各レイヤーの役割およびインフラ基礎

エンジニア内定者研修第2回

目次

- インターネットの構成を表現するモデル
- データリンク層
- ネットワーク層
- トランスポート層
- アプリケーション層
- AWSでインフラ構築してみよう

インターネットの構成を表現するモデル

インターネットの構成を表現するモデル

プロトコルが階層構造で表現される

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTP,無線

引用：https://www.itmanage.co.jp/column/osi-reference-model/

プロトコルとは？なぜ階層構造？

プロトコルとは

なぜ世界中がつながるのか？

異なるメーカーの機器が通信できる理由

- iPhone ⇔ Windowsサーバ ⇔ プリンタ ⇔ IoT機器
- 全て違うメーカー、違うOS、違う目的
- でも問題なく通信できる
- **その秘密は「プロトコル」という共通の約束事**

プロトコル = 機械同士の「厳密な約束事」

- 例：USB-C規格
  - 形状、ピン配置、電圧、データ転送方法を完全に定義
  - だからどのメーカーの機器も接続できる
- 人間の言語と違い、曖昧さが許されない
- 世界共通の「仕様書」に従って作られる

共通規格（ピン，電圧）

なぜ階層構造なのか

ネットワーク通信は複雑

- アプリケーションのデータ形式
- データの分割と組み立て
- エラーの検出と再送
- 経路の選択
- 物理的な電気信号への変換

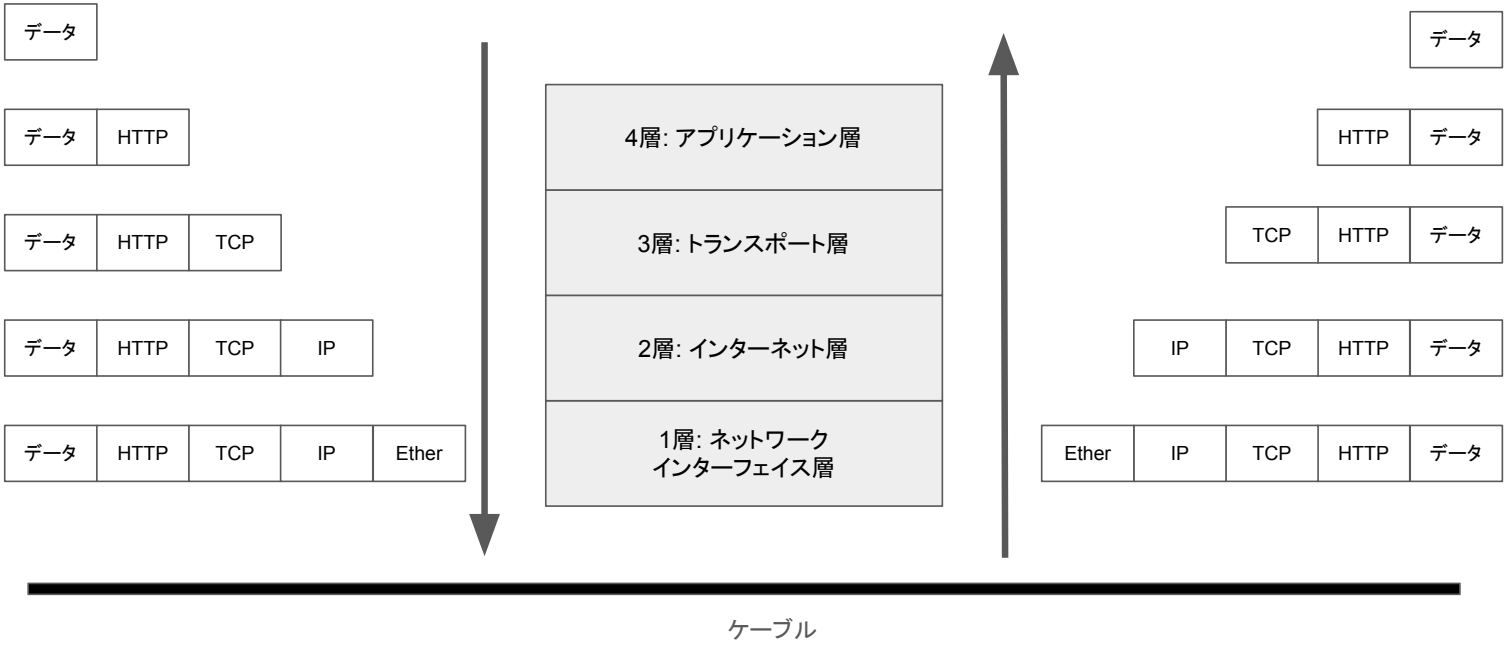
これを全部一度に考えるのは無理！

複雑な問題を「層」に分けて解決

- 建築の分業と同じ発想：
  - 基礎工事業者 → 大工 → 電気工事 → 内装
  - 各専門家は自分の仕事に集中
- **ネットワークも役割ごとに「層」に分ける**
  - 各層は特定の役割だけを担当
  - 下の層の複雑さを上の層から隠す（抽象化）

プロトコル層同士の関係

上層→下層の順に処理されていく= **カプセル化**



具体例:「Wi-FiからLTEに切り替わってもデリッシュキッチンが使える」

- デリッシュキッチンアプリ(上位層)は何も変更しなくていい
- 物理的な通信方法(下位層)だけが切り替わる
- **各層が独立** = 下の変更が上に影響しない
- メリット:
  - 技術の独立した進化が可能
  - 問題の切り分けが容易
  - 開発の分業化

代表的なモデル

TCP/IP（4層モデル）

- **1974年**：Vint Cerf、Robert Kahnが原型を発表
- **1983年1月1日**：インターネットの標準プロトコルとして採用
- 米国防総省DARPA主導で開発
- 実装から生まれた実用モデル

OSI参照モデル（7層モデル）

- **1984年**に標準化
- ISO（国際標準化機構）が策定
- 理論的に整理された参照モデル

TCP/IPとは
- 4層のプロトコルで表現されたモデル

TCP/IPの名前の由来
- TCPとIPという代表的なプロトコルの名前からきている
  - TCP(Transmission Control Protocol)
  - IP(Internet Protocol)

TCP/IPの階層構造のイメージ

レイヤー	階層
4層	アプリケーション層
3層	トランスポート層
2層	インターネット層
1層	ネットワーク インターフェイス層

OSI参照モデル：理想的なあるべき姿として作られた階層構造のモデル

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTPなど

OSI参照モデルとTCP/IP

理想

OSI参照モデル

レイヤー	階層
7層	アプリケーション層
6層	プレゼンテーション層
5層	セッション層
4層	トランスポート層
3層	ネットワーク層
2層	データリンク層
1層	物理層

現実

TCP/IP

レイヤー	階層
4層	アプリケーション層
3層	トランスポート層
2層	インターネット層
1層	ネットワーク インターフェイス層

OSI参照モデル

本研修では2～4層、7層について説明します

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTPなど

引用：https://www.itmanage.co.jp/column/osi-reference-model/

代表的なプロトコル

HTTP(HyperText Transfer Protocol)	WebサーバーとWebブラウザの間で、Web情報をやり取りするためのプロトコル
FTP(File Transfer Protocol)	ファイルを転送するためのプロトコル
DNS(Domain Name System)	ドメイン名を用いてインターネットを利用できるようにするプロトコル
SMTP(Simple Mail Transfer Protocol)	Eメールを送信するために使用されるプロトコル
Telnet	ネットワークに繋がれたコンピュータを遠隔操作するための仕組み
TLS(Transport Layer Security)	インターネット上のウェブブラウザとウェブサーバ間でのデータの通信を暗号化し、送受信する仕組み
NetBIOS(Network Basic Input/Output System)	ソフトウェアがコンピュータのネットワーク機能を利用するための標準的な規約(API)を定めた規格の一つ
TCP(Transmission Control Protocol)	1対1のセッションによる信頼性の高い通信を行うためのプロトコル
UDP(User Datagram Protocol)	相手との接続確立を行わずにデータを送信し、到達確認や再送処理などを行わないプロトコル

