minimisationに変更**され、**標準の動作**となっている

まとめ：プライバシー上の懸念点と名前解決の動作の変化

- DNSにおける**プライバシー上の懸念点**を解決し、**機密性**を確保するための仕組みの導入が進められた
  - 通信路の暗号化（**DoT・DoH・DoQ**）
  - 問い合わせ内容の最小化（**QNAME minimisation**）
- 仕組みは**既に普及しており、名前解決の動作にも影響を及ぼした**
- 次のパートでは電子署名の技術を使ってDNSの安全性を高める、**DNSSECの概要と仕組み**について解説する

4. DNSSECの概要と仕組み

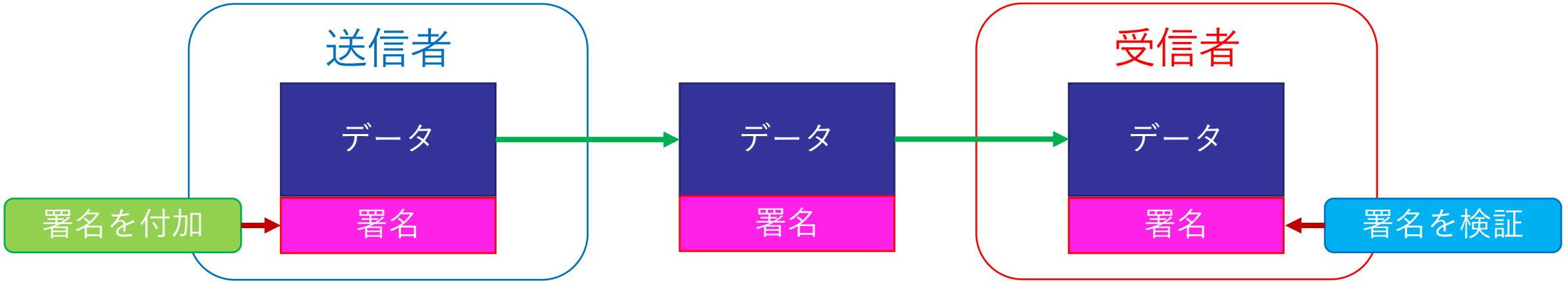
DNSSECとは？

- **DNS Security Extensions**
- DNS応答に**電子署名を追加**し、**受け取り側で検証する**ことでDNSの攻撃耐性を向上させる**セキュリティ拡張機能**

電子署名とは？

電子署名とは？

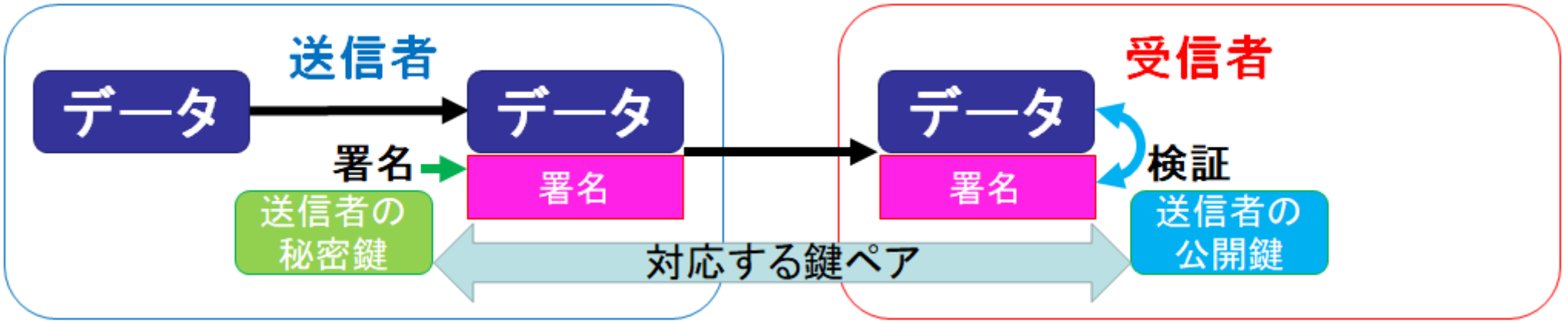
- データの送信者が、データに付加する（**署名する**）情報
- データの受信者が付加された電子署名を**検証する**ことで、**受け取ったデータに関する以下の2項目を確認できる**
  - **送信者が署名したデータである**こと（**出自の認証**）
  - 受け取ったデータに、**改ざんや欠落が見られない**こと（**完全性の検証**）



公開鍵暗号方式による電子署名

電子署名は暗号技術で実現されるが、
データの暗号化はしない

- 電子署名は、**公開鍵暗号方式**で実現できる
  - 対応する**公開鍵**と**秘密鍵**で構成される、**鍵ペア**を使う暗号方式
- **誰が・何で・何をするかを整理して理解する**ことが重要
  - **送信者**が**自身の秘密鍵**で**署名を付加する**（**署名できるのは送信者のみ**）
  - **受信者**が**送信者の公開鍵**で**署名を検証する**（**どの受信者も検証できる**）

