技術研修第9回
通信周り

本日のテーマ「通信」

・通信とは？

異なる端末間での情報のやりとり

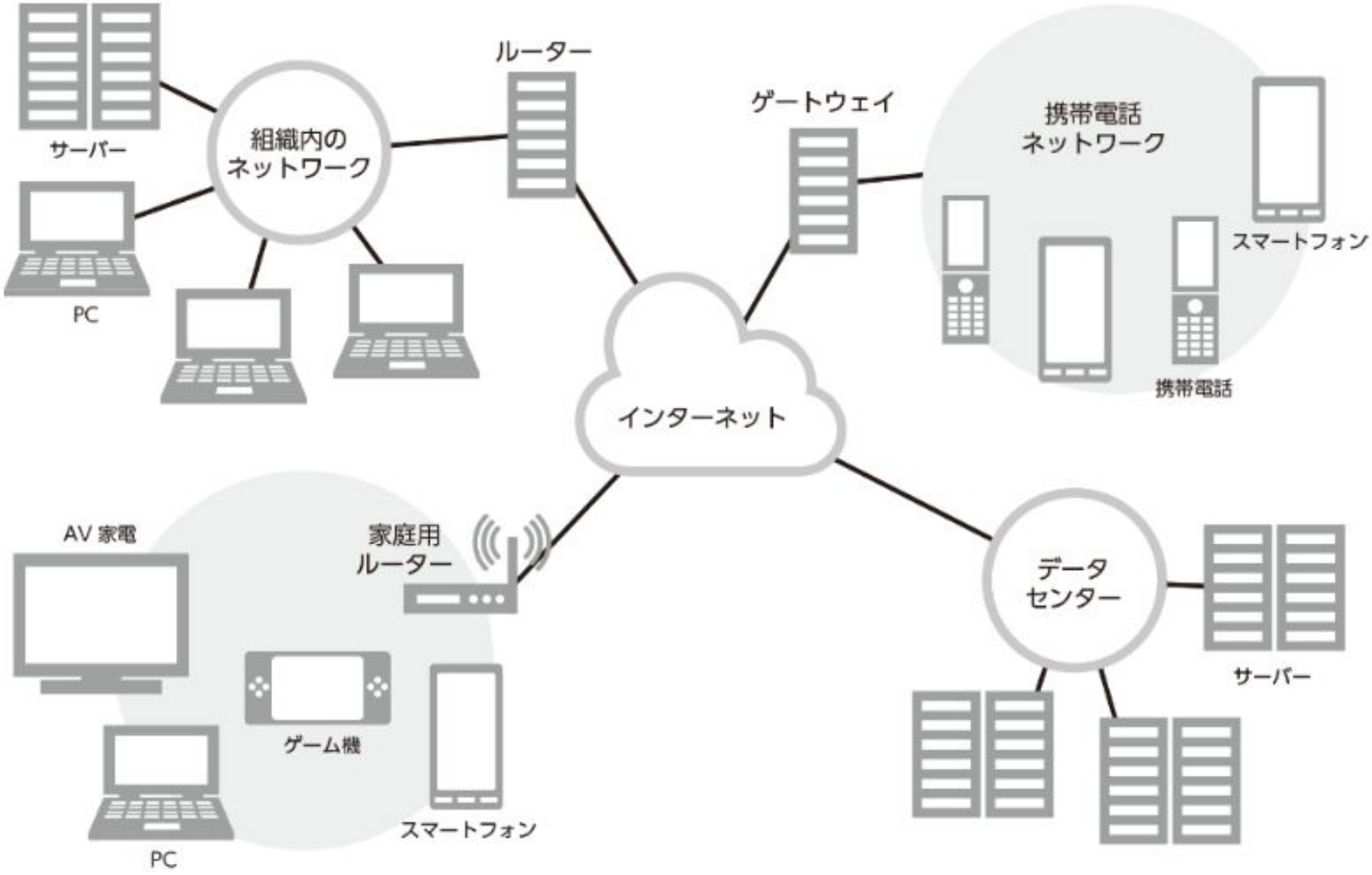
・ゴール

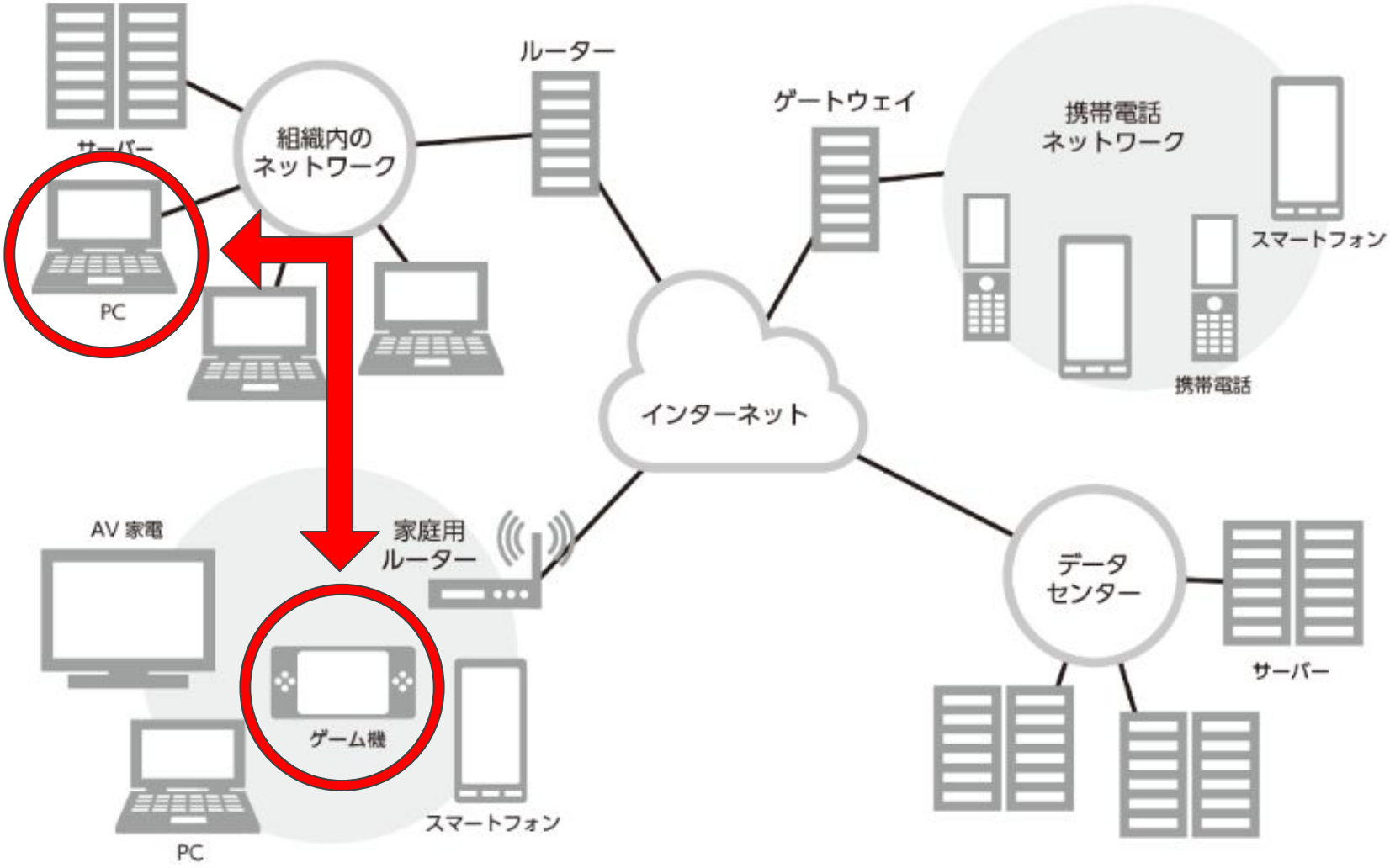
「通信技術の基礎を知り、実際の開発に当てはめて考えることができる」

前半：通信のベースとなる技術

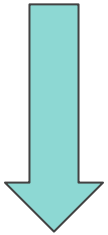
後半：開発視点での通信のお話

前半：通信のベースとなる技術


サーバー
組織内の
ネットワーク
ルーター
ゲートウェイ
携帯電話
ネットワーク
スマートフォン
PC
携帯電話
インターネット
AV 家電
家庭用
ルーター
データ
センター
サーバー
ゲーム機
スマートフォン
PC


サーバー
組織内の
ネットワーク
ルーター
ゲートウェイ
携帯電話
ネットワーク
スマートフォン
携帯電話
PC
インターネット
AV 家電
家庭用
ルーター
データ
センター
サーバー
ゲーム機
スマートフォン
PC