代表的なプロトコル

NetWare/IP	オペレーティングシステム
IP(Internet Protocol)	データパケットがネットワークを経由して正しい宛先に到達できるように、データパケットをルーティングおよびアドレス指定するためのプロトコル
ARP(Address Resolution Protocol)	IPアドレスからネットワーク上のMACアドレスを求めるためのプロトコル
RARP(Reverse Address Resolution Protocol)	コンピュータにIPアドレスを通知する仕組み
ICMP(Internet Control Message Protocol)	TCP/IPでネットワークの疎通がされているノード間で、通信状態に確認をするために使われるプロトコル
PPP(Point to Point Protocol)	コンピュータ同士が1対1の通信を行うための規約を定めたプロトコル
Ethernet	パソコンなどの機器を有線接続し、信号のやり取りをする際に使われている通信規格
RS-232(Recommended Standard 232)	パソコンに標準搭載されるなど、最も広く使われているシリアル通信規格
UTP(Unshielded Twisted Pair)	LANケーブル

データリンク層

データリンク層（OSIとTCP/IP）

- データリンク層は通信機器（ハードウェア）同士の通信を担当する
- キーワード：**イーサネット（Ethernet）、MACアドレス**

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTPなど

なぜデータリンク層が必要か？

物理層ができること

- 電気信号の送受信
- ビット列の伝送

物理層だけでは解決できない課題

- 誰から誰への通信か不明
- データの境界が不明
- エラーを検出できない
- 複数の機器が同時に送信すると衝突

→ データリンク層がこれらを解決

コンピュータや機器同士を有線で接続するために最も一般的な通信規格

イーサネットの特徴

- 1973年にXerox PARC（ゼロックス パロアルト研究所）で開発
- 有線ネットワークの標準プロトコル
- 10Mbps → 800Gbps（2024年標準化）

イーサネットの役割

- フレームに分割してデータを送信する
- 送受信タイミングの制御
- エラー検出機能

データリンク層（Ethernet フレーム）

- イーサネットでは、データを**フレーム**という単位に分割して順に送信していく
- フレームにはそれぞれの役割に応じたフィールドが存在

宛先MAC 6 byte	送信元MAC 6 byte	Type 2 byte	Payload（データ本体） 46 ~ 1500 byte	FCS 4 byte

各フィールドの役割

- 宛先/送信元MAC：機器の識別
- Type：上位プロトコルの識別
- Payload：実際の通信内容
- FCS：エラーチェック用

MAC（Media Access Control）アドレス

データリンク層物理的な機器の識別子

MACアドレスの特徴

- 48ビット（6バイト）の固定長
- ネットワーク機器に一意に割り当て
- 工場出荷時に設定

表記方法

例：**00:1B:63:84:45:E6　　00-1B-63-84-45-E6**

00:1B:63	84:45:E6
OUI（24ビット）	UAA（24ビット）
ベンダー識別子（Apple Inc. に割り当てなど）	製品固有番号

IPアドレス vs MACアドレス

- ネットワーク機器を識別するIDとして、ネットワーク層の**IPアドレス**がある
- **ARP（Address Resolution Protocol）**：IPアドレスをMACアドレスに変換する仕組み

ネットワーク層：IPアドレスで最終宛先を指定
データリンク層：隣接機器間はMACアドレスで通信

実際の通信経路
PC-A → ルータA → ルータB → **PC-B**
MAC1　MAC2　MAC3　MAC4　各区間でMACアドレスを使って転送

IPアドレスは最終目的地で、ARPで変換後MACアドレスで次の中継点を探索する

ARPの動作

1. IPアドレスは分かるが MACアドレスが不明な状況

2. ブロードキャストで同一ネットワーク内の全機器に問い合わせ

3.該当するIPアドレスを持つ機器が自分の MACアドレスを返答

4. 取得したMACアドレスをキャッシュに保存（効率化のため）

```
1. PC-A「192.168.1.2のMACアドレスは？」
    （ARP Request をブロードキャスト）

2. PC-B「それは私です。MACは 00:1B:63:84:45:E6」
    （ARP Reply を返信）

3. PC-A「了解。ARPテーブルに記録」
    （以降はこの情報を使用）
```

ネットワーク層

ネットワーク層（OSIとTCP/IP）

ネットワーク層の責任

- 異なるネットワーク間の接続
- 最適経路の選択（ルーティング）
- 論理アドレス（IPアドレス）の管理

データリンク層との違い

- データリンク層: 隣接機器間の通信
- ネットワーク層: 最終宛先までの通信

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTPなど

IPアドレス

MACアドレスの課題

- フラット構造（階層がない）
- ネットワークの所属が分からない
- 宛先への経路を計算できない

IPアドレスが解決すること

- **階層的な管理**：ネットワーク部とホスト部
- **ネットワークの識別**：所属ネットワークが明確
- **経路制御が可能**：ネットワーク単位でルーテング

IPアドレス

IPv4

- 32ビット（4バイト）
- ドット10進表記：192.168.1.1
- 約43億個のアドレス

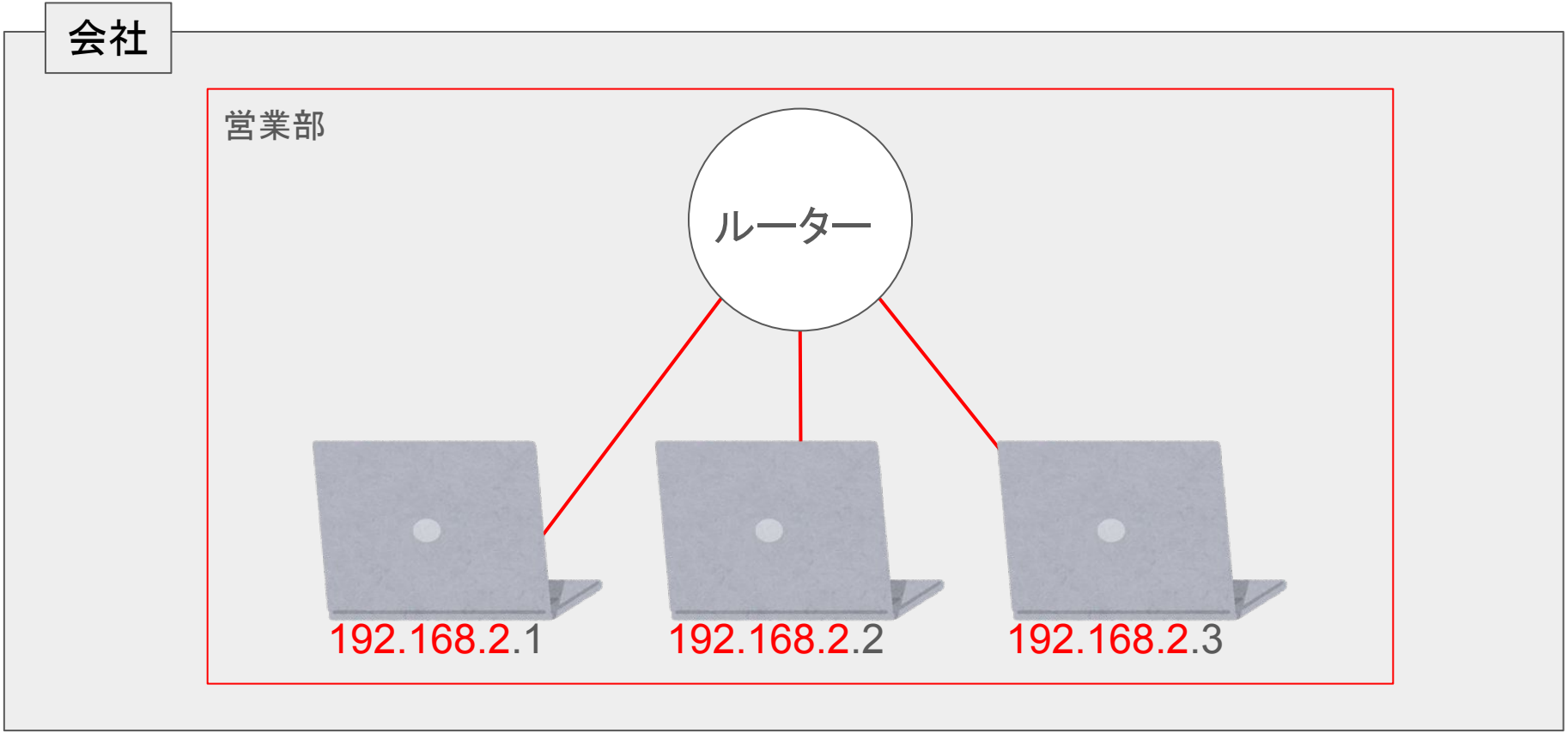
- ネットワーク部: 所属するネットワークを識別
- ホスト部: ネットワーク内の機器を識別
- サブネットマスク: ネットワーク部とホスト部の境界を識別

