TCP/IPチュートリアル

基本のTCP+UDP+ICMP/IP

株式会社シーイーシー
高田　寛

なぜこんなチュートリアルがあるのか

- 普通、TCP/IPって言うよね？
  - でもその中にはUDPもICMPも含んじゃってるよね？
  - じゃあ、UDPやICMPはオマケなの？
- オマケじゃないです
  - なければインターネット、動かないです
  - よく知らないで使っていると、障害やセキュリティ事故の要因に成り得ます
- 「HTTPは使ってるけどTCP/IPは使ってません」
  - はぁ？
  - バカなの？死ぬの？

というようなことにならないために

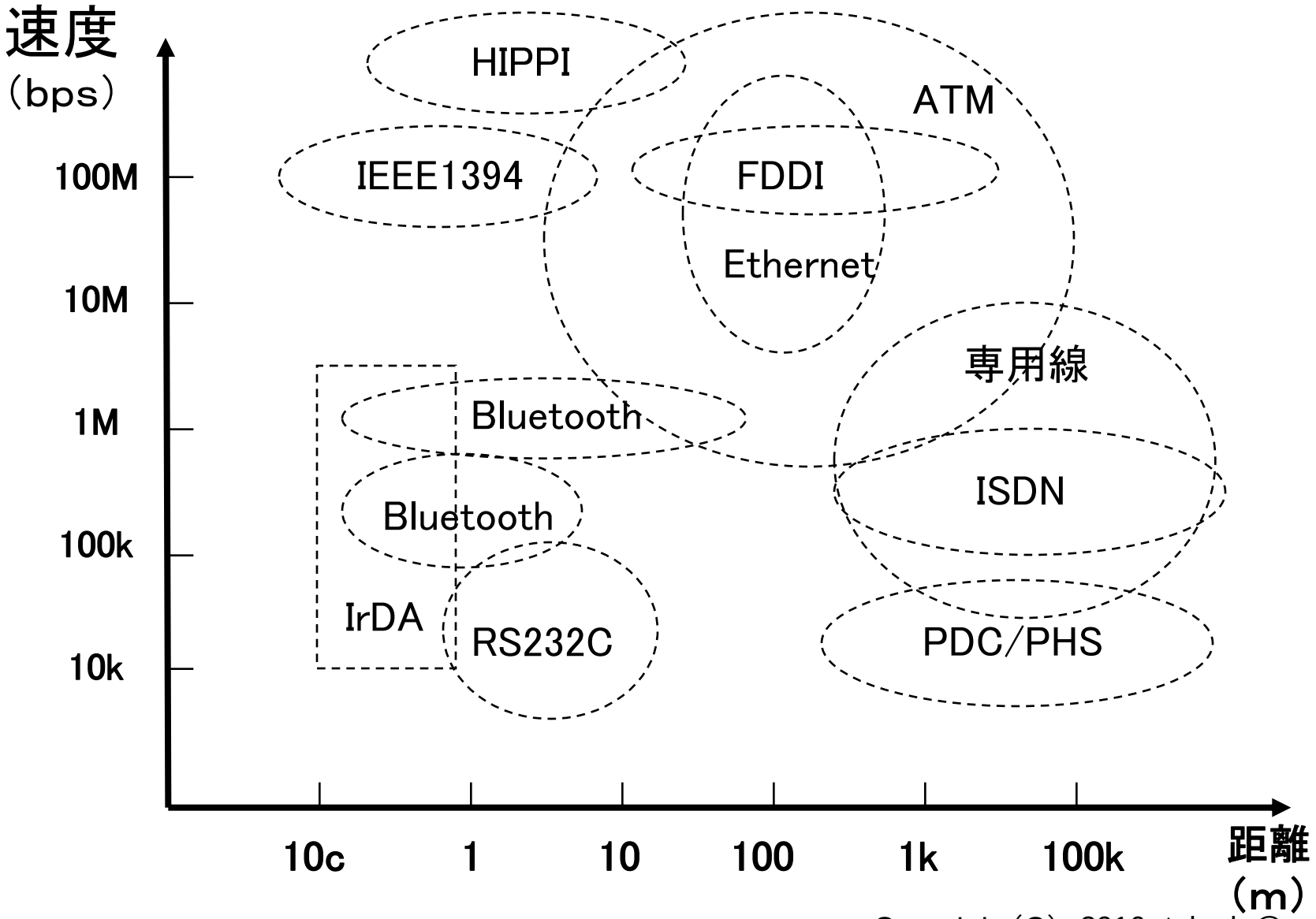
インターネットの標準

- 世界のインターネット技術者がオープンな場で検討
  - IETF ー Internet Engineering Task Force
  - IAB ー Internet Architecture Board
  - ISOC ー Internet Society インターネット学会
- 技術資料はRFCとして公開
  - Internet Draft
  - RFC ー Request for Comments
    - もはやRFCは7000番台
      - 7736まで（2016/01/14現在）
    - しかしTCP/IPに関係するものは変化が小さい
- 電話の世界のように、各国政府や通信事業者が決めているのではない

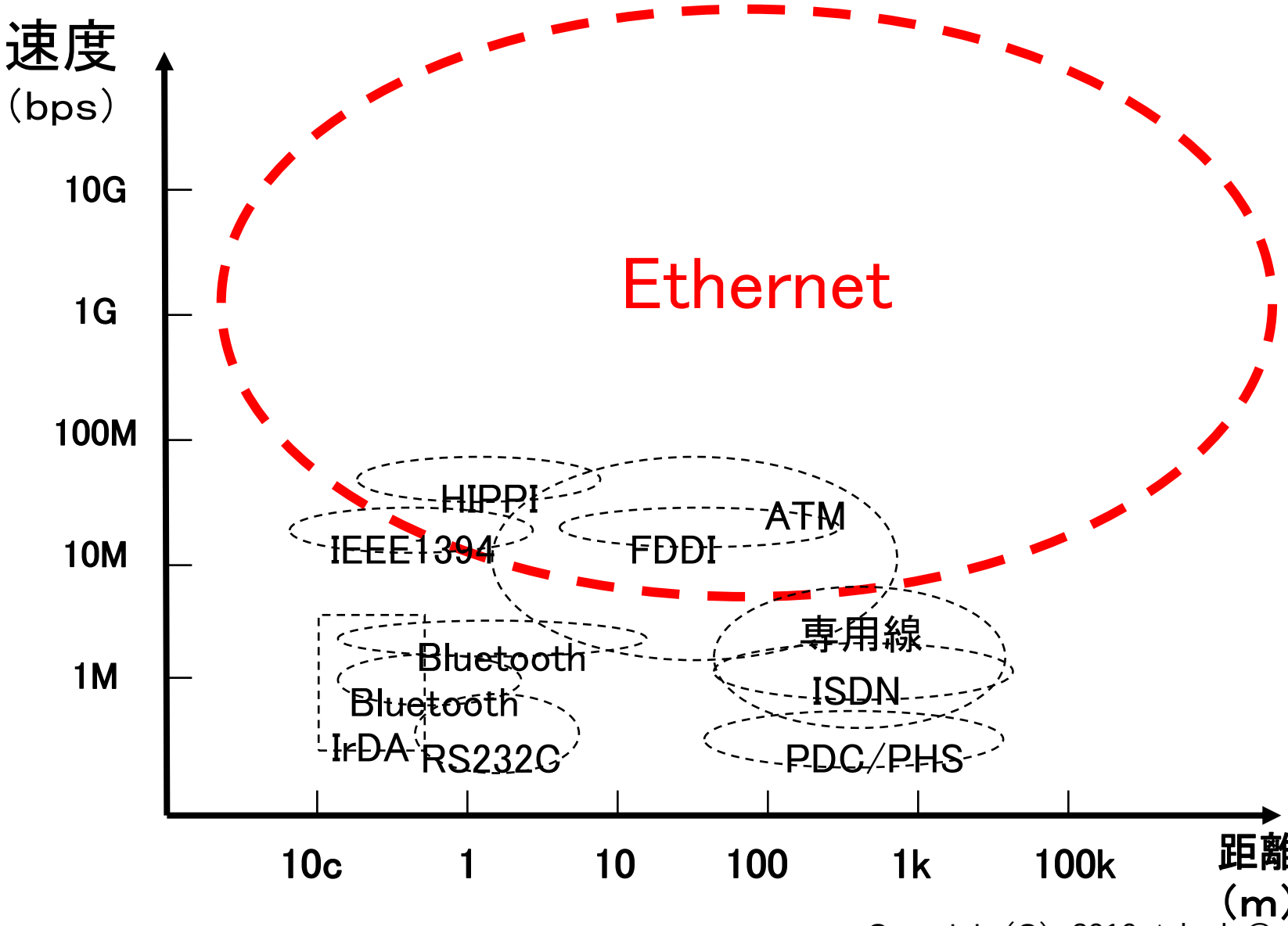
インターネットは

- 複数のネットワークを相互に接続
  - 論理的に一つのネットワークに見せる技術
  - 一つのネットワークアーキテクチャでは構成できないネットワークを構成
    - 地球規模での広域性
    - 低速から高速まで
- IPとIPアドレス
  - IPアドレスという論理識別子によって，インターネット上のノードを識別
  - ユーザはどう繋がっているのかを意識する必要はない

20年前のネットワークアーキテクチャ

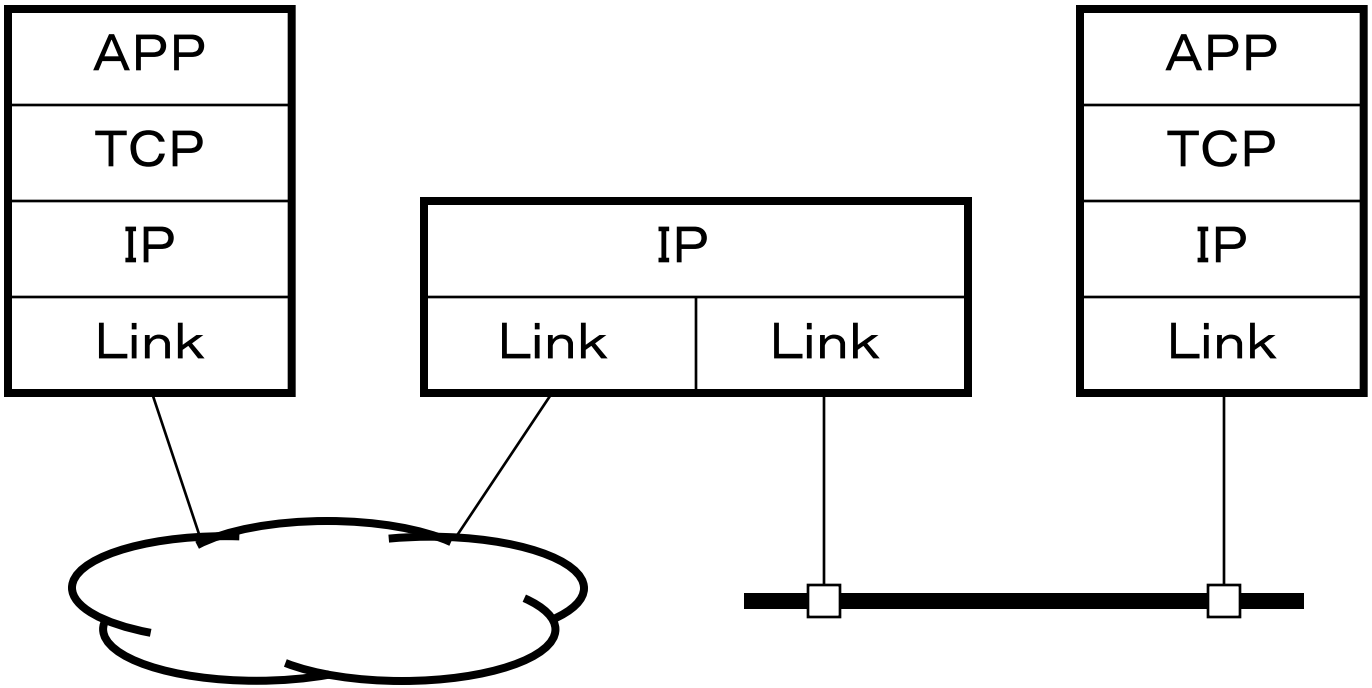


今のネットワークアーキテクチャ



インターネットプロトコル（IP）

- 異なったメディアを統合して論理的なネットワークに



IP: Internet Protocol

機能

- RFC 791 で定義
  - RFCは更新されることが多いが現役でRFC791
- データグラムを他のホストに配送する
  - 信頼性を保障しない
    - データグラムが紛失するかもしれない
    - データグラムの順番が保証されない
    - データグラムが重複するかもしれない
- フラグメンテーション
- 経路制御

