

情報ネットワーク

伊藤 嘉浩

講義内容

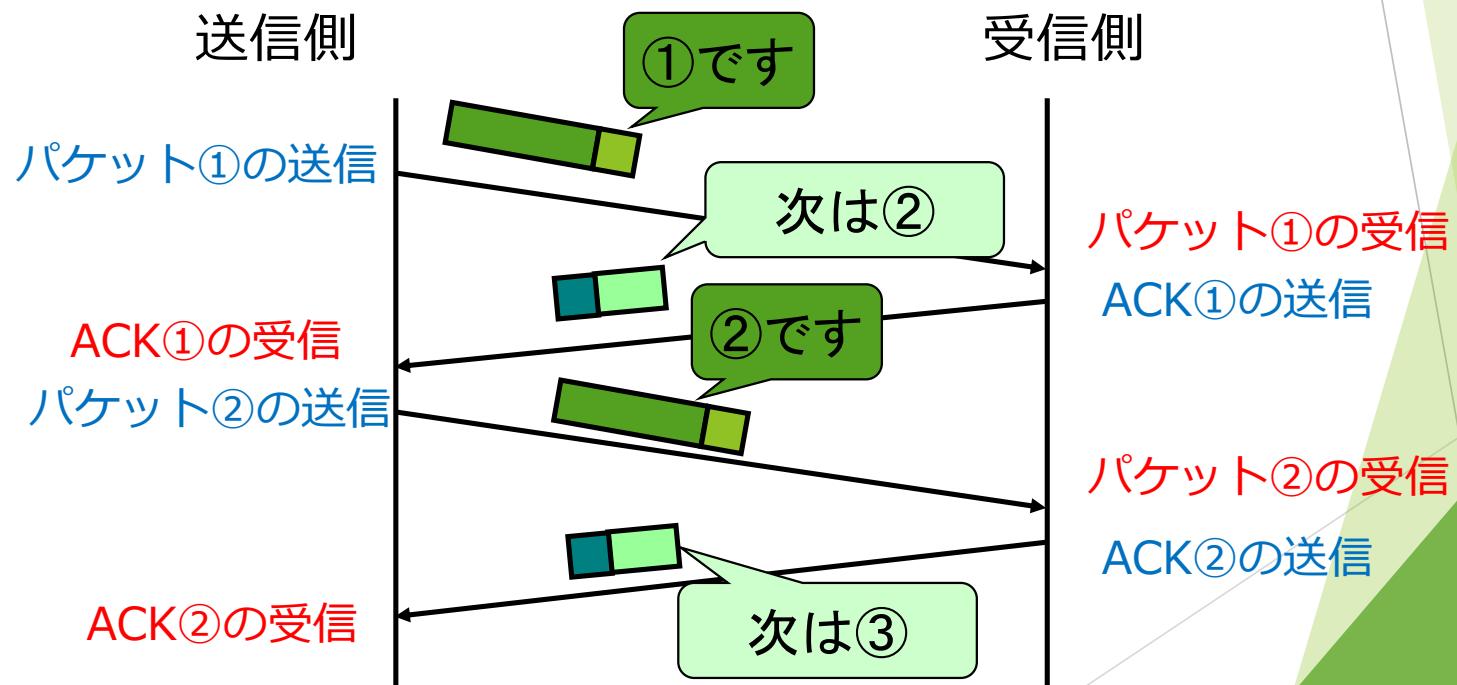
- ▶ 1) 序論
- ▶ 2) 3) ネットワークアーキテクチャ
- ▶ 4) 5) 伝送路と物理層
- ▶ 6) 誤り制御方式
- ▶ 7) MACプロトコル
- ▶ 8) 中間試験
- ▶ 9) データリンク層プロトコル
- ▶ 10) 11) データ交換とネットワーク層
- ▶ 12) 13) **14) TCP/IP**
- ▶ 15) 期末試験
- ▶ 16) 統括