IPv6

- 128ビット（43億の4乗個のアドレス）
- 例:2001:0db8:85a3:0000:0000:8a2e:0370:7334

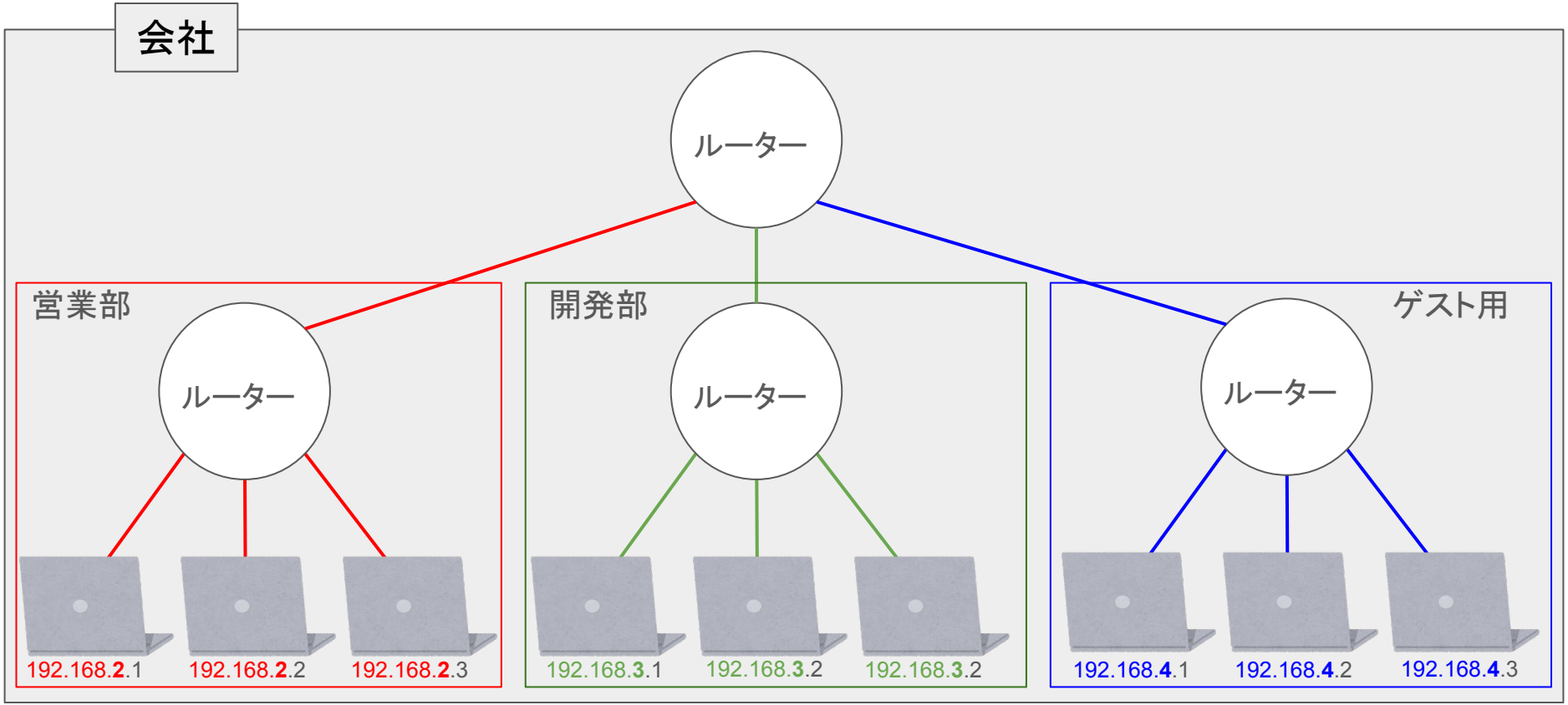
ネットワークセグメント

- 同一セグメントはIPアドレスのネットワーク部が同じ



ネットワークセグメント

- 異なるセグメントではIPアドレスのネットワーク部は異なる



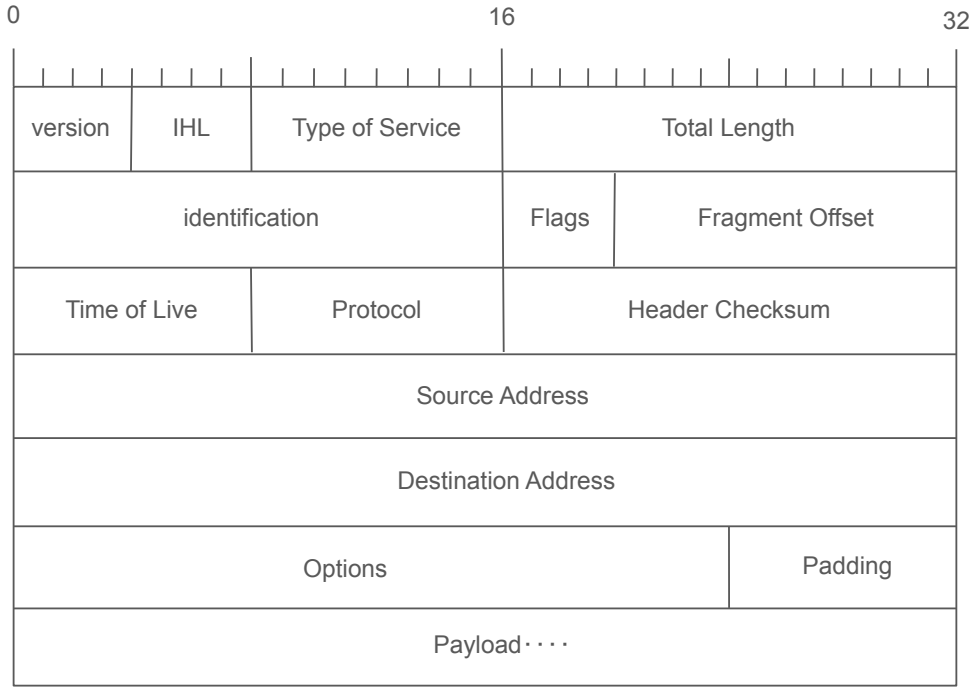
ネットワーク層の基本的なデータ形式

パケットの基本要素

- IPヘッダ: ルーティングに必要な情報
- ペイロード: 実際に送りたいデータ

- Source Address: 送信元IPアドレス
- Destination Address: 宛先IPアドレス

IPヘッダのフォーマット



ルーター

- 異なるネットワークを繋ぎ、パケットを中継する装置
- 別のルータ・ホストから送られてきたパケットを、次のルータ・ホストに送る
- パケットは目的地に届くまでたくさんのルーターを経由する

ルーティングテーブル

- あるネットワークへ行くために次どこに送るべきかを示した「経路表」
- 基本要素は以下の3つ
  - route: 宛先ルート
  - next hop: そのrouteに一致した時に送る先（IPアドレス）
  - interface: どのネットワークインターフェースから送るか（データリンク層の識別子）

