TCP

学習内容

1. TCPの概要
2. ポート番号の役割
3. TCPヘッダの構造
4. TCPの通信制御
5. TCPの状態遷移
6. 信頼性と効率の仕組み

01

TCPの概要

TCPとは何か？

インターネットで標準的に利用される信頼性の高い通信プロトコル

TCPは **Transmission Control Protocol** の略

OSI参照モデルの**トランスポート層**で動作

IPが「宛先のコンピュータまで届ける」役割を担うのに対し、TCPは「**アプリケーションに正しくデータを届ける**」役割を持つ

HTTP、FTP、メールなど、多くのアプリケーションプロトコルの土台となっている

TCPの主な特徴

TCPが「信頼性の高い通信」と呼ばれる理由を3つの特徴から解説します

信頼性	順序制御	コネクション型
確認応答や再送制御により、送信したデータが相手に確実に届くことを保証	分割して送られたデータがバラバラに届いても、送信時の正しい順番に並べ替える	データ送信の前に、送信側と受信側で通信経路を確立する「3ウェイハンドシェイク」を行う

TCPとUDPの違い

トランスポート層で使われるもう一つの代表的なプロトコル「UDP」との比較

TCP (信頼性重視)	UDP (速度・リアルタイム性重視)
コネクションを確立してから通信する	コネクションを確立せずに通信する
データが確実に届くことを保証（**再送あり**）	データが届く保証はない（再送なし）
データの順序を保証する	データの順序も保証しない
Webサイト閲覧、メール、ファイル転送など	動画配信、オンラインゲーム、DNSなど

02

ポート番号の役割

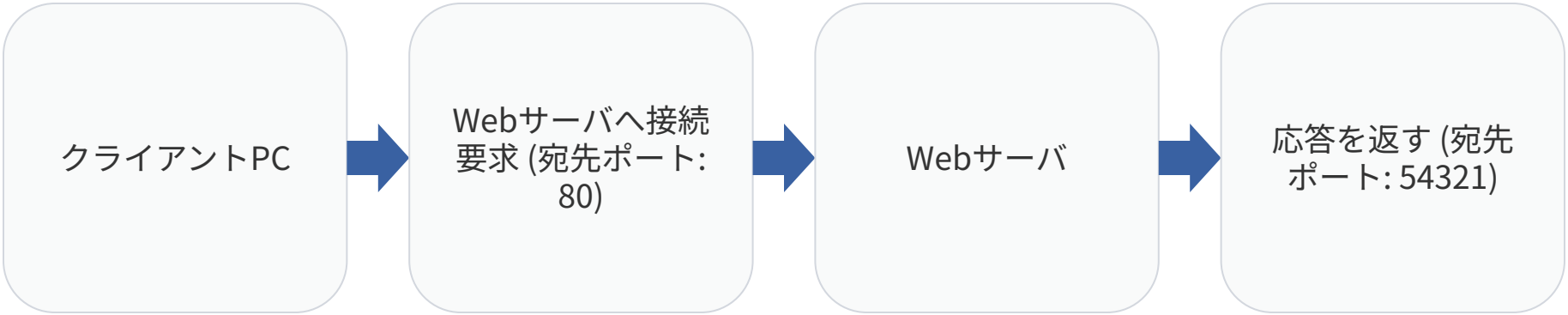
ポート番号の分類

ポート番号は用途に応じて3つの範囲に分類されています

種類	範囲	主な用途・例
ウェルノウンポート番号	0〜1023	広く知られた基本的なサービス用に予約済み (HTTP:80, HTTPS:443)
登録済みポート番号	1024〜49151	特定のアプリケーション用に登録・利用される (MySQL:3306)
ダイナミックポート番号	49152〜65535	クライアント側が通信の都度、動的に使用する

通信の流れの例（Webブラウザ）

ポート番号がどのように使われ、通信が振り分けられるかの流れを見てみましょう



03

TCPヘッダの構造

TCPセグメントの構成

TCPで送受信されるデータのかたまりは「TCPヘッダ」と「ペイロード」で構成される

TCPヘッダの主要フィールド

TCPヘッダには通信制御に不可欠な情報が含まれています

フィールド名	説明
送信元・宛先ポート番号	どのアプリケーション間の通信かを識別
シーケンス番号	データの順序を管理するための通し番号
確認応答番号 (ACK番号)	どこまでデータを受け取ったかを相手に伝える番号
コントロールフラグ	接続要求(SYN)や接続終了(FIN)など、通信の状態を制御する信号
ウィンドウサイズ	一度に受信できるデータ量を相手に通知

04

TCPの通信制御

コネクションの確立 (3ウェイハンドシェイク)

データを送受信する前に、3段階のやり取りで仮想的な通信路を確立します

STEP 1	**SYN**: クライアントからサーバへ接続を要求
STEP 2	**SYN/ACK**: サーバが要求を受け入れ、クライアントへも接続を要求
STEP 3	**ACK**: クライアントがサーバの要求を承諾し、接続が確立

コネクションの切断 (4ウェイハンドシェイク)

通信の終了は、お互いの送信データがなくなったことを確認し合うため4段階のやり取りで行われます

STEP 1	**FIN**: ホストAからBへ「もう送るデータはありません」と通知
STEP 2	**ACK**: ホストBが「了解しました」と応答
STEP 3	**FIN**: ホストBからAへ「こちらも送るデータがなくなりました」と通知
STEP 4	**ACK**: ホストAが「了解しました」と応答し、通信が完全に終了