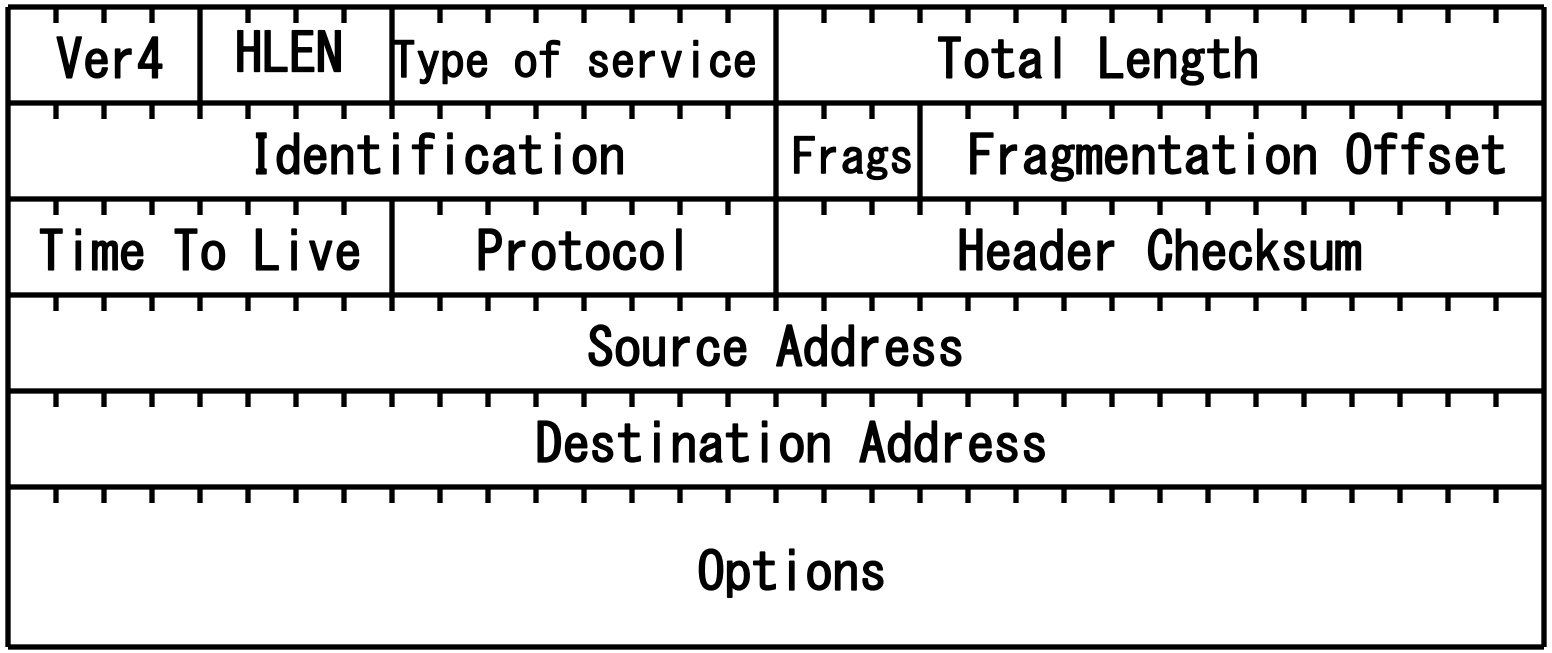
機能

- データ配送の基本単位
- ネットワークの抽象化
  - 物理層，リンク層の違いを隠蔽する
  - 仮想的に単一のネットワークに見せる

概要

- IPデータグラムを他のホストに配送
  - IPアドレスで相手を指定
- データをデータグラムに分割して配送
- コネクションレス
  - 誤り訂正，紛失，到着順序などの処理は上位層で行う
- 経路制御とアドレス
  - さまざまなネットワークを経由
  - ルータによる中継

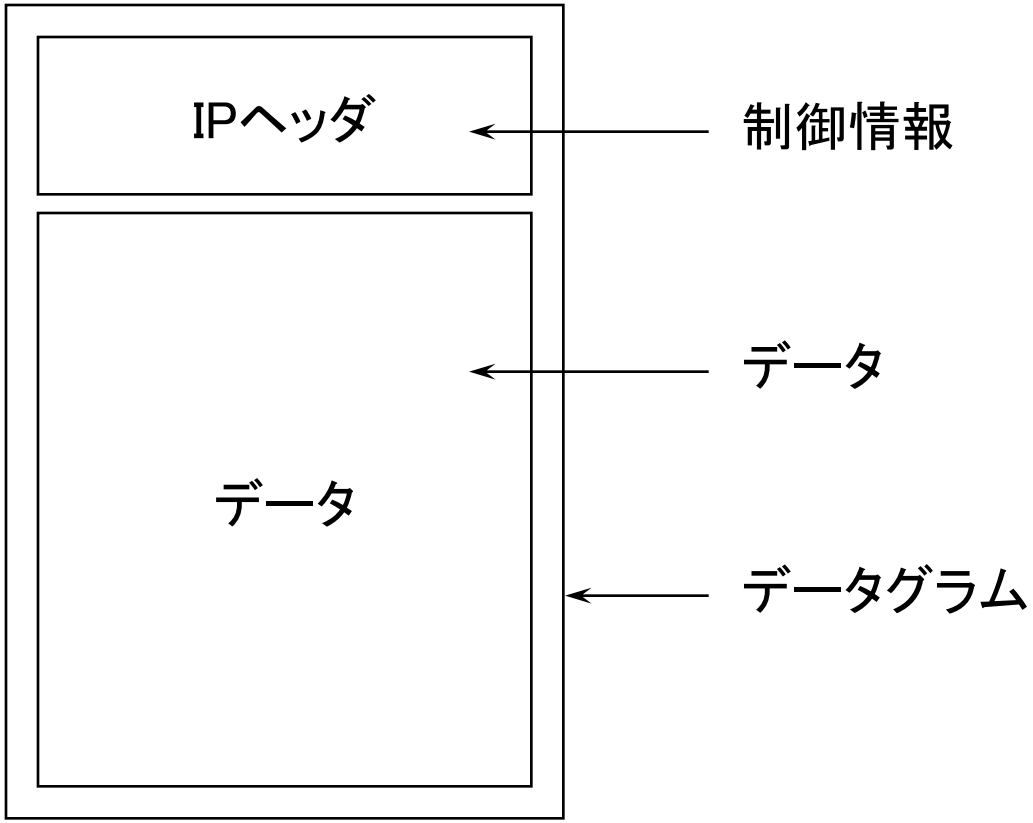
IPv4ヘッダフォーマット



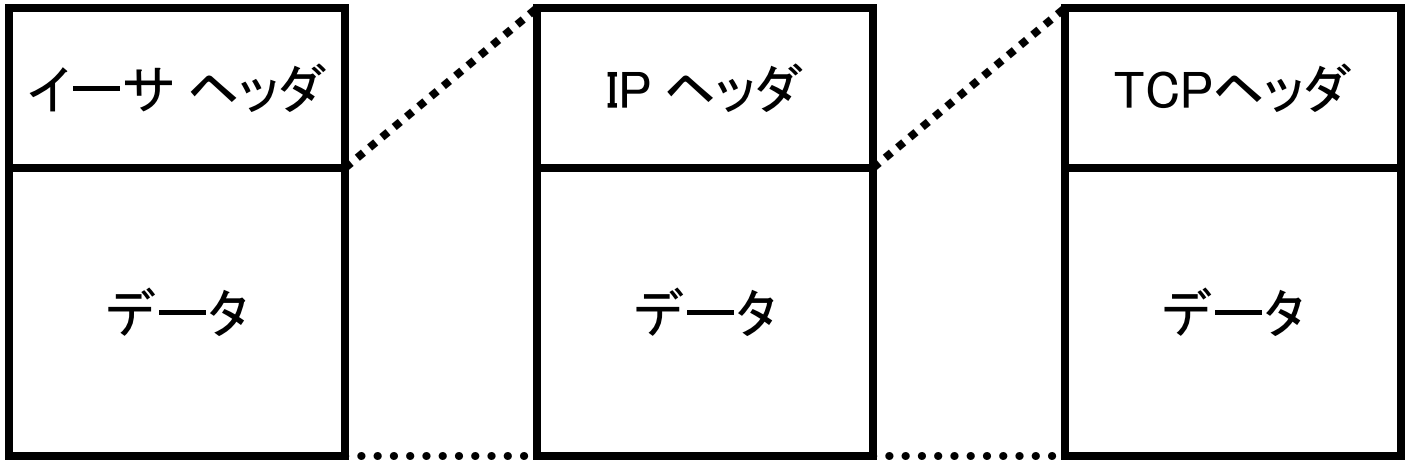
IPデータグラム

- 基本転送単位

  = ヘッダ + データ

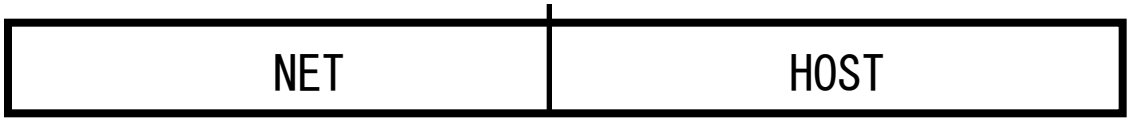


カプセル化



IPアドレス

- 固定長の論理アドレス
  - IPv4: 32ビット(IPv6 128bit)
  - ホストのネットワークインターフェースを識別
- 二つの部分
  - ネットワーク部 ― 経路指定，ネットワークを識別
  - ホスト部 ― ネットワーク内のホストを識別



- どこで区切られているかはユーザが認識する必要はない

TTL

- Time To Live
  - IP パケットの生存時間(単位: 秒　←仕様)
- (事故や設定ミスで)IP パケットがたらい回しにされても，そのうち消える
- ルータを経由すると TTL を 1 減らす(←実装)
- TTL が0 になったらルータは，
  - そのパケットを破棄する
  - 送信元にエラーメッセージを返す → ICMP　Time Exceeded

フラグメンテーション

- リンク層によってパケットの最大長が決まっている
- 最大転送単位: MTU (Maximum Transfer Unit)
- MTU より大きな IP パケットの中継
  - パケットを MTU に合わせて分割
- フラグメンテーションは，終点ノードで元のIPデータグラムに戻される
- PMTUd(Path MTU Discovery)
  - 予め、終点までの最小MTUを調べ、それ以下のサイズで送信する（途中経路でのフラグメンテーションを防ぐ）
  - こっちが主流（IPv6では必須）
  - ICMP Too big　が通らないと動作しない（後述）

経路制御

- IPデータグラムの中継経路の制御
  - 終点アドレスによる配送経路決定
    - ルータでの経路表の管理
    - hop by hop
  - 発信元が配送経路を指定
    - source routing
    - 通常使えません
- 経路はアドレスとマスク長により管理
- 経路表が大きくならないためのアドレス割り当て
  - CIDR（Classless InterDomain Routing）

経路表

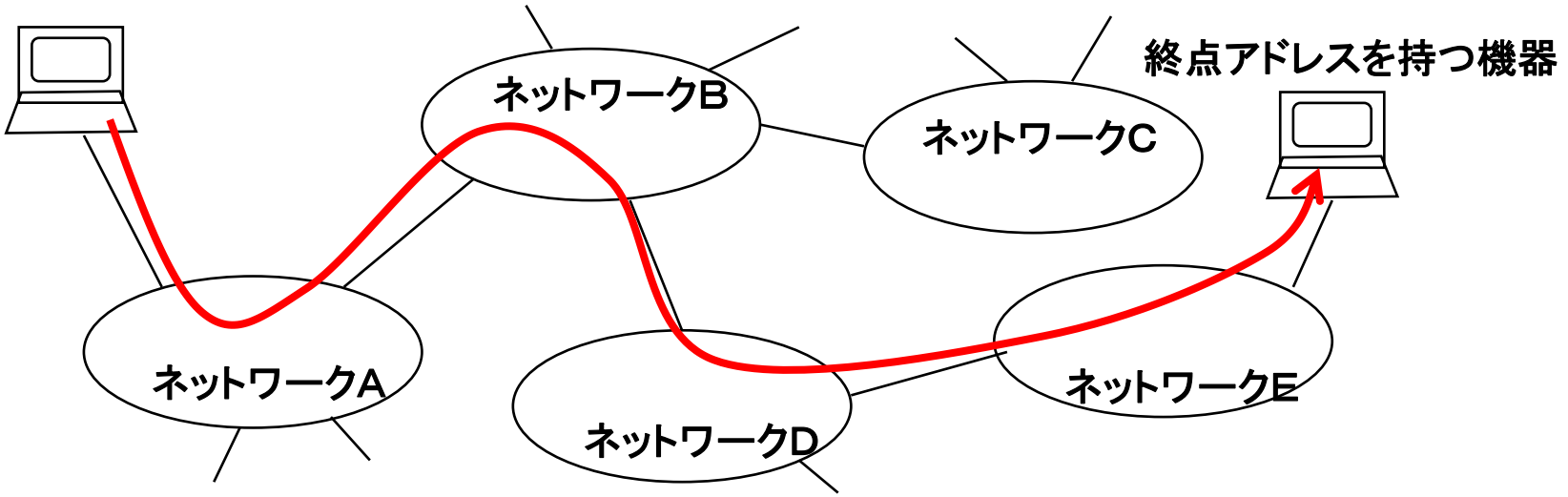
- インターネット上のノードはそれぞれ経路表を持ち，IPデータグラム中の終点アドレスによって，どのインターフェイスに転送するかを決めている