ルーティングと最長一致

- ルータは経路表を参照し、最長一致で次ホップを決定
- 転送経路の決定順序
  - 宛先IP → 経路表検索 → 最長一致の経路を選択
    → 次ホップIP → ARP/NDP解決 → 転送

宛先10.20.30.45

0.0.0.0/0	via 203.0.113.1	# デフォルト
10.0.0.0/8	via 10.0.0.1	
10.20.0.0/16	via 10.0.0.2	
10.20.30.0/24	via 10.0.0.3	# 宛先10.20.30.45が最長一致

グローバルIPアドレスとプライベートHPアドレス

グローバルIPアドレス

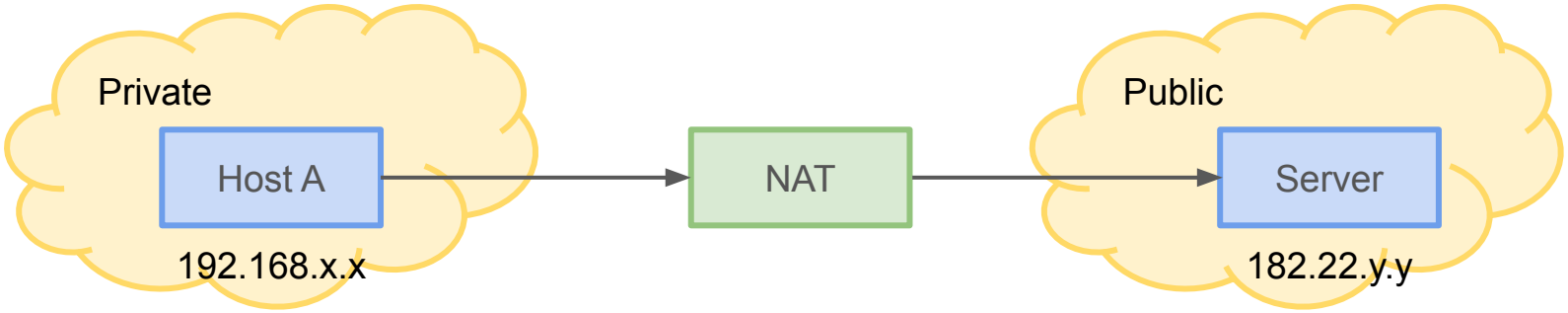
- インターネットに接続する際に割り当てられるIPアドレス
- プロバイダにと契約した際に割り振られる
- 重複しない

プライベートIPアドレス

- 特定のネットワーク内で割り当てられるIPアドレス
- 家や会社などで複数のパソコンルーターで繋げた場合に使われる
- 重複する

NAT(Network Address Translation)

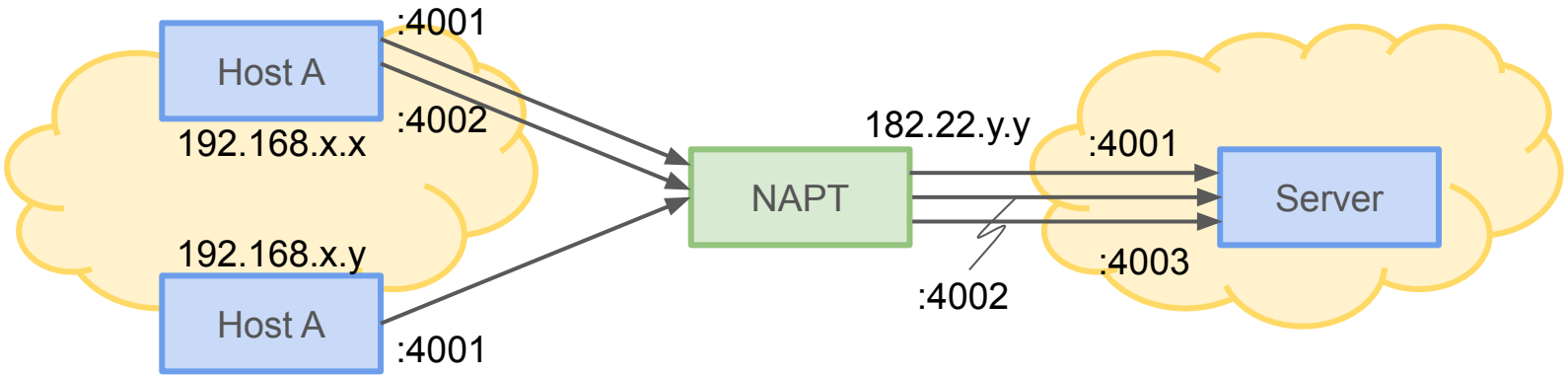
- NAT：アドレスの書き換え
  - プライベートIP ↔ グローバルIPの変換
  - 1対1の変換



NAPT (Network Address Port Translation)

- NAPT：TCP, UDPポートを含めて書き換え
  - 1つのIPアドレスで2つ以上のホストの同時接続が可能
  - 多対多の関係

NW接続を開始・終了するための仮想的な入口



トランスポート層

OSI参照モデル：理想的なあるべき姿として作られた階層構造のモデル

レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTP,無線

少しだけネットワーク層の復習

- ネットワーク層では、**どのコンピューターからどのコンピューターに送るか**　をIPアドレスで指定した
  しかし、1つのコンピューターでは大抵複数のアプリケーションが動いている

  →**どのアプリケーション（サービス）からどのアプリケーションに渡すか**　を決めないといけない

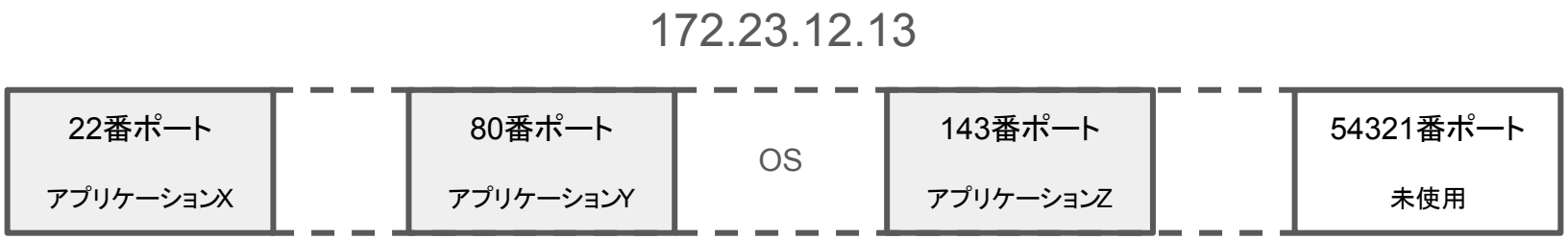
- ネットワーク層までは純粋に送ることに注力していた

  →**送り方**を決めないといけない

これらをトランスポート層が担っている

ポート・ポート番号

- **どのアプリケーション（サービス）からどのアプリケーションに渡すか**
- アプリケーションの出入り口
- OSが管理している
- 他のアプリケーションが使っているポートは基本的に別のアプリケーションから同時に使用できない



←ここでポート番号が
指定されている

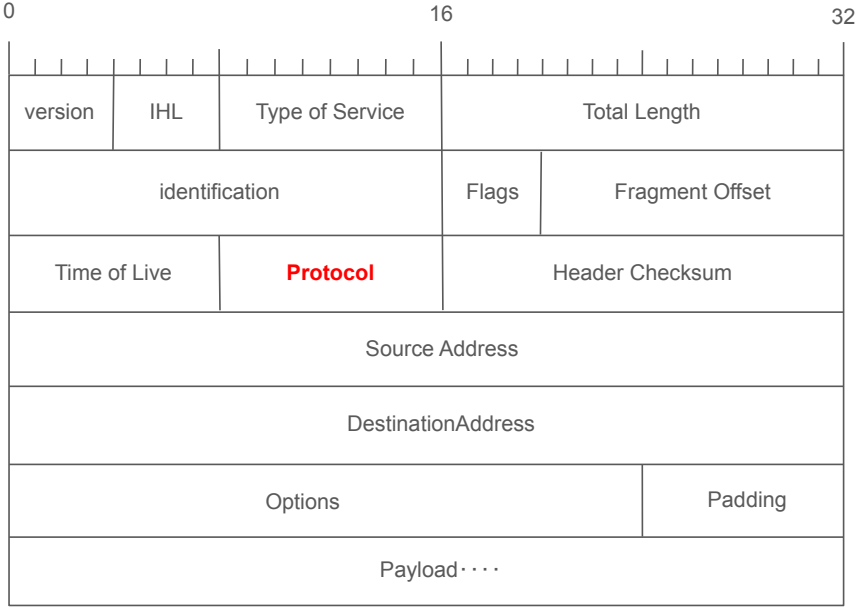
ポート番号の決まり方

- 使用するポート番号はおおまかに3つに分けられる
  - システムポート：0~1023（下記参照）
  - ユーザーポート：1024~49151（主にユーザーアプリケーション用）
  - ダイナミックポート：49152~65535（その他動的に割り当てられる）

- システムポートはアプリケーションごとにある程度決まっている
  - HTTP: 80
  - HTTPS: 443
  - DNS: 53
  - SSH: 22 …and more!

