

プログラミング応用

<http://bit.ly/ouyou3d>

コンピュータネットワーク（1）

後期 第14週

2019/1/17

本日は・・・

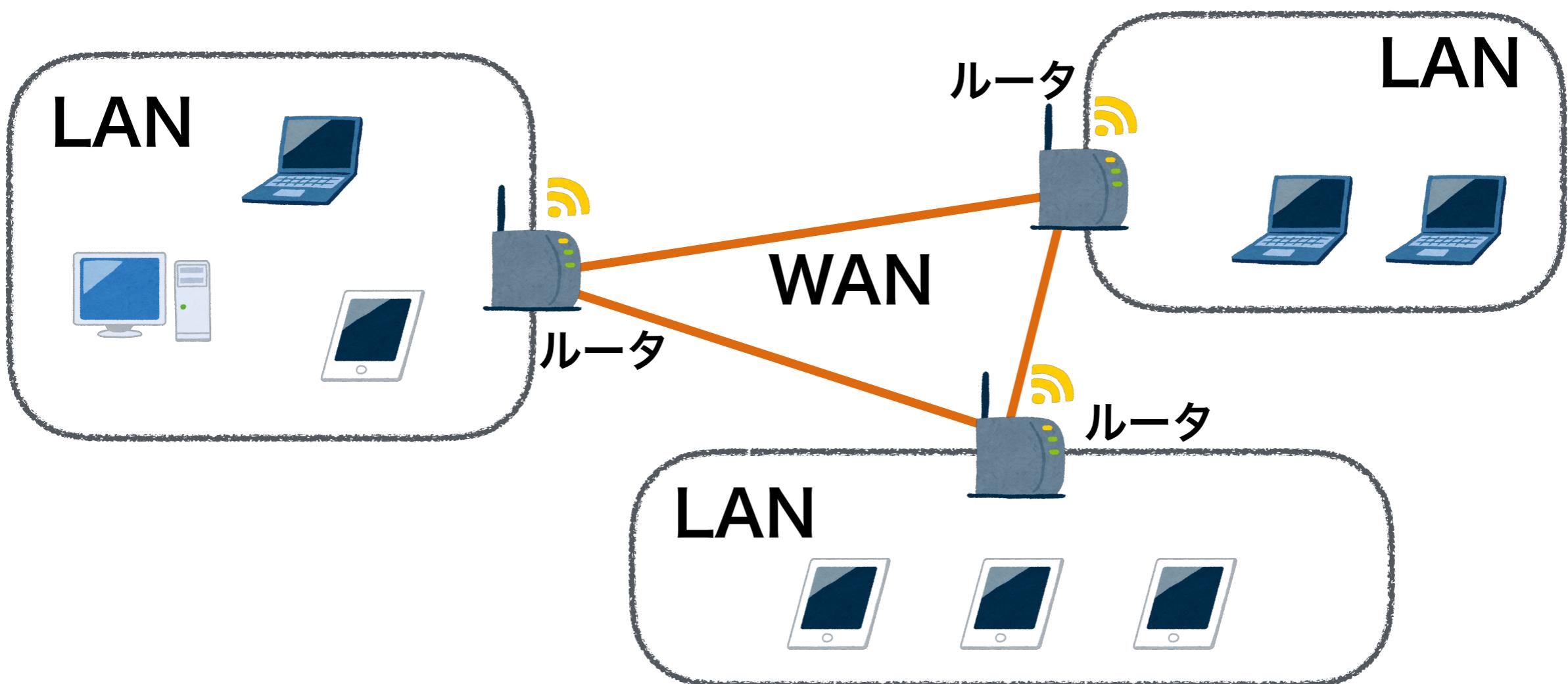
- ▶ ネットワークの基本概念
- ▶ ソケットを使ったプログラム

本日は・・・

- ▶ ネットワークの基本概念
- ▶ ソケットを使ったプログラム

ネットワーク

- ▶ **LAN (Local Area Network)**
大学や企業、一般家庭などの比較的狭い空間にある機器同士をつなぎだネットワーク
- ▶ **WAN (Wide Area Network)**
郊外や国際の範囲で地理的に離れた場所にある機器同士をつなぎだネットワーク
- ▶ **インターネット (Internet)**
LANやWANのネットワークを世界的な規模で相互に接続した通信ネットワーク