14) TCP/IP

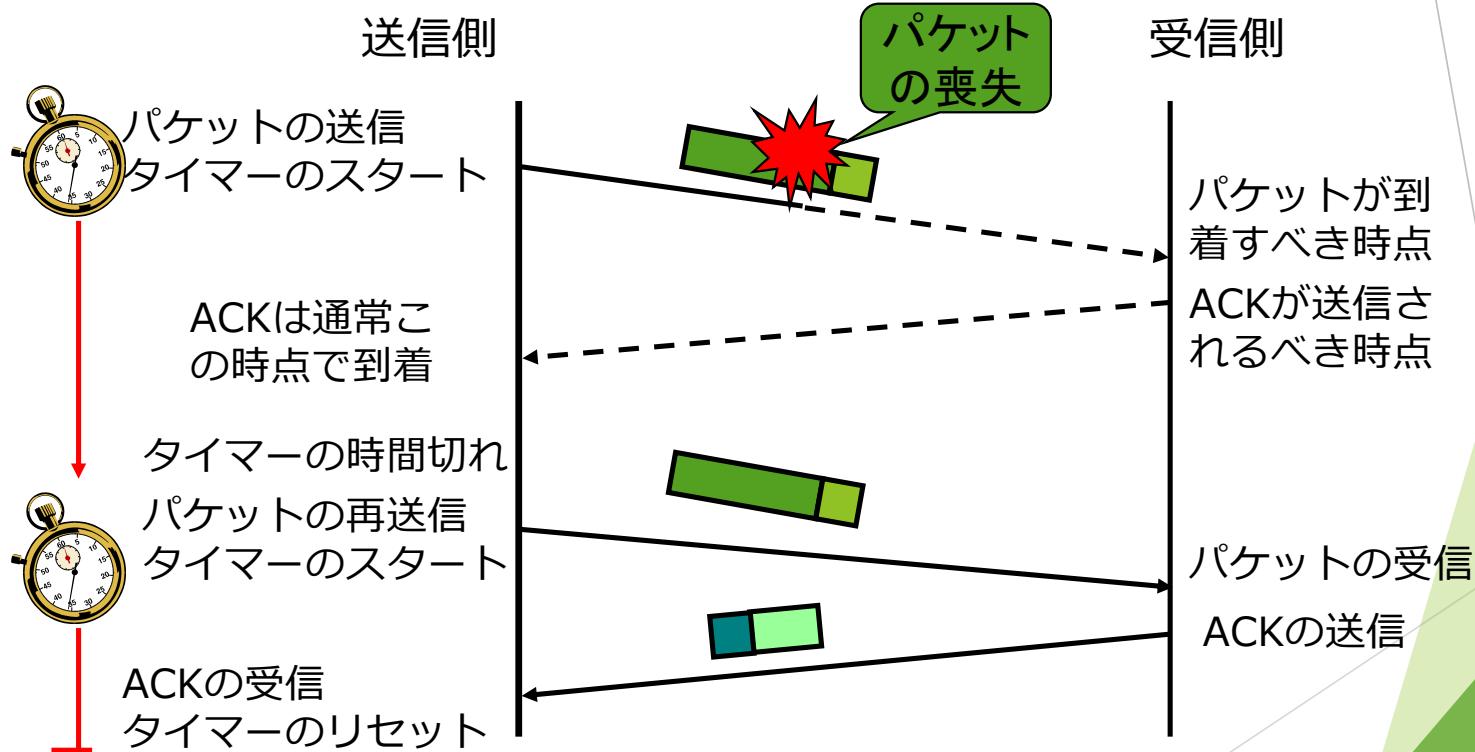
TCP続き

肯定型確認応答

ACK(Acknowledgment)の送信

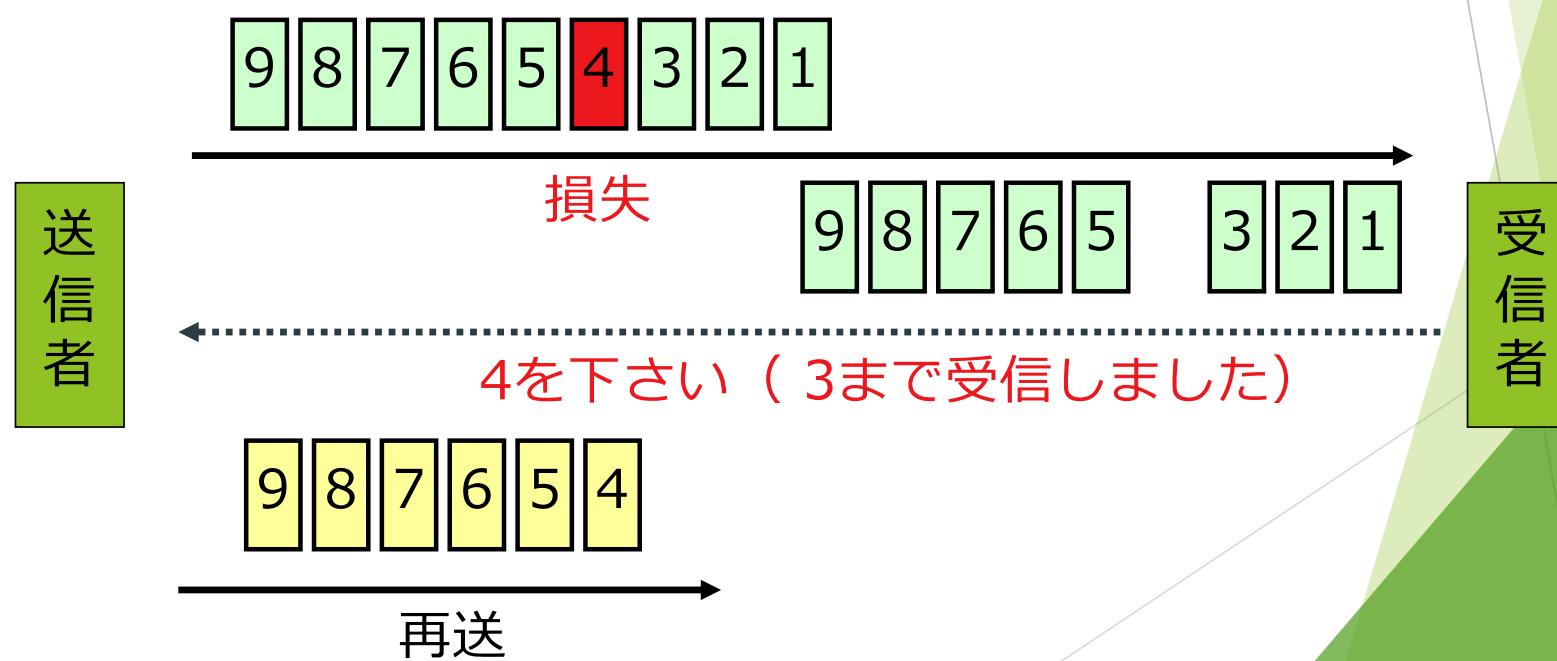


タイムアウト再送



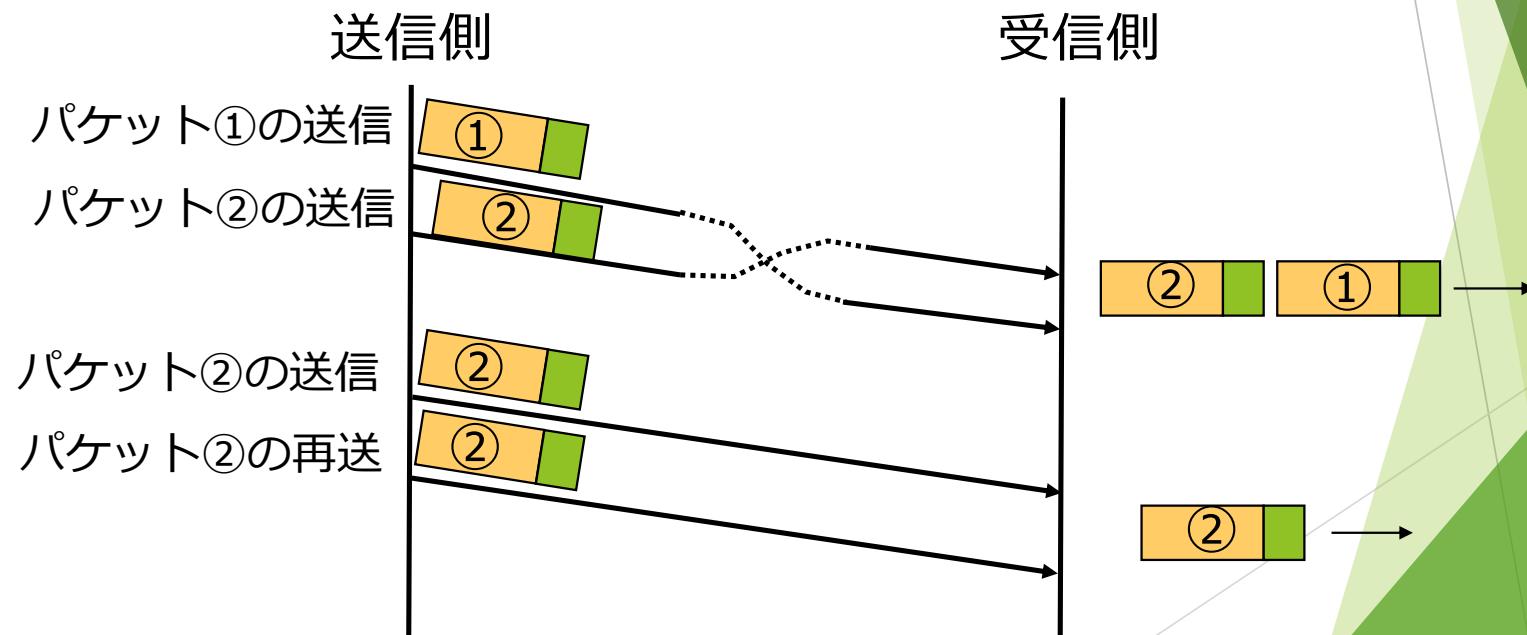
go-back-N再送

- ▶ データの損失を生じたところまで戻って、すべてのデータを再送



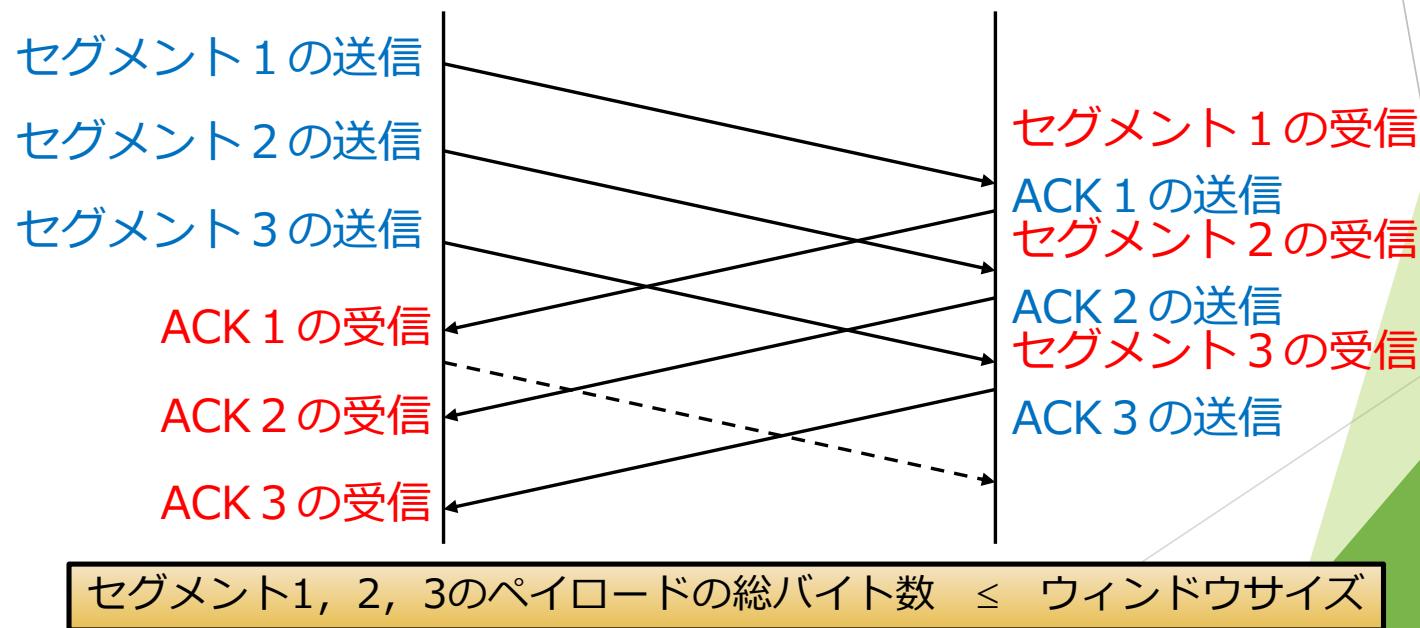
順序制御

パケット順序の入れ替わりや重複を修正



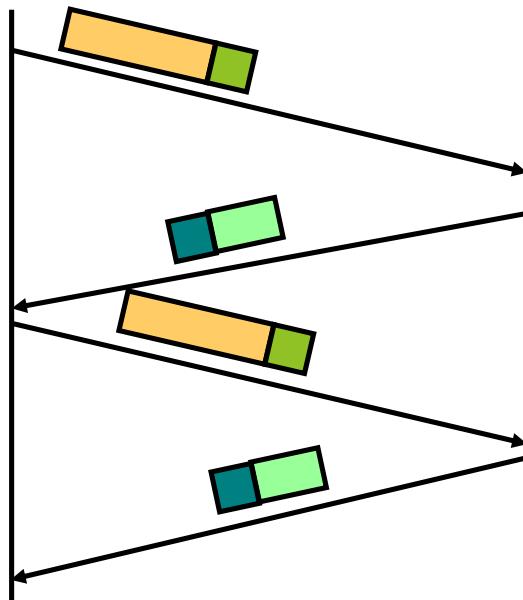
フロー制御（1）

- ▶ ウィンドウフロー制御
 - ▶ 送達確認を待つ前にウィンドウサイズ分のセグメントを送信
- ▶ ウィンドウサイズ
 - ▶ 送達確認を受け取る前に連續して送信できるバイト数の最大値