トランスポート層の代表的なプロトコル

- **送り方**
- TCP(Transmission Control Protocol)
- UDP(User Datagram Protocol)
- QUIC(Quick UDP Internet Connections)



IPヘッダのペイロードで指定されている

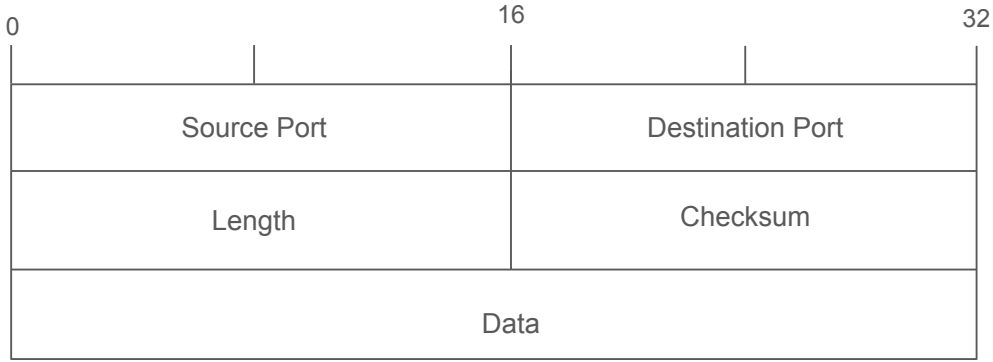
UDP

- コネクションレス（最初に通信を確立しない）
- シンプルな分、高速
- 再送制御なども行わない
- パケットの到着順序も考慮しない
- UDPを使う場合は、アプリケーション側で上記の考慮が必要

UDP

- Source Port（送信元ポート）16bit
  - 通信するアプリケーションが起動しているポート番号
- Destination Port（送信先ポート）16bit
  - 通信する先が起動している（はず）のポート番号

UDPのヘッダ



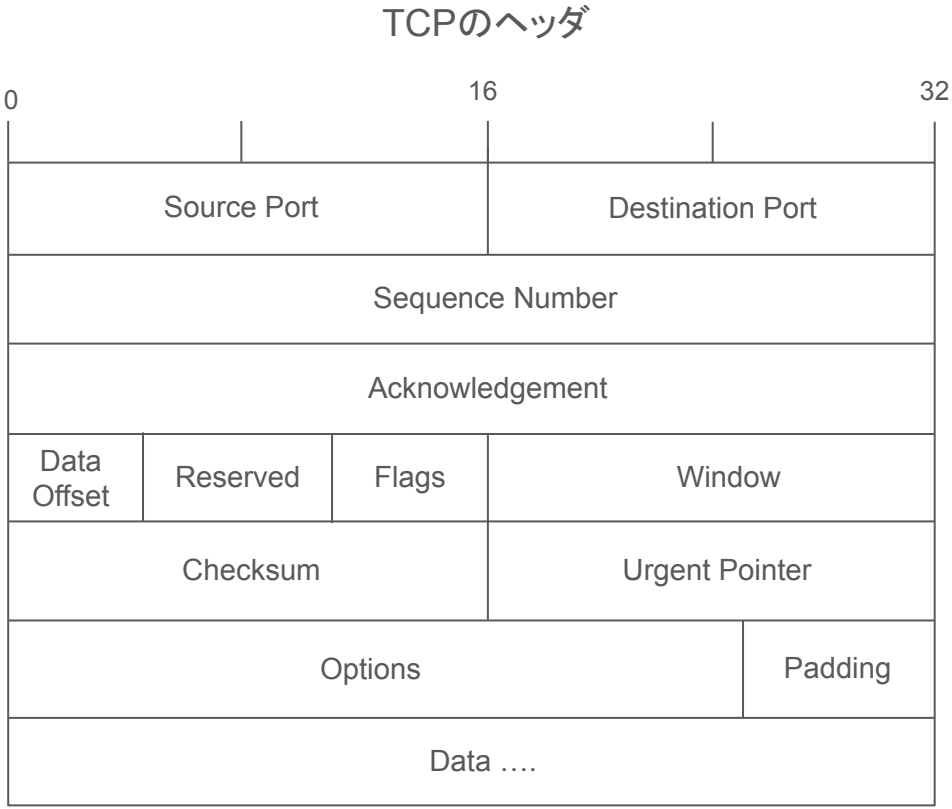
※概略図なので横幅のビット数は気にしないでください

TCP

- 信頼性◎
- 相手にデータが届いたこと確認し、もし届かなかった場合は再度送る
- 送った順番と到着する順番が入れ替わっても正しく認識して通信可能
- 数多くのプロトコルで下位層のプロトコルとして採用
    - Webブラウジング：HTTP(Hyper Text Transfer Protocol)
    - メールを転送：SMTP(Simple Mail Transfer Protocol)
    - など

TCPのヘッダ

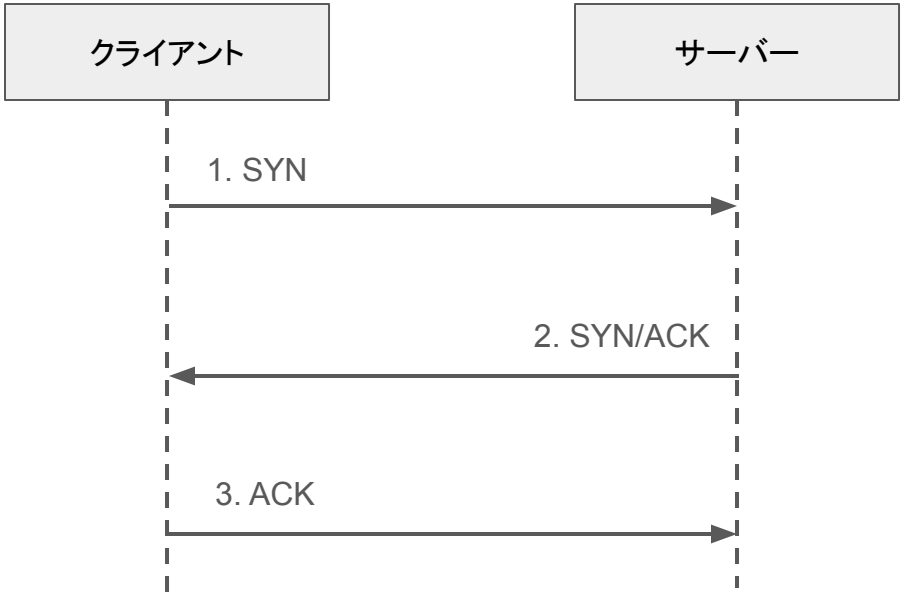
- 送信元・送信先ポートは UDPと同じ
- Sequence Number
  - シーケンス番号
  - データが到着する順番を管理するために使用
- Flags（コントロールフラグ）
  - ACK
    - 確認応答番号領域が有効であることを示す
  - SYN
    - 通信の初めだけ有効化する
    - これが有効化されていると受信側はコネクションの確立のための応答を返す
  - FIN
    - 通信の終わりを示す