別の機器との間で、どのように通信する？

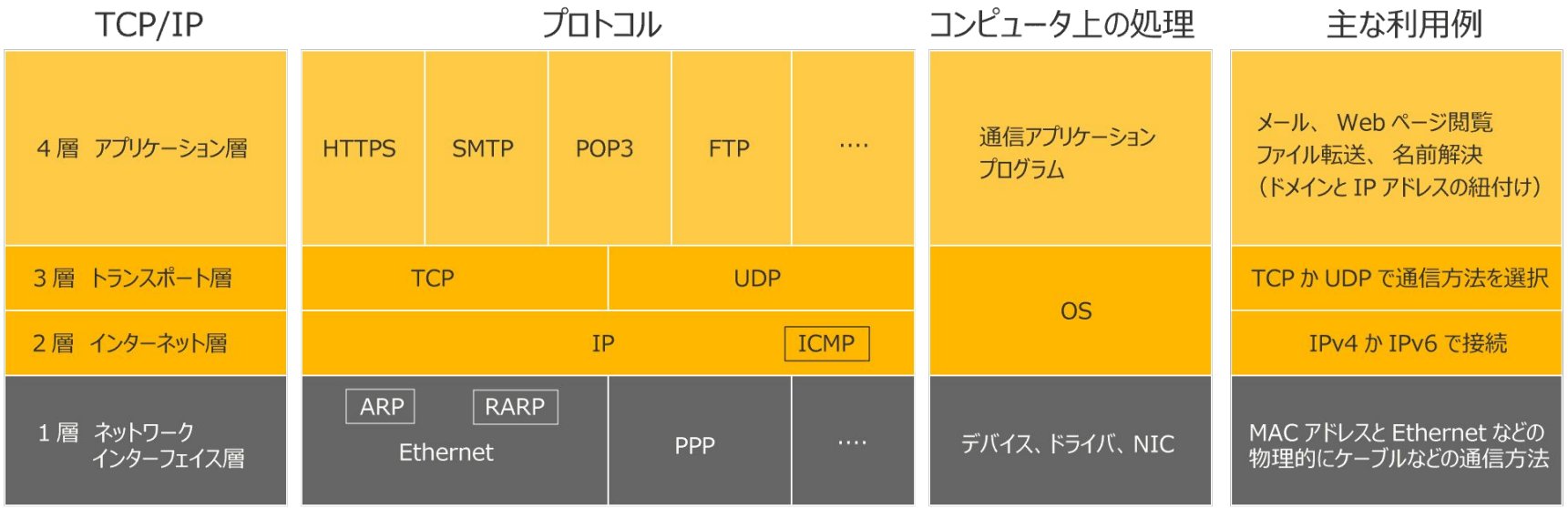


全機器共通でのルールが必要

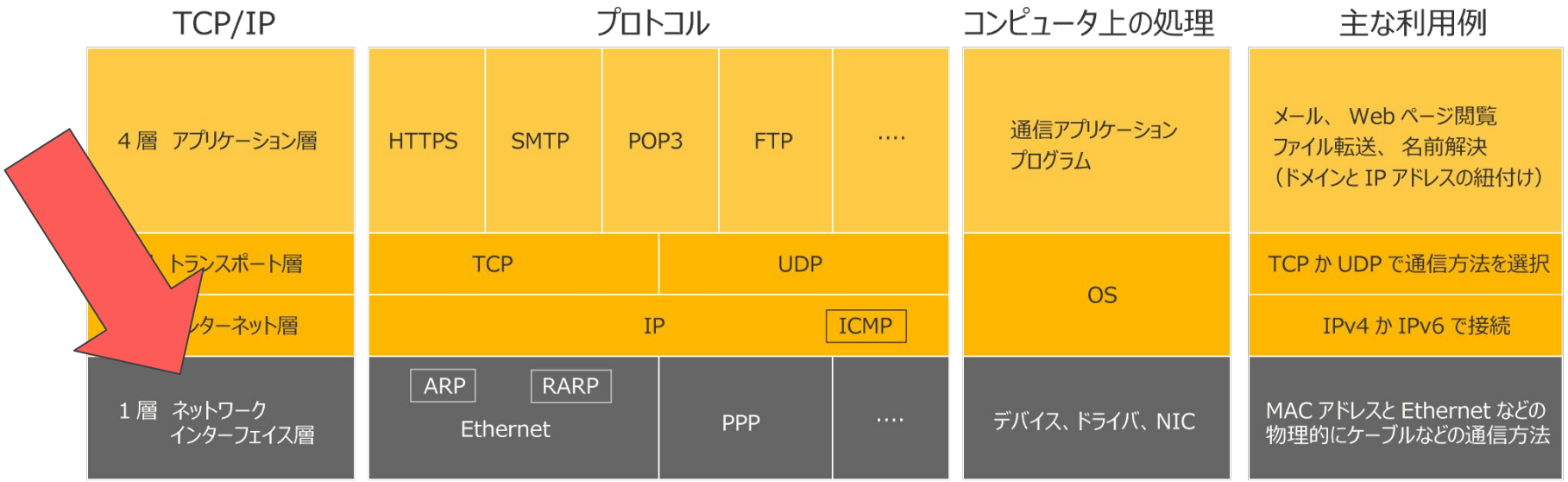
OSI参照モデル

層	名称	主な役割	領域	関連装置
7 層 (L7)	アプリケーション層	アプリケーション間でのやり取り	ソフトウェア	ゲートウェイ
6 層 (L6)	プレゼンテーション層	データの表現形式を定義		
5 層 (L5)	セッション層	接続の手順		
4 層 (L4)	トランスポート層	データ通信の制御	ネットワーク	
3 層 (L3)	ネットワーク層	インターネットでの通信		ルータ、L3 スイッチ
2 層 (L2)	データリンク層	同一ネットワークでの通信		ブリッジ、L2 スイッチ
1 層 (L1)	物理層	ケーブルや電気信号やコネクタなど	ハードウェア	LAN ケーブル、NIC

TCP/IPモデル


TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
4 層 アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
3 層 トランスポート層
TCP
UDP
OS
TCP か UDP で通信方法を選択
2 層 インターネット層
IP
ICMP
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

TCP/IPモデル


TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
4 層 アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
トランスポート層
TCP
UDP
OS
TCP か UDP で通信方法を選択
ンターネット層
IP
ICMP
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

1層 ネットワークインターフェース層

隣接した機器同士の通信を実現するための層

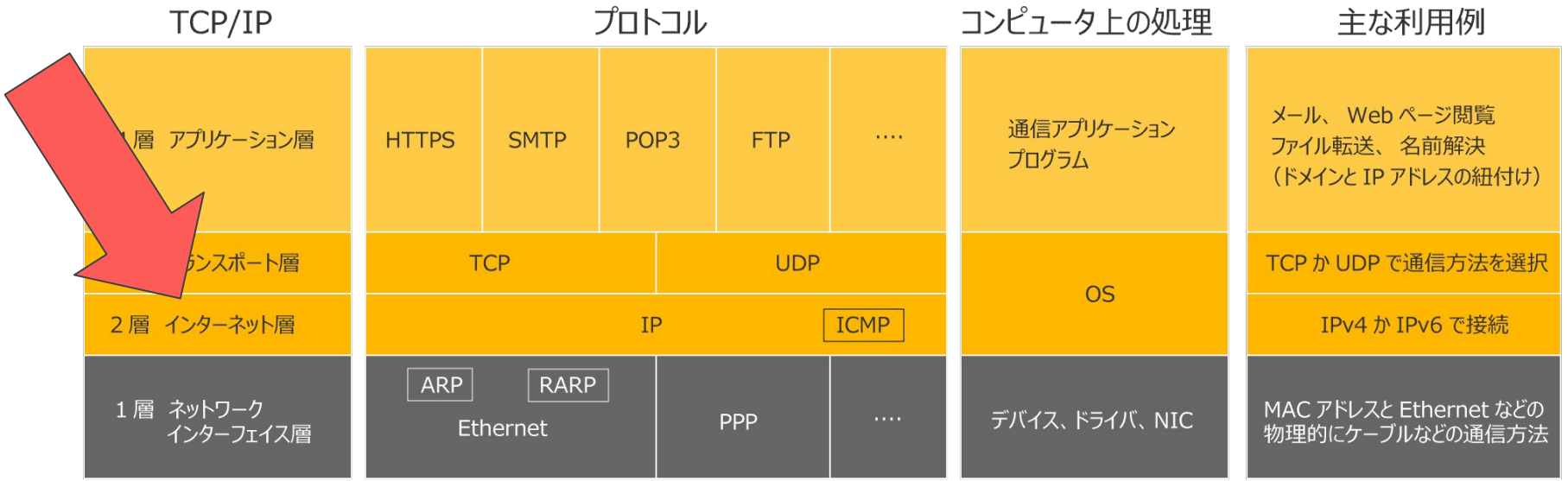
キーワード

「MACアドレス」

「イーサネット」

「LANケーブル」

TCP/IPモデル


TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
層 アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
ンスポート層
TCP
UDP
OS
TCP か UDP で通信方法を選択
2 層 インターネット層
IP
ICMP
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