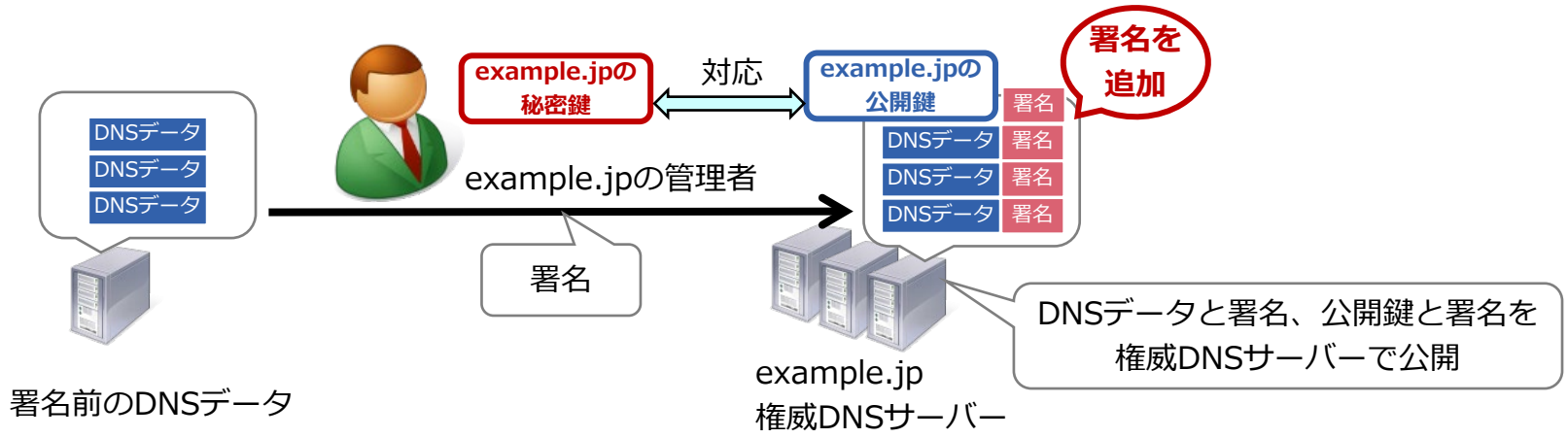


DNSSEC = 電子署名をDNSに適用

- DNSSECでは**ゾーンの管理者**がデータの送信者（**署名者**）となり、**DNSデータ**が署名対象となる
  - DNSデータは、リソースレコードセット（RRset）[*1]単位で署名される

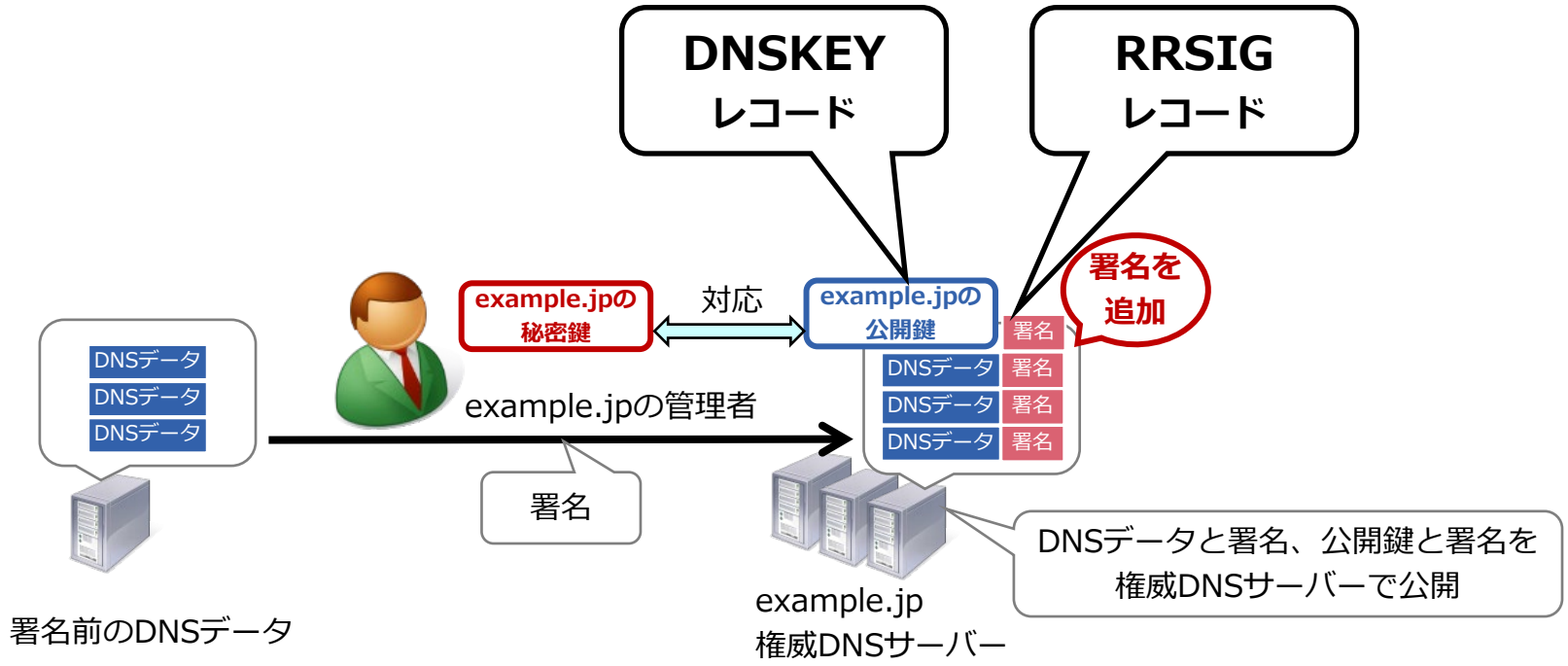
[*1] 同じドメイン名、タイプ（型）、クラスを持ち、データが異なるリソースレコードの集合。