※概略図なので横幅のビット数は気にしないでください

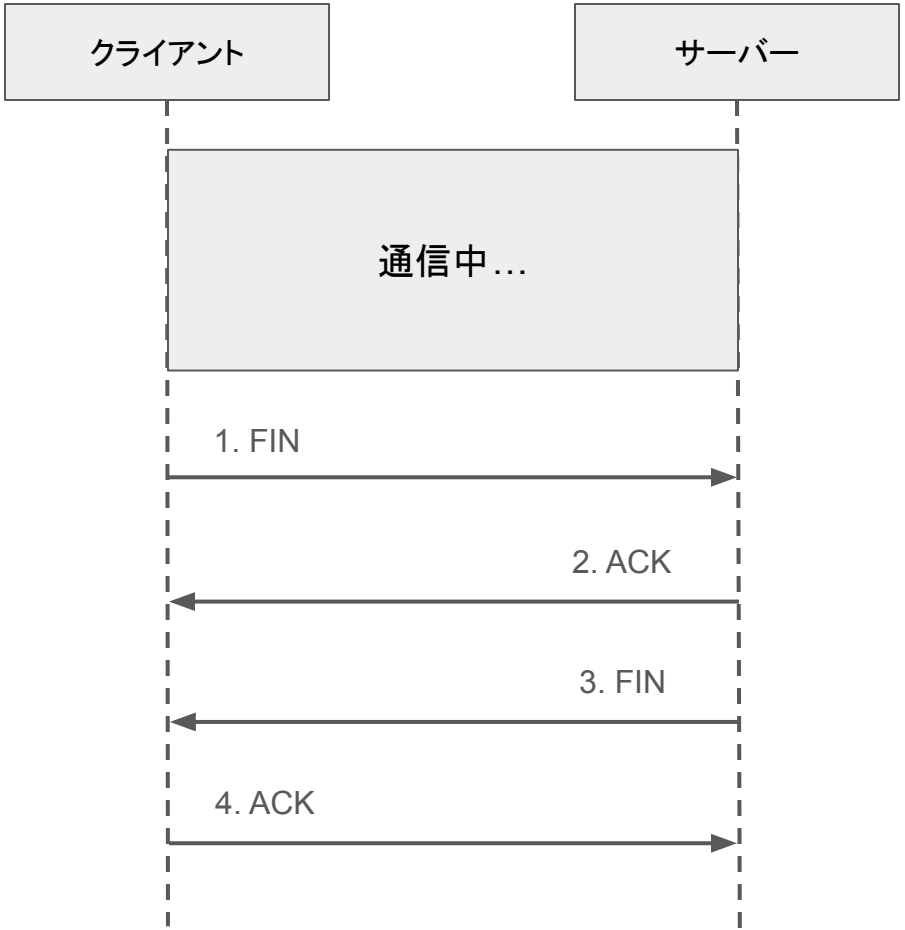
TCPのコネクション管理

- TCPでは通信をはじめるときに3つのフラグをやりとりする
  - この通信をスリーハンドシェイクと呼ぶ
- 目的:これからデータを送り合うことについて、お互いに確認する



1. クライアントからサーバーにコネクション確立の要求をする
2. サーバーはクライアントのコネクション要求に応える
   そしてサーバーからもコネクション確立を要求する
3. クライアントもサーバーからのコネクション要求に応える

TCPのコネクション管理



1. クライアントからサーバーにコネクション切断の要求をする
2. サーバーはクライアントからの切断要求に応える
3. そしてサーバーからもコネクション切断を要求する
4. クライアントもサーバーからの切断要求に応える

TCPとUDPのメリデメ

- ざっくり

	TCP	UDP
ヘッダ長	長い	短い
機能	多い	少ない
伝送速度	遅延・オーバーヘッド増	低オーバーヘッド低遅延

- TCPの特徴
  - 高信頼性でデータが届くまで再送される
  - オーバーヘッドや遅延が増えやすいが安定したデータ転送
    - 例：Webブラウジング、メール転送
- UDPの特徴
  - パケットロスによる低い信頼性
  - 高速なデータ転送
    - 例：リアルタイムのアプリケーションやゲーム、音声通話など

アプリケーション層

OSI参照モデル：理想的なあるべき姿として作られた階層構造のモデル

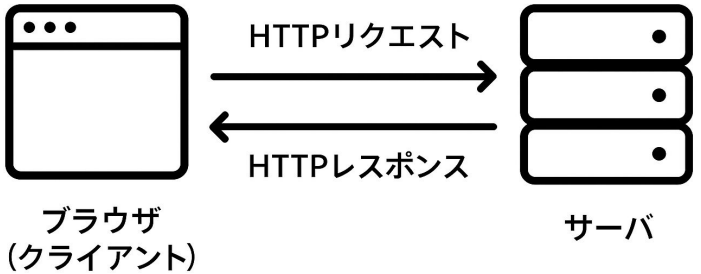
レイヤー	階層	概要	代表的なプロトコル
7層	アプリケーション層	個々のアプリケーション	HTTP,FTP,DNSなど
6層	プレゼンテーション層	データの表現形式	SMTP,FTP,Telnetなど
5層	セッション層	通信手段	TLS,NetBIOSなど
4層	トランスポート層	エンド間の通信	TCP,UDP ,NetWare/IPなど
3層	ネットワーク層	データを送る相手を決め、最適な経路で送信	IP,ARP,RARP,ICMPなど
2層	データリンク層	隣接する機器同士の通信を実現	PPP,Ethernetなど
1層	物理層	物理的な接続、電気信号	RS-232,UTP,無線

疑問

ブラウザでhttps://delishkitchen.tv/ と入力してエンターを押すと、何が起きるのか？

・サーバに問い合わせる！

・サーバからデータが返ってくる！



どうやってサーバを特定している？

DNSについて

人間用の”住所”と機械用の”座標”

- 郵便の世界でははがきに**住所**を書いて送るが、実際に届くのは友人の家の**座標**
- サーバの住所は「**ドメイン**」、そのサーバの座標は「**IPアドレス**」に値する
- 住所を座標に変換する仕組みが「**DNS（Domain Name Service）**」

東京都港区
六本木1-2-3 …..