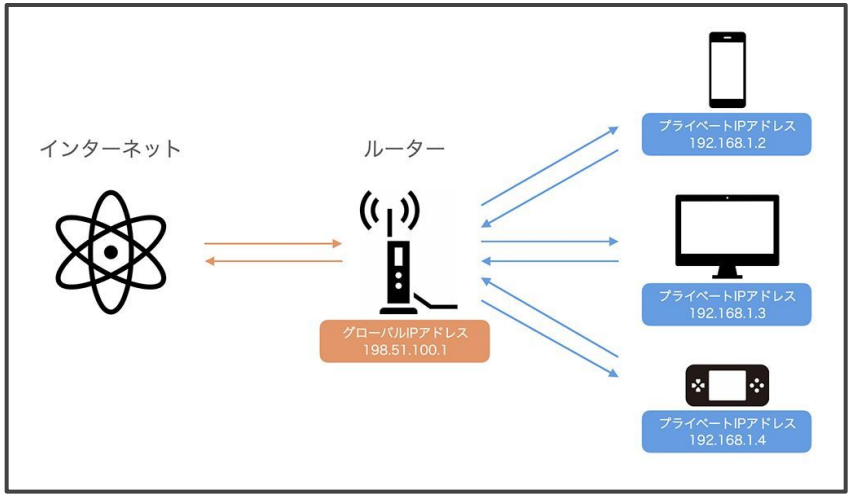
2層 インターネット層

どの端末が通信するかを識別し、送り届けるための層

キーワード

「IPアドレス」

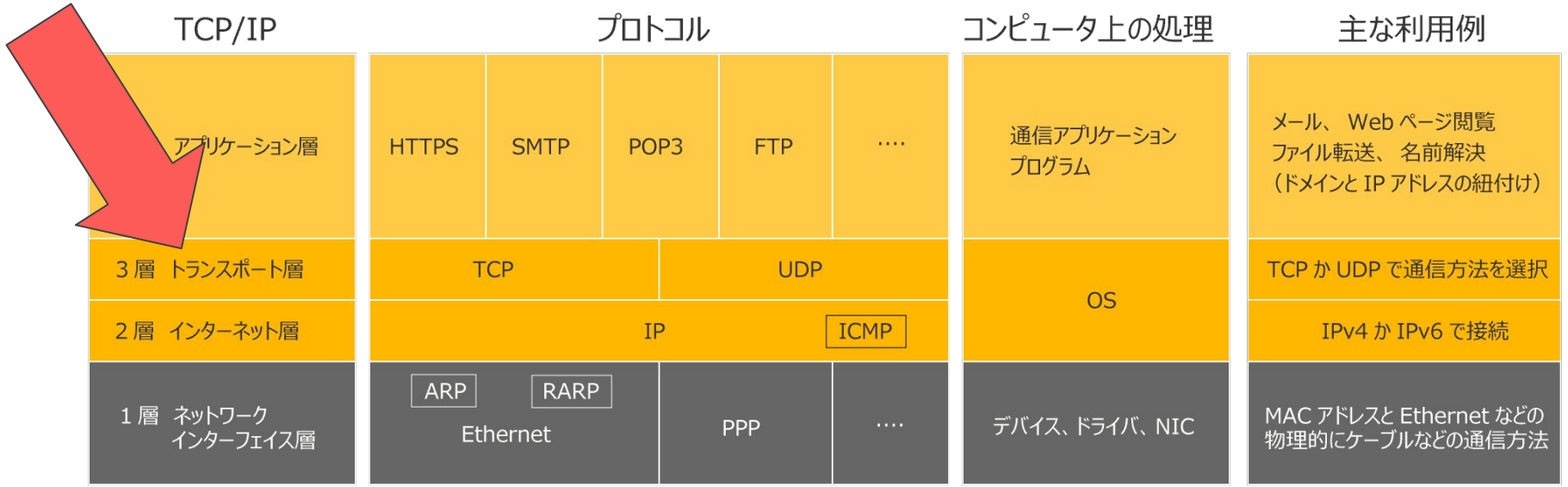
「グローバル / プライベート」



IPアドレス


IPv4アドレス
8ビット×4＝32ビット
192 . 168 . 0 . 1
0～255
256通りの4乗
＝
およそ43億通り
IPv6アドレス
16ビット×8＝128ビット
2001 : 0db8 : 0000 : 0000 : 1234 : 0000 : 0000 : 0abc
0000～ffff
6万5536通りの8乗＝3.4×10の38乗
＝
およそ340澗通り

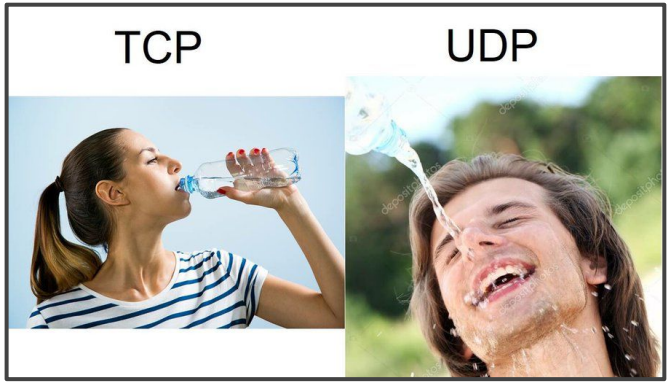
TCP/IPモデル


TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
3 層 トランスポート層
TCP
UDP
OS
TCP か UDP で通信方法を選択
2 層 インターネット層
IP
ICMP
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