プロトコル

機器同士が互いに通信する際に必要となる一連の手順

【手順の例】

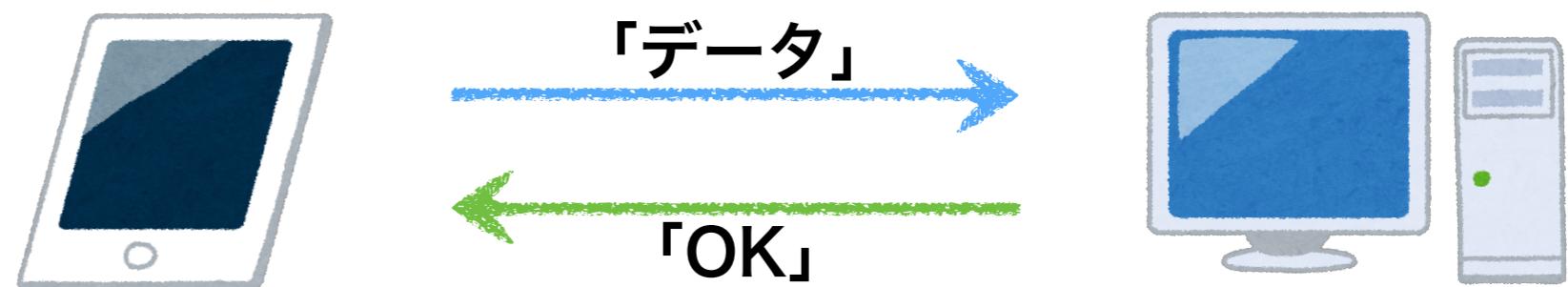
①開始

通信可能かどうか確認する



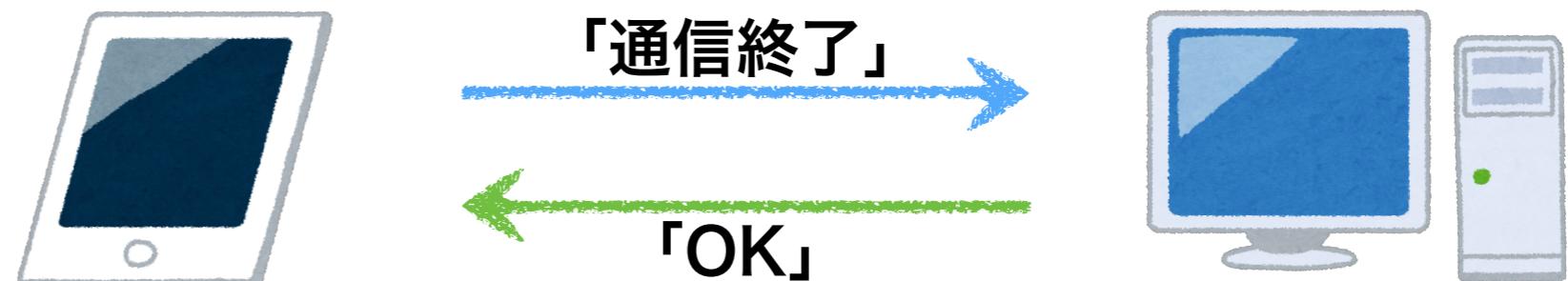
②データの送信

反応がない場合は再度送る



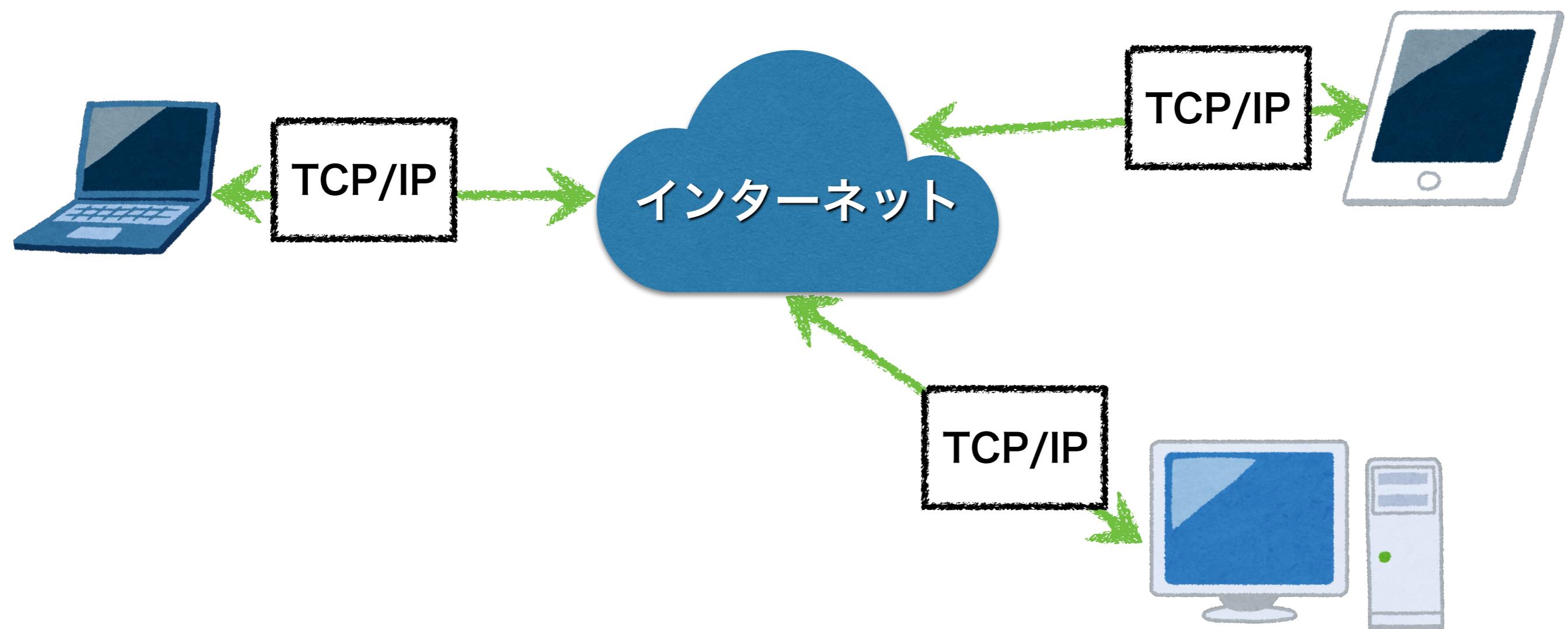
③終了