ジオコーディング

北緯 35.98328942度
東緯 139.6923094度

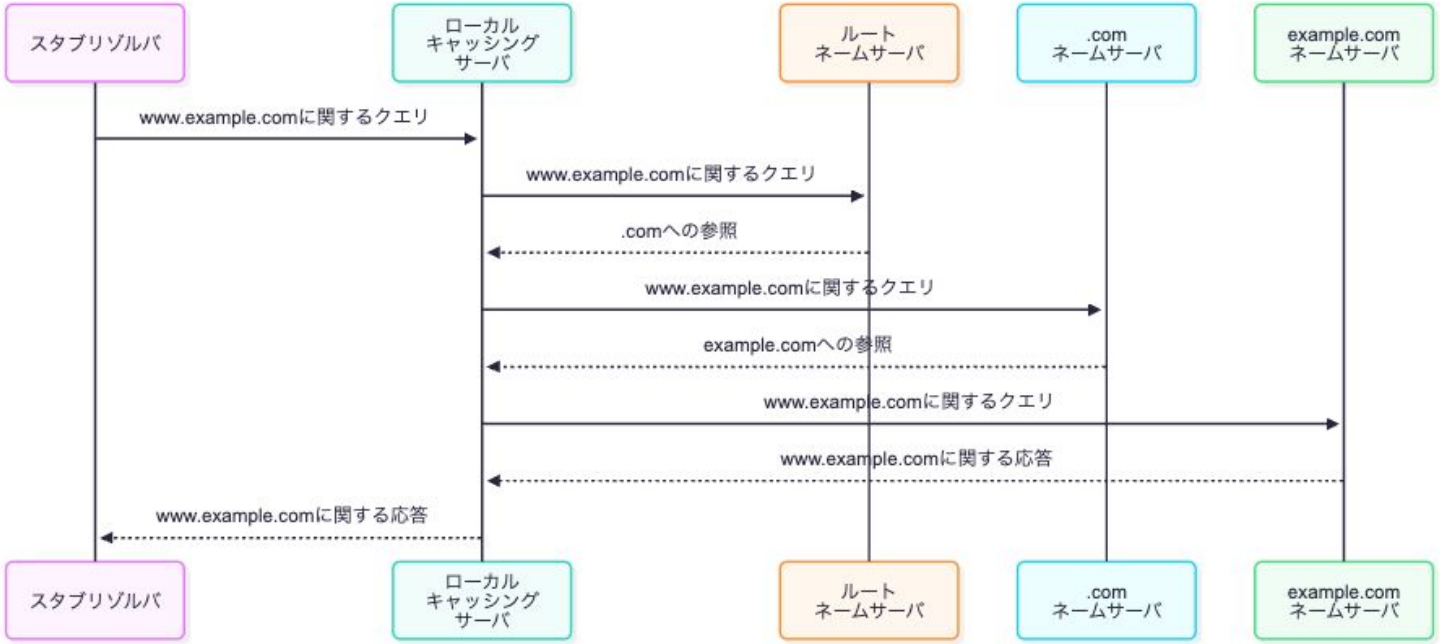
DNS

www.delishkitchen.tv

192.0.2.1

DNSの仕組み

- リゾルバというもの がネームサーバを順番に参照して　IPアドレスを特定していく



DNSレコードとは

- DNSレコードは、DNSサーバーに登録されるドメイン名に関する様々な情報
- 各レコードは「タイプ」「名前」「値」「TTL」(「クラス」)で構成される
- 用途に応じて適切なレコードタイプを使い分ける必要がある

レコードタイプ	役割	設定例	実際の用途
A	IPv4アドレスへの変換	example.com → 192.0.2.1	Webサイトのホスティング
AAAA	IPv6アドレスへの変換	example.com → 2001:db8::1	IPv6対応サイト
CNAME	別ドメインへのエイリアス	www.example.com → example.com	wwwありなしの統一
MX	メールサーバーの指定	example.com → mail.example.com	メール受信設定
NS	DNSサーバーの指定	example.com → ns1.example.com	ドメインの権威DNSサーバー

HTTPについて

HTTPとは

- HyperText Transfer Protocolの略
- Webブラウザとサーバー間の通信プロトコル
- リクエスト・レスポンス型の通信モデル
- ステートレス（状態を保持しない）

HTTPリクエスト/レスポンスの構造

HTTPリクエストの例

```
GET /api/users HTTP/1.1
Host: api.example.com
User-Agent: Mozilla/5.0
Accept: application/json
```

- メソッド + パス + バージョン
- ヘッダー情報
- ボディ（必要に応じて）

HTTPレスポンスの例

```
HTTP/1.1 200 OK
Content-Type:  application/json
Content-Length: 85
{"users": [...]}
```

- ステータス
- レスポンスヘッダー
- レスポンスボディ

HTTPメソッド

GET	取得
POST	作成
PUT	全体を更新
PATCH	一部を更新
DELETE	削除

ステータスコード

200	取得
400	リクエストが不正
401	認証が必要
403	アクセス禁止
404	リソースが見つからない
500	サーバー内部エラー
503	サービス利用不可

HTTPSについて

HTTPSとは

- HTTPで行う通信を暗号化する仕組み。**SSL/TLS**という暗号化技術を用いる
- 通信内容は暗号化され、正当な受信者（サーバー/クライアント）だけが復号できる
- 傍受されても暗号化されているため、第三者は内容を読み取れない

HTTPSが実現する3つのセキュリティ

1. **暗号化（Encryption）**
    - SSL/TLSプロトコルで通信内容を暗号化
2. **認証（Authentication）**
    - 認証局（CA）発行の証明書でサーバーの正当性を確認
    - フィッシングサイトや中間者攻撃を防ぐ
3. **完全性（Integrity）**
    - データが改ざんされていないことを保証
    - ハッシュ関数で通信内容の一貫性を検証

SSL/TLS証明書とは

ドメインの所有者が正しいことを第三者が証明する仕組み

- **問題**：通信相手のサーバーが本当に正規のサーバーか確認できない
- **リスク**：偽のサーバーに接続すると、情報が盗まれる可能性がある
- **解決策**：信頼できる第三者機関が「このドメインは正当な所有者のものです」と証明する

※OSやブラウザが信頼する第三者機関（ **認証局** ）は、PC内にある改ざんできない領域内のルート証明書ストアで管理されている

認証局（CA: Certificate Authority）の役割

- ドメイン所有者の身元を確認
- デジタル証明書を発行
- ブラウザはCAが発行した証明書を検証して、サーバーの正当性を確認

証明書の検証フロー

1. サーバーが証明書を提示
2. ブラウザが証明書の署名を CAの公開鍵で検証
3. 証明書の有効期限とドメイン名を確認
4. 検証成功で通信開始

AWSでインフラ構築してみよう

インフラとは？

- サーバー
- ネットワーク
- OS
- データベース

などなど、システムを動かすために必要な諸々のこと

- オンプレミス
  - 実際に機器を購入して運用する
  - 設置場所を社内でもどこでも自由に選べる、カスタマイズもできる
  - 一度購入すると簡単に変更・破棄したりできない
- IaaS (Infrastructure as a Service)
  - インフラを借りることができるクラウドサービス
  - サービス依存はあるが、インフラの変更、破棄が容易
  - AWS, Azure, GCPなど

AWSでインフラ構築してみよう

- ここまで、OSI参照モデルの代表的な層について学んできた
  - データリンク層: イーサネット、MACアドレス
  - ネットワーク層: IPアドレス、VPN、NAT
  - トランスポート層: TCP、UDP
  - アプリケーション層: HTTP、HTTPS, DNS

最後にまとめとして、これらを意識しながらAWSでWebサイトのためのインフラを構築してみる

VPC (Virtual Private Cloud)

- ネットワークセグメント、サブネットを作成できる
- パブリックサブネット
  - インターネットゲートウェイへの直接ルートがある
  - webサーバーを配置し、インターネットかアクセスを可能にする
- プライベートサブネット
  - インターネットゲートウェイへの直接ルートがない
  - データベースサーバーなどを配置する
- ルートテーブル
  - VPC内のルーティングを管理している

ルート (2)

両方 ▼ | ルートを編集

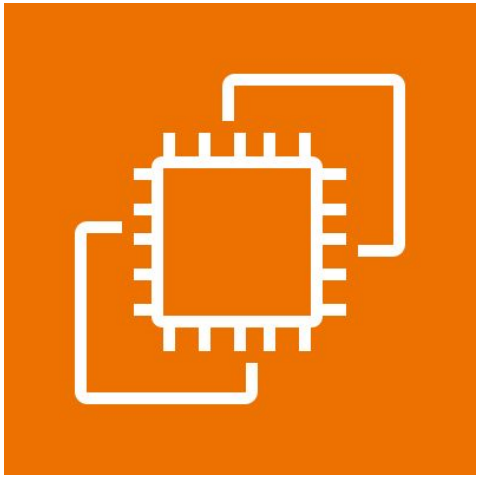
ルートをフィルタリング

1

送信先	ターゲット	ステータス	伝播済み	Route Origin
0.0.0.0/0	igw-07978b36a91651fa1	アクティブ	いいえ	Create Route
172.31.0.0/16	local	アクティブ	いいえ	Create Route Table