ウィンドウ制御

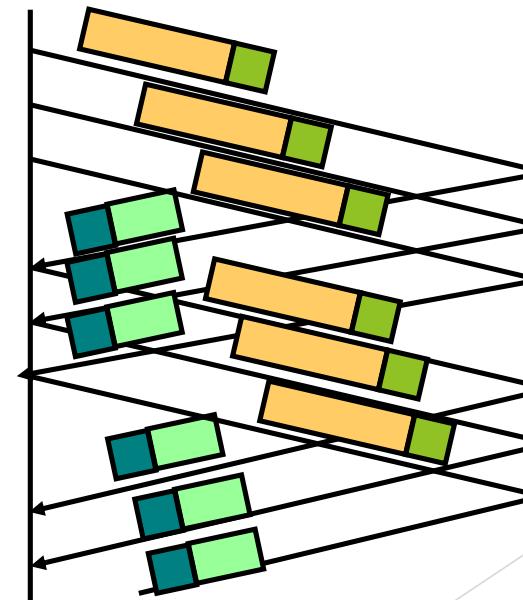
送信側



ストップアンドウェイトARQ

受信側

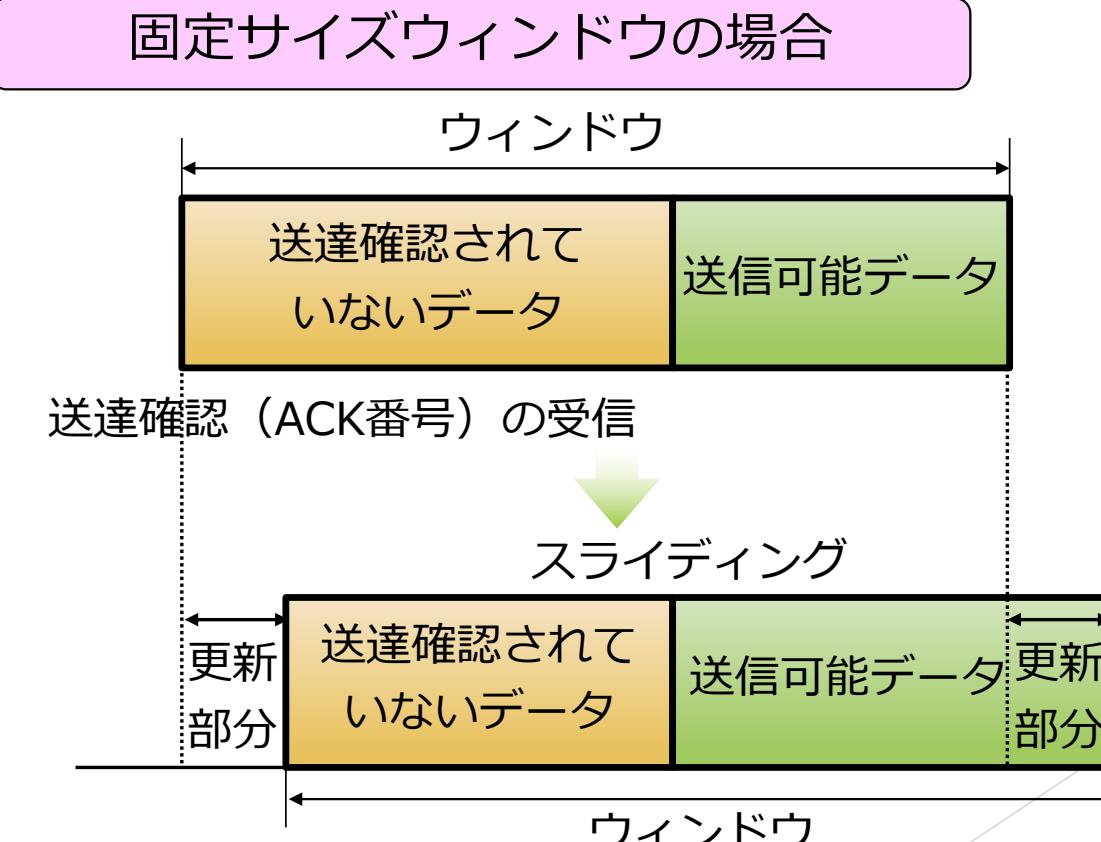
送信側



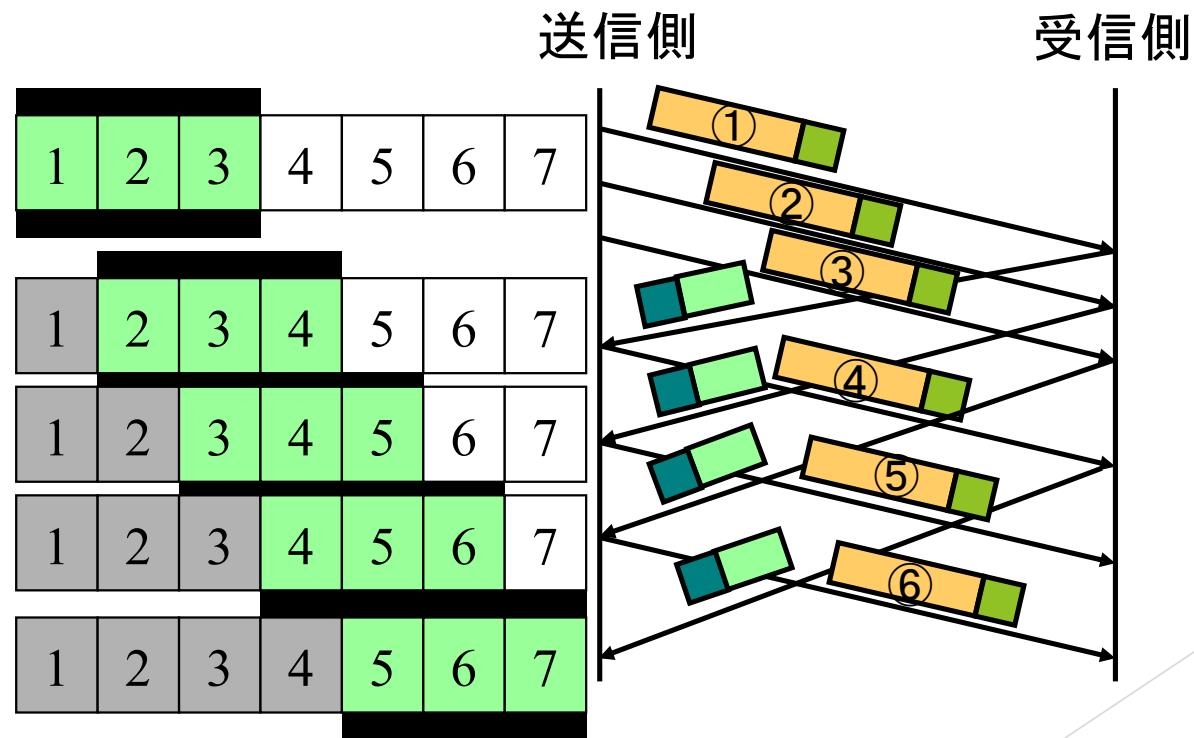
ウィンドウ制御を用いた
連続ARQ

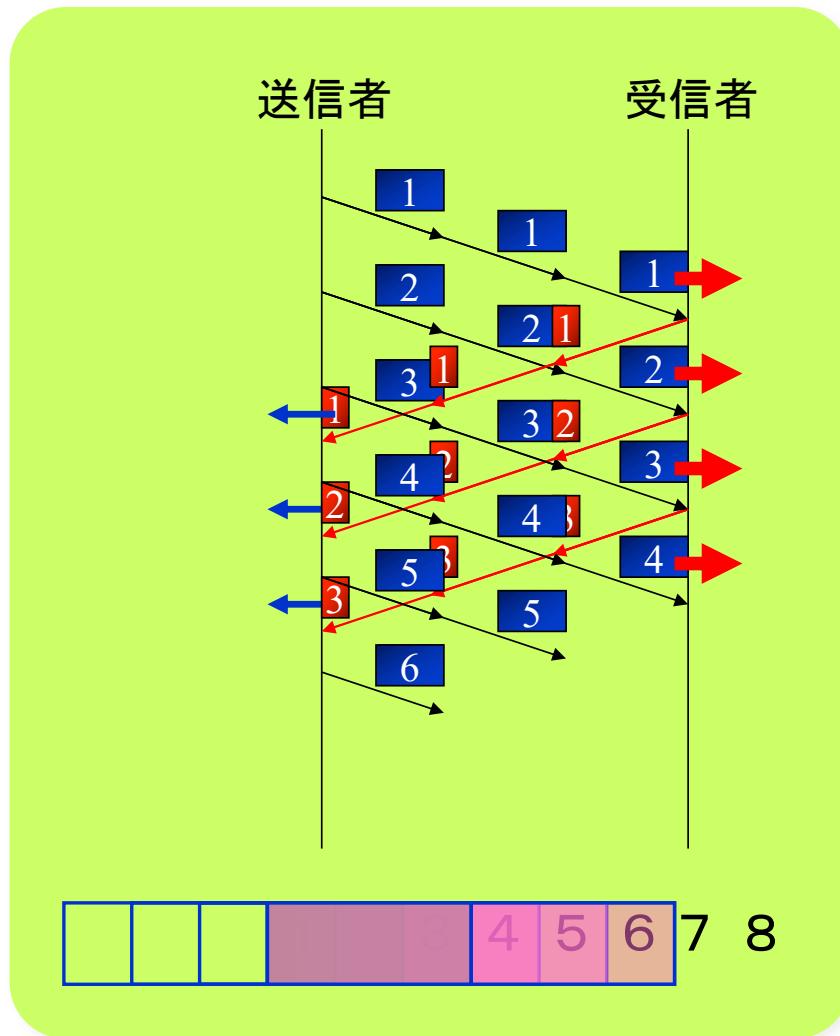
フロー制御 (2)