- 署名者は**DNSデータと対応する署名、署名に使った秘密鍵に対応する公開鍵とその署名**を、自分のゾーンの**権威DNSサーバーで公開**する



DNSKEYレコードとRRSIGレコード

- 公開鍵と署名も、DNSデータとして公開される
  - 公開鍵は**DNSKEYレコード**、署名は**RRSIGレコード**として公開される



DNSKEYレコードとRRSIGレコードの例

```
jprs.jp.                    86400 IN DNSKEY 256 3 8 (
                                    AwEAAcIDGoI2vY5POBJ+bQYofna89YoUqe9h7R+8xHY3
                                    euzIHMrjmT4e6/NWhzNaJjEXn9xXdhViUbJoHPvLWYvE
                                    lJlktKXEZnswfBt7n9Eh0XkMUkR2cPPL30xeYIC8wYLl
                                    c7+3rd83TAfWhWX2eou3oigCoQxsKTYDHFQwQ9hz4XFX
                                    ) ; ZSK; alg = RSASHA256 ; key id = 51633
```

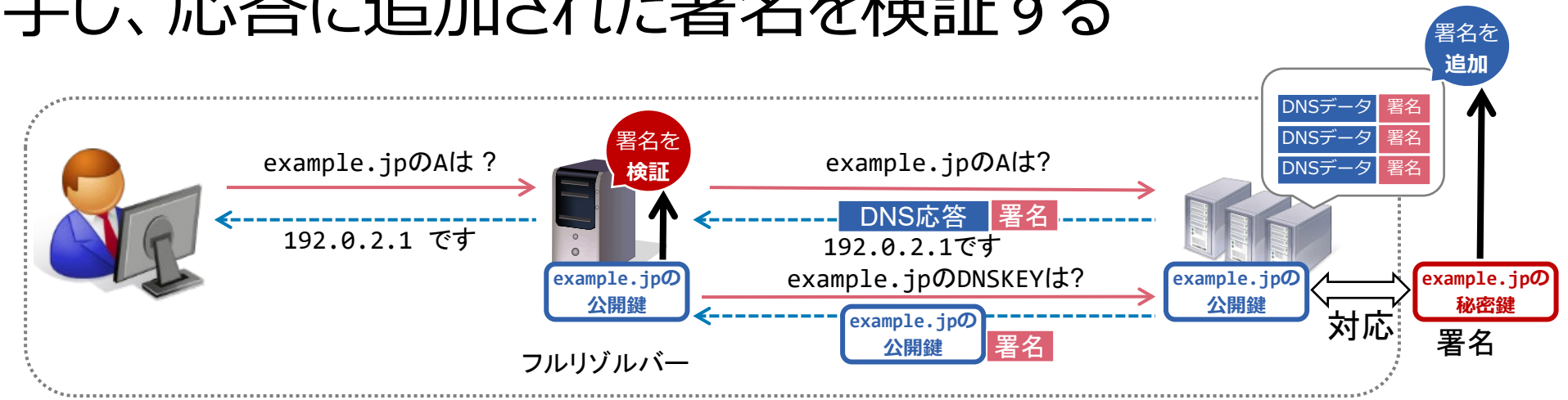
DNSKEYレコードの例

```
jprs.jp.                     300 IN RRSIG A 8 2 300 (
                                    20251124023003 20251025023003 51633 jprs.jp.
                                    gQAQLUoT8Fi2uBP1L8jcb51+KvUMoXpzPt0fKsyeO4n7
                                    86FG+2SeF8Yu0nZxDte4TMbRLHTtSV4NP9lznI7Jlu6f
                                    9dM1S8jHmR1Uijveau3oGyMCxCxmjZzLjH9JQe8kIbXc
                                    HP8dt/7MaviO6DQQlniDPGao50NMrlyxlmtc4QY= )
```

RRSIGレコードの例

署名の検証

- DNSSECではデータの受信者である**リゾルバー**が**署名を検証**する
  - **フルリゾルバー**で検証することが一般的
- フルリゾルバーは検証に必要なDNSKEYレコードをDNS問い合わせで入手し、応答に追加された署名を検証する



権威DNSサーバー側で応答に署名を追加し、フルリゾルバー側で応答の署名を検証

信頼の連鎖

- 署名を検証するためには、入手した公開鍵（**DNSKEYレコード**）を**信頼できる**必要がある
- **DNSKEYレコードを信頼できるようにする**ため、DNSSECでは、
  - 自分の**DNSKEYレコードに対応するDSレコード**を、**親ゾーンに登録**する
  - 親ゾーンは**DSレコード**を**自身の秘密鍵で署名**し、**自身のゾーンで公開**する