Destination	Gateway	Flags	Interface
127.0.0.1	127.0.0.1	UH	lo0
192.168.3.179	192.168.3.11	UGH	ed0
133.2.0.0	133.2.2.253	UG	ed1
202.2.8.16	202.2.8.26	UG	vx0
172.16.0.0	133.2.2.3	UG	ed1
202.2.8.24	202.2.8.27	U	vx0
131.41.0.0	133.2.2.253	UG	ed1
203.18.98.0	202.3.8.26	UG	vx0
133.2.74.0	133.2.2.253	UG	ed1
133.2.2.0	133.2.2.7	U	ed1
133.3.0.0	133.2.2.253	UG	ed1
192.168.3.0	192.168.3.2	U	ed0
202.2.6.0	202.3.8.26	UG	vx0

IPアドレスと経路制御

- IPパケットに含まれるあて先アドレス（終点アドレス）によって経路が決まり，正しくあて先に配送される．

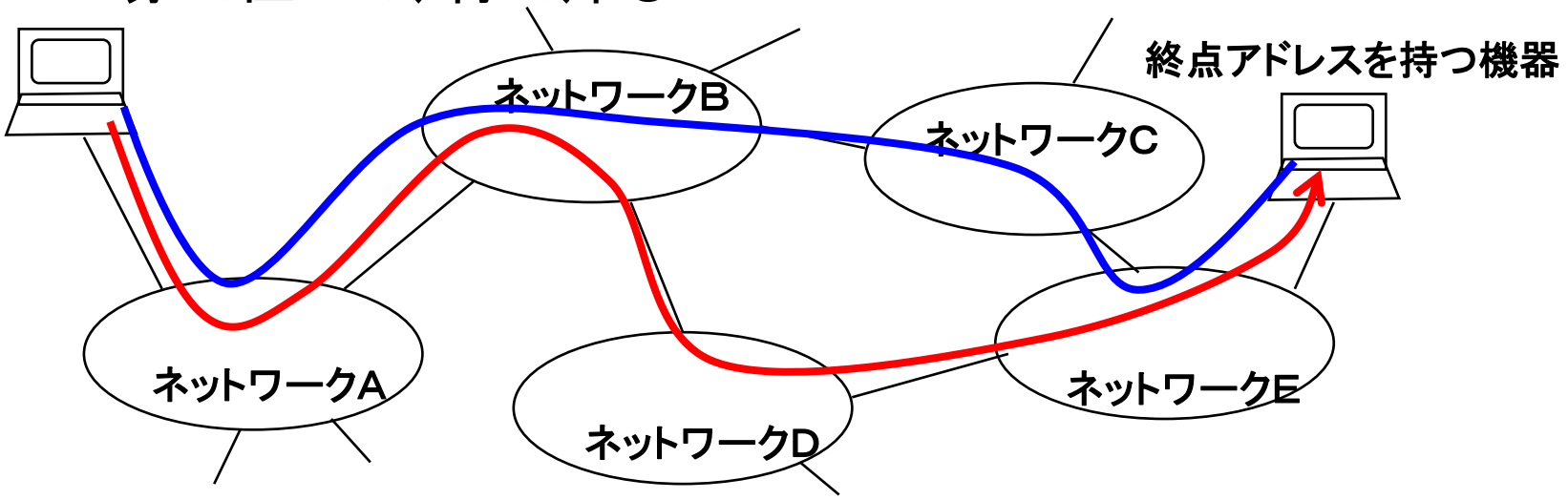
始点アドレス	終点アドレス	データ



- どこに転送するのかは、経路上のそれぞれの機器で経路表に従って判断される（Hop by Hop）

往きと帰り

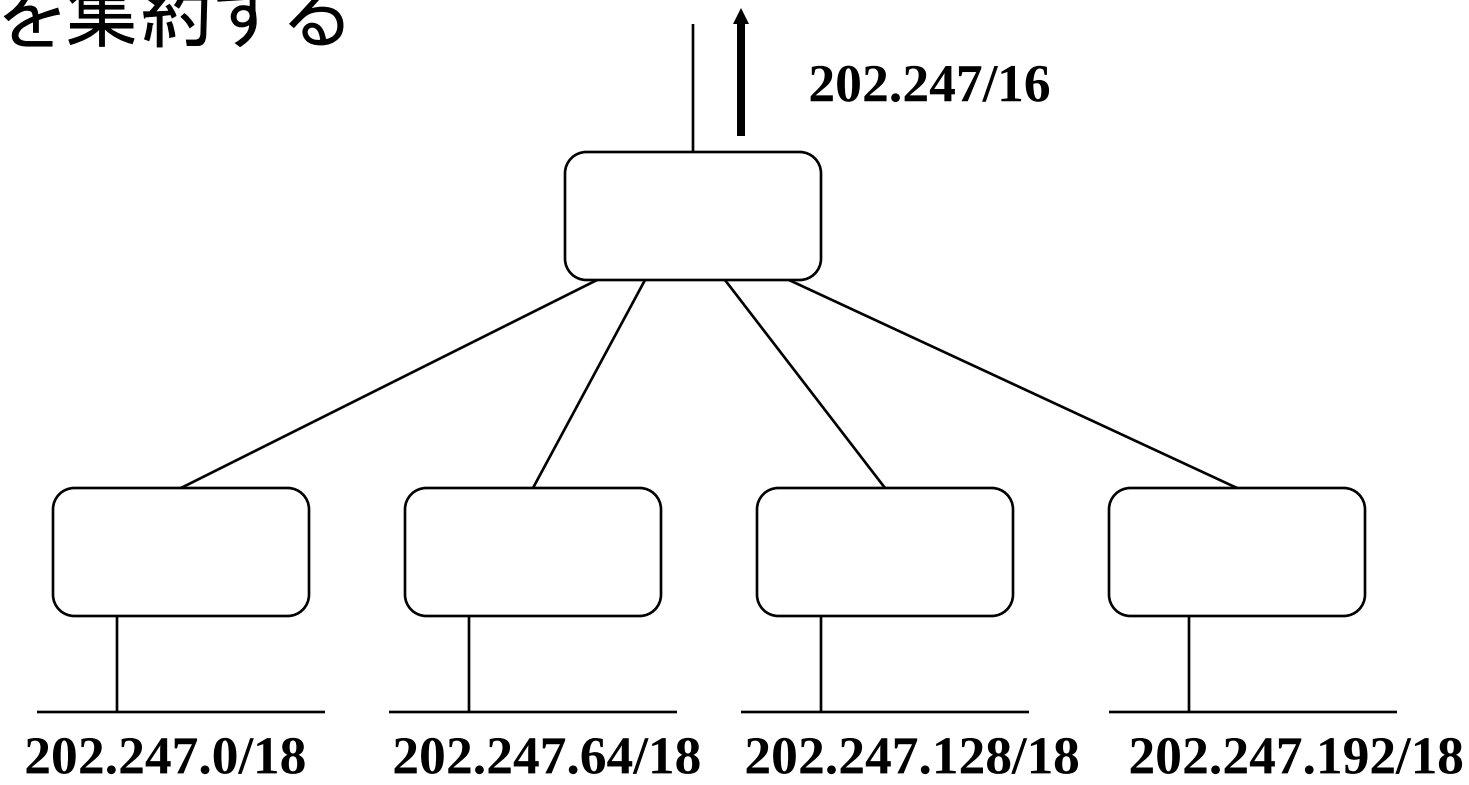
- 複数の経路がある場合
  - 往きの経路と帰りの経路は違う場合がある
  - 赤で往って、青で帰る



- tracerouteでは往きの経路しかわからない
  - アジア方面で、往きは近いけど帰りは北米経由とか
  - よくハマります

アドレスの階層構造

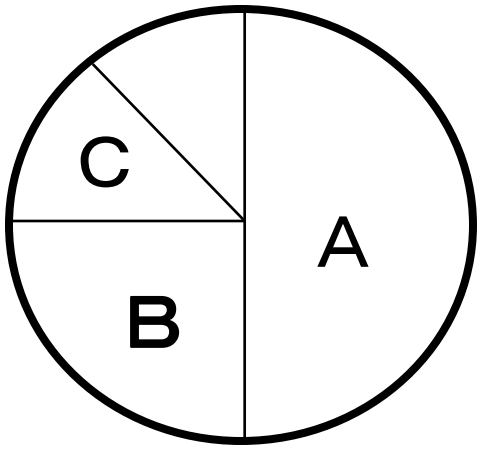
- アドレスを階層的に割り振ることによって経路情報を集約する

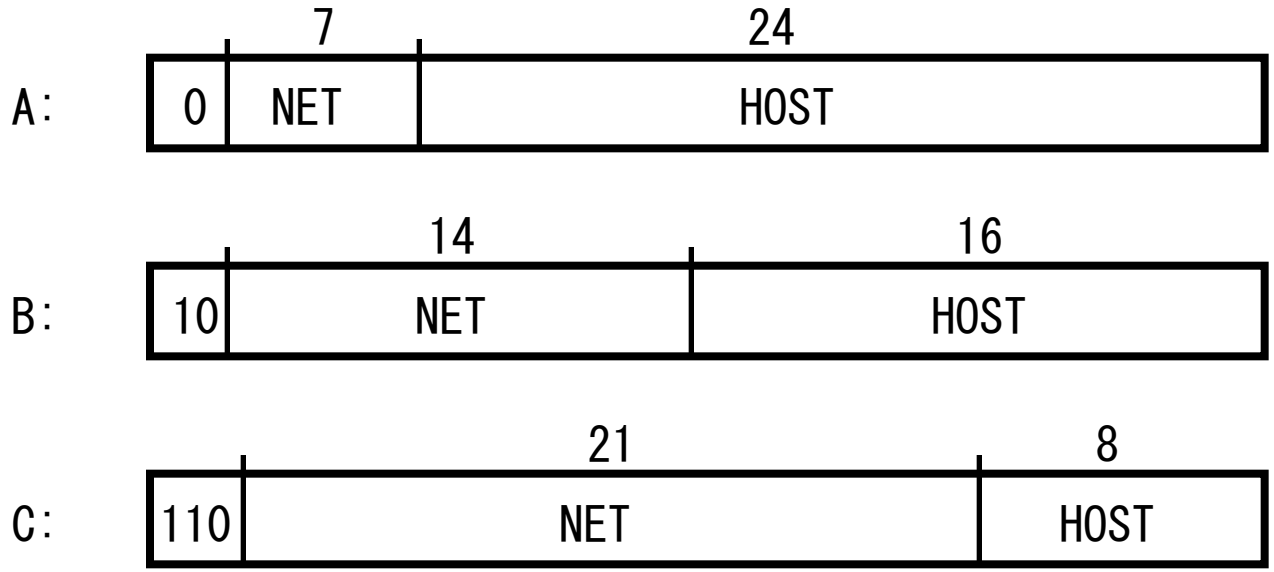


- 地域や国，プロバイダにアドレスブロックを割り振り，下位ネットワークの経路情報を１つにまとめる

アドレスのクラス

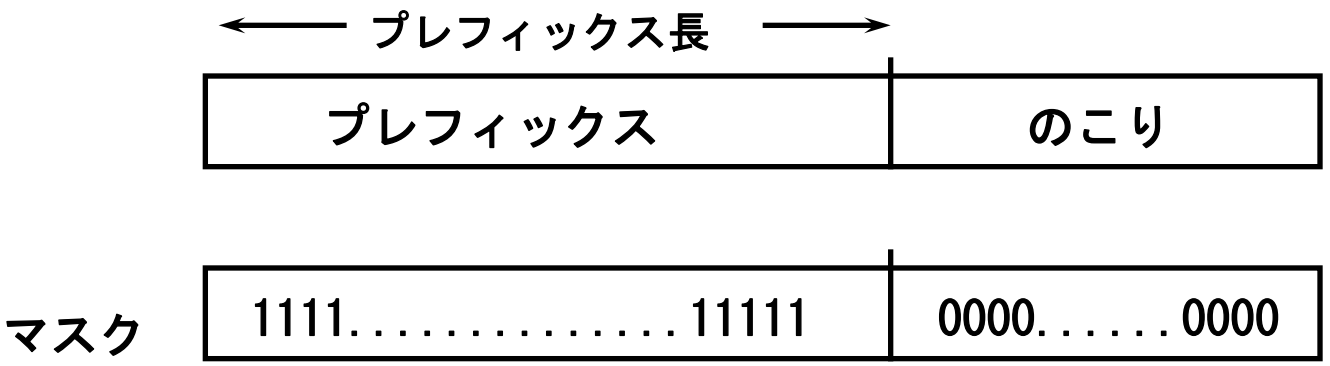
- 最初のビットでクラスを指示
- クラスフルアドレスとも
  - その非効率さからクラスレスへ