3層 トランスポート層

データを適切なアプリケーションに振り分ける層

（データ転送の信頼性を保証する層）



キーワード

「TCP, UDP」

「パケットロス」

「3Way ハンドシェイク」

TCP通信

高信頼性

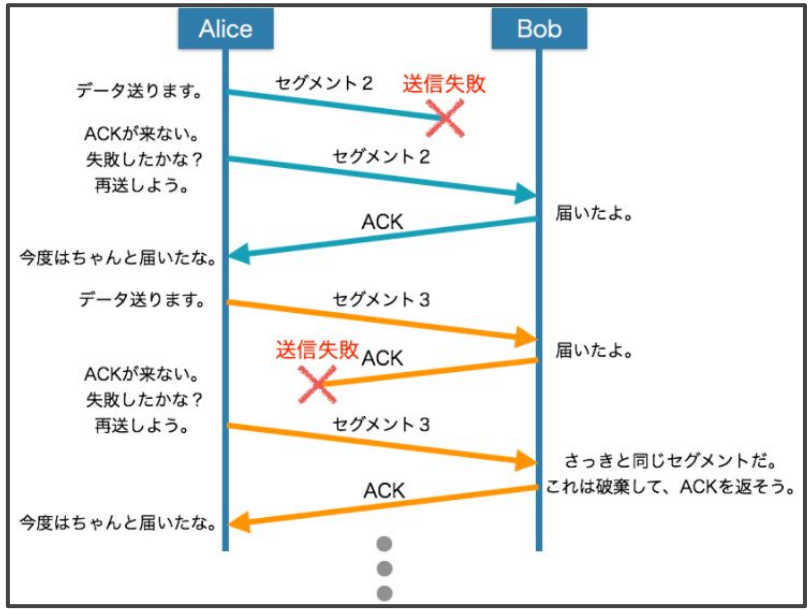
・3 way ハンドシェイクによる通信前の打診

・ack による相手の受信確認や再送処理

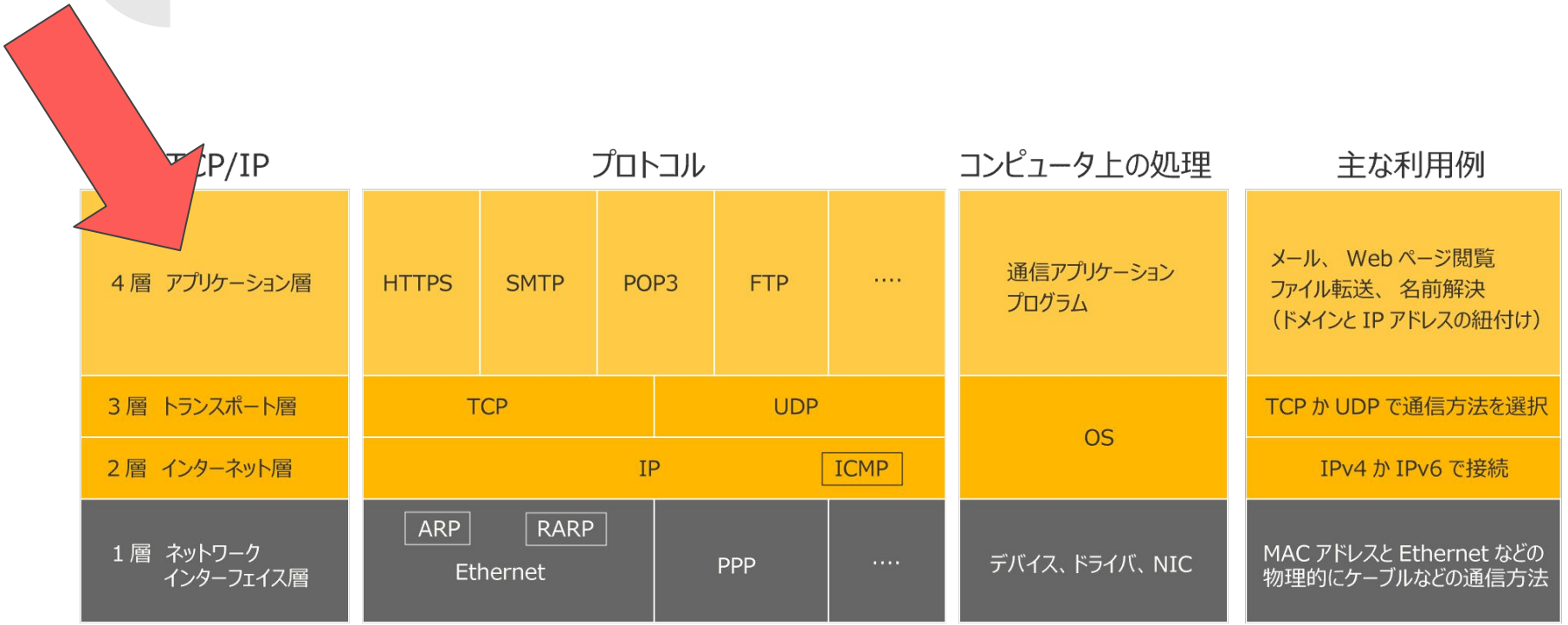
通信効率の最適化機能

・Windowによるフロー制御や輻輳制御

引き換えに、負荷は高い



TCP/IPモデル

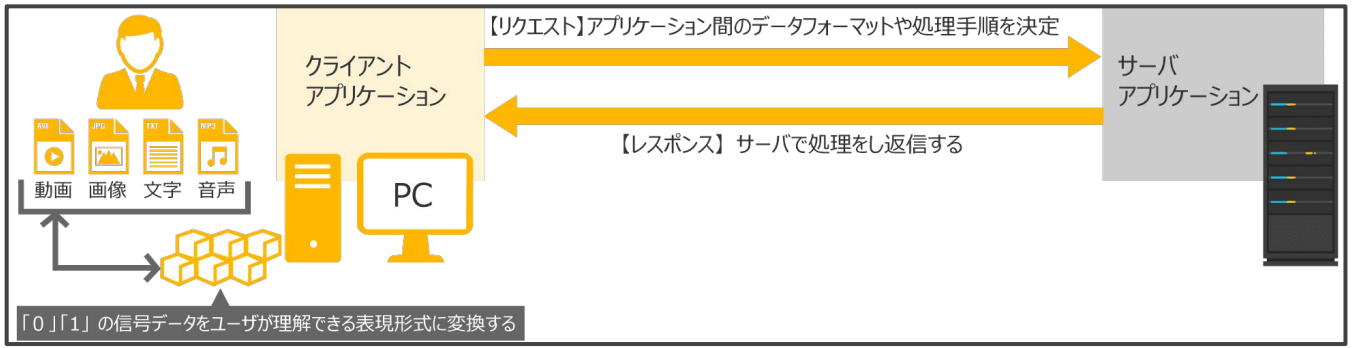

TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
4 層 アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
3 層 トランスポート層
TCP
UDP
OS
TCP か UDP で通信方法を選択
2 層 インターネット層
IP
ICMP
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

4層 アプリケーション層

アプリケーションで扱うデータのフォーマットや手順を決める層

キーワード

「通信プロトコル」



通信プロトコルのバリエーション

HTTP　：Webページの送受信

SMTP　：電子メールの送信

POP3　：電子メールの受信

DHCP　：IPアドレス情報の割り当て

DNS　：ドメインをIPアドレスへ変換する

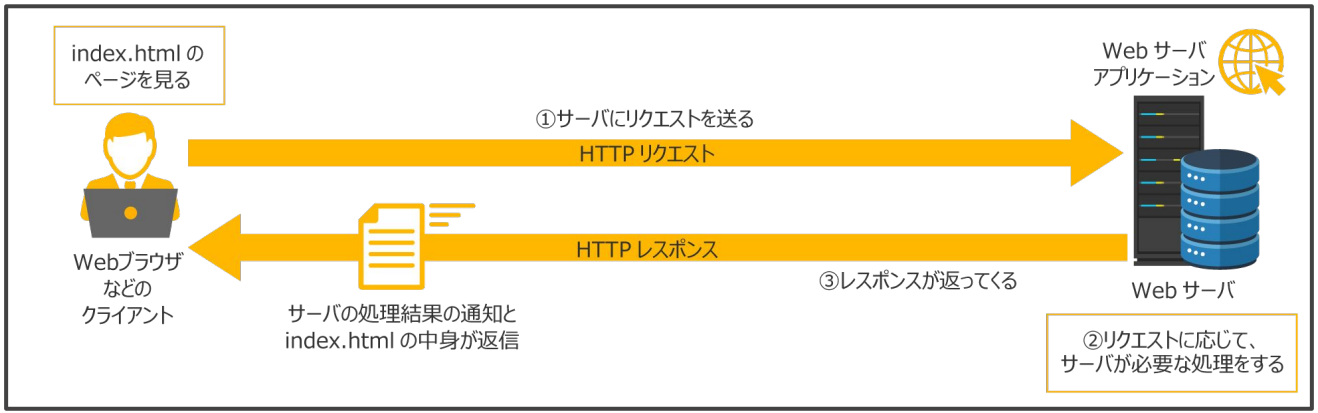
SSH　：機器の遠隔操作　など

HTTP(Hypertext Transfer Protocol)