通信を終了する



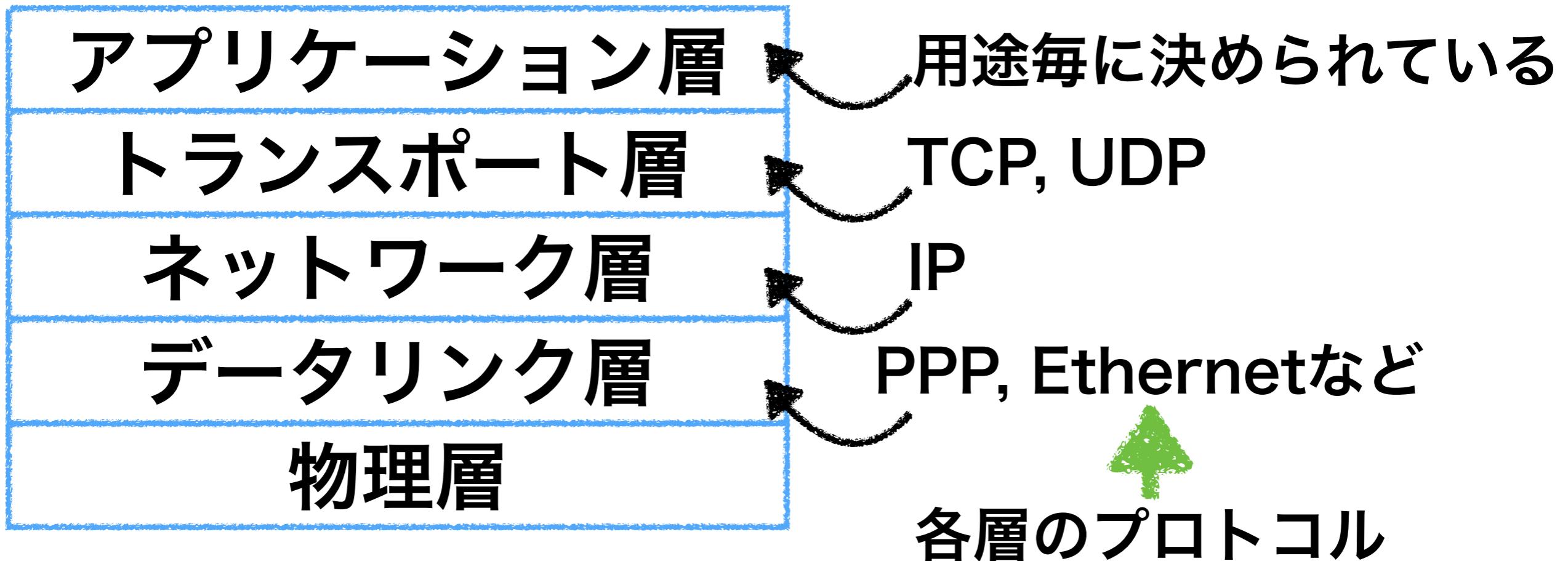
TCP/IP

インターネットにおいて**世界共通**で利用されているプロトコル



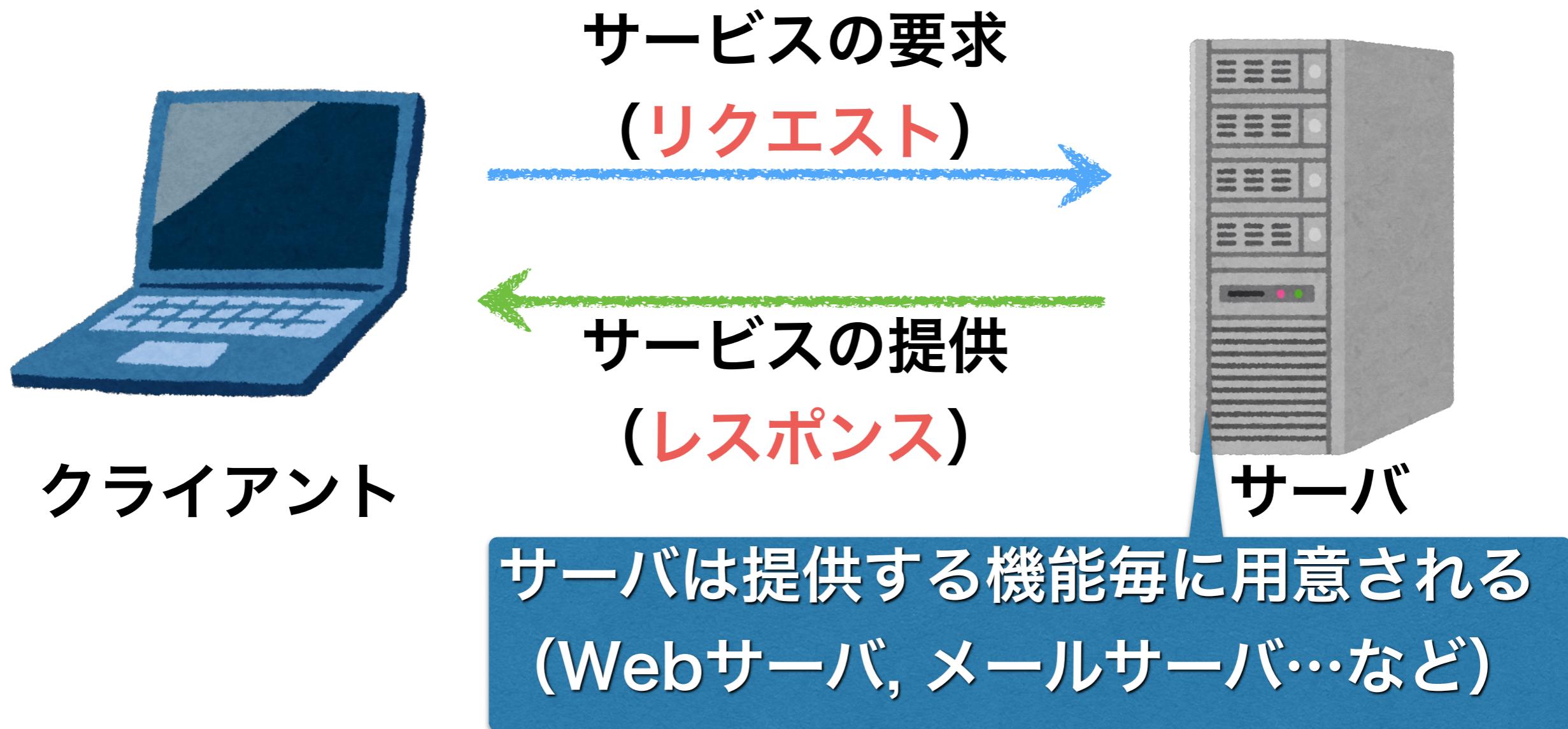
TCP/IPの階層

役割ごとにプロトコルが層に分かれている



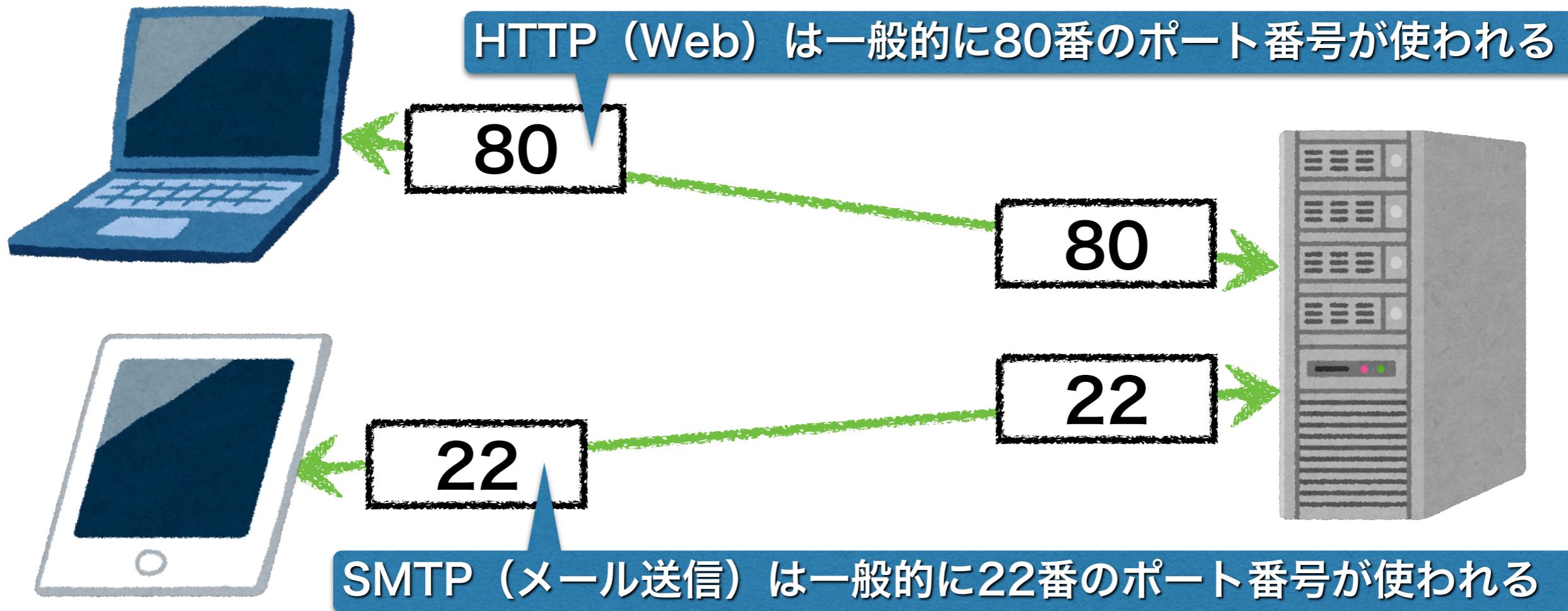