  ことで、公開鍵の信頼を**親ゾーンに担保してもらう**ようになっている
- この仕組みを、親子間の**信頼の連鎖**と呼ぶ

DSレコードの例

```
jprs.jp.                    7200 IN DS 7240 8 2 (
                                    E147A85589E24FE0DBB5980C73501B5DD656BE555071
                                    4F150BE574AE8777B77D )
```

DSレコードの例（jpゾーンに登録・設定）

鍵の番号（鍵タグ）が同じ

```
jprs.jp.                    86400 IN DNSKEY 257 3 8 (
                                    AwEAAcmfhDGesW94oy+xPJTdEsYsp+w1y4t+MoteuhEv
                                    J1PAhuVCf6uy5fC9uu0ctxgtPb19ZOzE5HZOvWHJ4Tix
                                    EHQiJHAsqB0izTiWM1Sx00PQdvAJamBOgFbjprTAtDaR
                                    R3lTQv/8tiFlLxPyYK/FMS+BZb9QBFU79RhjtJQaERV/
                                    bkc/J6FEc8DA11algcVBpP/X9SnyussY5J/6FBU95GIJ
                                    CfZIDavg0uGh/43o91Y7DS4AtEJEtzBBRVcIln5cdE7G
                                    f9og+WEagqhU9vcr9pMPxsx53JWYEiHbq1yRCtW7RCZ8
                                    p0kDuc+UVwYNZ2LnwzsimnRzvD2TlBMtLgKqxas=
                                    ) ; KSK; alg = RSASHA256 ; key id = 7240
```