05

TCPの状態遷移

TCPの主な状態（ステート）

LISTEN

サーバがクライアントからの接続要求（SYN）を待ち受けている状態

ESTABLISHED

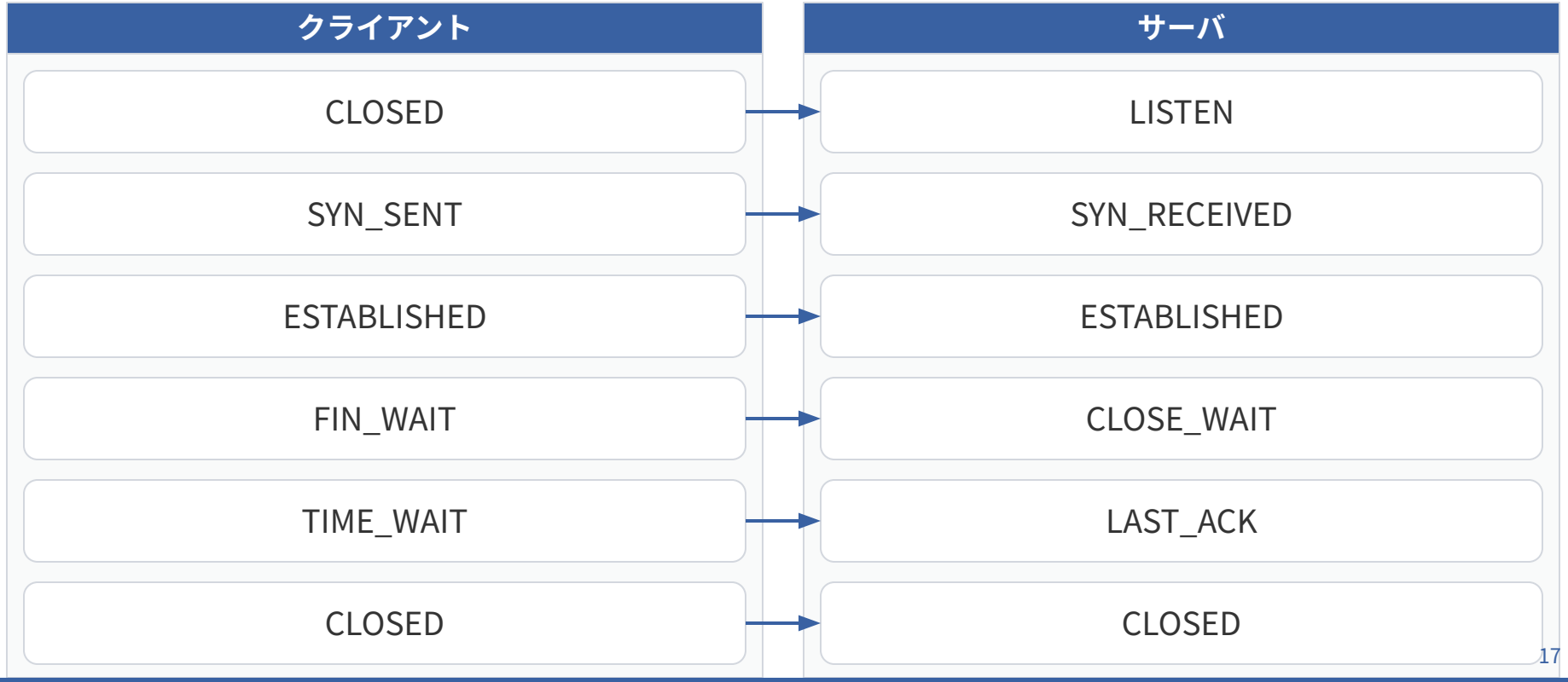
3ウェイハンドシェイクが完了し、データ通信が可能な状態

TIME_WAIT

接続終了後、最後のACKが相手に届かなかった場合に備えて一定時間待機する状態

状態遷移の流れ（イメージ）

3ウェイハンドシェイクと4ウェイハンドシェイクにおけるクライアントとサーバの状態変化



06

信頼性と効率の仕組み

信頼性の確保

TCPの信頼性は「順序制御」と「再送制御」という2つの重要な仕組みで支えられています

順序制御

送信するパケットに「**シーケンス番号**」を付与

受信側はシーケンス番号を基にデータを正しい順序に並べ替え

パズルのピースを番号通りに組み立てるイメージ

再送制御

受信側はデータを受け取ると「**ACK（確認応答）**」を返す

一定時間ACKが返らない場合、送信側はパケットが消失したと判断し、**データを再送**

「届いたよ」の返事がなければ送り直すイメージ

通信効率の向上

信頼性を保ちつつ高速通信を実現するための「ウィンドウ制御」と「フロー制御」

ウィンドウ制御

ACKを待たずに連続でデータを送信できる量を決める仕組み（**ウィンドウサイズ**）

一度にまとめてデータを送り、まとめてACKを受け取ることで効率化

スライディングウィンドウで待ち時間を最小化

フロー制御

受信側の処理能力に応じて送信量を調整する仕組み

受信側が忙しい時、ウィンドウサイズを**小さく通知**して送信を抑制

受信側のバッファ溢れ（パンク）を防ぐ

初心者向けイメージ

ウィンドウ制御は「**まとめて荷物を送る許可**」。
フロー制御は「**受け取る側の都合に合わせて荷物の量を調整すること**」

— TCPの仕組みの比喩