クライアントとサーバ

サービス（機能）を要求する側（クライアント）と、
サービスを提供する側（サーバ）に分ける方式



ポート

プロトコルごとにデータの出入口（**ポート**）が用意されており、各ポートには番号（**ポート番号**）が割り振られている



0~1023番は予め予約されたウェルノウン・ポート番号

ポートとIPアドレス

TCP/IPでは通信先としてIPアドレスにポート番号を付けて
どのサーバのどの機能との通信なのかを明示する

IPアドレスが172.16.52.235のコンピュータの
80番ポート (Webの機能) ヘアクセスする場合

172.16.52.235:80

172.16.52.235:22

IPアドレスが172.16.52.235のコンピュータの
22番ポート (メール送信の機能) ヘアクセスする場合

同じコンピュータが複数の機能を同時に提供することができる

本日は・・・

- ▶ ネットワークの基本概念
- ▶ ソケットを使ったプログラム

ソケット

アプリケーションプログラムからネットワーク
通信サービスを実現するために利用するAPI

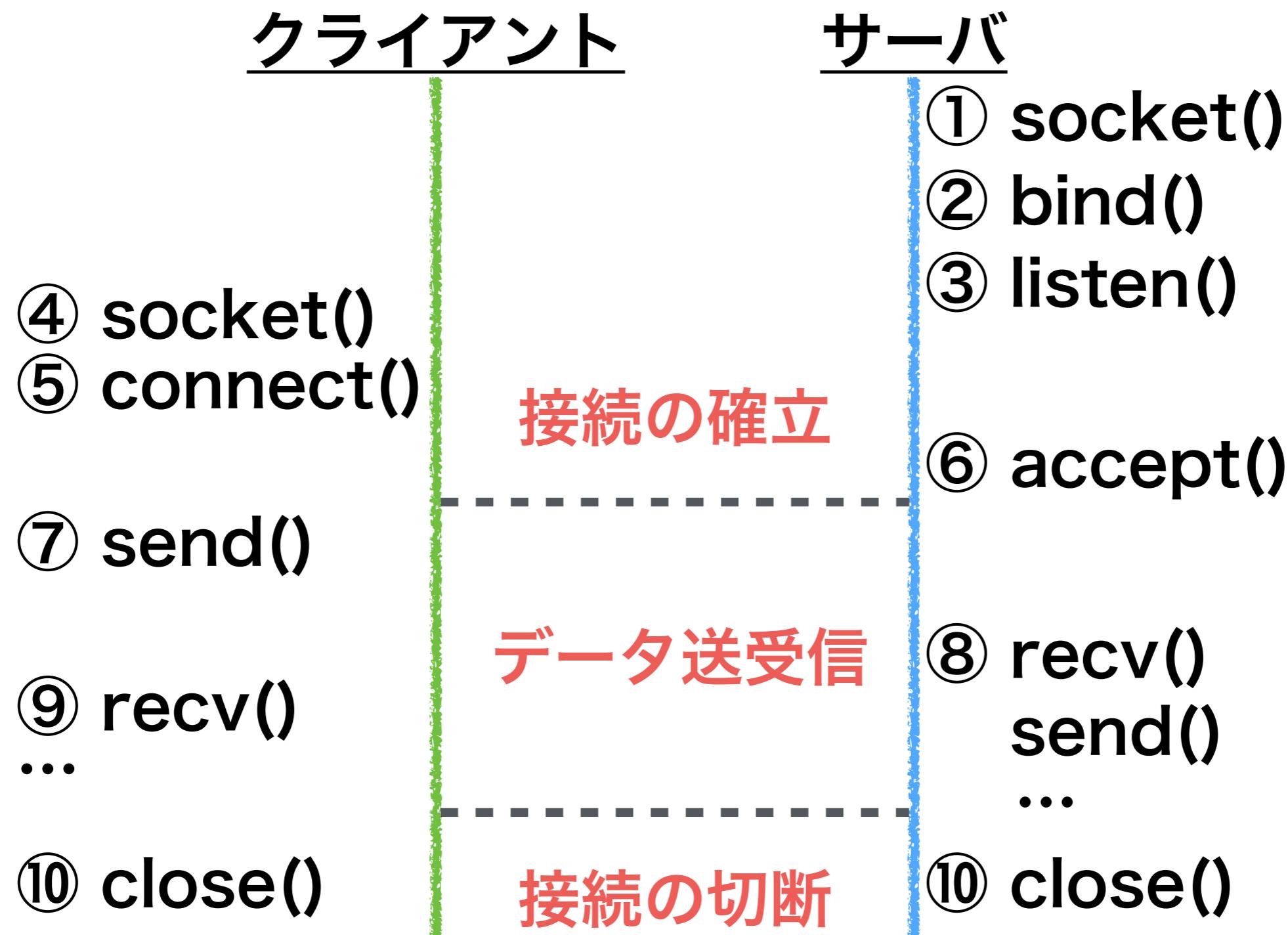
- ▶ OSが機能を提供している
- ▶ システムコールとして様々な関数が用意されている
- ▶ プログラムはC言語で書くことができる
- ▶ WindowsではCygwinで機能を追加できる