クラスレスのアドレス体系

- ネットワーク部は可変



- プレフィックス長によるネットワーク番号表記

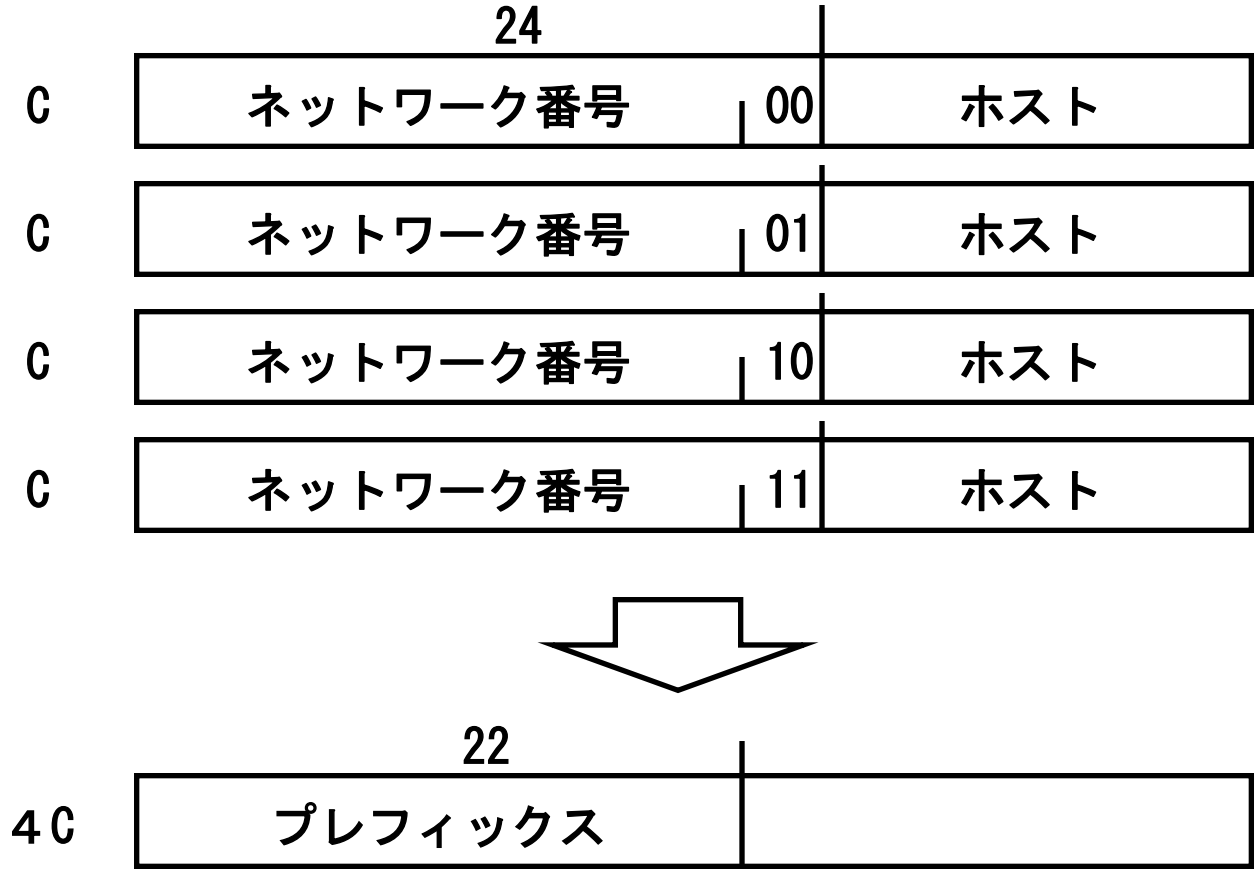
例）133.201.2/24

プレフィックス長

ネットワーク番号

アグリケーション(集約)

- 連続したネットワークのブロック化



アグリゲーションの特徴と問題点

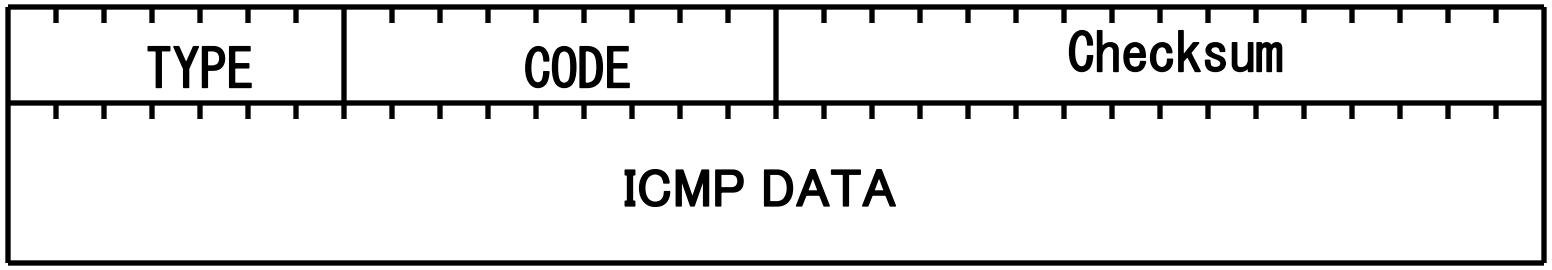
- アドレス空間を広くして，経路情報の増加を押える
  - 経路爆発問題：
    - １年ちょっと前に50万経路を超えました
    - 57万経路ぐらいあります(2016/01)
    - 51万２千経路問題
- ネットワークトポロジに対応して割り当て
  - 上流のトポロジの変更でアドレスを変更する必要
    - IPアドレス直書きするのやめようね
  - ISPの変更でも

ICMP: Internet Control Message Protocol

機能

- IP データグラム配送のエラー報告と制御
- RFC 777, 792 で定義
- 配送時のエラーを発見
- 網状態の問い合わせ

ICMPフォーマット



配送方式

- IPデータグラムのデータ
- データが紛失する可能性あり
- ICMP パケットがエラーの原因になったら?
  - エラーの通知はしない

主なメッセージの種類

- エコー要求とエコー応答（echo，　echo reply）
- 終点到達不可能報告 (host, net, port, protocol)
- 始点抑制 (source quench)
- 経路変更要求（redirect）
- 時間経過済み（time exceed）
- データグラムが大きすぎる（Too big）

ICMPをフィルタすると

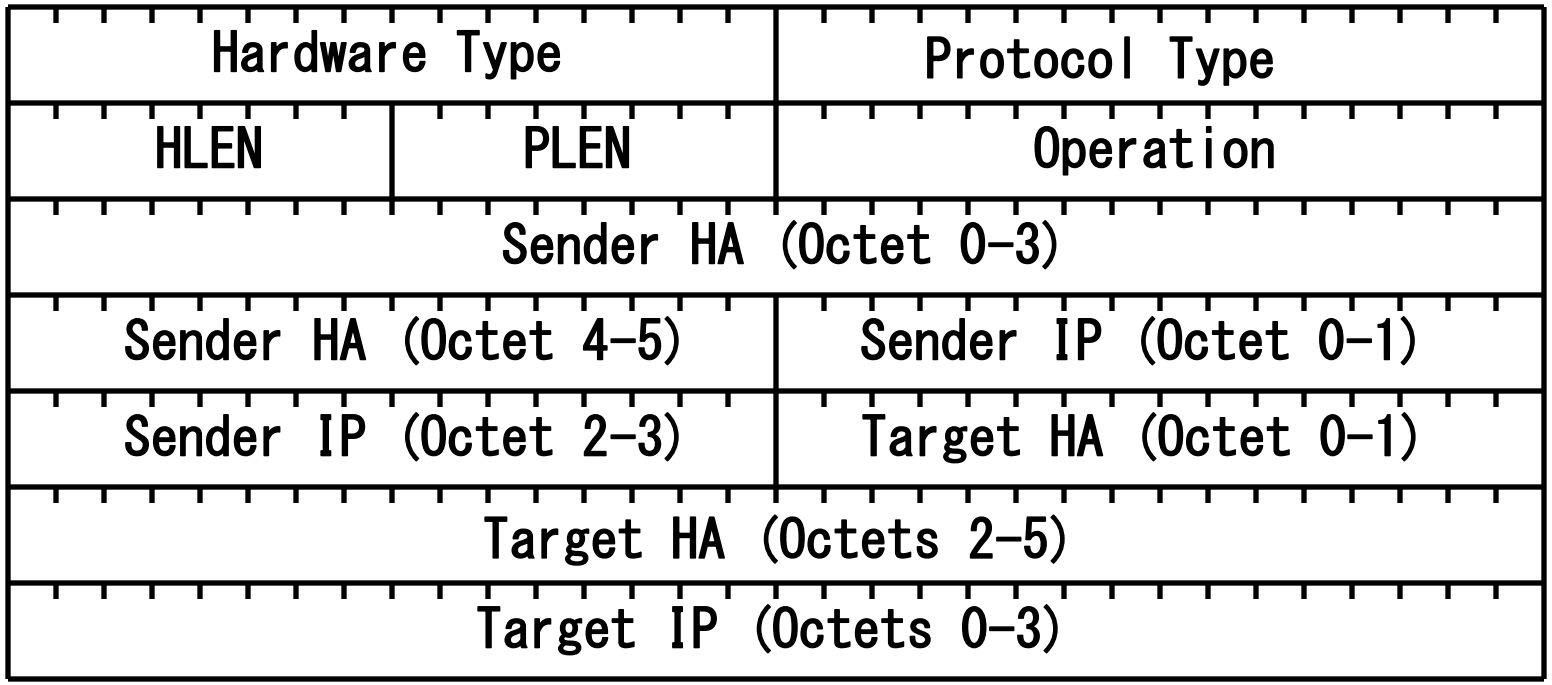
- FW等でICMPをフィルタすることはよくあると思いますが
  - Too big
    - フィルタしてしまうとPath MTU Discoveryが動作しない
    - LinkのMTU(Ethernetでは1500オクテットが標準)で送られたデータグラムが失われる
    - Too bigが返れば、小さく千切って送り直せるが、フィルタされているとできない
  - Time Exceed
    - フィルタしてしまうとtracerouteが動かない
    - トラブルシューティングの時は困ります

ARP: Address Resolution Protocol

機能

- TCP/IPとは直接関係ありませんが
  - IPアドレスから物理アドレスへの問い合わせを行うプロトコル
  - RFC 826 で定義

パケット形式



物理層アドレス解決

- 実際の通信では，リンク層の識別子を使う
  - イーサネットアドレス
  - ATM アドレス
  - FDDI アドレス
- IPアドレスからリンク層識別子への変換機構が必要
  - それぞれのインタフェイスモジュールが対応

UDP: User Datagram Protocol

機能

- 信頼性を保証しないデータグラム配送
  - = IPパケット+ポート番号+チェックサム
- RFC 768 で定義
- TCPと比較して高速
  - コネクションを張る動作が必要ない
  - 最近になってオンラインゲームなどに需要が高まる
  - でも信頼性がまったく確保されていない・・・

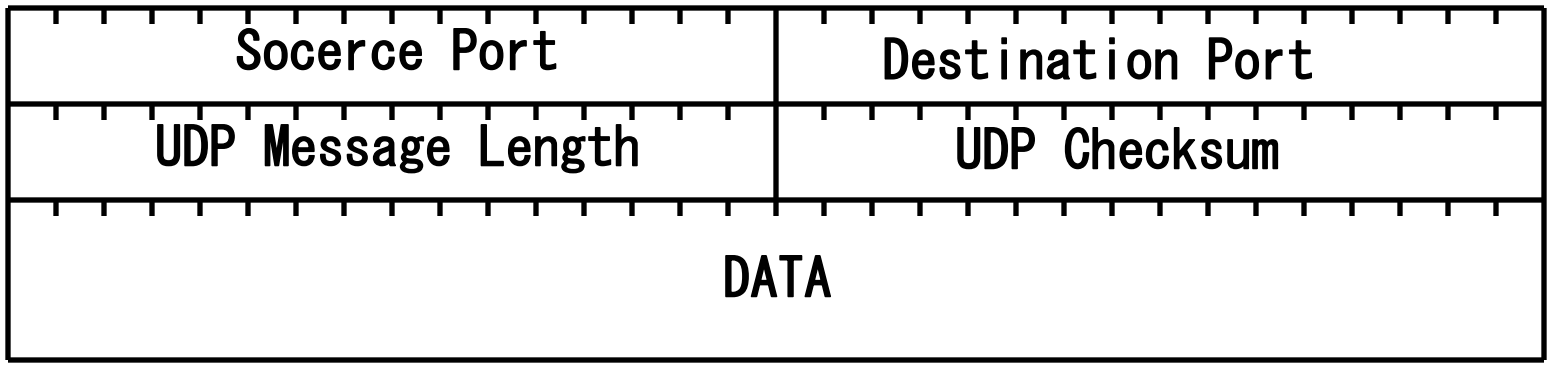
目的

- ユーザ(プログラマ) が利用できるデータグラム通信
  - NFS, TFTP, DHCP, DNS, ...