スライディングウィンドウ機構



スライディングウィンドウ

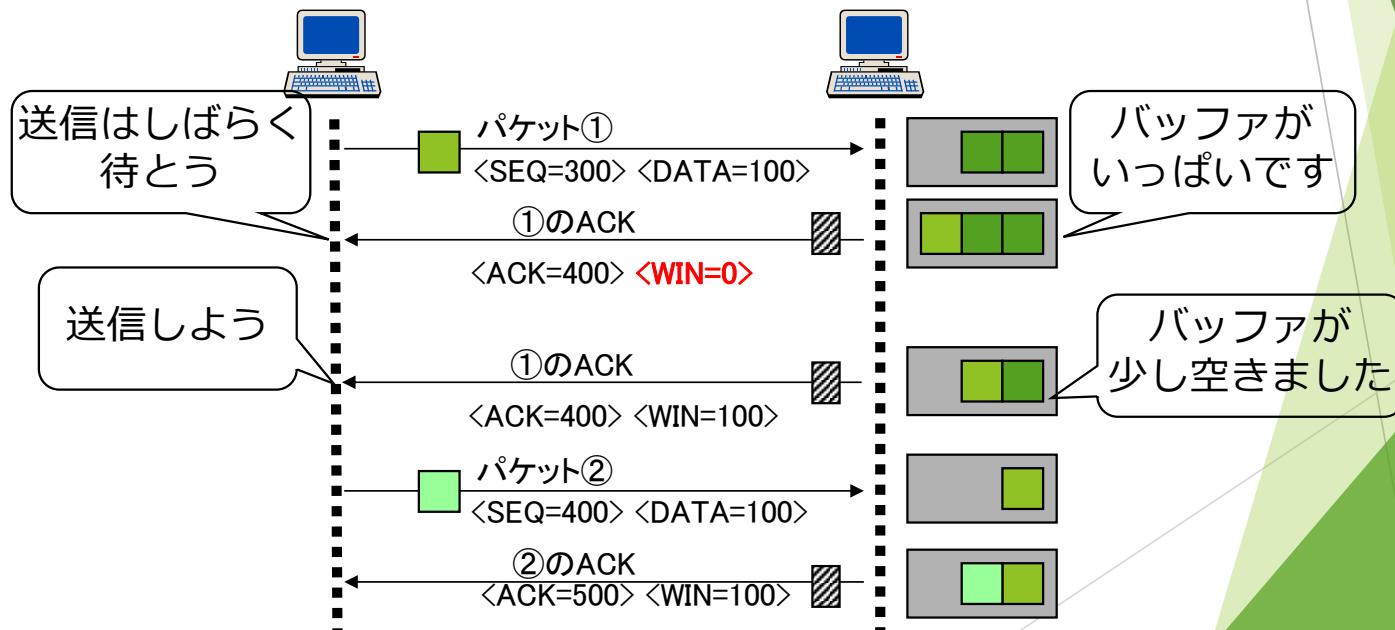




ウインドウサイズが
3の場合

フロー制御（3）

▶送信速度に比べ受信処理速度が遅い場合、
(広告) ウィンドウサイズで送信速度を制御



14) TCP/IP

UDP, ソケットシステムコール

UDP概要

UDP (User Datagram Protocol)

ユーザデータグラム : UDPでの転送の単位

- ▶ コネクションレス型
 - ▶ コネクションを確立することなく、いきなりデータ転送可能
- ▶ 信頼性を保証しない
 - ▶ ユーザデータグラムの順序制御、誤り制御、フロー制御など通信における信頼性を保証する機能を持たない
 - ▶ ユーザデータグラムの重複、順序違いなどの正常化、消失や誤りの回復は、上位層に任される

UDPメッセージフォーマット

- ▶ ユーザデータグラム
 - ▶ UDPメッセージのこと
- ▶ ユーザデータグラムフォーマット



- ▶ UDPヘッダのフォーマット



ソケットインターフェース（1）

- ▶ TCP/IP通信のための代表的なAPI
 - ▶ ソケット (**Berkeley sockets**)
 - ▶ カリフォルニア大学バークレー校でUNIX向けに開発
 - ▶ TLI(Transport Layer Interface)
 - ▶ AT&Tによって開発
- ▶ ソケット
 - ▶ データを送信及び受信するための通信端点
 - ▶ ソケットはUNIXのファイル・デバイスと同じように動作
 - ▶ あたかもファイルを扱うように、readやwriteのような従来からのオペレーションで使用可能
 - ▶ ソケット識別子とファイル識別子は同じ整数の集合から重複しないように割り当て

ソケットインターフェース（2）

- ▶ ネットワーク用APIのデファクトスタンダード
- ▶ ほとんどすべてのOSで実装
- ▶ ソケットの種類
 - ▶ **ストリームソケット**
 - ▶ 信頼性の高い通信を実現するためのソケット（TCP用）
 - ▶ **データグラムソケット**
 - ▶ ストリーム型の品質は保証されない（UDP用）
- ▶ C言語のAPIであるが、他の言語でも類似のAPI
- ▶ TCPとUDPはソケットシステムコールで操作