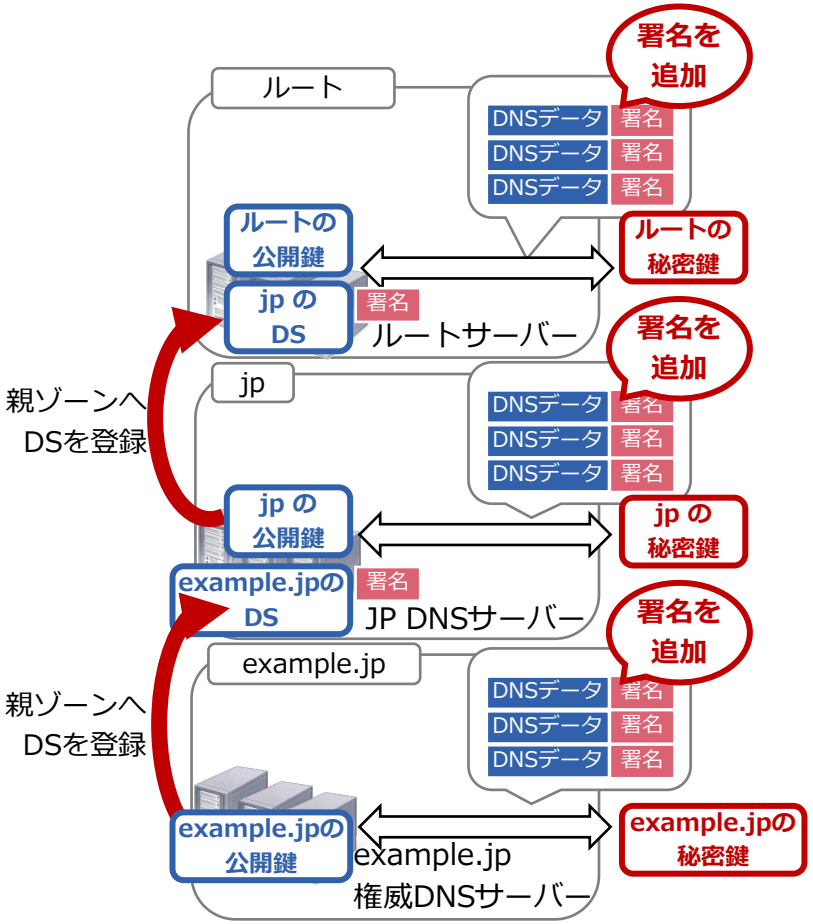
参考：このDSに対応するjprs.jpゾーンのDNSKEYレコード

信頼の連鎖の構築

- 子ゾーンが自身の公開鍵から**DSレコード**を作成し、**親ゾーンに登録する**

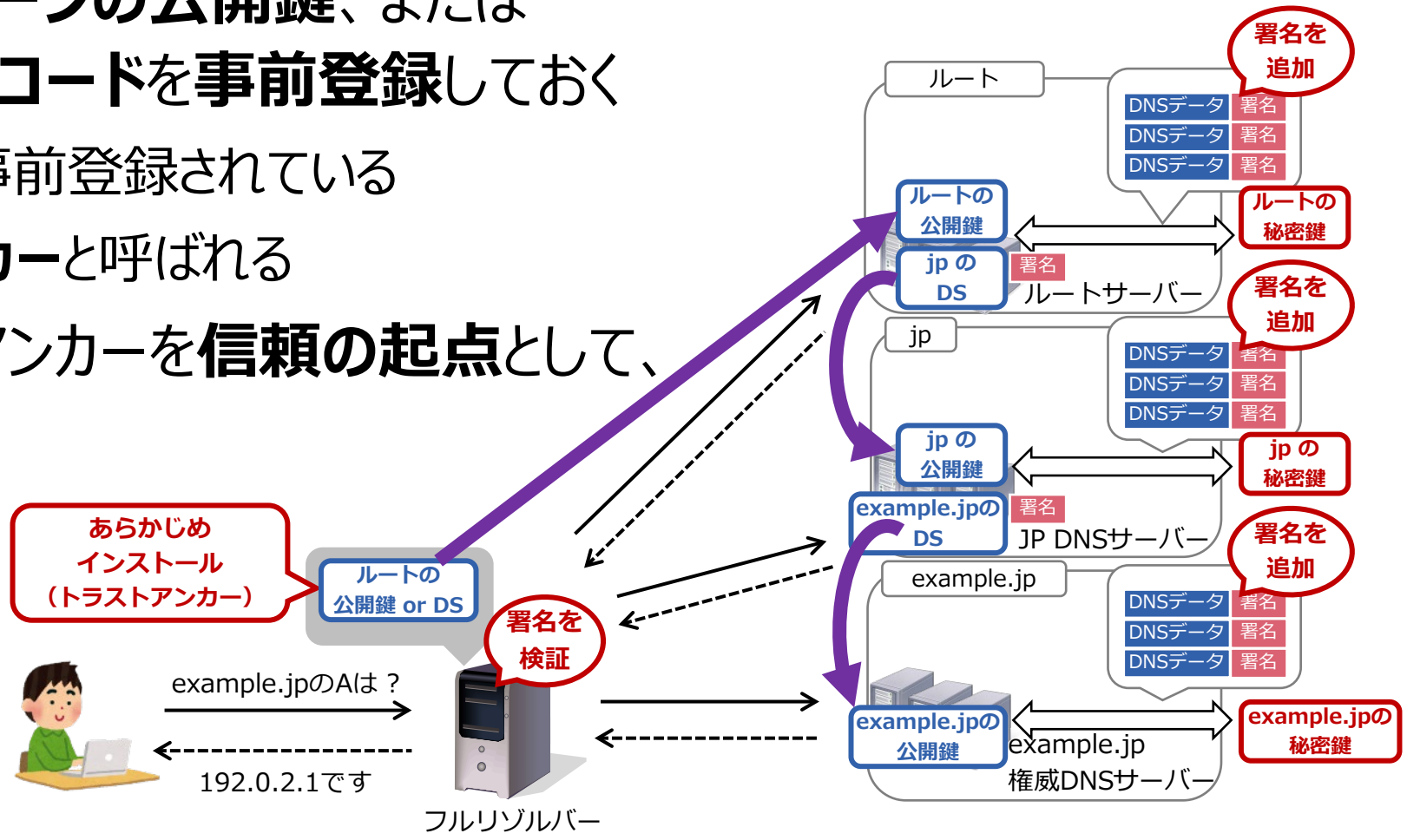
子ゾーンの公開鍵（DNSKEYレコード）と親ゾーンのDSレコードが一対一で対応する

- 親ゾーンは登録されたDSレコードを**自身の公開鍵で署名**し、**DNSデータとして公開する**



信頼の連鎖の検証

- フルリゾルバーに**ルートゾーンの公開鍵**、または**公開鍵に対応するDSレコード**を**事前登録**しておく
  - 主なフルリゾルバーには事前登録されている
  - この情報は**トラストアンカー**と呼ばれる
- フルリゾルバーはトラストアンカーを**信頼の起点**として、信頼の連鎖を**検証する**



DNSSECで使われる2対の鍵

- 電子署名の仕組みをDNSSECにそのまま適用するのであれば、ゾーンごとの鍵は**1対**（公開鍵と秘密鍵のセットが**1セット**）でよい
- しかし、実際のDNSSECの運用では、ゾーンごとに**2対の鍵**（公開鍵と秘密鍵のセットが**2セット**）が使われる
- なぜ、2対の鍵が使われるのか？

2対の鍵が使われる理由（1/2）

- DNSSECでは安全性の確保のため、**定期的な鍵の更新（ロールオーバー）**が必要になる
- 安全性の観点から見ると、鍵は**頻繁に更新**したい
- しかし、鍵が1対だと鍵の更新の際に、親ゾーンに登録した**DSレコードの更新**が常に必要になる
  - DSリソースレコードの更新には**親ゾーンへの登録**と**適切な手順**が必要で、**手間がかかる**

2対の鍵が使われる理由（2/2）

- そのため、DNSSECでは
  ① **親ゾーンに登録する（信頼の連鎖を構築する）**ための鍵
  ② **自分のゾーンを署名する**ための鍵
  を分け、運用をしやすくするため、**2対の鍵を使う方式**が採用された
- それぞれの鍵を、
  ① **Key Signing Key（KSK：鍵署名鍵）**
  ② **Zone Signing Key（ZSK：ゾーン署名鍵）**
  と呼ぶ