概要

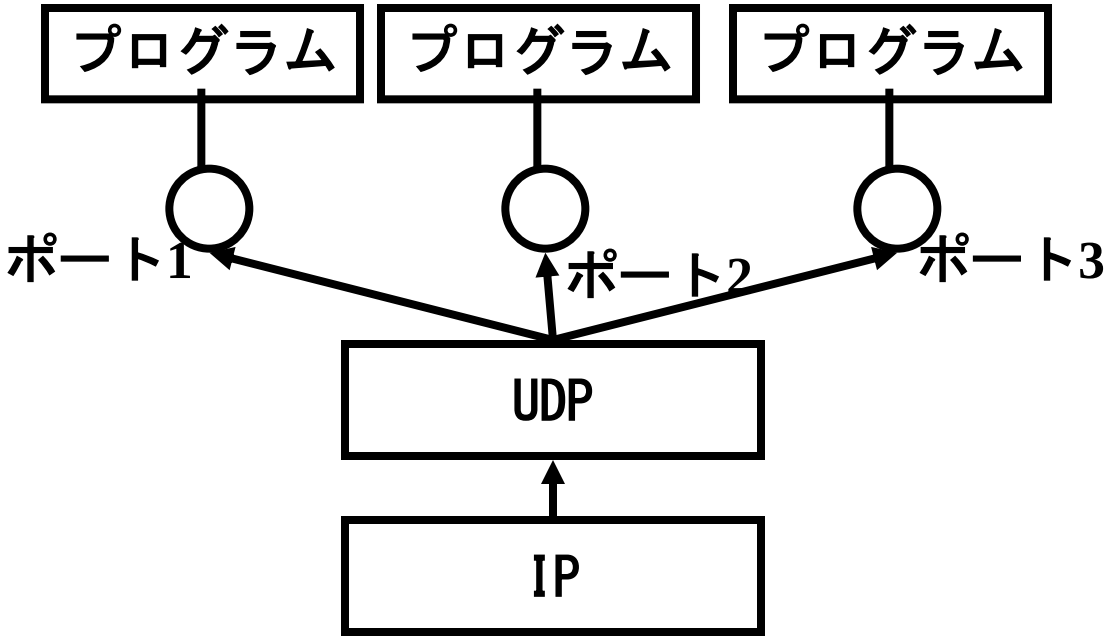
- コネクションレス型の通信形態
- ホスト間のオーバヘッドの少ないデータ転送を提供
  - コネクション開設・解放の手続きが不要
  - データの完全性（順序の保証、エラーパケットの再送など）を保証せず → 上位層で解決
- 状態の管理はアプリケーション任せ
- ポート番号によりサービスを選択

パケット形式



ポート

- 終点のサービスを識別するのに利用
  - IPアドレスは終点ホストだけを識別



代表的なサービス

- domain(DNS)　53/UDP
- DHCP　67, 68/UDP
- SNMP　161/UDP
- NFS　2049/UDP

TCP: Transmission Control Protocol

機能

- 信頼性のあるデータ配送
  - ストリーム型
  - バーチャルサーキット(コネクション型)
  - バッファ付き転送
  - 全二重

目的

- ユーザが下位プロトコルを意識せずに通信を行う

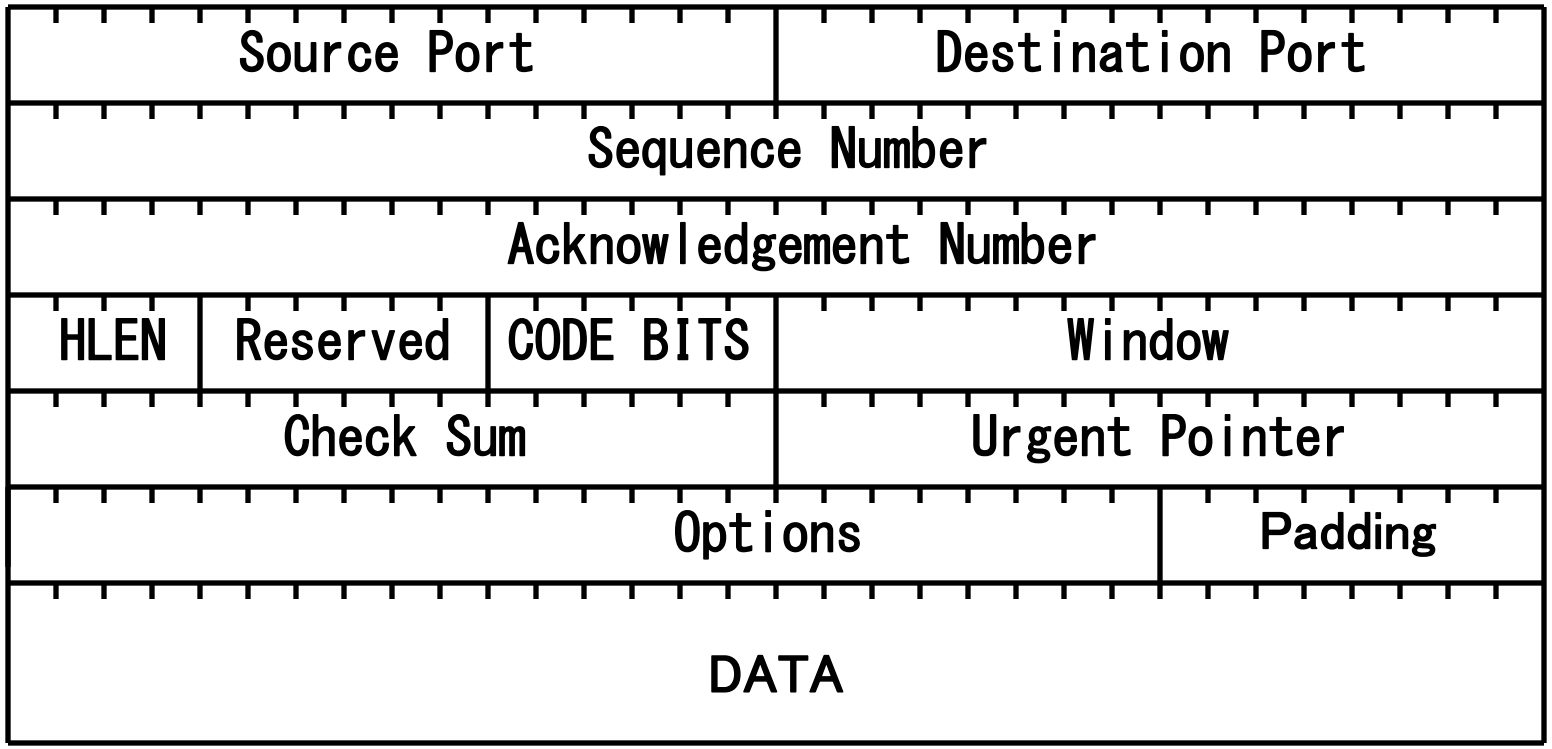
概要

- RFC 793 で定義
- 転送単位: セグメント
- ホスト間の信頼性の高い通信を提供
- 再転送付き肯定応答
  - 肯定応答: ACK
- スライディングウィンドウ
- バイト単位のウィンドウによるフロー制御
- 輻輳制御
- コネクションとポート

概要(2)

- コネクション型の通信形態
- ストリーム型のデータ転送
  - 単純バイト列
  - レコードの概念は無い

パケット形式



フィールド

- Source Port: 始点ポート
- Destination Port: 終点ポート
- Sequence Number: シーケンス番号
- Acknowledgement Number: ACK 番号
- HLEN: ヘッダー長
- CODE BITS: セグメントの内容を示すビット
  - SYN
  - FIN
  - RST
  - ACK
  - URG
  - PSH

フィールド(2)

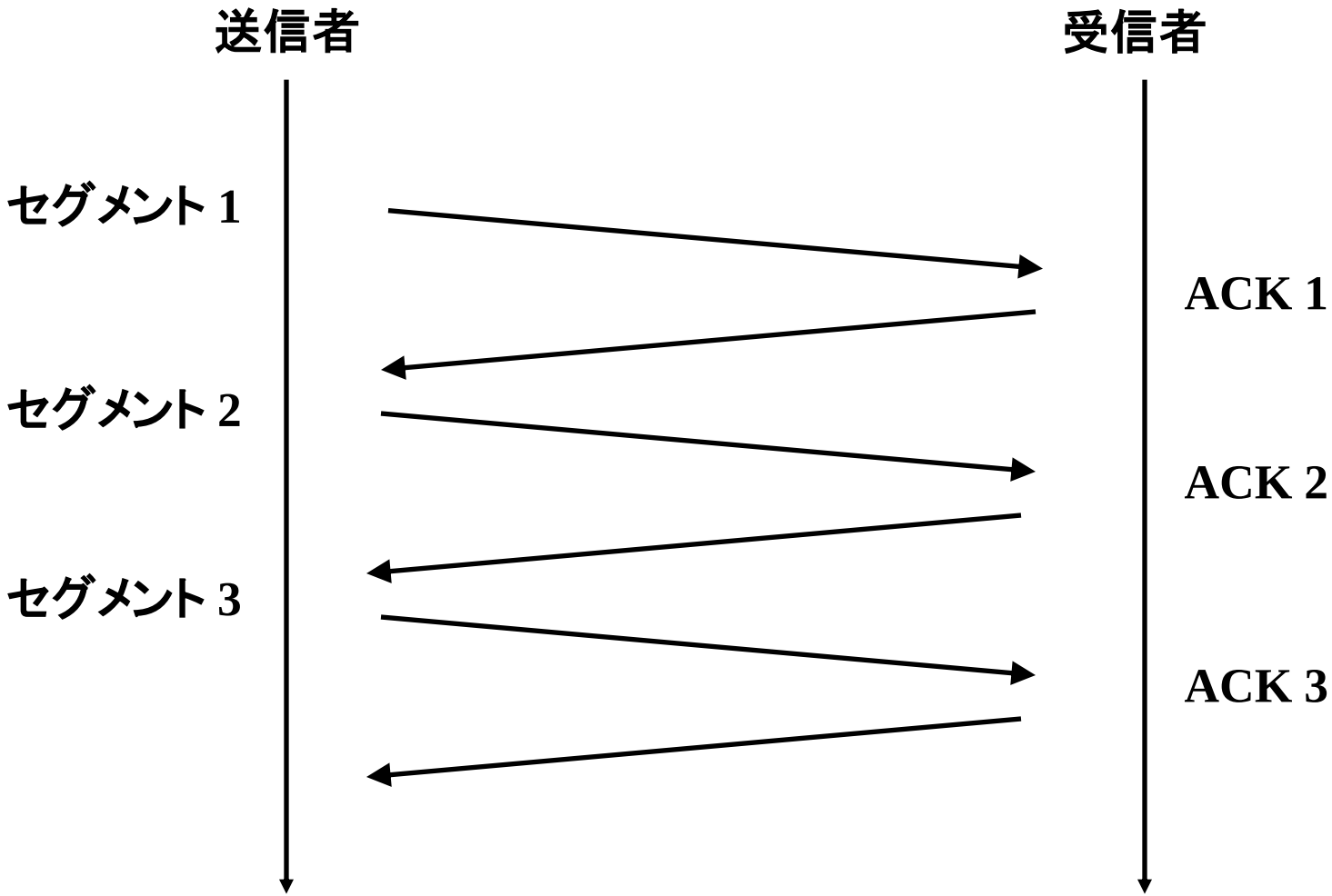
- Window: ウィンドウサイズ
- Check Sum: チェックサム
- Urgent Pointer: 緊急データポインタ
- Options: オプション