WebサーバとWebブラウザの間で、Web情報をやりとりするためのプロトコル

＞ 普段、ホームページで情報収集したり、ブログを読んだりする時、このHTTPを使ってサーバとクライアント（ユーザ）間でやり取りが行われます

1つの要求（リクエスト）には**1つの応答**（レスポンス）を返すルール



HTTPメソッド	冪等	安全	内容
GET	○	○	特定のリソースの表現をリクエストします。データを受け取るだけです。
HEAD	○	○	指定されたリソースをGET メソッドで返されるヘッダーをリクエストします。
POST	×	×	サーバーにデータを送信します
PUT	○	×	新しいリソースを作成するか、 指定したリソースの表現をリクエストのペイロードで置き換えます。
DELETE	○	×	特定のリソースを削除します。
CONNECT	×	×	双方向のコミュニケーションを開始します。 これはトンネリングを開始するときに使用されます。
OPTIONS	○	○	対象リソースの通信オプションを記述するために使用します。
TRACE	○	×	対象リソースまでのパスに沿ってメッセージのループバックテストを行い、 便利なデバッグの仕組みを提供します。
PATCH	×	×	リソースへの部分的な変更を適用します。

HTTPの要素

ヘッダ：Webコンテンツの伝送に用いられるHTTPで、メッセージの前半にある制御情報を記した領域

ステータスコード：リクエストの結果を大別したもの

コード	分類	意味
1xx	情報系	リクエストを受信して、その処理を継続中
2xx	成功系	リクエストの処理に成功
3xx	リダイレクト系	リクエストを終えるには、さらに動作が必要
4xx	クライアントエラー系	リクエストの構文に問題がある。または実行できない
5xx	サーバーエラー系	リクエストの構文は正常だが、サーバーが実行に失敗