ZSKとKSK

- **KSK：鍵署名鍵**
  - 親ゾーンにDSレコードを登録して、**信頼の連鎖を構築する**ための鍵
  - 自分のゾーンの**公開鍵（DNSKEYレコードセット）のみに署名**する
- **ZSK：ゾーン署名鍵**
  - 自分のゾーンの**DNSKEY以外のDNSデータに署名する**ための鍵
  - ZSKの情報は**親ゾーンに登録されない**ため、高頻度に更新できる

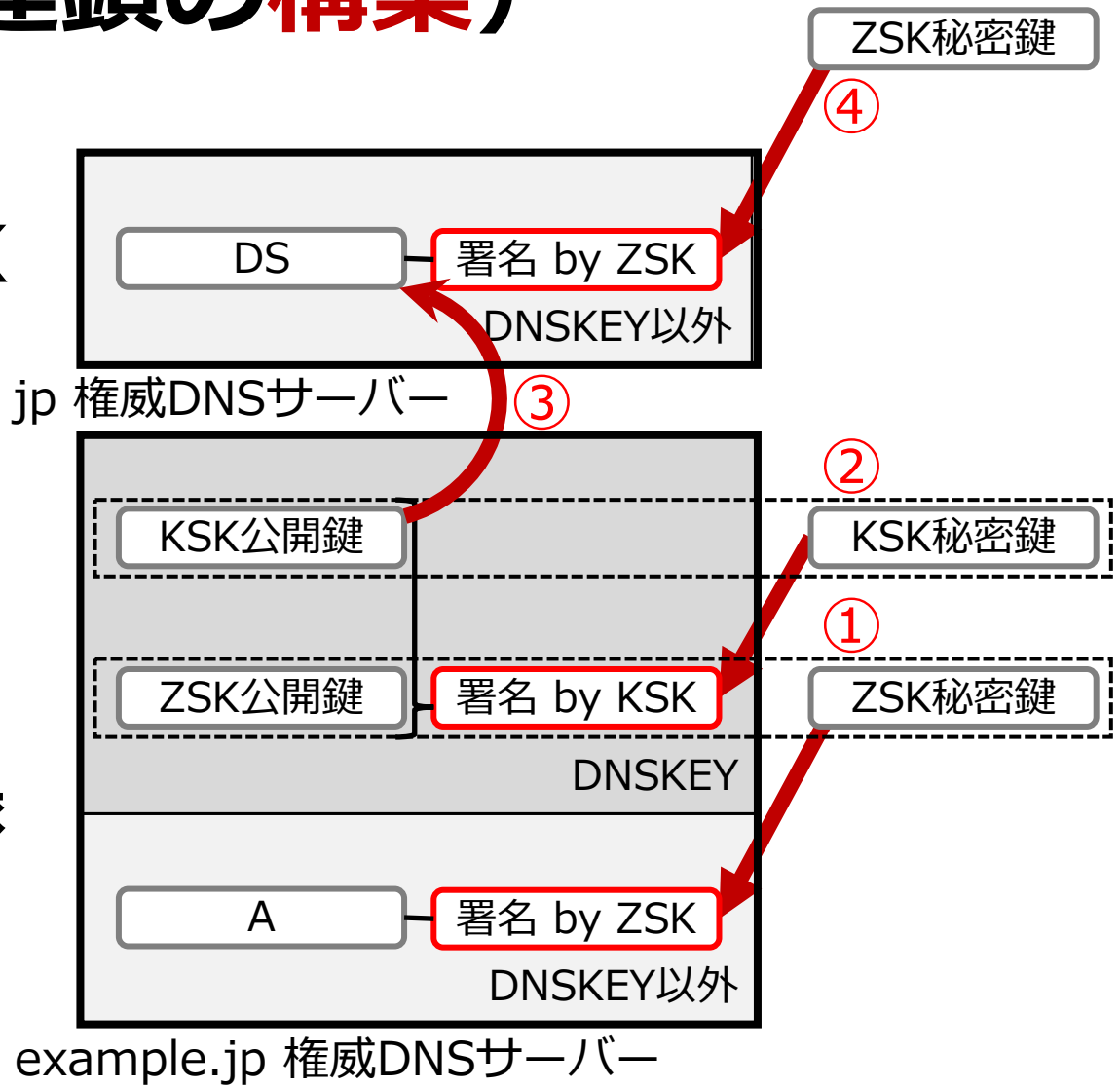
KSKとZSKを使ったDNSSEC（署名と信頼の連鎖の構築）

① そのゾーンの**DNSKEY以外のリソースレコードセット**（図ではA）を、**ZSKの秘密鍵で署名**する

② そのゾーンの**DNSKEYリソースレコードセット**（KSKとZSKが含まれる）を、**KSKの秘密鍵で署名**する

③ KSKに対応する**DSを親ゾーンに登録**する

④ DSは**親ゾーンのZSKで署名**される



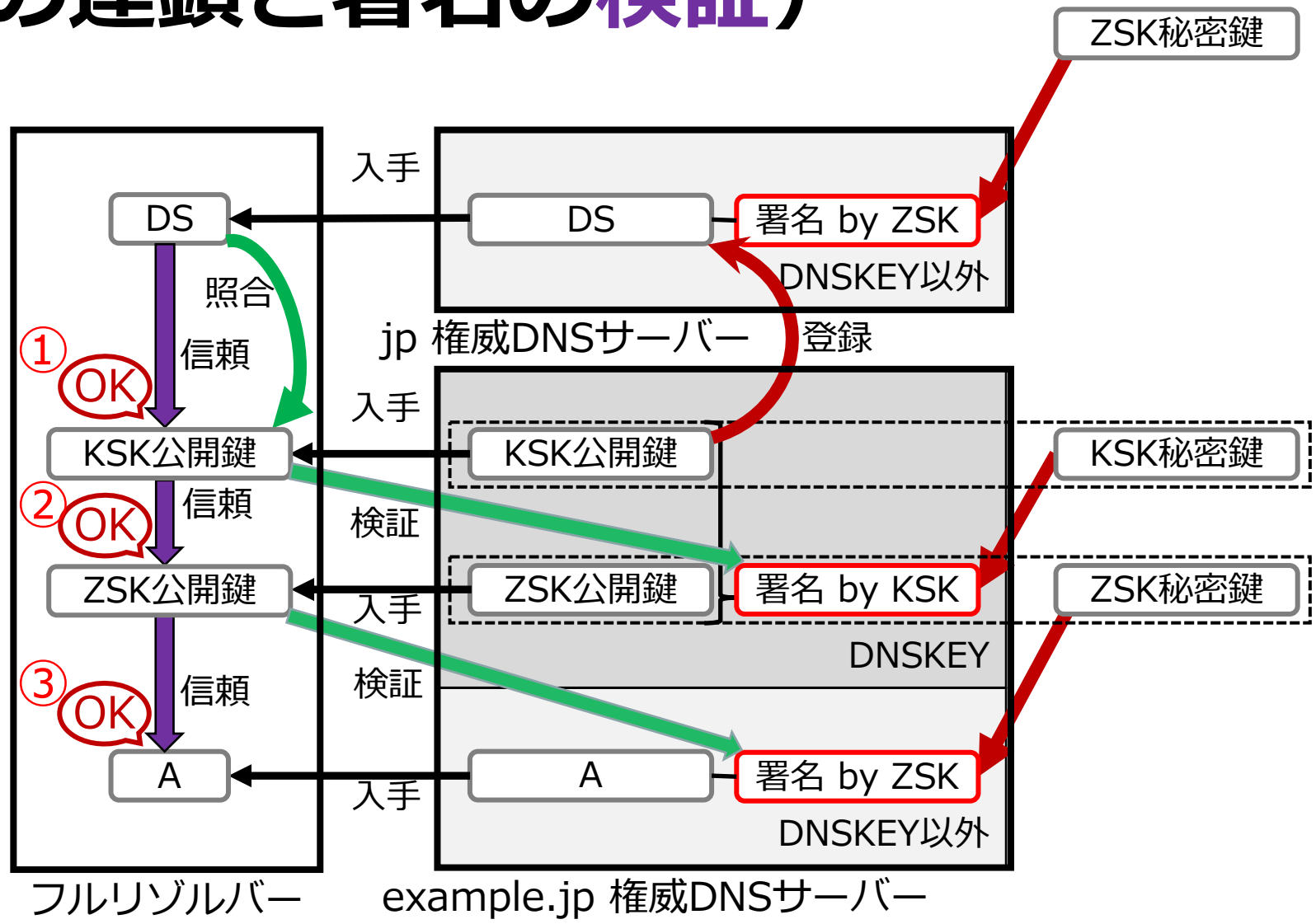
KSKとZSKを使ったDNSSEC（信頼の連鎖と署名の検証）

① **親のDSと子のKSKを照合**し、照合できたら**KSKを信頼**する

② 信頼したKSKで**DNSKEYレコードの署名を検証**し、検証できたら**ZSKを信頼**する

③ 信頼したZSKで**Aレコードの署名を検証**し、検証できたら**Aを信頼**する

トラストアンカーから最後のAレコードまでたどり着ければ検証成功



不在証明

- DNSの応答には「**問い合わされたデータは存在しない**」という応答も存在する（**不在応答**）
- **存在しないデータには署名できない**ため、 存在しないことを証明する際には**存在するデータを示し、そのデータに署名することで不在を証明する**ようになっている
- この方法を、**不在証明**と呼ぶ

存在する別のデータを使って、存在しないことを証明している

不在証明の方法

- 不在証明には、**NSECレコード**が使われる
- NSECレコードには、以下の情報が設定される
  - **そのドメイン名の「次のドメイン名」**
    - ドメイン名の**大文字を小文字に変換し、ASCIIコード順にソート**（以下を参照）
  - **そのドメイン名に存在するリソースレコードタイプの一覧**

【参考】「次のドメイン名」の決め方

例えば、**example.jp**ゾーンに「**www.example.jp**」 「**Mail.example.jp**」「**abc.example.jp**」のドメイン名が存在する場合、これらは **abc.example.jp** → **mail.example.jp** → **www.example.jp** の順にソートされる。