セグメント

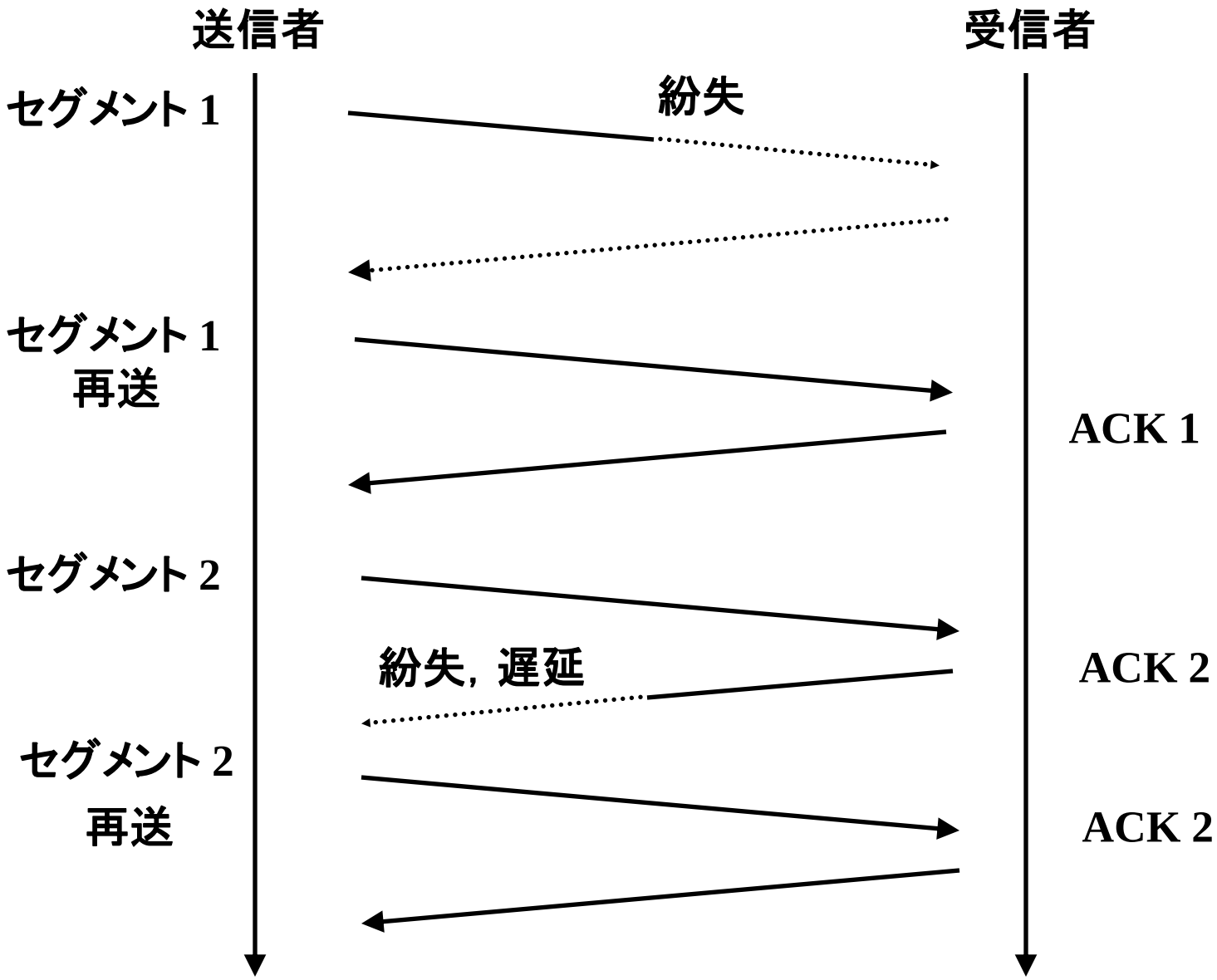
- TCP における転送単位
- シーケンス番号: データ(オクテット)に割り当てられた順序数
  - 32ビットの整数
  - $2^{32}-1$ の次は 0 (循環する)
- ネットワークの状況で大きさが変化

再転送付き ACK

- セグメントごとに ACK が必要



再転送付き ACK(2)

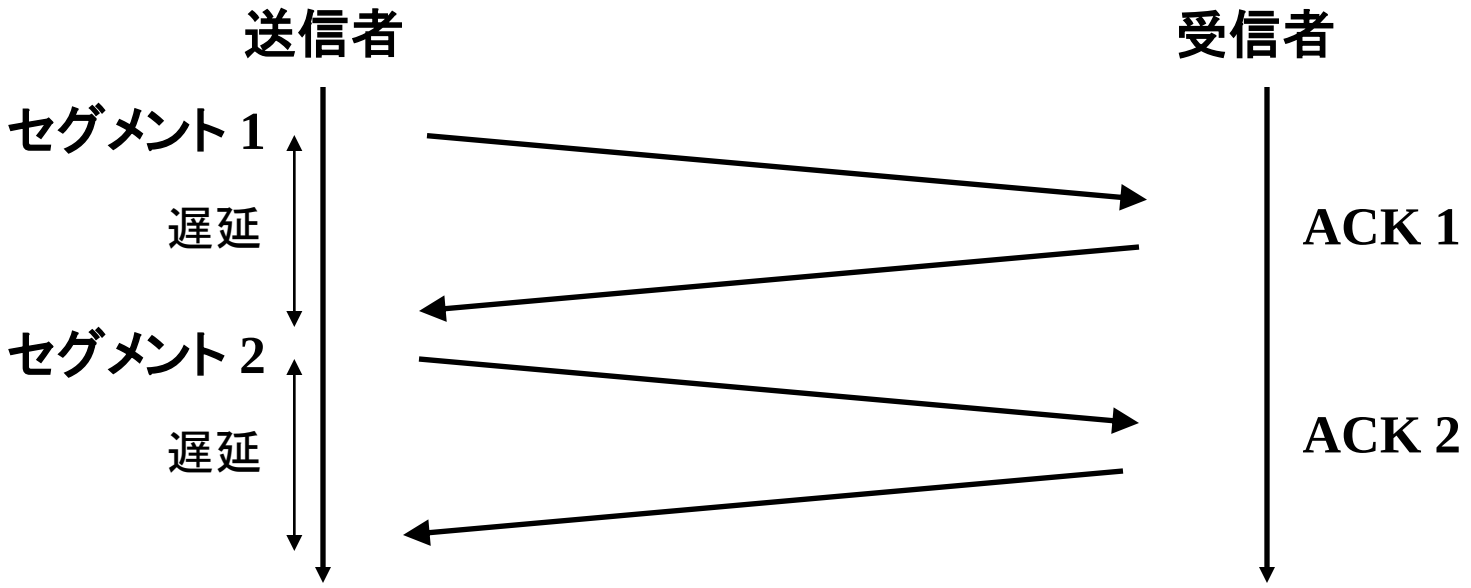


再転送付き ACK(3)

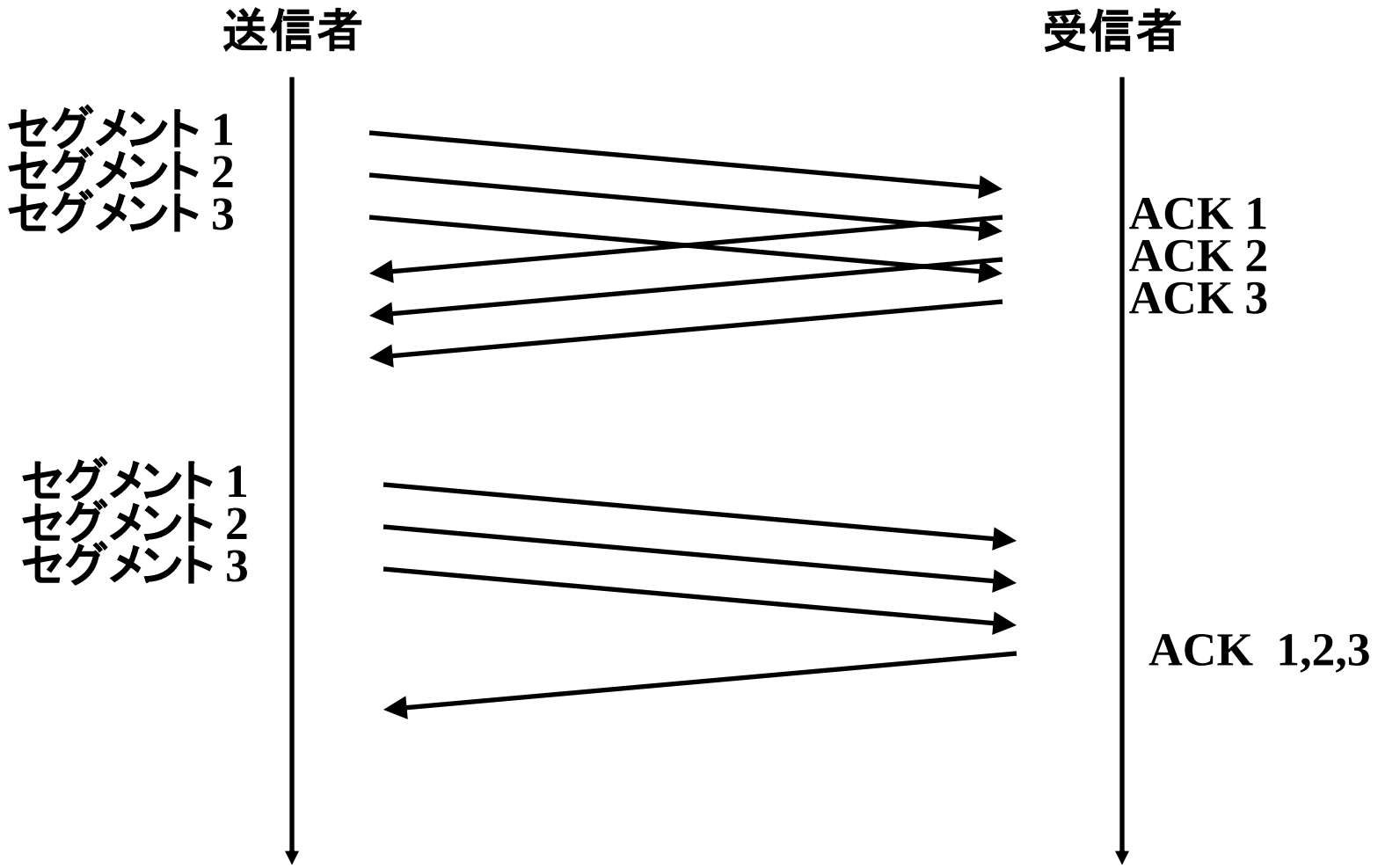
- ある時間以内に ACK が無ければ(タイムアウト)再転送する
- タイムアウト値はネットワークの状況で変化

スライディングウィンドウ

- 基本は「セグメント一つ送信して ACK を待つ」
  - でも効率が悪い
- ネットワーク内ををセグメントで埋める
- 確認応答を待たずに，送信して良いバイト列の集合
- 複数のセグメントの確認応答をまとめることも可能

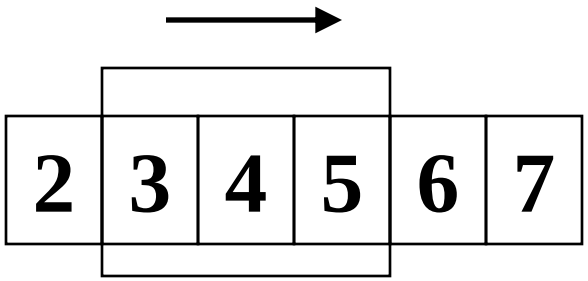


スライディングウィンドウ(2)



スライディングウィンドウ(3)

- ウィンドウの左側
  - ACK 受信済み
- ウィンドウの右側
  - 未送信
- ウィンドウ中
  - 送信済み
  - ウィンドウの一番左のセグメントの ACK を受信したら，ウィンドウが左に移動



ウィンドウ通知

- ウィンドウは，可変長
- 大きさは受信側が決定権をもつ
  - 受信側のバッファサイズ
- TCP ヘッダの Window フィールドで指定
  - 単位：オクテット
  - フィールドは16 ビット = 最大64 Kオクテット