【TIPS】HTTPの歴史

1991年：HTTP/0.9（HTTPの始まり。GETメソッドしかなかった）

1996年：HTTP/1.0（GET以外のメソッドが増えヘッダーやボディ、ステータスコードが誕生！セッションが生まれた）

1997年：HTTP/1.1（通信効率化、Keep-Aliveの誕生。暗号化やバーチャルホスト、chunk送信などが爆誕）

2015年：HTTP/2（元祖SPDYプロトコルを元にストリームが誕生）

202X年：HTTP/3（UDPベース、パケットロス時の効率化）

WireSharkでみてみようのコーナー

WireSharkとは…

・LAN上に流れているパケットを「見える化」するパケットキャプチャツール

・オープンソースのツール

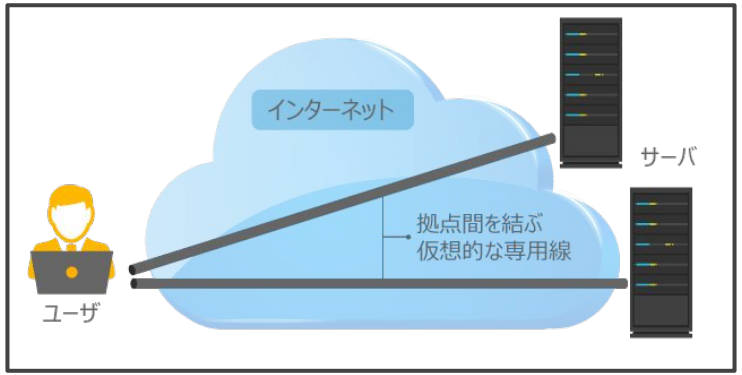

実際にやってみた

VPN（Virtual Private Network）

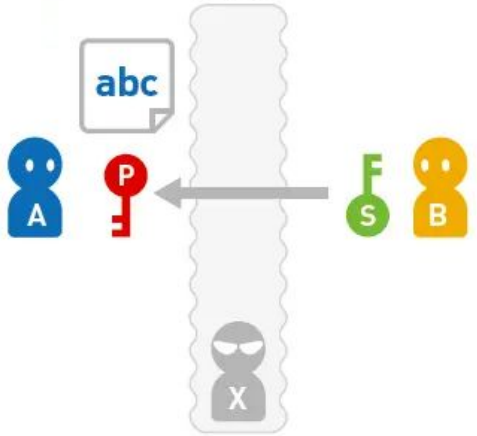
ネットワークの中に、別のプライベートなネットワークを**仮想的に**作る技術

トンネリングにより盗聴・改ざんから保護

→**セキュリティ上安全な経路を用いて通信が可能**



暗号鍵



共通鍵暗号方式（AES）

送信者が暗号鍵を作成し、暗号化・復号を行う

暗号化の処理速度は早い が、鍵の数が増えやすいかつ第三者に盗まれる危険性

公開鍵暗号方式（RSA）

受信者が暗号鍵を2つ作成する（**公開鍵・秘密鍵**）