ソケットのシステムコール

クライアント	動作
(1) socket()	TCP とアプリケーション間の通信路を作る
(2) connect()	サーバに接続する
(3) send()	サーバに要求メッセージを送信する
(4) recv()	サーバから応答メッセージを受信する
(5) close()	接続を終了する

サーバ	動作
(1) socket()	TCP とアプリケーション間の通信路を作る
(2) bind()	ポート番号と IP アドレスを指定する
(3) listen()	接続の受付を開始する
(4) accept()	受け付けた接続用のソケットを作る
(5) recv()	クライアントから要求メッセージを受信する
(6) send()	クライアントに応答メッセージを送信する
(7) close()	接続を終了する

ソケットによる通信の流れ



ソケットで利用する構造体

構造体名	説明
sockaddr	アドレス情報を持つ汎用的な構造体
sockaddr_in	sockaddr を IP アドレスとポート番号の情報を持つようにした構造体
in_addr	sockaddr_in で IP アドレスを持つための構造体

ソケットで利用する主な関数

socket()：ソケットを作成する

- ▶ 第1引数：通信プロトコル（通常はPF_INET）
- ▶ 第2引数：通信の種類（通常はSOCK_STREAM）
- ▶ 第3引数：プロトコル詳細設定（通常はIPPROTO_TCP）

- ▶ 戻り値：ソケットの作成に失敗 → -1
ソケットの作成に成功 → 整数值（ソケットディスクリプタ）

【記述例】

```
sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

close()：ソケットを閉じる

- ▶ 第1引数：ソケットディスクリプタ

ソケットで利用する主な関数

bind(): ソケットにIPアドレス等を設定

- ▶ 第1引数：ソケット
- ▶ 第2引数：sockaddr_in構造体へのポインタ
- ▶ 第3引数：sockaddr_in構造体のサイズ

- ▶ 戻り値：ソケットの作成に失敗 → -1
ソケットの作成に成功 → 0

【記述例】

```
bind(sSock, (struct sockaddr *)&sAddr, sizeof(struct sockaddr));
```

ソケットで利用する主な関数

listen(): クライアントからの接続待ちを準備

- ▶ 第1引数：ソケット
- ▶ 第2引数：同時接続できるクライアント数

- ▶ 戻り値：
失敗 → -1
成功 → 0

【記述例】

```
listen(sSock, 5);
```

ソケットで利用する主な関数

accept(): クライアントからの接続を承認する

- ▶ 第1引数：ソケット
- ▶ 第2引数：クライアントのsockaddr_in構造体へのポインタ
- ▶ 第3引数：構造体のサイズを格納した変数へのポインタ

- ▶ 戻り値：失敗 → -1
成功 → 整数値（データを送受信用のソケットディスクリプタ）

【記述例】