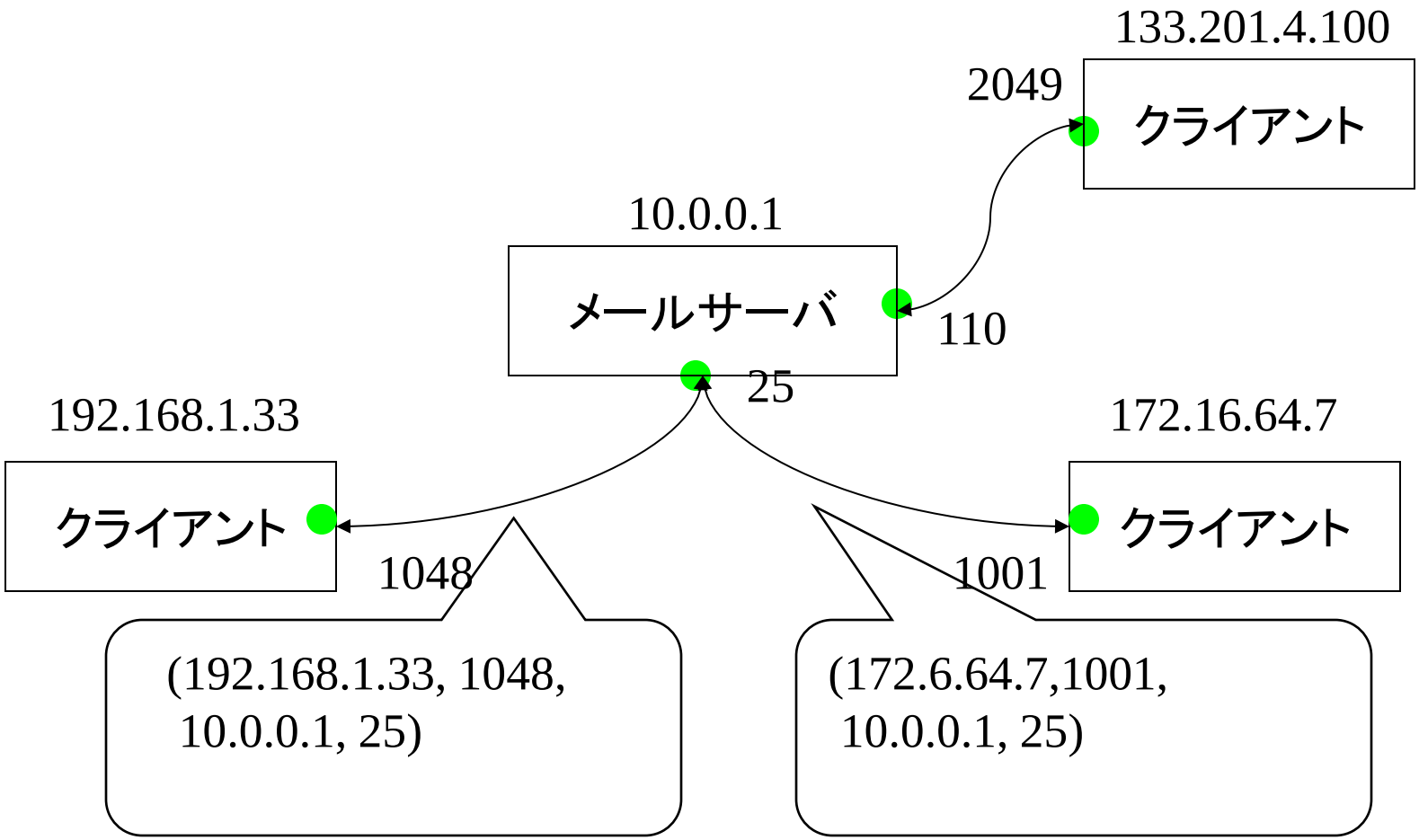
コネクションとポート

- ポート：サービスを指定
- 終点ホストのサービスを識別するのに利用
  - IPアドレスは終端ホストを識別
- コネクションで複数の同一サービスを識別
  - 同じポートを持つプロセスが存在
- コネクション
  - （始点アドレス，始点ポート，終点アドレス，終点ポート，プロトコル）

コネクションとポート(2)

- 代表的なサービス
  - SMTP　25/TCP
  - TELNET　23/TCP
  - SSH　22/TCP
  - FTP　20, 21/TCP
  - domain(DNS)　53/TCP
  - HTTP　80/TCP
  - HTTPS　443/TCP
  - POP　110/TCP

コネクションとポート(3)

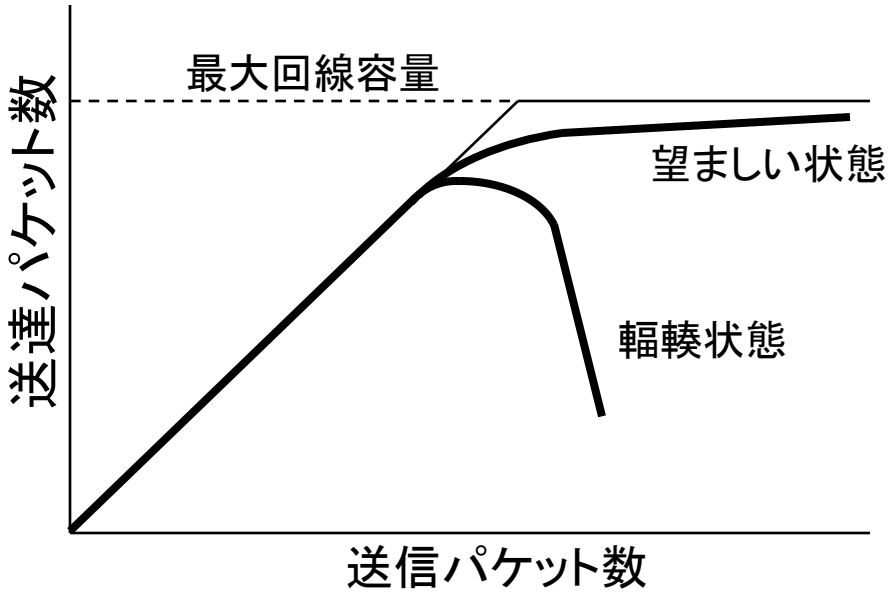


再送

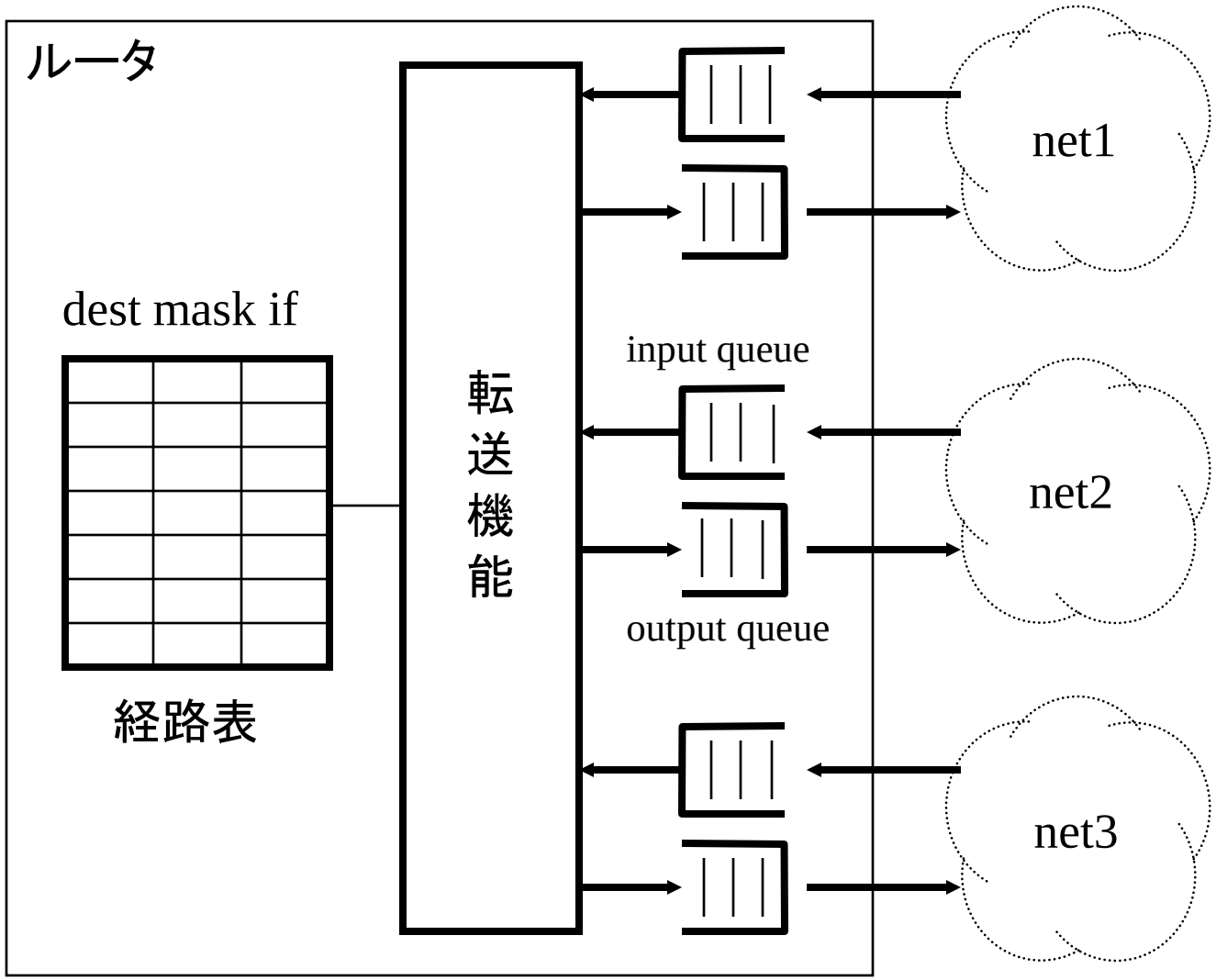
- ある時間内に確認応答が無ければ(=タイムアウト)再送
- 「ある時間」は，どのように決定?
  - リンク層の性能や現在のネットワークの状況に応じて
- 各セグメントの送信時刻と，それに対する確認応答の受信時刻の差
  - ラウンドトリップ時間 (RTT)

輻輳

- 輻輳（congestion）
  - ネットワーク上に過度のトラフィックが存在している状態
  - パケット喪失，性能の急激な劣化が発生

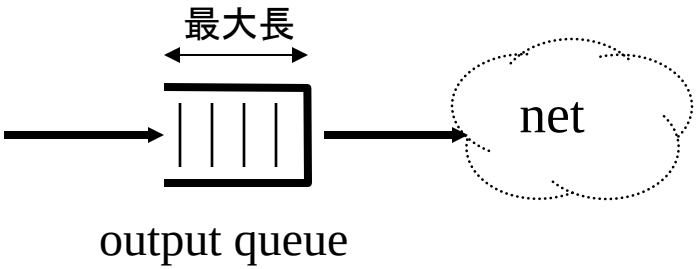


ルータの構成



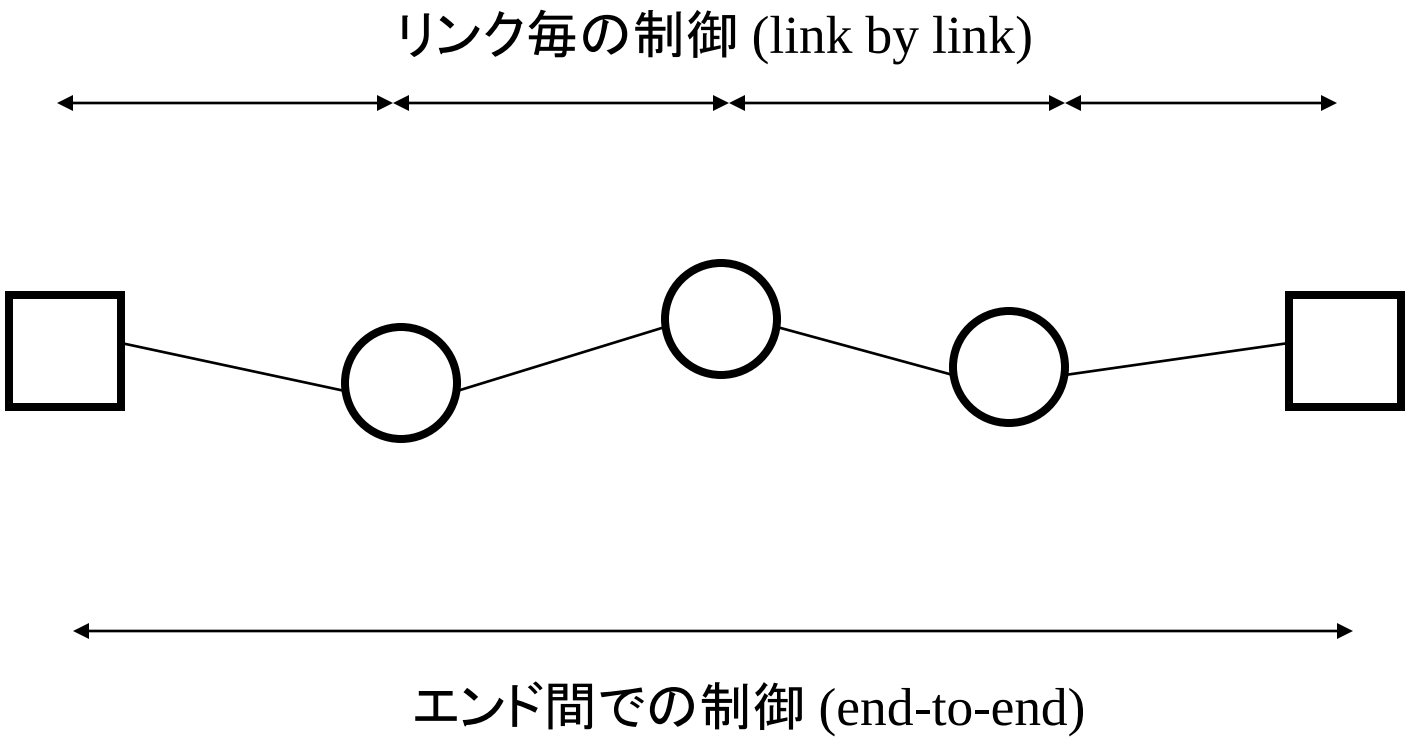
出力キューと溢れ

- 輻輳のほとんどは出力キューが溢れること



  - キューの大きさをむやみに大きくしてはいけない
    - ルータのリソースは有限
    - 大きな遅延が発生
  - どのように溢れさせるか(捨てるか)がポイント