NSECレコードの例①

- **ルートサーバー**に**jpaのAレコード**を問い合わせた際の応答

```
$ dig +norec +dnssec jpa a @a.root-servers.net
 (略)
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 27877
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
 (略)
;; AUTHORITY SECTION:
jp.                     86400 IN NSEC jpmorgan. NS DS RRSIG NSEC
```

jpの次のドメイン名はjpmorganである

jpと**jpmorgan**の間には何も存在しない ⇒ **jpaは存在しない**

NSECレコードの例②

- **seの権威DNSサーバー**に**seのAレコード**を問い合わせた際の応答

```
$ dig +norec +dnssec se a @a.ns.se
 (略)
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17883
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1
 (略)
;; AUTHORITY SECTION:
se.                     7200 IN NSEC 0.se. NS SOA TXT RRSIG NSEC DNSKEY ZONEMD
```

seにはNS・SOA・TXT・RRSIG・NSEC・DNSKEY・ZONEMDレコードが存在する

seに存在するのは**NS・SOA・TXT・RRSIG・NSEC・DNSKEY・ZONEMD** ⇒ **Aは存在しない**

ゾーン列挙

- NSECレコードには**そのドメイン名の次のドメイン名**と**存在するリソースレコードタイプの一覧**が記載されるため、NSECレコードの情報をもとに**外部からDNS検索を繰り返す**ことで、**そのゾーンに存在するすべてのDNSデータを入手する**ことが可能になる
- この行為を**ゾーン列挙（zone enumeration）**と呼ぶ
  - 管理者が自分のゾーン情報を秘匿したい場合、DNSSEC導入の妨げとなる

対策：NSEC3レコードの追加導入

- この問題に対応するため、**名前そのものに替えて名前のハッシュ値を利用した不在証明**を追加導入し、**ゾーン列挙のコストを上げる**対策が実施された
- この不在証明には、**NSEC3レコード**が使われる
  - 管理者は自分のゾーンをDNSSEC署名する際、**NSECとNSEC3のどちらを使うかを選択できる**
  - ルートゾーンや逆引き用ゾーン（in-addr.arpa/ip6.arpa）のゾーンデータは公開情報であるため、NSECが使われている

NSEC3レコードの出力例

- **jpの権威DNSサーバー**に**example.jpのAレコード**を問い合わせた際の**不在応答**（抜粋）

```
$ dig +norec +dnssec example.jp a @a.dns.jp
 (略)
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 35926
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1
 (略)
;; AUTHORITY SECTION:
tdlmdp7ltobedc177lpcnee1na35a0c4.jp. 900 IN NSEC3 1 1 0 - (
                                    TE1O6F2U5UJJB88TL9V9L1H2U549M6F8
                                    NS DS RRSIG )
 (略)
```

ハッシュ値を示すNSEC3レコードが出力されている

まとめ：DNSSECの概要と仕組み（1/3）

- DNS応答に**電子署名を追加**し、**受け取り側で検証**する**拡張機能**
  - **公開鍵**と**秘密鍵**の**鍵ペア**を使う、**公開鍵暗号**方式で実現される
- 署名を検証することで、**以下の2項目を確認**できる
  - **送信者が署名したデータである**こと（**出自の認証**）
  - 受け取ったデータに、**改ざんや欠落が見られない**こと（**完全性の検証**）
- 各ゾーンの公開鍵は**DNSKEYレコード**、署名は**RRSIGレコード**として公開される

まとめ：DNSSECの概要と仕組み（2/3）

- **ゾーンの管理者**はDNSKEYレコードに対応する**DSレコード**を**親ゾーンに登録・公開**することで、**信頼の連鎖を構築**する
- リゾルバーは**トラストアンカー**を起点として、**信頼の連鎖を検証**する
- 運用をしやすくするため、ゾーンごとに**2対の鍵**が使われる
  - **KSK**（Key Signing Key：**鍵署名鍵**）
  - **ZSK**（Zone Signing Key：**ゾーン署名鍵**）

まとめ：DNSSECの概要と仕組み（3/3）

- **データの不在**を証明するため、**不在証明**という仕組みが存在する
- 不在証明の方式には、**NSECレコード**によるものと**NSEC3レコード**によるものの2種類が存在する
  - 自分のゾーンをDNSSEC署名する際、**どちらを使うかを選択できる**
  - ルートゾーンや逆引き用ゾーンでは、NSECが使われている

おわりに

- 本チュートリアルでは「**90分で学び直すDNSとDNSSECの基本**」と題し、前半では**DNSの構成要素と分散管理の仕組み**、**名前解決の概要と具体的な動作**について解説しました。
- 加えて、**プライバシー上の懸念点を解決するために導入された仕組み**と、それによる**名前解決の動作の変化**についても解説しました。
- 後半では**DNSのセキュリティ拡張**である、**DNSSECの概要と仕組み**についても解説しました。
- 本チュートリアルが**インターネットを支える重要な基盤技術の一つ**である、**DNSについて学び直すきっかけ**となれば幸いです。

ご視聴ありがとうございました！

<https://jprs.jp/tech/>

@JPRS_official　JPRSofficial　JPRSpress