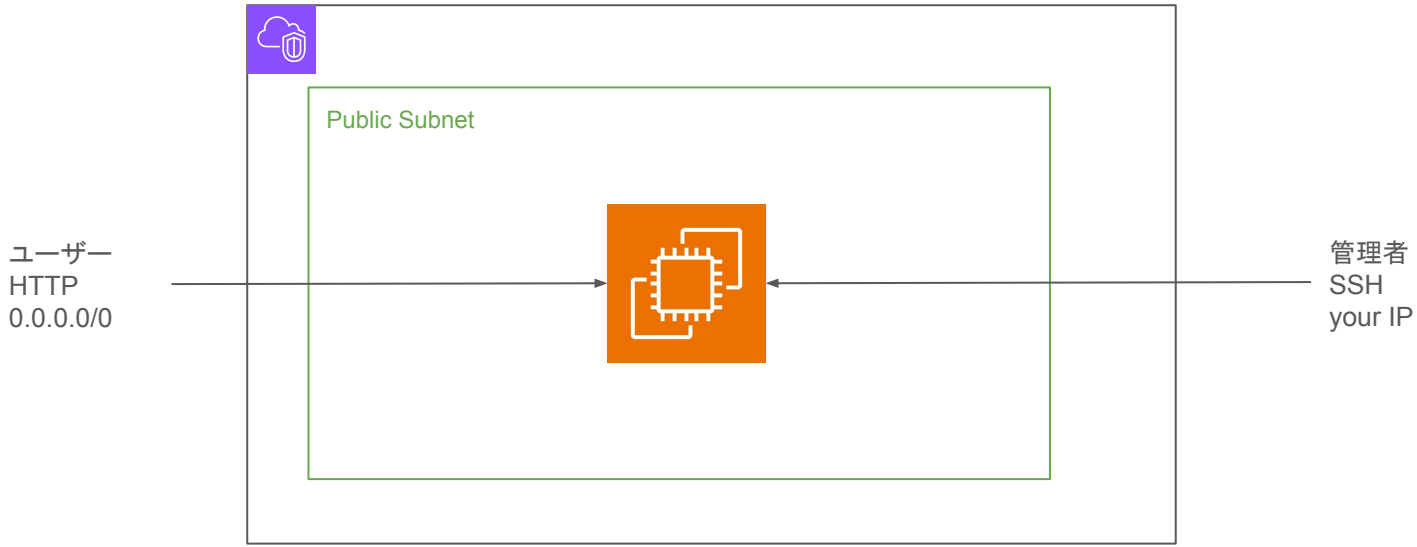
例：パブリックサブネットのルートテーブル

EC2 (Elastic Compute Cloud)



- OSやCPU、メモリの組を選んでサーバーを作成できる
- Elastic IP
  - 固定のIPアドレス
  - EC2インスタンスを停止、再起動しても変わらない
- セキュリティグループ
  - IP、ポート、プロトコルごとに通信の許可、制御を定義する
  - 例：自分のIPからのSSH接続と、任意のIPからのHTTP接続（つまりインターネット全体からのアクセス）だけ許可
  - ENIに紐づく
- Elastic Network Interface
  - 仮想的なNetwork Interface Card
  - MACアドレス、IP、セキュリティグループなど、ネットワークに関する情報を持っている

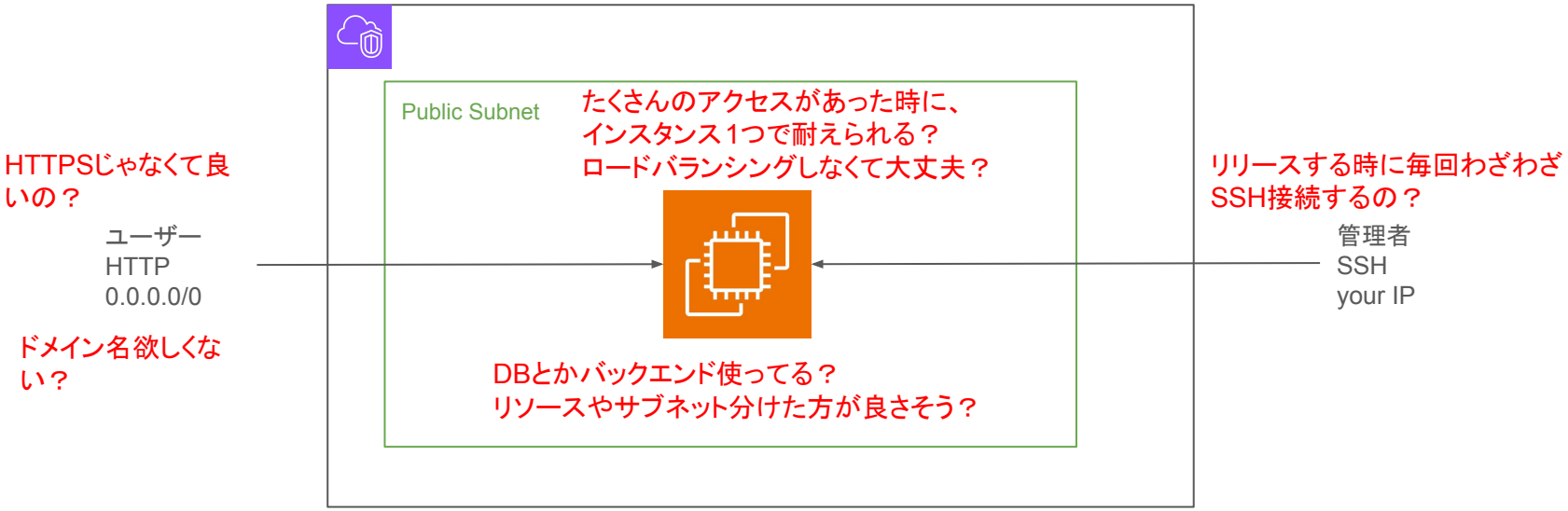
ミニマムにインフラ構成するならこれだけ！



でも、実際にWebサービスを運用していくときは、これだけだと困ることがたくさんある

この図で思いつく問題は何か？

想定される問題



他にもきっといろいろな問題が起きる

以上の問題に使えそうなAWSサービス

- Certificate Manager
  - SSL/TLS証明書を簡単に発行・管理できるサービス
  - CloudFrontやELBと連携
- Elastic Load Balancing
  - 複数のサーバーに負荷を分散するロードバランサー
  - Application Load Balancer(ALB) と Network Load Balancer(NLB) がある
- Route 53
  - マネージドDNSサービス
  - ドメイン名登録、DNSルーティング
- Elastic Container Service
  - フルマネージドなコンテナオーケストレーションサービス
  - Dockerコンテナを簡単にデプロイ・管理できる
  - Fargate（サーバーレスでコンテナ実行）と EC2の2種類がある
- CodeBuild
  - GitHubなどと連携してテスト、CI/CDを自動化
- …and more!




インフラの構築で考えること

- 拡張性：将来的なユーザー増加、機能追加にどの程度対応できるか？
- 堅牢性：障害や予期せぬ状況でもシステムが安定して動作し続けられるか？
- 可用性：なるべくシステムがダウンしない、したとしてもすぐに復旧できるか？
- 運用・保守性：日々の管理や障害対応、バージョンアップなどを容易に行えるか？
- セキュリティ：不正アクセスやデータの漏洩の防止
- 監視：システムの状態や稼働状況をリアルタイムで把握し、異常や障害を早期に検知するための仕組み
- コスト
- …etc

インフラ構築の際には、想定されるユーザー規模や、どの時間帯にアクセスが多くなるのか、アプリケーションで使っている技術（DB、コンテナ等）、障害が起きた時にどのように対処するのか等を考えた上で検討をしなければならない

参考

- Linuxで動かしながら学ぶTCP/IPネットワーク入門
  もみじあめ
- マスタリング TCP/IP 入門編 第6版
  井上 直也、村山 公保、竹下 隆史、荒井 透、苅田 幸雄
- 徹底攻略　ネットワークスペシャリスト教科書
  瀬戸美月
- OSI参照モデルとは？ TCP/IPとの違いを図解で解説 | ITコラム | アイティーエム株式会社

引用

- AWS シンプルアイコン - AWS アーキテクチャーセンター | AWS

以下の状況で何が問題か、どう解決するか回答してください

Webサイトにアクセスできないが、IPアドレス直打ちなら接続できる

回答形式

- 考えられる原因（2つ以上）
- 確認すべきこと
- 解決方法


everu