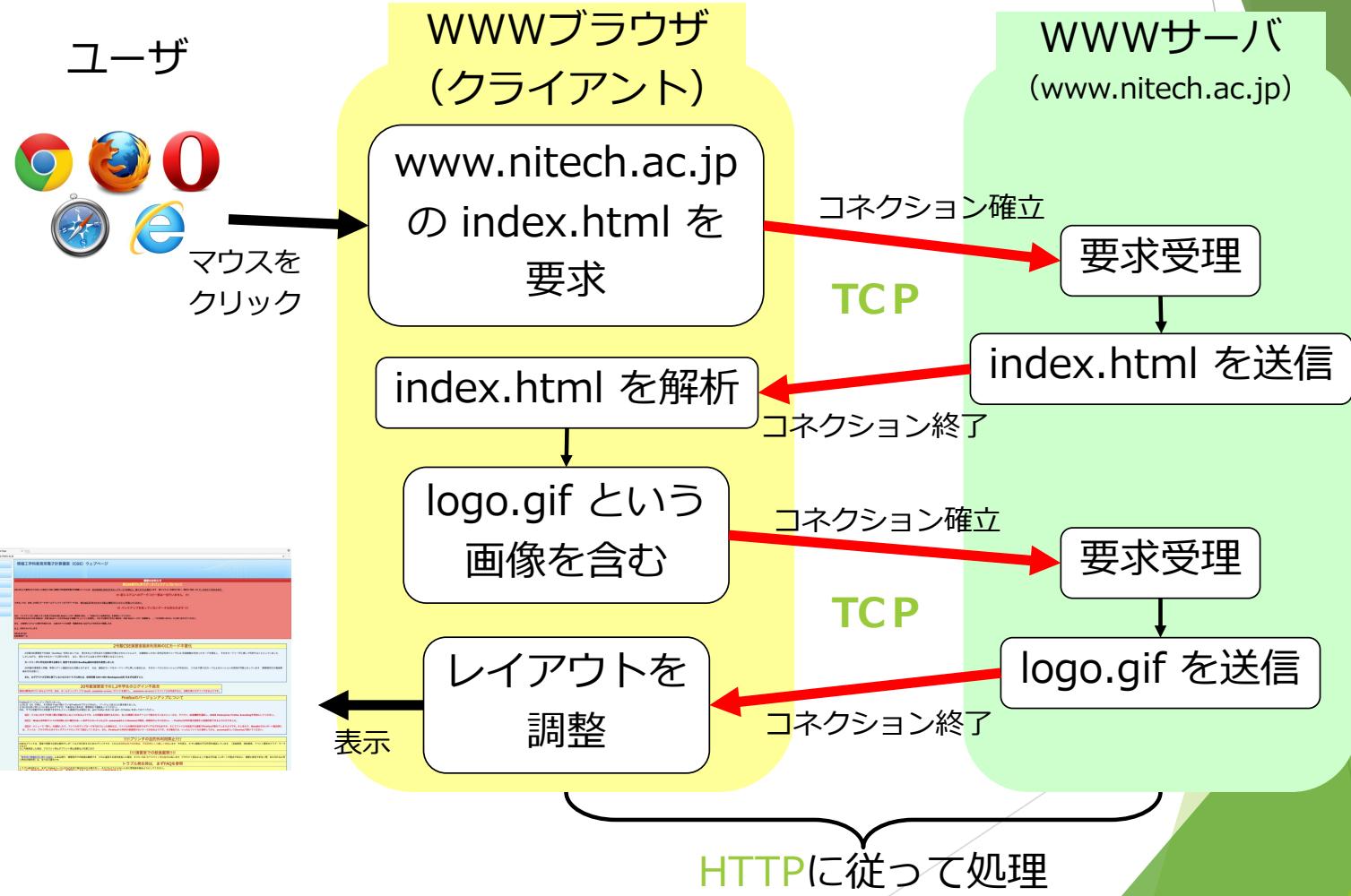
ソケットインターフェース（3）

- ▶ `socket()`
 - ▶ 片方の終端を作成し、それを表すファイル記述子を返す
 - ▶ `socket(プロトコルファミリー, ソケットタイプ, プロトコル)`
 - ▶ プロトコルファミリー : `AF_INET`(IPv4), `AF_INET6`(IPv6),
`AF_UNIX` (UNIXドメインソケット)
 - ▶ ソケットタイプ : `SOCK_STREAM` (ストリーム)
`SOCK_DGRAM` (データグラム) , `SOCK_RAW` (IPで直接)
 - ▶ プロトコル : 通常 は 0
 - ▶ `gethostbyname()` , `gethostbyaddr()`
 - ▶ 名前やアドレスで指定されたホストを返す
 - ▶ `connect()`
 - ▶ コネクションの確立
 - ▶ `bind()`
 - ▶ ソケットにアドレスを設定
 - ▶ `listen()`, `accept()`

14) TCP/IP

応用例：HTTP

応用例 (WWWのしくみ)



HTTP(Hypertext Transfer Protocol)

- ▶ WWWに用いられるアプリケーション層プロトコル
- ▶ クライアントとサーバ間でメッセージの交換を行う
- ▶ HTTPでは、このメッセージの構造や交換方法を定義
- ▶ [HTTP/1.0](#)と[HTTP/1.1](#)、そして[HTTP/2](#) というバージョンがある
 - ▶ HTTP/1.0 : RFC1945 (May 1996)
 - ▶ HTTP/1.1 : RFC7230-7235 (June 1999)
 - ▶ HTTP/2 : RFC7540 (May 2015)
- ▶ HTML (Hypertext Markup Language)
 - ▶ HTML2.0はRFC1866(Nov. 1995)で規定。ただし、RFC2854(June 2000)によって廃棄
 - ▶ HTML3.0以降は、W3C (World Wide Web Consortium)によって策定。HTML4.0をベースに、XHTMLも規定された
 - ▶ HTML5.2 ([Dec. 2017に改訂](#))
 - ▶ HTML5はすべて廃止 ([Jan. 2021](#))
 - ▶ ⇒WHATWGの[HTML Living Standard](#)が標準へ

HTML5

Taxonomy & Status on January 20, 2013

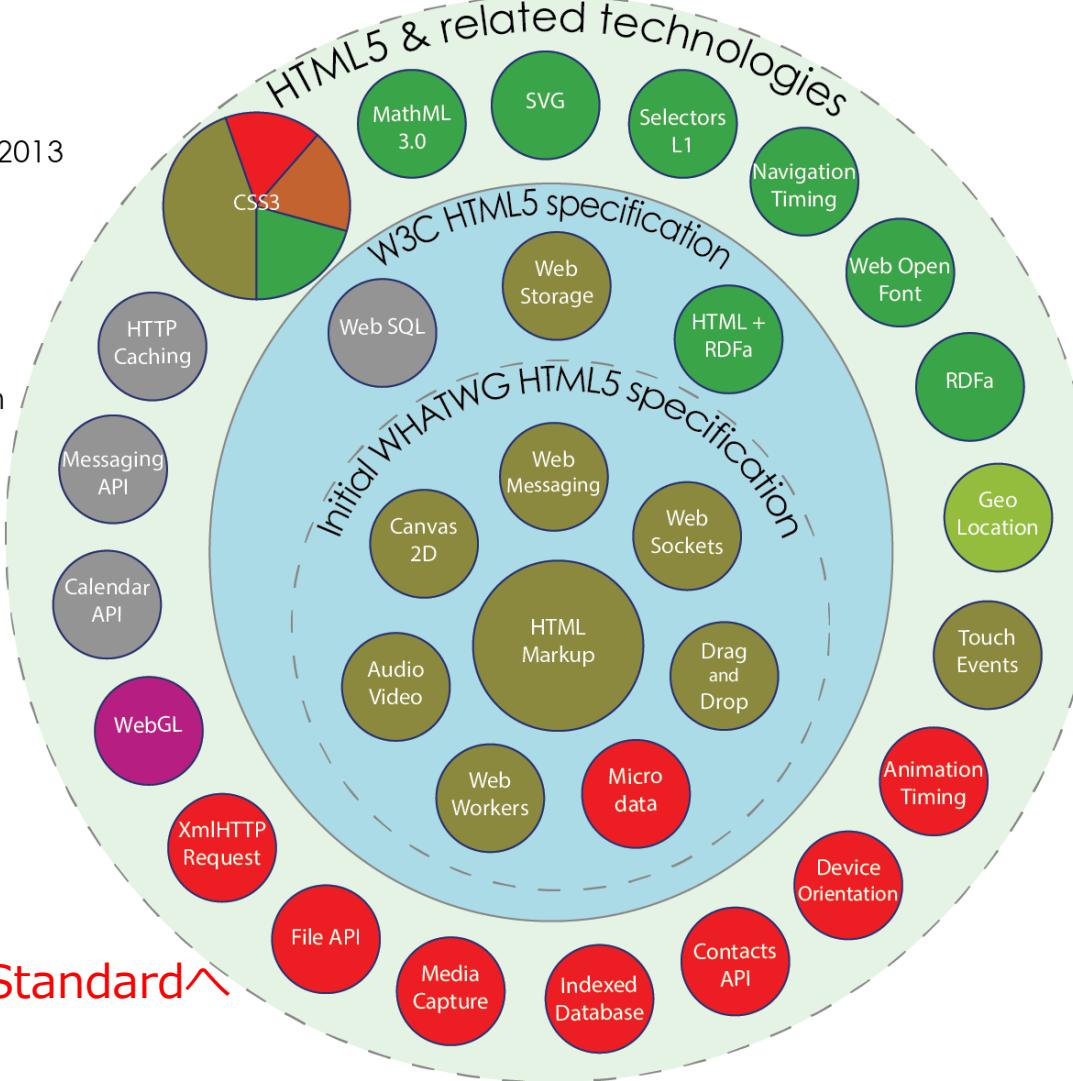
- W3C Recommendation
- Proposed Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated

2021年1月28日に廃止

WHATWGのHTML Living Standardへ

by Sergey Mavrody  BY · SA

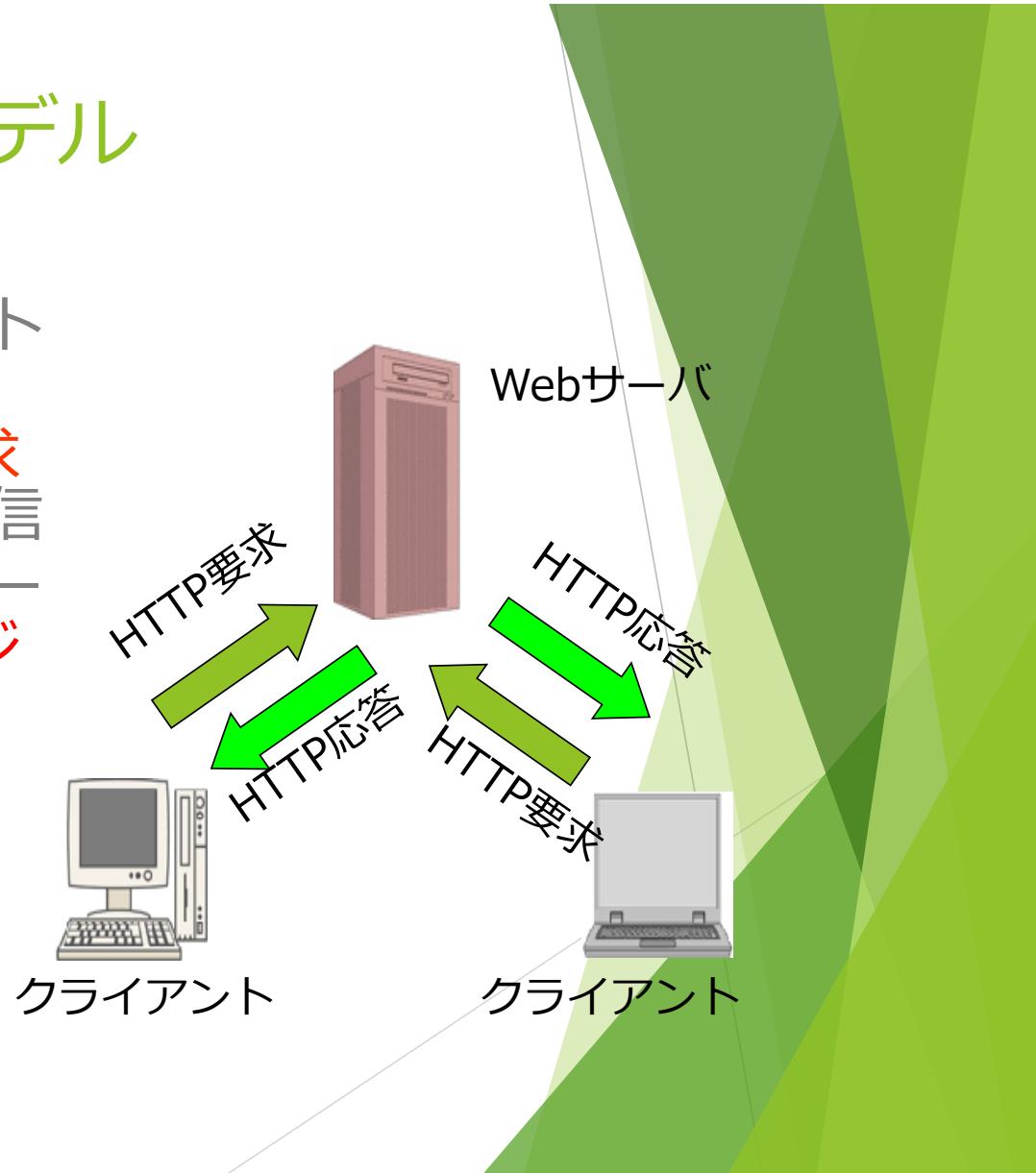
W3C(World Wide Web Consortium)が規定



クライアント・サーバモデル

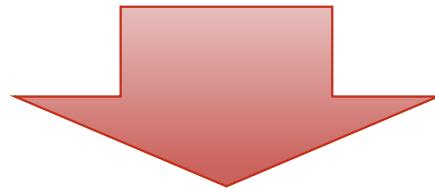
1. ユーザがあるWebページ(いくつかのオブジェクトから成る)を要求
2. クライアントがHTTP要求メッセージをサーバに送信
3. サーバは要求のあったページをHTTP応答メッセージを用いて返信

サーバは、ファイル送信の際、
クライアント情報を記録に残さない
↓
HTTPは、stateless protocol



HTTPのトランSPORT層プロトコル

HTTP : 信頼性のあるトランSPORT層
プロトコル上で提供
TCP/IPの場合はTCP
(デフォルトのポート番号は80)



HTTPは、ネットワーク内でのデータ損失や
伝送誤りを一切考慮しない

TCPコネクションの確立（1）

HTTPにおける2種類のTCPコネクション確立方法

- 非持続的(nonpersistent)コネクション

- ・・・ HTTP/1.0, HTTP/1.1

- 持続的(persistent)コネクション

- ・・・ HTTP/1.1(デフォルト)

非持続的コネクション

一本のTCPコネクションで一つのファイルのみを転送

ファイル数が多くなると、TCPコネクション確立のための処理時間や交換されるパケット数が多くなる

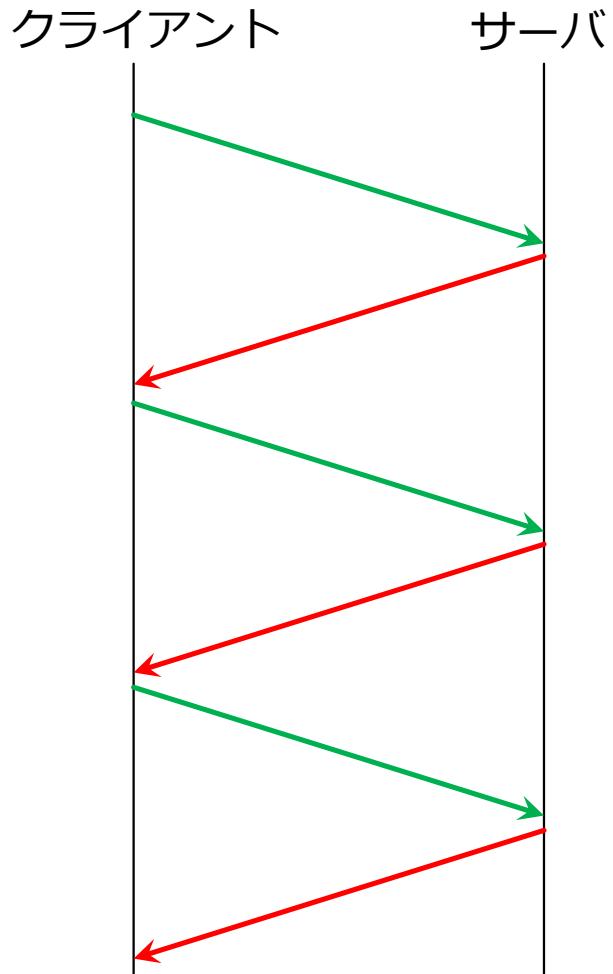
TCPコネクションの確立（2）

持続的コネクション

1個のTCPコネクションで複数のファイルを転送
♦パイプライン（piplining）有り（デフォルト）と無しの2方式

- 2回目以降の要求メッセージ送信時には、TCPコネクションが既に確立しているので、コネクション確立のための時間がなく、応答時間が短縮される
- TCPコネクション確立の回数が減少し、サーバの負担が軽くなる

HTTPパイプライン



クライアント サーバ

HTTP1.1では、クライアントからのリクエストの順序とサーバからのレスポンスの順序は同期

HTTP/2

- ▶ GoogleのSPDYプロトコルをベース
- ▶ RFC 7540 (May 2015)として標準化
- ▶ 特徴
 - ▶ HTTPヘッダの圧縮
 - ▶ サーバプッシュ型通信
 - ▶ HTTPパイプライン機能の拡張によるHoL(Head-of-Line)ブロッキングの解消
 - ▶ リクエスト順序とレスポンス順序の非同期化

HTTP/3 : HTTP over QUIC

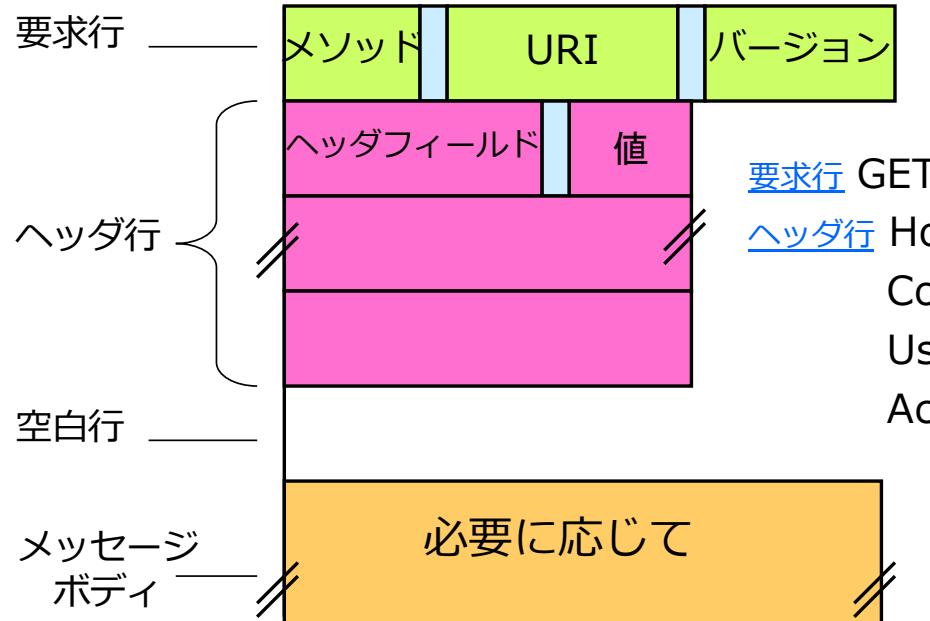
QUIC: googleが開発したトランスポート層プロトコル

要求メッセージ

クライアントがサーバに送信するメッセージ

要求メッセージの基本フォーマット

各行末には、改行符号（CRとLF）がある



(例)

要求行 GET /somedir/page.html HTTP/1.1

ヘッダ行 Host:www.someschool.edu

Connection: close

User-agent: Mozilla/4.0

Accept-Language: fr

メソッド

メソッドとは、URLと組み合わせて使用されるもので、その URL先にあるリソースに対してどのように振る舞って欲しいかを働きかけるためのものである

メソッドの種類 (HTTP/1.1)