```
cSock =accept(sSock, (struct sockaddr *)&cAddr, &cLen);
```

ソケットで利用する主な関数

connect(): クライアントからサーバへ接続する

- ▶ 第1引数：ソケット
- ▶ 第2引数：サーバのsockaddr_in構造体へのポインタ
- ▶ 第3引数：構造体のサイズを格納した変数へのポインタ

- ▶ 戻り値：失敗 → -1
成功 → 整数值

【記述例】

```
connect(sock, (struct sockaddr *)&sAddr, sizeof(struct sockaddr));
```

ソケットで利用する主な関数

send() / recv() : データを送信 / 受信する

- ▶ 第1引数 : 送り先のソケット
- ▶ 第2引数 : 文字列配列
- ▶ 第3引数 : 文字列の長さ
- ▶ 第4引数 : 動作の設定 (通常は0)

- ▶ 戻り値 : 失敗の場合 → -1
成功の場合 → 送受信したバイト数

【記述例】

```
send(cSock, buffer, msgSize, 0);
recv(sock, buffer, sizeof(buffer), 0);
```

コンパイルと実行手順

1. プログラムのコンパイル

```
$ cc 「クライアントプログラム名」 -o client  
$ cc 「サーバプログラム名」 -o server
```

2. サーバプログラムの実行

(一度起動したら Control-C で停止するまで動く)

```
$ ./server  
ポート番号 > 「ポート番号 例: 5000」
```

クライアントの受付を開始しました。

コンパイルと実行手順

3. クライアントプログラムの実行

まず、サーバの IP アドレスを調べます。Linux の場合「ifconfig」コマンドで見ることができます。画面に表示された「inet アドレス」の項目が、そのコンピュータの現在の IP アドレスとなります。

(この演習室の場合、最初の「172.16.206.」はどのコンピュータでも共通です。)

```
$ /sbin/ifconfig eth0
```

(「inet アドレス」欄の IP アドレスを確認する。一度確認すれば、再度実行する必要はない。)

```
$ ./client
```

IPアドレス > 「サーバのIPアドレス」

ポート番号 > 「サーバで入力したポート番号」

送信文字列 > 「送信する文字列」

コンパイルと実行手順

4. 他の人のサーバプログラムと自分のクライアントプログラムで通信してみる

サーバプログラムが実行している IP アドレスと、サーバプログラムで入力したポート番号がわかれば、サーバとクライアントのプログラムを異なるコンピュータで実行しても通信できます。

【練習14-1】

以下のようにサンプルプログラムをダウンロードして、プログラムを実行してみましょう。

クライアントプログラム：client.c
サーバプログラム：server.c

このプログラムは、「クライアントで文字列を入力し、文字列をサーバへ送り、サーバは受け取った文字列をそのままクライアントへ送り返す」処理をします。

【課題14-1】

サンプルプログラムを次のような動作になるように改良して下さい。（サーバのみの変更だけで完成します。）

TCP サーバプログラムで、受信した文字列データを表示した後に、**文字列を入力し**、クライアントにその文字列を送信する。

- ▶ recv()で受信した文字列をprintf()で表示したあとに、scanf() を使って文字列を入力する。
- ▶ send()箇所を、クライアントプログラムでのsend()の使い方を参考に、入力した文字列の長さを調べて長さ分だけ送るようにする。

【課題14-2】

課題14-1のプログラムを次のように改良して下さい。
(クライアントとサーバのプログラムを変更する必要
があります。)

クライアントが送信した**整数値**をサーバで受信し表示
した後に、 サーバで入力した**整数値**を加算し、 クラ
イアントに送信する。

- ▶ クライアントとサーバで送受信するデータを文字列から整数に変更する
(char型の配列からint型の変数に変更する)
- ▶ サーバのrecv()で受信した整数値をprintf()で表示したあとに、 scanf()
を使って整数を入力し、 受信した整数値と加算して、 send()で送信する