このうち公開鍵を用いて送信者は暗号化をし、受信者は秘密鍵を用いて複合

他者に通信を覗かれにくい

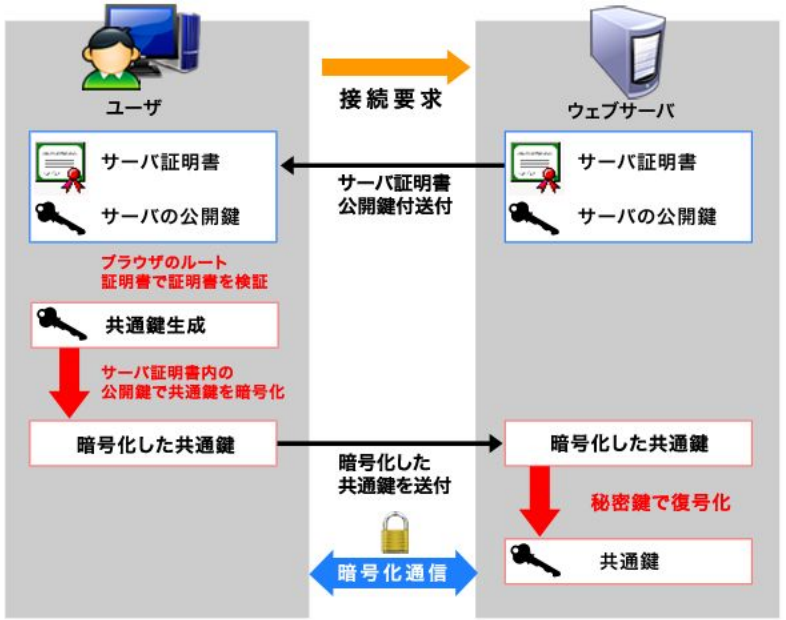
HTTPS

HTTP + SSL/TLS

通信が覗き見られないように、
セキュリティ性が高い

「公開鍵暗号方式」と「秘密鍵暗号方式」の

ハイブリッドな方式



前半戦ここまで

後半：開発の視点での通信

ここまで説明して...

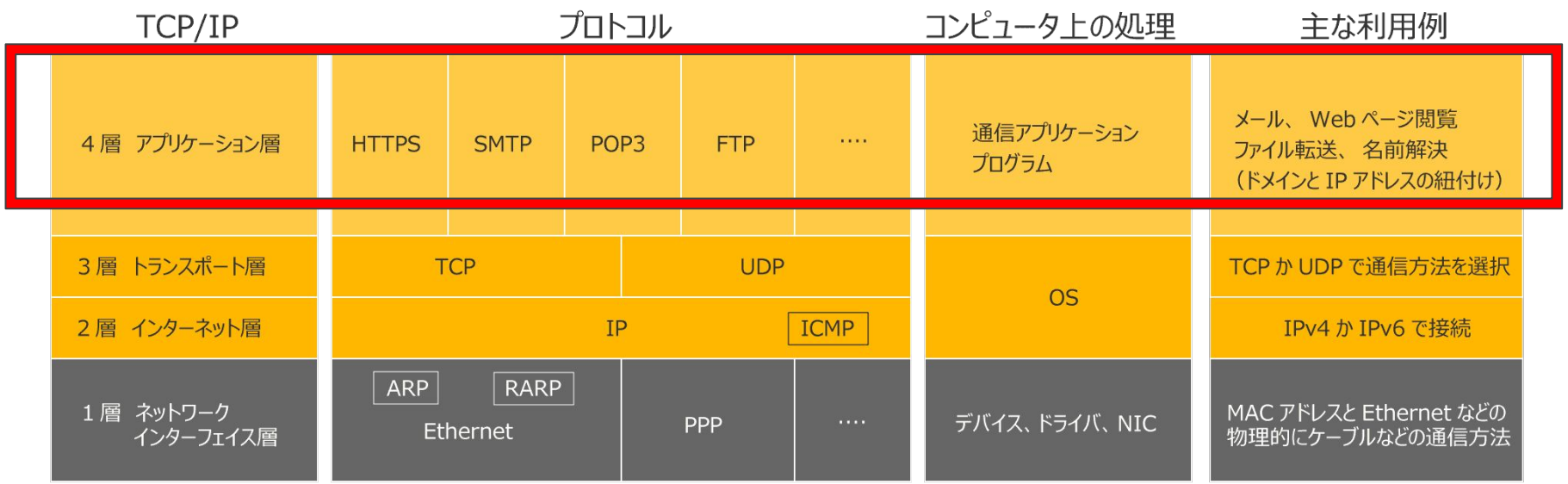
開発時にほとんど意識しない、聞いたことがない

→**共通ルールとして無意識に利用**

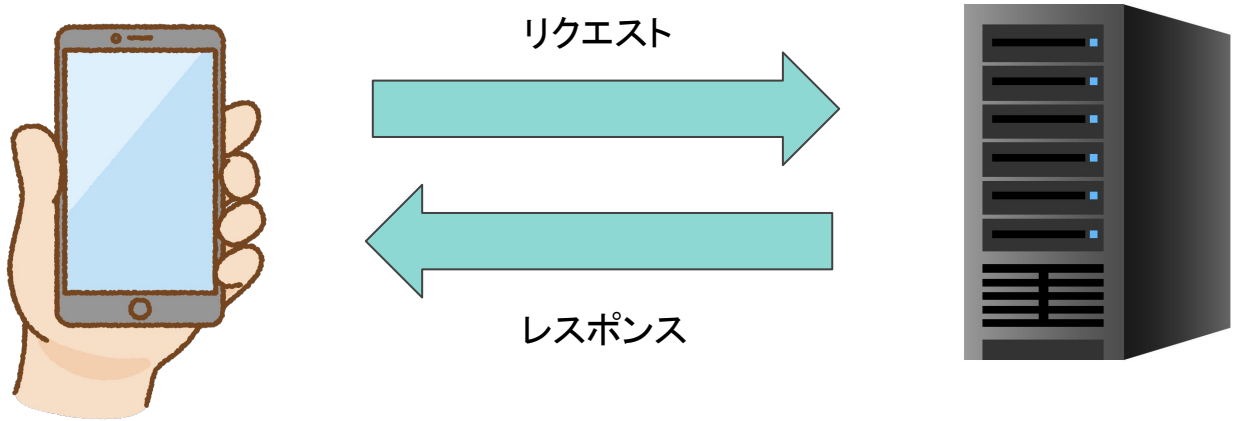
例えば...）

アプリケーションの裏側で行っているのは、大体HTTP通信

ここからの話


TCP/IP
プロトコル
コンピュータ上の処理
主な利用例
4 層 アプリケーション層
HTTPS
SMTP
POP3
FTP
....
通信アプリケーション
プログラム
メール、 Web ページ閲覧
ファイル転送、 名前解決
（ドメインと IP アドレスの紐付け）
3 層 トランスポート層
TCP
UDP
TCP か UDP で通信方法を選択
2 層 インターネット層
IP
ICMP
OS
IPv4 か IPv6 で接続
1 層 ネットワーク
インターフェイス層
ARP
RARP
Ethernet
PPP
....
デバイス、ドライバ、NIC
MAC アドレスと Ethernet などの
物理的にケーブルなどの通信方法