- OPTIONS (オプションの設定)
- GET (指定したURIの情報取得)
- HEAD (ヘッダ情報取得)
- POST (指定したURIにデータ登録)
- PUT (指定したURIにデータ保存)
- DELETE (指定したURIのデータ削除)
- TRACE (リクエストメッセージを戻す)
- CONNECT (動的なPROXYなどのため)

要求ヘッダフィールド

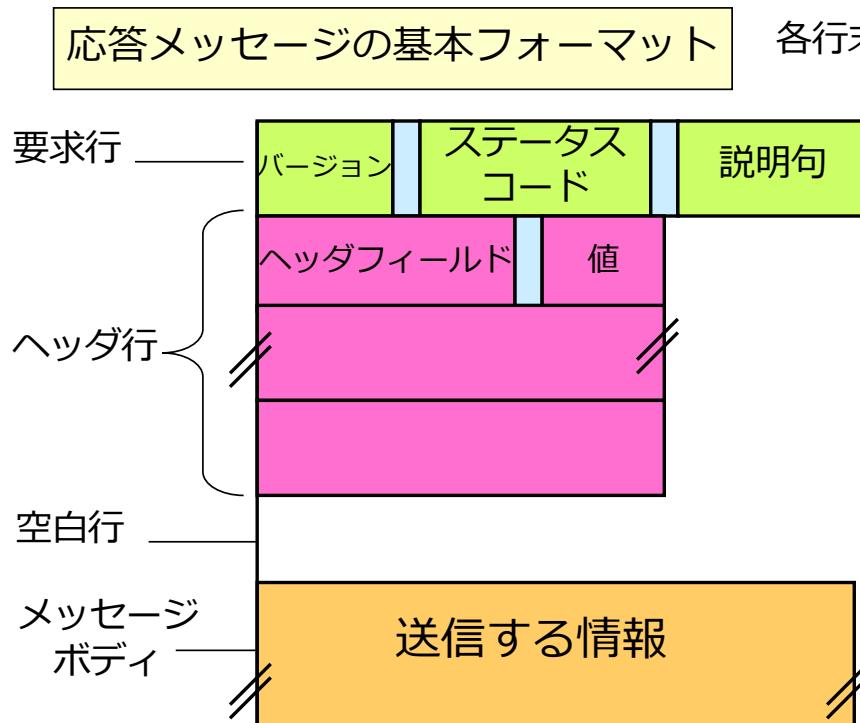
要求ヘッダフィールドにより、要求に関する追加的な情報（クライアントに関する情報など）をサーバに渡す事ができる

ヘッダの例

- Accept (応答することができるメディアタイプを指定)
- Host (相手のHostとポート番号を記述)
- User-Agent (ユーザエージェントの名前などの記述)
- Referer (参照元を記述)

応答メッセージ

サーバが、要求メッセージに対して返すメッセージ



各行末には、改行符号（CRとLF）がある

(例)

ステータス行 HTTP/1.1 200 OK

ヘッダ行

Connection: close
Date:Thu,06 Aug 2003 12:10:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Length:6821
Content-Type: text/html

(data.....)

ステータスコード

クライアントからの要求を理解し満足しようと
した結果を表す3桁の数字

ステータスコードの最初の数字は応答メッセージのクラスを定義する

- 1xx: Informational - 要求は受け入れられ、処理を続けている
- 2xx: Success - 要求は正常に受信され、理解され、受け入れられた
- 3xx: Redirection - 要求を完了するためには、
さらに動作を行わなければならない
- 4xx: Client Error - 要求は間違った構文か、満足させることのできない
ものを含んでいる
- 5xx: Server Error - サーバは要求を満足させるのに明らかに失敗した

ステータスコードの一部

- ▶ 100 Continue 繙続
- ▶ 200 OK リクエストは成功
- ▶ 300 Multiple Choices 複数の選択
- ▶ 400 Bad Request リクエストが不正
- ▶ 403 Forbidden 禁止
- ▶ 404 Not Found 未検出
- ▶ 500 Internal Server Error サーバ内部エラー
- ▶ 503 Service Unavailable サービス利用不可
- · · · ·

応答ヘッダフィールド

応答ヘッダフィールドにより、サーバは、
ステータス行に置けない応答に関する追加的な
情報を渡すことができる

ヘッダの例

- Accept-Ranges(サーバが許容可能な要求の範囲を記述)
- Age(サーバで応答が生成されてからの推定時間)
- Location(新しい接続先を記述)
- Server(要求を処理するためにサーバで用いられているソフトウェアに関する情報)

14) TCP/IP

応用例：DNS

IPアドレスとドメイン名

- ▶ **IPアドレス**

インターネット上のホストを
区別するための32ビットのアドレス

- ▶ **ドメイン名**

人間が覚えやすい名前

www.nitech.ac.jp www.apple.com

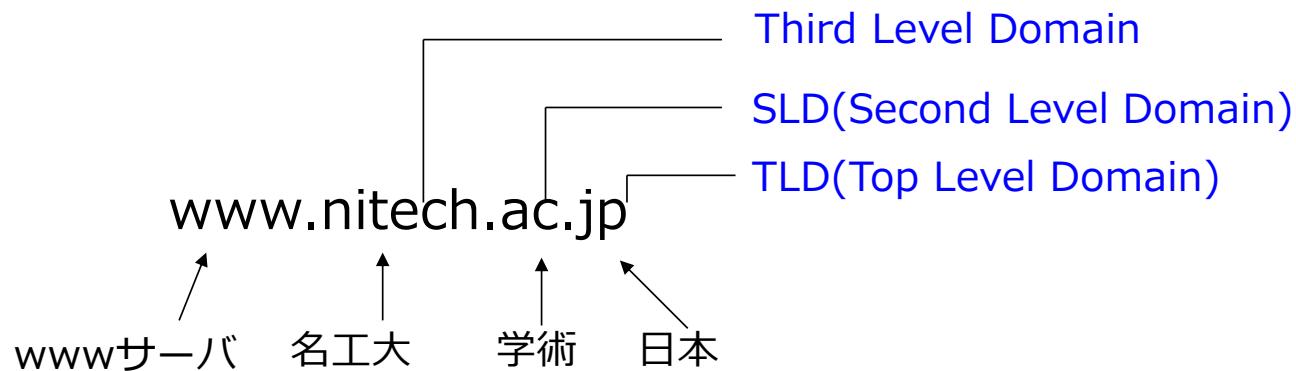
コンピュータは、数値であるIPアドレスを用いる
人は名前の方が覚えやすい

↓
ドメイン名をIPアドレスに変換するサービスが必要

DNS(Domain Name System)

ネームサーバによる階層的分散
データベース(RFC1034,1035)

ドメイン名



◆ ドメイン名は階層構造をしている

- 名前が一つであると、名前が重なる可能性がある
- 住所の原理と同じ(階層化することで一意性を保証)
- `www.nitech.ac.jp`は世界中に一つだけ

ドメイン名の階層構造

▶ TLD(Top Level Domain)

汎用ドメイン(gTLD:general TLD) : com, edu, gov, int, mil, net, org など

国別ドメイン(ccTLD:Country Code TLD) : jp, kr, de, uk, fr など

▶ SLD(Second Lecel Domain)