リンク毎の制御，エンド間での制御



輻輳回避

- TCP はエンド−エンドで輻輳を回避する
  - リンク間では行わない
  - 自律分散
  - ネットワーク全体の性能を上げる
- 輻輳が起こったら TCP は転送量を減らす必要がある
- 輻輳を避ける二つの技術
  - 倍数減少輻輳回避
  - スロースタート
- 転送量を一旦大きく減らして，少しずつ増やす

倍数減少輻輳回避

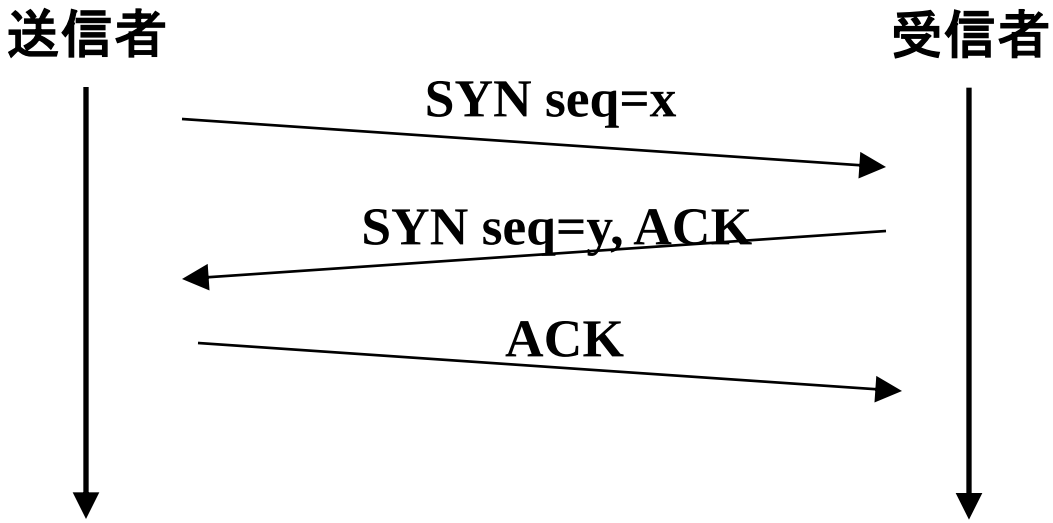
- 輻輳ウィンドウを使った制限
- 送信ウィンドウ = min(輻輳ウィンドウ，受信者ウインドウ)
- 安定状態では受信者のウィンドウと同じ
- 遅延(パケットの喪失)が起こると
  - 輻輳ウィンドウを半分にする(最小1セグメントまで)
  - タイムアウト値を大きくする
- 増やす時はスロースタート

スロースタート

- 初めからウィンドウサイズいっぱいの送信をしない
  - 1セグメントから始めてACKが返る毎に大きくする

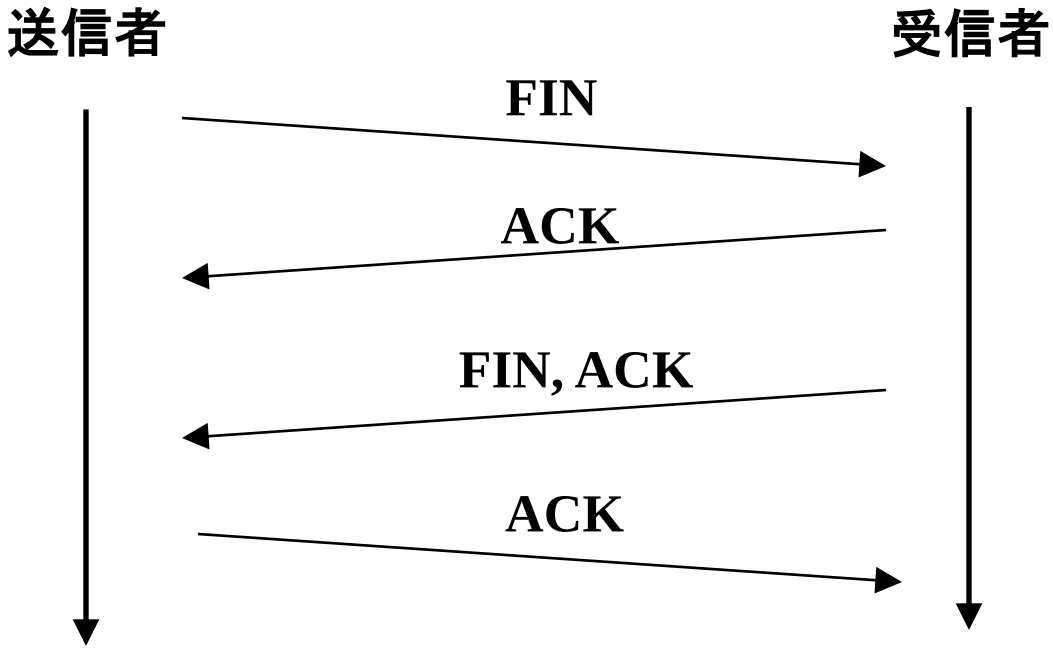
TCPコネクションの確立

- 3 ウェイハンドシェイク
  - SYN パケットの送信
- 初期シーケンス番号の一致
  - シーエケンス番号はランダムに決定
- 規格では，「4 μ秒に1つ加算されるカウンタ」を使う
  - 4.55 時間で一周



コネクションの終了方法

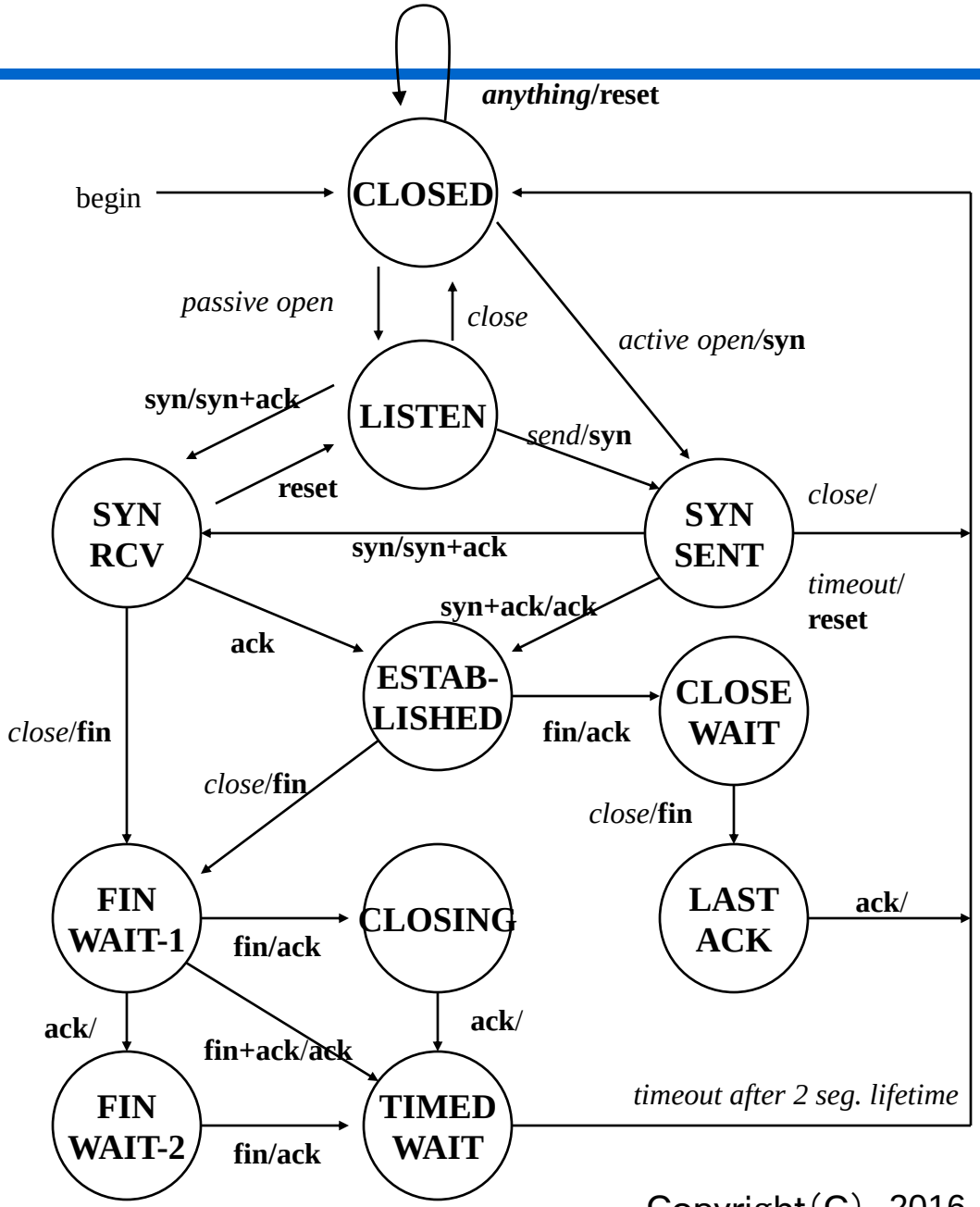
- FIN パケットを送信する
  - TCP は全二重なので片方向のみ閉じられる
    - half-open
  - 相手が異常終了した場合も half-open
- 両方向とも閉じられたら終了



コネクションのリセット

- 異常なコネクション(相手が異常状態など)の強制終了
- RST パケットを送信する

状態遷移



高速通信

- ギガビットイーサ，ATM などのメディアでは…
  - シーケンス番号をすぐに消費してしまう
- シーケンス番号は循環($2^{32}-1$ の次は0)する
- TTL時間内に同一シーケンス番号のデータグラムが回ってくるため順序関係が分からなくなる

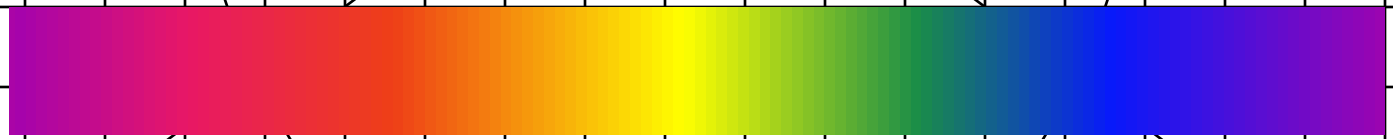
1000km，1Gbps

- ファイバの伝送速度 2．1x10^{8}m/s
- 片道 5msecの遅延，往復10msecの遅延
- 単純計算で 6．4Mbps のスループット

- 1Gbps を出すためには
  - 100Mバイトのウインドウサイズ
    - 100Mバイトのバッファを用意
    - TCPの拡張が必要
      - RFC1323 TCP Extensions for High Performance

ご質問は？

CECメディアエクスチェンジサービス



takada@mex.ad.jp