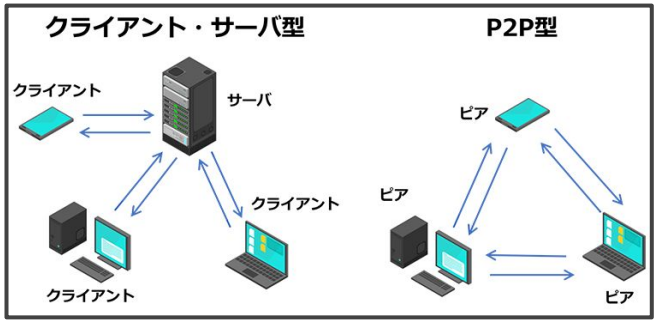
サーバ・クライアント通信



サーバ：特定のポートを開放し、接続待ち状態として待機

クライアント：IPアドレスとポートを指定し接続要求

他の通信モデル



P2P方式（Peer to Peer）

サーバを経由せず、ピア同士がデータを送受信する通信方式

メリット

・直接接続なので、**通信速度が速い**

・匿名性が高く、標的型攻撃を受けづらい

・ゼロダウンタイム

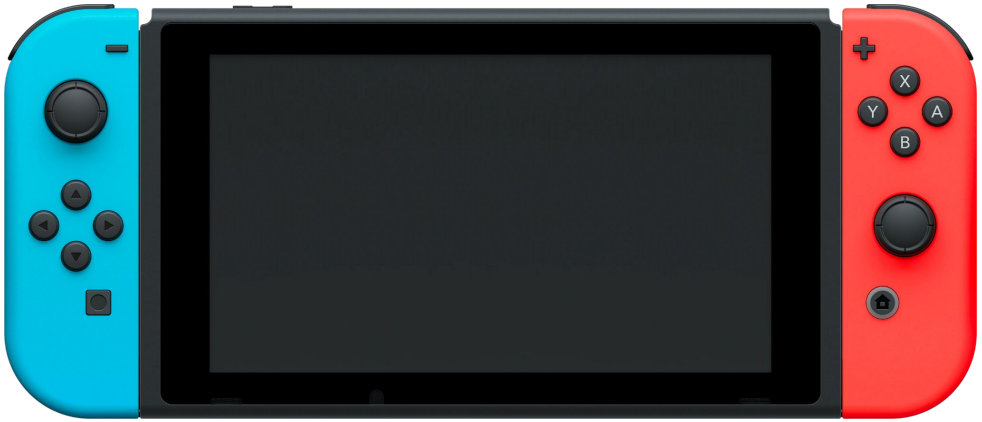
・構築が安価

デメリット

・セキュリティ面での問題（**データ送信元の安全性を確認できない**）

・ネットワーク帯域の圧迫

他の通信モデル



Switchのリアルタイム通信にも利用されている

（スプラトゥーン，ポケモンユナイトなど）

・ネットワーク帯域の圧迫

通信方式

各アプリケーションは、ルールに基づき通信を行っている

さまざまなモデルがある

例）REST, gRPC, SOAP ……

現状、最も普及しているのは**REST方式**

REST方式

URI	HTTP Verb	Outcome
.../api/employees	GET	Gets list of employees
.../api/employees/1	GET	Gets employee with Id = 1
.../api/employees	POST	Creates a new employee
.../api/employees/1	PUT	Updates employee with Id = 1
.../api/employees/1	DELETE	Deletes employee with Id = 1

REST(**RE**presentational **S**tate **T**ransfer)

Webサービスの設計モデルであり、リソース中心な考え方

1. **アドレス指定可能なURI**で公開されていること
2. インターフェースの統一がされていること(HTTPメソッドの利用)
3. **ステートレス**であること
4. 処理結果が**HTTPステータスコード**で通知されること

バイナリ形式：JSON

JSON（JavaScript Object Notation）

軽量なテキストベースのデータ交換用フォーマットであり、言語を問わず利用可能

```
[
 {
  "メニュー": "さばのみそ煮",
  "値段": 750,
  "カロリー": 858
 },
 {
  "メニュー": "豚肉のしょうが焼き",
  "値段": 800,
  "カロリー": 1024
 },
 {
```

gRPC

Googleが開発を開始した、オープンソースのリモートプロシージャコール　(RPC) システム

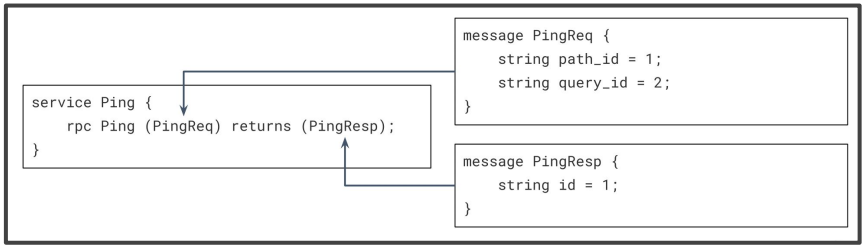
- HTTP2を利用
- Protobufをベースに記述
- サードパーティツールに依存しないコード生成
- RESTより早い通信、遅い実装

バイナリ形式：Protocol Buffer

インタフェース定義言語(IDL)で構造を定義する通信や永続化での利用を目的としたシリアライズフォーマットであり、Googleにより開発されている。

静的な構造化データを対象とする
シリアライズを行うためのデータは、事前にその構造を定義しておく必要がある。構造の定義には、専用の書式とデータ型を用いる

データアクセスコードを自動生成
Protocにより、様々な言語にコンパイル可能



バイナリ形式：Message Pack

効率の良いバイナリ形式のオブジェクト·シリアライズフォーマット。JSONの置き換えとして使うことができ、様々なプログラミング言語をまたいでデータを交換することが可能です。

JSONと比べると、保存した状態の可読性を犠牲にする代わりに、より早くて小さいフォーマットになっています。(Master Memory内部でも使用)

MagicOnionではgRPC + MessagePackでリアルタイム通信が実装されている。

Unity & サーバー間通信でのMessagePack導入奮闘記(Unity編)https://blog.applibot.co.jp/2016/05/09/messagepack-unity/

まとめ

ぜひ皆さんのプロジェクトの通信周りも見てみてください！

何か質問があればぜひ聞いてください