gTLDでは、yahoo.comやcnn.comなど

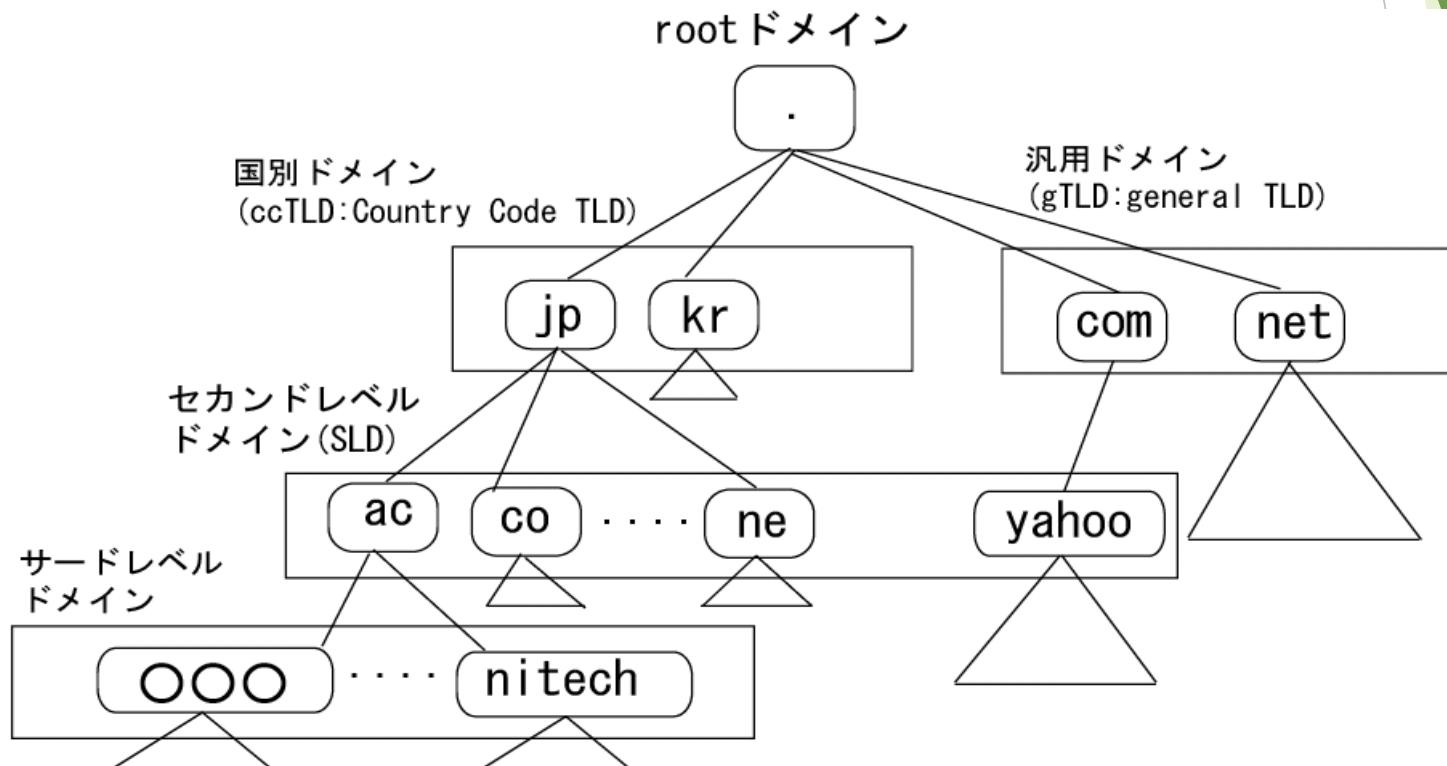
ccTLDでは、ac.jp, ad.jp, co.jp, ne.jpなど

▶ Third Level Domain

nitech.ac.jpやntt.co.jpなど

ドメインツリー

階層構造をした名前空間



ドメイン名の管理

- ▶ TLD(gTLDとccTLD)は、 ICANN(Internet Corporation for Assigned Names and Numbers)が管理
- ▶ jpドメインやcomドメイン以下の管理は委任
- ▶ jpドメインはJPRS(日本レジストリサービス)が管理
- ▶ サードレベルドメインまでをJPRSが管理し、それ以下は各組織が管理
- ▶ nitech.ac.jp以下のドメインは、名工大が管理

DNSの概要

DNSプロトコルは、ポート番号 53 で UDP を利用

▶ DNS の機能

ドメイン名から IP アドレスへと
IP アドレスからドメイン名への変換

ドメイン名
en.web.nitech.ac.jp



IPアドレス
133.68.98.44

DNSサーバ

DNSの詳細（1）

メールサーバのエイリアス名登録

cse宛のメールを受け取るサーバは、 mailgw.mains.nitech.ac.jp

メールを送る時はsomeone@mail.hoge.nitech.ac.jp

@mail.hoge.nitech.ac.jpからメールサーバ名を知る方法

- ▶ DNSのMX(Mail eXchange)レコード(Name=メールサーバのエイリアス, Value=対応する公式ホスト名)に記述
- ▶ mail.hoge.nitech.ac.jpのMXレコードに,
mailgw.mains.nitech.ac.jpを記述
- ▶ @mail.hoge.nitech.ac.jpの時には、 mailgw.mains.nitech.ac.jp
に送ればよい

[参考] DNSのResource Record=(Name, Value, Type, TTL)
Type=A, NS, CNAME, **MX**

DNSの詳細（2）

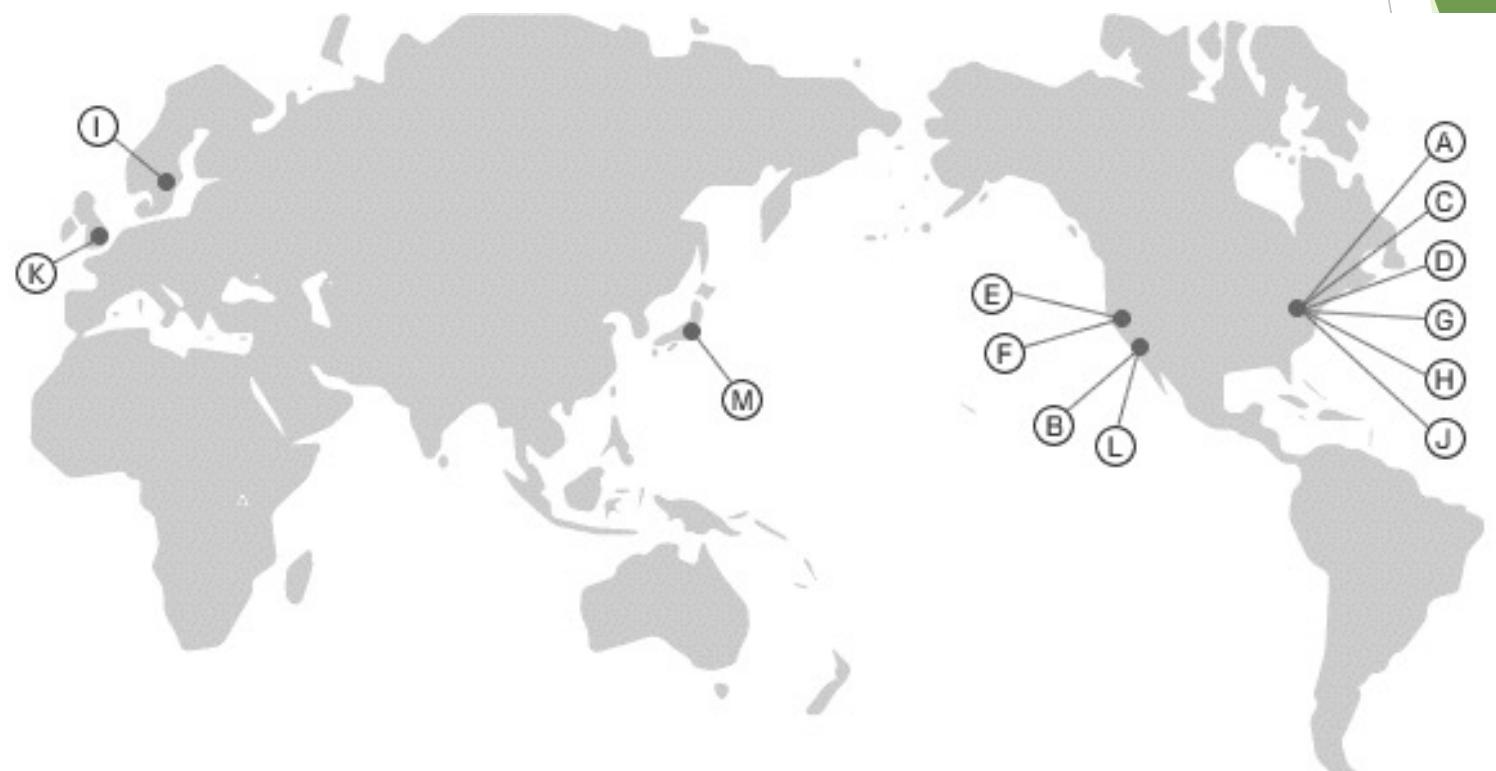
- ▶ rootネームサーバ
 - ▶ 全世界に13台(A~M)
 - ▶ rootネームサーバはセカンドネームサーバのIPアドレスを知っている
- ▶ セカンド/サードネームサーバ
 - ▶ 日本では6台
 - ▶ 日本ではセカンド/サードネームサーバは同一

各ネームサーバは、下位(子)のネームサーバのIPアドレスとrootネームサーバのIPアドレスを知っている



rootネームサーバからたどることで世界中すべてのネームサーバにたどりつける

ルートサーバの配置図



ルートサーバー一覧

| ルート サーバ | 運用組織 | 所在地 |
|------------|--|------------|
| A | VeriSign Naming and Directory Services | 米国バージニア州 |
| B | 南カリフォルニア大学情報科学研究所(ISI) | 米国カリフォルニア州 |
| C | Cogent Communications | 米国バージニア州 |
| D | メリーランド大学 | 米国メリーランド州 |
| E | 米航空宇宙局(NASA)エイムズ研究所 | 米国カリフォルニア州 |
| F | Internet Systems Consortium, Inc.(ISC) | 米国カリフォルニア州 |
| G | 米国防総省ネットワークインフォメーションセンター | 米国バージニア州 |
| H | 米陸軍研究所 | 米国メリーランド州 |
| I | Autonomica | ストックホルム |
| J | VeriSign Naming and Directory Services | 米国バージニア州 |
| K | Reseaux IP Europeens-Network Coordination Centre(RIPE NCC) | ロンドン |
| L | Internet Corporation for Assigned Names and Numbers(ICANN) | 米国カリフォルニア州 |
| M | WIDE Project | 東京 |